National University of Engineering (UNI)

School of Sciences

School of Computer Science

Curricular Plan 2026

– Reporte Final–

Última modificación: 2026

This document is based on the international proposal of ACM/IEEE-CS *Computing Curricula*, which also served as the basis for the Spanish version created by the Peruvian Computer Society (SPC).

ISBN:

All rights reserved. Ernesto Cuadros-Vargas, National University of Engineering (UNI).

Task Force

Dr. Ciro Nuñez Iturri
Acting Director of the School of Computer Science
email: *ciro.nunez.i@uni.edu.pe*

Dr. Carlos Javier Solano Salinas
Director of the Department of Computer Science
e-mail: *jsolano@uni.edu.pe*

Dr. José Luis Segovia-Juárez
PhD in Computer Science. Wayne State University (Detroit, USA) (2001)
Professor, Department of Computer Science. UNI
email: *jsegoviaj@uni.edu.pe*
http://www.tecnociencia9.com/jlsegovia/es

Dr. Marcos Antonio Alania Vicente
PhD in Sciences, University of Antwerp, Belgium (2017)
Professor, Department of Computer Science. UNI
Developer of AI algorithms for complex interdisciplinary processes
email: *alania.vicente@gmail.com*

MSc. Eduardo Yauri Lozano
MSc. in Computer Science. UNI (2024)
Professor, Department of Computer Science. UNI
email: *eduardo.yauri@uni.edu.pe*

MSc. Ronald Ricardo Martinez Chunga
MSc. Electronic Engineering Sciences (2021)
Professor, Department of Computer Science. UNI
email: *rmartinezch@uni.edu.pe*
https://www.linkedin.com/in/rmartinezchunga/

Cesar Martin Cruz-Salazar
Candidate for Master's in Automation and Instrumentation, UNI
Professor of Computer Science, UNI, Lima-Peru
email: *ccruz@uni.edu.pe*

Prof. Miguel Arrunategui Angulo
Professor, Department of Computer Science
email: *garrunateguia@uni.edu.pe*

Mag. Yuri Ccoicca
Department of Computer Science, UNI, Lima-Peru
email: *yccoicca@uni.edu.pe*

M.Sc. Raul Huarote Zegarra
Master's in Computer Science, UNT (2014)
Professor, Department of Computer Science, UNI, Lima-Peru
email: *rhuarotez@uni.edu.pe*

M.Sc. Carlos Alberto Bazán Cabanillas
Master of Science in Mathematics, National University of Engineering (UNI), 1980
Bachelor of Science in Mathematics, National University of Engineering (UNI), 1977
email: *cbazanc@uni.edu.pe*

Dr. Jose Alfredo Herrera-Quispe
PhD in Computer Science, UNSA (2013)
Faculty member, Department of Computer Science

Mag. Sidney Weisser Valer Quispe
Master of Science, concentration in Telematics, FIEE-UNI (2026)
Faculty member, Department of Computer Science
email: *svalerq@uni.edu.pe*

Mag. Johnny Quispe Obregón
Faculty member, Department of Computer Science
email: *johnny.quispe.o@uni.edu.pe*

Dr. Ernesto Cuadros-Vargas <*ecuadros@spc.org.pe*>
PhD in Computer Science, University of Sao Paulo, Brazil (2004)
Distinguished Speaker for the Association of Computing Machinery (ACM) (2025-2027)
Member of the Board of Governors of the IEEE Computer Society (2020-2023)
Member of the Steering Committee ACM/ IEEE-CS Computing Curricula 2020 (CS2020)
Member of the Steering Committee ACM/ IEEE-CS Computing Curricula/Computer Science (CS2013)
Executive Secretary of the Latin American Center for Computing Studies (CLEI) 2008-2016
email: *ecuadros@spc.org.pe*

Executive Summary

Este documento representa el informe final de la nueva malla curricular 2026 del Escuela Profesional de Ciencia de la Computación de la Universidad Nacional de Ingeniería (UNI) (*https://www.uni.edu.pe*) en la ciudad de Lima-Perú.

This document presents the curricular design of the Computer Science program under the international standards of the *Computing Curricula* CC2020, a reference framework developed jointly by the *Association for Computing Machinery* (ACM) and the *IEEE Computer Society* (IEEE-CS). CC2020, available at http://www.acm.org/education, synthesizes three decades of disciplinary evolution (from CS2001 to CC2020) and responds to the current demands of the fourth industrial revolution.

The proposal adapts this global reference to our regional context, incorporating:

- Local labor market requirements in technology
- Institutional strengths in applied research
- Emerging trends in computing education

Computing today presents the following undergraduate professional training profiles:

- Computer Engineering (CE) [Soldan et al., 2016],
- Computer Science (CS) [ACM/IEEE-CS/AAAI Joint Task Force, 2023],
- Information Systems (IS) [The Joint ACM/AIS IS2020 Task Force, 2020],
- Software Engineering (SE) [LeBlanc et al., 2015],
- Information Technology (IT) [ACM and IEEE-CS, 2017].
- Cybersecurity (CY) [ACM/IEEE-CS Joint Task Force on Cybersecurity Education, 2017].
- Data Science (DS) [ACM Data Science Task Force, 2021].

The pedagogical model is supported by three transformative pillars:

- **Technical excellence**: Mastery of computational fundamentals and progressive specialization
- **Disruptive innovation**: Agile methodologies, computational thinking, and startup creation
- **Social responsibility**: Professional ethics, technological sustainability, and community impact

This curriculum will produce professionals with:

- Ability to solve complex problems in global environments
- Competencies to lead digital transformation
- Sensitivity to the ethical challenges of artificial intelligence, privacy, and technological governance

This proposal incorporates continuous update mechanisms, ensuring relevance in the face of rapid technological obsolescence and alignment with international accreditation standards.

Contents

Acknowledgments **xxi**

Acronyms **xxiii**

1 Introduction **1**
1.1 Basic definitions . . . 1
1.2 Occupational Field and Job Market . . . 3
1.3 Importance of the Career in Society . . . 3
1.4 Institutional Info . . . 4
1.4.1 Mission . . . 4
1.4.2 Vision . . . 4
1.5 Accreditation information . . . 4
1.5.1 Career Outcomes . . . 4
1.5.2 Educational objectives . . . 5
1.5.3 Learning Outcomes . . . 5
1.6 Profiles . . . 6
1.6.1 Applicant profile . . . 6
1.6.2 Graduate profile . . . 6
1.6.3 ProfessionalProfile . . . 7
1.6.4 Faculty profile . . . 7
1.7 Degrees and Titles . . . 7
1.8 Resources to teach . . . 8
1.9 Document Organization . . . 8

2 Body of knowledge of Computer Science **9**
2.1 Artificial Intelligence (AI) . . . 11
2.1.1 AI/Introduction to AI: Concepts and Intelligent Behavior (CS Core: 2 hrs, KA Core: 1 hr) . . . 12
2.1.2 AI/Problem Characteristics and Agent Nature (CS Core: 1 hr, KA Core: 1 hr) 13
2.1.3 AI/State and Uninformed Search (CS Core: 1 hr, KA Core: 2 hrs) . . . 13
2.1.4 AI/Heuristic Search and CSP (CS Core: 1 hr, KA Core: 2 hrs) . . . 13
2.1.5 AI/Adversarial Search and Stochastic Optimization (CS Core: 1 hr, KA Core: 3 hrs) . . . 14
2.1.6 AI/Knowledge Representation and Probabilistic Reasoning Foundations (CS Core: 1 hr, KA Core: 1 hr) . . . 14
2.1.7 AI/Independence, Markov Models and Decision Making (CS Core: 1 hr, KA Core: 1 hr) . . . 15
2.1.8 AI/Machine Learning Fundamentals (CS Core: 1 hr, KA Core: 2 hrs) . . . 15
2.1.9 AI/Machine Learning Data, Representations and Evaluation (CS Core: 1 hr, KA Core: 2 hrs) . . . 15
2.1.10 AI/Neural Networks and Ethics (CS Core: 1 hr, KA Core: 1 hr) . . . 16
2.1.11 AI/Machine Learning Formulation and Evaluation (CS Core: 1 hr, KA Core: 2 hrs) . . . 16
2.1.12 AI/Deep Learning, RL and Ethics (CS Core: 1 hr, KA Core: 2 hrs) . . . 17
2.1.13 AI/Neural Network Training and Optimization (CS Core: 1 hr, KA Core: 1 hr) 17

2.1.14 AI/Regularization Techniques for Deep Networks (CS Core: 1 hr, KA Core: 1 hr) 17
2.1.15 AI/Convolutional and Residual Network Architectures (CS Core: 1 hr, KA Core: 1 hr) . . . 17
2.1.16 AI/Transformer Architectures and Generative Models (CS Core: 1 hr, KA Core: 1 hr) . . . 18
2.1.17 AI/AI Applications and Societal Impact (CS Core: 2 hrs, KA Core: 2 hrs) . . 18
2.1.18 AI/Additional AI Applications and Societal Impact (CS Core: 2 hrs, KA Core: 2 hrs) . . . 18
2.1.19 AI/Logical Representation and Reasoning . . . 19
2.1.20 AI/Probabilistic Representation and Reasoning . . . 19
2.1.21 AI/Planning . . . 20
2.1.22 AI/Agents and Cognitive Systems . . . 20
2.1.23 AI/Natural Language Processing . . . 21
2.1.24 AI/Robotics . . . 21
2.1.25 AI/Perception and Computer Vision . . . 22
2.2 Algorithmic Foundations (AL) . . . 23
2.2.1 AL/Foundational Data Structures (CS Core: 5 hrs, KA Core: 3 hrs) . . . 24
2.2.2 AL/Foundational Algorithms (CS Core: 4 hrs, KA Core: 2 hrs) . . . 24
2.2.3 AL/Advanced Algorithms (CS Core: 3 hrs, KA Core: 2 hrs) . . . 25
2.2.4 AL/Algorithmic Strategies (CS Core: 6 hrs) . . . 25
2.2.5 AL/Complexity Analysis Framework (CS Core: 2 hrs, KA Core: 1 hr) . . . 26
2.2.6 AL/Asymptotic Notation and Complexity Classes (CS Core: 1 hr, KA Core: 1 hr) . . . 26
2.2.7 AL/Complexity Analysis: Recursion, Amortization, and Tight Bounds (CS Core: 2 hrs, KA Core: 1 hr) . . . 27
2.2.8 AL/Computational Complexity Theory (CS Core: 2 hrs, KA Core: 1 hr) . . . 27
2.2.9 AL/Formal Languages and Automata (CS Core: 4 hrs, KA Core: 10 hrs) . . . 27
2.2.10 AL/Computability and Decidability (CS Core: 6 hrs, KA Core: 14 hrs) . . . 28
2.2.11 AL/Society, Ethics, and the Profession . . . 29
2.3 Architecture and Organization (AR) . . . 29
2.3.1 AR/Digital Logic and Digital Systems (KA Core: 3 hrs) . . . 30
2.3.2 AR/Machine-Level Data Representation (CS Core: 1 hr) . . . 30
2.3.3 AR/Assembly Level Machine Organization (CS Core: 1 hr, KA Core: 2 hrs) . 31
2.3.4 AR/Memory Hierarchy (CS Core: 6 hrs) . . . 31
2.3.5 AR/Interfacing and Communication (CS Core: 1 hr) . . . 32
2.3.6 AR/Functional Organization (KA Core: 2 hrs) . . . 32
2.3.7 AR/Performance and Energy Efficiency (KA Core: 3 hrs) . . . 32
2.3.8 AR/Heterogeneous Architectures (KA Core: 2 hrs) . . . 33
2.3.9 AR/Secure Processor Architectures (KA Core: 2 hrs) . . . 33
2.3.10 AR/Quantum Architectures (KA Core: 2 hrs) . . . 34
2.3.11 AR/Sustainability Issues . . . 35
2.4 Data Management (DM) . . . 35
2.4.1 DM/The Role of Data and the Data Life Cycle (CS Core: 2 hrs) . . . 36
2.4.2 DM/Core Database System Concepts (CS Core: 2 hrs, KA Core: 1 hr) . . . 36
2.4.3 DM/Data Modeling (CS Core: 2 hrs, KA Core: 3 hrs) . . . 37
2.4.4 DM/Relational Databases (CS Core: 1 hr, KA Core: 3 hrs) . . . 37
2.4.5 DM/Query Construction (CS Core: 2 hrs, KA Core: 4 hrs) . . . 38
2.4.6 DM/Query Processing (KA Core: 4 hrs) . . . 38
2.4.7 DM/DBMS Internals (KA Core: 4 hrs) . . . 39
2.4.8 DM/NoSQL Systems (KA Core: 2 hrs) . . . 39
2.4.9 DM/Data Security and Privacy (CS Core: 1 hr, KA Core: 2 hrs) . . . 40
2.4.10 DM/Data Analytics (KA Core: 3 hrs) . . . 40
2.4.11 DM/Distributed Databases/Cloud Computing . . . 41
2.4.12 DM/Semi-structured and Unstructured Databases . . . 41
2.4.13 DM/Society, Ethics, and the Profession . . . 41
2.5 Foundations of Programming Languages (FPL) . . . 42

2.5.1 FPL/Object-Oriented Programming Fundamentals (CS Core: 2 hrs, KA Core: 1 hr) . . . 43
2.5.2 FPL/Object-Oriented Programming: Encapsulation, Subtyping, and Reflection (CS Core: 2 hrs, KA Core: 1 hr) . . . 44
2.5.3 FPL/Functional Programming (CS Core: 4 hrs, KA Core: 3 hrs) . . . 44
2.5.4 FPL/Logic Programming (KA Core: 2 hrs) . . . 45
2.5.5 FPL/Shell Scripting (CS Core: 2 hrs) . . . 46
2.5.6 FPL/Event-Driven and Reactive Programming (CS Core: 2 hrs, KA Core: 2 hrs) 46
2.5.7 FPL/Parallel and Distributed Computing (CS Core: 2 hrs, KA Core: 2 hrs) . . 46
2.5.8 FPL/Aspect-Oriented Programming . . . 47
2.5.9 FPL/Type Systems Fundamentals (CS Core: 2 hrs, KA Core: 2 hrs) . . . 47
2.5.10 FPL/Type Systems: Polymorphism and Complementary Typing (CS Core: 1 hr, KA Core: 1 hr) . . . 48
2.5.11 FPL/Advanced Type Theory (CS Core: 2 hrs, KA Core: 2 hrs) . . . 48
2.5.12 FPL/Systems Execution and Memory Model (CS Core: 2 hrs) . . . 49
2.5.13 FPL/Memory Management (CS Core: 1 hr) . . . 49
2.5.14 FPL/Language Translation and Execution (CS Core: 2 hrs, KA Core: 3 hrs) . 50
2.5.15 FPL/Program Abstraction and Representation (KA Core: 3 hrs) . . . 50
2.5.16 FPL/Syntax Analysis . . . 51
2.5.17 FPL/Compiler Semantic Analysis . . . 51
2.5.18 FPL/Program Analysis and Analyzers . . . 52
2.5.19 FPL/Code Generation . . . 52
2.5.20 FPL/Run-time Behavior and Systems . . . 53
2.5.21 FPL/Advanced Programming Constructs . . . 53
2.5.22 FPL/Language Pragmatics . . . 54
2.5.23 FPL/Formal Semantics . . . 54
2.5.24 FPL/Formal Development Methodologies . . . 55
2.5.25 FPL/Design Principles of Programming Languages . . . 55
2.5.26 FPL/Society, Ethics, and the Profession . . . 56
2.6 Graphics and Interactive Techniques (GIT) . . . 56
2.6.1 GIT/Fundamental Concepts of Graphics and Interactive Techniques (CS Core: 4 hrs, KA Core: 3 hrs) . . . 56
2.6.2 GIT/Visualization (KA Core: 6 hrs) . . . 57
2.6.3 GIT/Applied Rendering and Techniques (KA Core: 15 hrs) . . . 57
2.6.4 GIT/Geometric Modeling (KA Core: 6 hrs) . . . 57
2.6.5 GIT/Shading and Advanced Rendering (KA Core: 6 hrs) . . . 58
2.6.6 GIT/Computer Animation (KA Core: 6 hrs) . . . 58
2.6.7 GIT/Simulation (KA Core: 6 hrs) . . . 58
2.6.8 GIT/Immersion (KA Core: 6 hrs) . . . 59
2.6.9 GIT/Interaction (KA Core: 4 hrs) . . . 59
2.6.10 GIT/Image Processing (KA Core: 6 hrs) . . . 59
2.6.11 GIT/Tangible/Physical Computing (KA Core: 6 hrs) . . . 59
2.6.12 GIT/Society, Ethics, and the Profession . . . 60
2.7 Human-Computer Interaction (HCI) . . . 60
2.7.1 HCI/Understanding the User: Individual goals and interactions with others (CS Core: 2 hrs, KA Core: 5 hrs) . . . 61
2.7.2 HCI/Accountability and Responsibility in Design (CS Core: 2 hrs, KA Core: 2 hrs) . . . 61
2.7.3 HCI/Accessibility and Inclusive Design (CS Core: 2 hrs, KA Core: 2 hrs) . . . 61
2.7.4 HCI/Evaluating the Design (CS Core: 1 hr, KA Core: 2 hrs) . . . 62
2.7.5 HCI/System Design (CS Core: 1 hr, KA Core: 5 hrs) . . . 62
2.7.6 HCI/Society, Ethics, and the Profession . . . 62
2.8 Mathematical and Statistical Foundations (MSF) . . . 63
2.8.1 MSF/Discrete Mathematics (CS Core: 15 hrs, KA Core: 6 hrs) . . . 63
2.8.2 MSF/Advanced Discrete Mathematics (CS Core: 15 hrs, KA Core: 6 hrs) . . . 64
2.8.3 MSF/Probability (CS Core: 11 hrs, KA Core: 29 hrs) . . . 64

2.8.4 MSF/Statistics (CS Core: 10 hrs, KA Core: 30 hrs) . . . 64
2.8.5 MSF/Linear Algebra (CS Core: 5 hrs, KA Core: 35 hrs) . . . 65
2.8.6 MSF/Calculus (KA Core: 40 hrs) . . . 65
2.9 Networking and Communication (NC) . . . 66
2.9.1 NC/Fundamentals of Networks and Communications (CS Core: 3 hrs) . . . 67
2.9.2 NC/Networked Applications (CS Core: 4 hrs) . . . 67
2.9.3 NC/Reliability Support (KA Core: 6 hrs) . . . 68
2.9.4 NC/Routing and Forwarding (KA Core: 4 hrs) . . . 68
2.9.5 NC/Single Hop Communication (KA Core: 3 hrs) . . . 68
2.9.6 NC/Network Security (KA Core: 3 hrs) . . . 68
2.9.7 NC/Mobility Support (KA Core: 4 hrs) . . . 69
2.9.8 NC/Emerging Topics (KA Core: 4 hrs) . . . 69
2.10 Operating Systems (OS) . . . 69
2.10.1 OS/Role and Purpose of Operating Systems (CS Core: 2 hrs) . . . 70
2.10.2 OS/Principles of Operating System (CS Core: 2 hrs) . . . 71
2.10.3 OS/Concurrency (CS Core: 2 hrs, KA Core: 1 hr) . . . 71
2.10.4 OS/Protection and Safety (CS Core: 2 hrs, KA Core: 1 hr) . . . 72
2.10.5 OS/Scheduling (KA Core: 2 hrs) . . . 73
2.10.6 OS/Process Model (KA Core: 2 hrs) . . . 73
2.10.7 OS/Memory Management (KA Core: 1 hr) . . . 74
2.10.8 OS/Device management (KA Core: 1 hr) . . . 74
2.10.9 OS/File Systems API and Implementation (KA Core: 2 hrs) . . . 75
2.10.10 OS/Advanced File systems (KA Core: 1 hr) . . . 75
2.10.11 OS/Virtualization (KA Core: 1 hr) . . . 76
2.10.12 OS/Real-time and Embedded Systems (KA Core: 1 hr) . . . 76
2.10.13 OS/Fault tolerance (KA Core: 1 hr) . . . 77
2.10.14 OS/Society, Ethics, and the Profession . . . 77
2.11 Parallel and Distributed Computing (PDC) . . . 78
2.11.1 PDC/Programs: Declarative Parallelism (CS Core: 1 hr, KA Core: 1 hr) . . . 79
2.11.2 PDC/Programs: Starting Activities (CS Core: 1 hr, KA Core: 1 hr) . . . 79
2.11.3 PDC/Programs: Execution Properties (CS Core: 1 hr, KA Core: 1 hr) . . . 79
2.11.4 PDC/Programs: Distribution (CS Core: 1 hr, KA Core: 1 hr) . . . 79
2.11.5 PDC/Programs: Implementation Mappings (CS Core: 1 hr, KA Core: 1 hr) . . . 80
2.11.6 PDC/GPU Programming . . . 80
2.11.7 PDC/Communication (CS Core: 2 hrs, KA Core: 4 hrs) . . . 81
2.11.8 PDC/Communication: Properties and Extensions (CS Core: 1 hr, KA Core: 1 hr) . . . 81
2.11.9 PDC/Memory and Consistency (CS Core: 1 hr, KA Core: 1 hr) . . . 82
2.11.10 PDC/Coordination (CS Core: 1 hr, KA Core: 3 hrs) . . . 82
2.11.11 PDC/Coordination: Synchronization and Atomicity (CS Core: 1 hr, KA Core: 2 hrs) . . . 82
2.11.12 PDC/Coordination: Advanced Properties (CS Core: 1 hr, KA Core: 2 hrs) . . . 83
2.11.13 PDC/Evaluation (CS Core: 1 hr, KA Core: 3 hrs) . . . 83
2.11.14 PDC/Evaluation: Formal Methods (CS Core: 1 hr, KA Core: 1 hr) . . . 84
2.11.15 PDC/Evaluation: Testing and Measurement (CS Core: 1 hr, KA Core: 1 hr) . . . 84
2.11.16 PDC/Algorithms (CS Core: 1 hr, KA Core: 3 hrs) . . . 84
2.11.17 PDC/Algorithms: Application Domains Survey (CS Core: 1 hr, KA Core: 3 hrs) 85
2.11.18 PDC/Algorithms: Algorithmic Domains (CS Core: 1 hr, KA Core: 3 hrs) . . . 85
2.12 Software Development Fundamentals (SDF) . . . 86
2.12.1 SDF/Basic Programming Concepts and Constructs (CS Core: 10 hrs) . . . 87
2.12.2 SDF/Programming Practices: Testing, Debugging, and Documentation (CS Core: 10 hrs) . . . 87
2.12.3 SDF/Fundamental Data Structures (CS Core: 12 hrs) . . . 88
2.12.4 SDF/Algorithms (CS Core: 6 hrs) . . . 88
2.12.5 SDF/Software Development Practices (CS Core: 5 hrs) . . . 88
2.12.6 SDF/Society, Ethics, and the Profession . . . 89

2.13 Software Engineering (SE) . . . 89
2.13.1 SE/Team Collaboration and Dynamics (CS Core: 2 hrs, KA Core: 2 hrs) . . . 90
2.13.2 SE/Diversity, Stakeholders and Distributed Teams (CS Core: 1 hr, KA Core: 1 hr) . . . 91
2.13.3 SE/Version Control and CI/CD (CS Core: 1 hr, KA Core: 2 hrs) . . . 91
2.13.4 SE/Development Environment and IDE (CS Core: 1 hr, KA Core: 2 hrs) . . . 92
2.13.5 SE/Requirements Engineering (KA Core: 2 hrs) . . . 92
2.13.6 SE/Requirements Evolution and Estimation (KA Core: 1 hr) . . . 93
2.13.7 SE/Principles and Software Architecture (CS Core: 1 hr, KA Core: 2 hrs) . . . 93
2.13.8 SE/API Design and Modeling (CS Core: 1 hr, KA Core: 1 hr) . . . 93
2.13.9 SE/Software Design Quality and Evaluation (CS Core: 1 hr, KA Core: 2 hrs) . 94
2.13.10 SE/Coding Practices (CS Core: 1 hr, KA Core: 2 hrs) . . . 94
2.13.11 SE/Large-Scale Construction and Process (CS Core: 1 hr, KA Core: 2 hrs) . . 95
2.13.12 SE/Verification and validation Concepts (CS Core: 1 hr, KA Core: 1 hr) . . . 96
2.13.13 SE/Test Planning and Types (CS Core: 1 hr, KA Core: 1 hr) . . . 96
2.13.14 SE/Advanced Testing Practices (CS Core: 1 hr, KA Core: 1 hr) . . . 96
2.13.15 SE/Testing Tools and Analysis (CS Core: 1 hr, KA Core: 1 hr) . . . 97
2.13.16 SE/Performance Testing and Benchmarking (CS Core: 1 hr, KA Core: 1 hr) . 97
2.13.17 SE/API Compatibility and Versioning (KA Core: 2 hrs) . . . 98
2.13.18 SE/Refactoring Techniques (KA Core: 1 hr) . . . 98
2.13.19 SE/Reliability Concepts (KA Core: 2 hrs) . . . 99
2.13.20 SE/Reliability Engineering (KA Core: 1 hr) . . . 99
2.13.21 SE/Formal Methods . . . 99
2.14 Security (SEC) . . . 100
2.14.1 SEC/Security Mindset and Principles (CS Core: 1 hr, KA Core: 3 hrs) . . . 101
2.14.2 SEC/Access Control and Applications (CS Core: 1 hr, KA Core: 3 hrs) . . . 102
2.14.3 SEC/AI and Hardware Security (CS Core: 1 hr, KA Core: 2 hrs) . . . 102
2.14.4 SEC/Security Ethics, Law and Society (CS Core: 1 hr, KA Core: 2 hrs) . . . 102
2.14.5 SEC/Privacy and Security by Design (CS Core: 1 hr, KA Core: 1 hr) . . . 103
2.14.6 SEC/Injection and Input Validation (CS Core: 1 hr, KA Core: 2 hrs) . . . 103
2.14.7 SEC/Memory Safety and Types (CS Core: 1 hr, KA Core: 1 hr) . . . 104
2.14.8 SEC/Information Flow and Non-Interference (CS Core: 1 hr, KA Core: 1 hr) . 104
2.14.9 SEC/Malware Analysis and Advanced Security (CS Core: 1 hr, KA Core: 3 hrs) 104
2.14.10 SEC/Cryptography Fundamentals (CS Core: 1 hr, KA Core: 2 hrs) . . . 105
2.14.11 SEC/Cryptographic Algorithms (CS Core: 1 hr, KA Core: 2 hrs) . . . 105
2.14.12 SEC/Protocols, Applications and Advanced Cryptography (CS Core: 1 hr, KA Core: 2 hrs) . . . 106
2.14.13 SEC/Threat Analysis and Security Engineering (CS Core: 1 hr, KA Core: 4 hrs) 106
2.14.14 SEC/Security Design and Controls Engineering (CS Core: 1 hr, KA Core: 3 hrs) 107
2.14.15 SEC/Trusted Computing and Privacy Engineering (CS Core: 1 hr, KA Core: 3 hrs) . . . 107
2.14.16 SEC/Digital Forensics Techniques (KA Core: 4 hrs) . . . 108
2.14.17 SEC/Legal Aspects and Incident Handling (KA Core: 3 hrs) . . . 108
2.14.18 SEC/Security Governance, Management and Policy (KA Core: 2 hrs) . . . 109
2.14.19 SEC/Risk and Data Management (KA Core: 1 hr) . . . 109
2.15 Society, Ethics, and the Profession (SEP) . . . 109
2.15.1 SEP/Social Impact and AI (CS Core: 2 hrs, KA Core: 1 hr) . . . 110
2.15.2 SEP/Digital Equity and Accessibility (CS Core: 2 hrs, KA Core: 1 hr) . . . 111
2.15.3 SEP/Ethical Foundations and Frameworks (CS Core: 2 hrs, KA Core: 1 hr) . 111
2.15.4 SEP/Ethical Analysis Methods and Tools (CS Core: 1 hr, KA Core: 1 hr) . . . 112
2.15.5 SEP/Professional Values, Nature and Continuing Development (CS Core: 1 hr, KA Core: 1 hr) . . . 112
2.15.6 SEP/Ethics Codes and Legal Accountability (CS Core: 1 hr, KA Core: 1 hr) . 113
2.15.7 SEP/Professional Responsibility and Public Policy (CS Core: 1 hr, KA Core: 1 hr) . . . 113
2.15.8 SEP/Ethical Dissent and Cultural Context (CS Core: 1 hr, KA Core: 1 hr) . . 114

2.15.9 SEP/Workplace Ethics (CS Core: 1 hr, KA Core: 1 hr) . . . 114
2.15.10 SEP/Intellectual Property Foundations (CS Core: 1 hr, KA Core: 1 hr) . . . 114
2.15.11 SEP/Licensing, Plagiarism and Authorship (CS Core: 1 hr, KA Core: 1 hr) . . 115
2.15.12 SEP/Digital IP Rights and Copyright (CS Core: 1 hr, KA Core: 1 hr) . . . 115
2.15.13 SEP/Open Source, Piracy and DRM (CS Core: 1 hr, KA Core: 1 hr) . . . 115
2.15.14 SEP/Data, Surveillance and Privacy Technology (CS Core: 1 hr, KA Core: 1 hr) 115
2.15.15 SEP/Privacy Philosophical Foundations and Legislation (CS Core: 1 hr, KA Core: 1 hr) . . . 116
2.15.16 SEP/Civil Liberties and Expression (CS Core: 1 hr, KA Core: 1 hr) . . . 116
2.15.17 SEP/Technical and Professional Communication (CS Core: 1 hr, KA Core: 1 hr) 117
2.15.18 SEP/Team Communication and Culture (CS Core: 1 hr, KA Core: 1 hr) . . . 117
2.15.19 SEP/Environmental Footprint of Computing Systems (CS Core: 1 hr, KA Core: 1 hr) . . . 118
2.15.20 SEP/Systemic Effects and Social Context (CS Core: 1 hr, KA Core: 1 hr) . . . 118
2.15.21 SEP/Sustainable Design and Pervasive Computing (CS Core: 1 hr, KA Core: 1 hr) . . . 118
2.15.22 SEP/Foundations and Early History of Computing (CS Core: 1 hr, KA Core: 1 hr) . . . 119
2.15.23 SEP/Modern History: PC, Internet and AI (CS Core: 1 hr, KA Core: 1 hr) . . 119
2.15.24 SEP/Economic Models and Strategies (KA Core: 1 hr) . . . 120
2.15.25 SEP/Automation, AI and Economic Ethics (KA Core: 1 hr) . . . 120
2.15.26 SEP/Computer Crimes and Attacks (CS Core: 1 hr, KA Core: 1 hr) . . . 120
2.15.27 SEP/Security Laws, Policies and Responsibility (CS Core: 1 hr, KA Core: 1 hr) 121
2.15.28 SEP/Identity and Diversity in Computing (CS Core: 1 hr, KA Core: 1 hr) . . 121
2.15.29 SEP/DEIA Historical and Cross-Cultural Context (CS Core: 1 hr, KA Core: 1 hr) . . . 122
2.15.30 SEP/Inclusion and Computing Practices (CS Core: 1 hr, KA Core: 1 hr) . . . 122
2.15.31 SEP/Accessible Design (CS Core: 1 hr, KA Core: 1 hr) . . . 122
2.16 System Fundamentals (SF) . . . 123
2.16.1 SF/Hardware Components and Logic (CS Core: 2 hrs) . . . 123
2.16.2 SF/Layers, Abstractions and Scalability (CS Core: 2 hrs) . . . 124
2.16.3 SF/Digital Systems and Logic (CS Core: 2 hrs) . . . 124
2.16.4 SF/State Machines (CS Core: 2 hrs) . . . 124
2.16.5 SF/Sequential and Parallel Processing (CS Core: 2 hrs) . . . 125
2.16.6 SF/Resource Management (CS Core: 1 hr, KA Core: 1 hr) . . . 125
2.16.7 SF/Latency, Cache and Memory Hierarchy (CS Core: 2 hrs, KA Core: 2 hrs) . 126
2.16.8 SF/Virtualization and Isolation (CS Core: 1 hr, KA Core: 1 hr) . . . 126
2.16.9 SF/Performance Metrics and Benchmarks (CS Core: 1 hr, KA Core: 1 hr) . . 127
2.16.10 SF/Analytical Performance Models (CS Core: 1 hr, KA Core: 1 hr) . . . 127
2.16.11 SF/Events, Tools and Experimentation (CS Core: 1 hr, KA Core: 1 hr) . . . 127
2.16.12 SF/System Reliability (CS Core: 2 hrs, KA Core: 1 hr) . . . 128
2.16.13 SF/System Security (CS Core: 2 hrs, KA Core: 1 hr) . . . 128
2.16.14 SF/System Design (CS Core: 2 hrs, KA Core: 1 hr) . . . 129
2.16.15 SF/Society, Ethics, and the Profession . . . 129
2.17 Specialized Platform Development (SPD) . . . 129
2.17.1 SPD/Common Aspects: Platforms, APIs and Constraints (CS Core: 3 hrs, KA Core: 2 hrs) . . . 130
2.17.2 SPD/Common Aspects: Platform Languages and Web Programming Patterns (CS Core: 2 hrs, KA Core: 1 hr) . . . 131
2.17.3 SPD/Frontend, Frameworks and Web Standards (KA Core: 3 hrs) . . . 131
2.17.4 SPD/Cloud Architecture and Data Management (KA Core: 3 hrs) . . . 131
2.17.5 SPD/Mobile Platforms: Development Fundamentals (KA Core: 2 hrs) . . . 132
2.17.6 SPD/Mobile Platforms: Architecture and Constraints (KA Core: 2 hrs) . . . 132
2.17.7 SPD/Mobile Platforms: Capabilities and Asynchronous Computing (KA Core: 1 hr) . . . 132
2.17.8 SPD/Robot Platforms (KA Core: 4 hrs) . . . 132

2.17.9 SPD/Embedded Characteristics and Architectures (KA Core: 1 hr) 133
2.17.10 SPD/Systems, Programming and Sensors (KA Core: 2 hrs) 133
2.17.11 SPD/Real-Time, Memory and Verification (KA Core: 3 hrs) 134
2.17.12 SPD/Platform History, Types and Characteristics (KA Core: 1 hr) 134
2.17.13 SPD/Sensors, Controllers and eSports Ecosystems (KA Core: 1 hr) 135
2.17.14 SPD/Architectures and Rendering (KA Core: 1 hr) 135
2.17.15 SPD/Algorithms, Media and Virtual Worlds (KA Core: 1 hr) 135
2.17.16 SPD/Languages, APIs and Development Tools (KA Core: 1 hr) 135
2.17.17 SPD/Game Engines (KA Core: 1 hr) 136
2.17.18 SPD/Game Design Foundations (KA Core: 1 hr) 136
2.17.19 SPD/Genres, Audiences and Social Reach (KA Core: 1 hr) 136
2.17.20 SPD/Interactive Data and Analytics Platforms (KA Core: 2 hrs) 137
2.17.21 SPD/AI Programming and Quantum Platforms (KA Core: 1 hr) 137
2.17.22 SPD/SPD-SEP/Mobile 138
2.17.23 SPD/SPD-SEP/Web 138
2.17.24 SPD/Intellectual Property in Games 138
2.17.25 SPD/Game Interface and Accessibility 139
2.17.26 SPD/Health, Safety and Sustainability 139
2.17.27 SPD/Representation, Inclusion and Equality 139
2.17.28 SPD/SPD-SEP/Robotics 140
2.17.29 SPD/SPD-SEP/Interactive 140

3 Body of knowledge of Basic Sciences for Computing 141
3.1 Calculus and Analysis (CAN) 142
3.1.1 CAN/Line Integrals and Vector Fields (CS Core: 1 hr, KA Core: 1 hr) 142
3.1.2 CAN/Multiple Integrals (CS Core: 1 hr, KA Core: 1 hr) 143
3.1.3 CAN/Multivariable Transformations (CS Core: 1 hr, KA Core: 1 hr) 143
3.1.4 CAN/Surface Integrals and Fundamental Theorems (CS Core: 1 hr, KA Core: 1 hr) 143
3.1.5 CAN/Vector Functions of a Real Variable (CS Core: 1 hr, KA Core: 1 hr) . . 144
3.1.6 CAN/Parametric Equations and Polar Coordinates (CS Core: 1 hr, KA Core: 1 hr) 144
3.1.7 CAN/Limits and Continuity (CS Core: 1 hr, KA Core: 1 hr) 144
3.1.8 CAN/Differential Calculus (CS Core: 1 hr, KA Core: 1 hr) 144
3.1.9 CAN/Antiderivatives and Integration Techniques (CS Core: 1 hr, KA Core: 1 hr) 145
3.1.10 CAN/Improper Integrals (CS Core: 1 hr, KA Core: 1 hr) 145
3.1.11 CAN/Sequences, Series, and Power Series (CS Core: 1 hr, KA Core: 1 hr) . . . 145
3.1.12 CAN/Multivariable Differential Calculus (CS Core: 1 hr, KA Core: 1 hr) . . . 145
3.1.13 CAN/Harmonic Analysis and Fourier Theory (CS Core: 1 hr, KA Core: 1 hr) . 146
3.2 Algebra and Number Theory (ANT) 146
3.2.1 ANT/Matrices and Systems of Linear Equations (CS Core: 1 hr, KA Core: 1 hr) 146
3.2.2 ANT/Vector Spaces and Linear Transformations (CS Core: 1 hr, KA Core: 1 hr) 147
3.2.3 ANT/Eigenvalues, Inner Product Spaces, and Quadratic Forms (CS Core: 1 hr, KA Core: 1 hr) 147
3.2.4 ANT/Group Theory (CS Core: 1 hr, KA Core: 1 hr) 148
3.2.5 ANT/Rings, Fields, and Galois Theory (CS Core: 1 hr, KA Core: 1 hr) 148
3.2.6 ANT/Cryptography and Coding Theory (CS Core: 1 hr, KA Core: 1 hr) . . . 148
3.3 Discrete Mathematics and Combinatorics (DMC) 149
3.3.1 DMC/Enumerative Combinatorics (CS Core: 1 hr, KA Core: 1 hr) 149
3.3.2 DMC/Graph Theory (CS Core: 1 hr, KA Core: 1 hr) 149
3.3.3 DMC/Discrete Optimization (CS Core: 1 hr, KA Core: 1 hr) 149
3.3.4 DMC/Mathematical Logic and Set Theory (CS Core: 1 hr, KA Core: 1 hr) . . 150
3.4 Probability and Statistics (PST) 150
3.4.1 PST/Probability Axioms and Distributions (CS Core: 1 hr, KA Core: 1 hr) . . 150
3.4.2 PST/Limit Theorems (CS Core: 1 hr, KA Core: 1 hr) 151
3.4.3 PST/Stochastic Processes (CS Core: 1 hr, KA Core: 1 hr) 151

3.4.4 PST/Estimation Theory (CS Core: 1 hr, KA Core: 1 hr) . . . 151
3.4.5 PST/Bayesian Inference (CS Core: 1 hr, KA Core: 1 hr) . . . 152
3.4.6 PST/Time Series Analysis (CS Core: 1 hr, KA Core: 1 hr) . . . 152
3.5 Numerical and Scientific Analysis (NSA) . . . 152
3.5.1 NSA/Approximation Theory and Interpolation (CS Core: 1 hr, KA Core: 1 hr) 153
3.5.2 NSA/Numerical Integration and Quadrature (CS Core: 1 hr, KA Core: 1 hr) . 153
3.5.3 NSA/Error Analysis and Floating-Point Arithmetic (CS Core: 1 hr, KA Core: 1 hr) . . . 154
3.5.4 NSA/Root-Finding for Nonlinear Equations (CS Core: 1 hr, KA Core: 1 hr) . 154
3.5.5 NSA/Numerical Linear Algebra (CS Core: 1 hr, KA Core: 1 hr) . . . 154
3.5.6 NSA/Numerical Methods for Differential Equations (CS Core: 1 hr, KA Core: 1 hr) . . . 155
3.5.7 NSA/Optimization Algorithms (CS Core: 1 hr, KA Core: 1 hr) . . . 155
3.5.8 NSA/Finite Element Method (CS Core: 1 hr, KA Core: 1 hr) . . . 155
3.5.9 NSA/Parallel and High-Performance Computing (CS Core: 1 hr, KA Core: 1 hr) 156
3.6 Mathematical Modeling and Simulation (MMS) . . . 156
3.6.1 MMS/First-Order Ordinary Differential Equations (CS Core: 1 hr, KA Core: 1 hr) . . . 156
3.6.2 MMS/Higher-Order Differential Equations and Systems (CS Core: 1 hr, KA Core: 1 hr) . . . 157
3.6.3 MMS/Dynamical Systems and Chaos (CS Core: 1 hr, KA Core: 1 hr) . . . 157
3.6.4 MMS/Mathematical Control Theory (CS Core: 1 hr, KA Core: 1 hr) . . . 157
3.6.5 MMS/Stochastic Differential Equations (CS Core: 1 hr, KA Core: 1 hr) . . . 158
3.7 Mathematical Foundations of Data Science (FDS) . . . 158
3.7.1 FDS/Statistical Learning Theory (CS Core: 1 hr, KA Core: 1 hr) . . . 158
3.7.2 FDS/Optimization for Machine Learning (CS Core: 1 hr, KA Core: 1 hr) . . . 159
3.7.3 FDS/Information Theory (CS Core: 1 hr, KA Core: 1 hr) . . . 159
3.7.4 FDS/Dimensionality Reduction (CS Core: 1 hr, KA Core: 1 hr) . . . 159
3.7.5 FDS/Mathematical Foundations of Deep Learning (CS Core: 1 hr, KA Core: 1 hr) . . . 160
3.8 Quantum Information and Computing (QIC) . . . 160
3.8.1 QIC/Quantum Mechanics Foundations for Computing (CS Core: 1 hr, KA Core: 1 hr) . . . 160
3.8.2 QIC/Quantum Circuits and Gates (CS Core: 1 hr, KA Core: 1 hr) . . . 161
3.8.3 QIC/Quantum Algorithms (CS Core: 1 hr, KA Core: 1 hr) . . . 161
3.8.4 QIC/Entanglement and Bell Inequalities (CS Core: 1 hr, KA Core: 1 hr) . . . 162
3.9 Electromagnetism (ELM) . . . 162
3.9.1 ELM/Electrostatics: Electric Field and Gauss's Law (CS Core: 1 hr, KA Core: 1 hr) . . . 162
3.9.2 ELM/Electric Current, Circuits, and Kirchhoff's Laws (CS Core: 1 hr, KA Core: 1 hr) . . . 163
3.9.3 ELM/Maxwell's Equations and Gauge Transformations (CS Core: 1 hr, KA Core: 1 hr) . . . 163
3.10 Thermodynamics and Statistical Mechanics (TSM) . . . 163
3.10.1 TSM/Laws of Thermodynamics and State Functions (CS Core: 1 hr, KA Core: 1 hr) . . . 164
3.10.2 TSM/Kinetic Theory of Gases (CS Core: 1 hr, KA Core: 1 hr) . . . 164
3.10.3 TSM/Heat Engines, Entropy, and the Second Law (CS Core: 1 hr, KA Core: 1 hr) . . . 164
3.11 Computational and Data-Driven Physics (CDP) . . . 165
3.11.1 CDP/Numerical Methods for Differential Equations in Physics (CS Core: 1 hr, KA Core: 1 hr) . . . 165
3.11.2 CDP/Monte Carlo Methods and Simulations (CS Core: 1 hr, KA Core: 1 hr) . 165
3.11.3 CDP/Machine Learning for Physical Discovery (CS Core: 1 hr, KA Core: 1 hr) 166
3.12 Classical Mechanics (CME) . . . 166

3.12.1 CME/Rotational Motion and Angular Momentum (CS Core: 1 hr, KA Core: 1 hr) . . . 166
3.12.2 CME/Systems of Particles and Linear Momentum (CS Core: 1 hr, KA Core: 1 hr) . . . 167
3.12.3 CME/Kinematics and Dynamics of Particles (CS Core: 1 hr, KA Core: 1 hr) . 168
3.12.4 CME/Work, Energy, and Conservation Laws (CS Core: 1 hr, KA Core: 1 hr) . 168
3.12.5 CME/Rigid Body Dynamics (CS Core: 1 hr, KA Core: 1 hr) . . . 168
3.12.6 CME/Oscillations and Waves (CS Core: 1 hr, KA Core: 1 hr) . . . 169
3.12.7 CME/Lagrangian and Hamiltonian Mechanics (CS Core: 1 hr, KA Core: 1 hr) 169
3.12.8 CME/Elasticity (CS Core: 1 hr, KA Core: 1 hr) . . . 170
3.12.9 CME/Fluids: Hydrostatics, Hydrodynamics, and Viscosity (CS Core: 1 hr, KA Core: 1 hr) . . . 170
3.12.10 CME/Vibrations and Waves (CS Core: 1 hr, KA Core: 1 hr) . . . 170
3.12.11 CME/Transport Phenomena (CS Core: 1 hr, KA Core: 1 hr) . . . 171
3.13 General Chemistry (GCH) . . . 171
3.13.1 GCH/Atomic Structure (CS Core: 1 hr, KA Core: 1 hr) . . . 172
3.13.2 GCH/Chemical Bonding (CS Core: 1 hr, KA Core: 1 hr) . . . 172
3.13.3 GCH/Matter and Energy (CS Core: 1 hr, KA Core: 1 hr) . . . 173
3.13.4 GCH/States of Matter (CS Core: 1 hr, KA Core: 1 hr) . . . 173
3.13.5 GCH/Stoichiometry (CS Core: 1 hr, KA Core: 1 hr) . . . 173
3.13.6 GCH/Environmental and Water Chemistry (CS Core: 1 hr, KA Core: 1 hr) . . 174
3.13.7 GCH/Polymers and Geosynthetics (CS Core: 1 hr, KA Core: 1 hr) . . . 174
3.13.8 GCH/Chemical Equilibrium (CS Core: 1 hr, KA Core: 1 hr) . . . 174
3.13.9 GCH/Ionic Equilibrium and Salt Hydrolysis (CS Core: 1 hr, KA Core: 1 hr) . 175
3.13.10 GCH/Salts, Solubility, and Precipitation Equilibria (CS Core: 1 hr, KA Core: 1 hr) . . . 175
3.13.11 GCH/Electrochemistry and Electrolysis (CS Core: 1 hr, KA Core: 1 hr) 175
3.13.12 GCH/Cement Chemistry (CS Core: 1 hr, KA Core: 1 hr) . . . 176
3.14 Computational and Theoretical Chemistry (CTC) . . . 176
3.14.1 CTC/Molecular Modeling and Simulation (CS Core: 1 hr, KA Core: 1 hr) . . . 176
3.14.2 CTC/Machine Learning in Chemical Discovery (CS Core: 1 hr, KA Core: 1 hr) 177
3.14.3 CTC/Computational Drug Design (CS Core: 1 hr, KA Core: 1 hr) . . . 177
3.15 Biochemistry and Molecular Biology (BMB) . . . 177
3.15.1 BMB/Structure and Function of Biomolecules (CS Core: 1 hr, KA Core: 1 hr) 178
3.15.2 BMB/Molecular Genetics and Enzymology (CS Core: 1 hr, KA Core: 1 hr) . . 178
3.15.3 BMB/Bioinformatics and Computational Biology (CS Core: 1 hr, KA Core: 1 hr)179
3.16 Probability Theory (PRO) . . . 179
3.16.1 PRO/Sample Spaces and Events (CS Core: 1 hr, KA Core: 1 hr) . . . 179
3.16.2 PRO/Probability Axioms and Properties (CS Core: 1 hr, KA Core: 1 hr) . . . 180
3.16.3 PRO/Conditional Probability and Independence (CS Core: 1 hr, KA Core: 1 hr)180
3.16.4 PRO/Discrete Random Variables and Distributions (CS Core: 1 hr, KA Core: 1 hr) . . . 180
3.16.5 PRO/Continuous Random Variables and Distributions (CS Core: 1 hr, KA Core: 1 hr) . . . 181
3.16.6 PRO/Joint, Marginal, and Conditional Distributions (CS Core: 1 hr, KA Core: 1 hr) . . . 181
3.16.7 PRO/Limit Theorems: LLN and CLT (CS Core: 1 hr, KA Core: 1 hr) 181
3.17 Descriptive Statistics and Data Analysis (DSC) . . . 182
3.17.1 DSC/Numerical Summaries: Location, Spread, and Shape (CS Core: 1 hr, KA Core: 1 hr) . . . 182
3.18 Statistical Inference (INF) . . . 182
3.18.1 INF/Point Estimation (CS Core: 1 hr, KA Core: 1 hr) . . . 182
3.18.2 INF/Interval Estimation and Confidence Intervals (CS Core: 1 hr, KA Core: 1 hr) . . . 183
3.18.3 INF/Hypothesis Testing Foundations (CS Core: 1 hr, KA Core: 1 hr) 183
3.18.4 INF/Bayesian Inference (CS Core: 1 hr, KA Core: 1 hr) . . . 183

3.19 Regression and Statistical Models (REG) . . . 184
3.19.1 REG/Simple Linear Regression (CS Core: 1 hr, KA Core: 1 hr) . . . 184

4 Curricula 2026 **185**
4.1 Course Coding . . . 185
4.2 Curriculum Structure . . . 186
4.3 Credits for Diverse Activities . . . 191
4.4 Graduation Requirements . . . 191
4.5 Topics distributed by course . . . 193
4.5.1 Computer Science topics/course . . . 193
4.5.2 Basic Sciences for Computing topics/course . . . 211
4.6 Expected outcomes distributed by course . . . 215
4.7 Courses by Outcome . . . 222
4.7.1 Outcome: AG-C01) Social impact of computing solutions. . . . 222
4.7.2 Outcome: AG-C02) Professional ethics in computing. . . . 222
4.7.3 Outcome: AG-C03) Individual and teamwork. . . . 222
4.7.4 Outcome: AG-C04) Effective communication in computing. . . . 222
4.7.5 Outcome: AG-C05) Project management in computing. . . . 223
4.7.6 Outcome: AG-C06) Continuous learning in computing. . . . 223
4.7.7 Outcome: AG-C07) Computing knowledge. . . . 223
4.7.8 Outcome: AG-C08) Analysis of computing problems. . . . 223
4.7.9 Outcome: AG-C09) Design and development of solutions. . . . 224
4.7.10 Outcome: AG-C10) Inquiry in computing. . . . 224
4.7.11 Outcome: AG-C11) Use of computing tools. . . . 224
4.7.12 Outcome: AG-C12) Management of secure computing technologies. . . . 225
4.8 Course distribution in the Career . . . 225
4.9 Departments . . . 227
4.9.1 DAISI . . . 227
4.10 Compatibility of the Career with International Standards . . . 228

5 Detailed content by course **239**
First Semester . . . 239
BIC101 Introduction to Computing (Mandatory) . . . 239
BMA101 Linear Algebra (Mandatory) . . . 244
BMA102 Differential Calculus (Mandatory) . . . 246
BCH101 Chemistry I (Mandatory) . . . 249
BEI101 Technical and professional English I (Mandatory) . . . 254
Second Semester . . . 257
CS100 Introduction to Computer Science (Mandatory) . . . 257
CS112 Objects-oriented programming I (Mandatory) . . . 268
CS1D1 Discrete Structures (Mandatory) . . . 273
BMA103 Integral Calculus (Mandatory) . . . 276
BFI101 Physics I (Mandatory) . . . 278
BEI102 Technical and professional English II (Mandatory) . . . 283
Third Semester . . . 287
CS113 Objects-oriented programming II (Mandatory) . . . 287
CS2B1 Platform Based Development (Mandatory) . . . 293
AI161 Applied AI (Mandatory) . . . 297
BMA104 Advanced Differential and Integral Calculus (Mandatory) . . . 301
STA251 Probability Calculation (Mandatory) . . . 304
BEI201 Technical and professional English III (Elective) . . . 309
Fourth Semester . . . 313
CS210 Algorithms and Data Structures (Mandatory) . . . 313
CS211 Theory of Computation (Mandatory) . . . 318
CS221 Computer Systems Architecture (Mandatory) . . . 323
CS271 Data Management (Mandatory) . . . 327

CS401 Research Methodology (Mandatory) . . . 331
MA202 Numerical Methods (Mandatory) . . . 335
BEI202 Technical and professional English IV (Elective) . . . 340
Fifth Semester . . . 344
CS212 Analysis and Design of Algorithms (Mandatory) . . . 344
CS261 Artificial Intelligence (Mandatory) . . . 349
CS272 Databases II (Mandatory) . . . 356
CS291 Software Engineering I (Mandatory) . . . 360
CS2S1 Operating systems (Mandatory) . . . 365
FG106 Theater (Mandatory) . . . 370
BEI203 Technical and professional English V (Elective) . . . 374
Sixth Semester . . . 378
CS231 Networking and Communication (Mandatory) . . . 378
CS311 Algorithms for Complex Problems (Mandatory) . . . 383
CS312 Advanced Data Structures (Mandatory) . . . 386
CS342 Compilers (Mandatory) . . . 389
AI263 Introduction to Machine Learning (Mandatory) . . . 392
FI201 Computational Physics (Mandatory) . . . 395
Seventh Semester . . . 398
CS251 Computer Graphics (Mandatory) . . . 398
CS292 Software Engineering II (Mandatory) . . . 401
CS2H1 User Experience (UX) (Mandatory) . . . 406
AI264 Deep Learning (Mandatory) . . . 411
FG211 Professional Ethics (Mandatory) . . . 414
FG350 Leadership and Performance (Mandatory) . . . 416
Eighth Semester . . . 419
CS281 Computing in Society (Mandatory) . . . 419
CS3I1 Computer Security (Mandatory) . . . 425
CS3P1 Parallel and Distributed Computing (Mandatory) . . . 430
CS402 Capstone Project I (Mandatory) . . . 439
EX301 Extracurricular Activities (Mandatory) . . . 443
ET201 Entrerpreneurship I (Mandatory) . . . 445
AI268 Computational Vision (Elective) . . . 449
CS391 Software Engineering III (Elective) . . . 455
CS393 Information systems (Elective) . . . 459
CS394 Software Architecture and DevOps for Cloud Software (Elective) . . . 462
CS395 Software Quality (Elective) . . . 468
Ninth Semester . . . 475
CS370 Big Data (Mandatory) . . . 475
CS400 Pre-professional internships (Mandatory) . . . 478
CS403 Capstone Project II (Mandatory) . . . 480
AI365 Advanced Generative AI Models (Mandatory) . . . 484
CB309 Bioinformatics (Mandatory) . . . 486
ET301 Entrerpreneurship II (Mandatory) . . . 489
AI369 Robotics (Elective) . . . 492
CS351 Topics in Computer Graphics (Elective) . . . 496
CS353 Quantum Computing (Elective) . . . 500
CS392 Tópicos en Ingeniería de Software (Elective) . . . 504
CS3P3 Internet of Things (Elective) . . . 511
Tenth Semester . . . 513
CS3P2 Cloud Computing (Mandatory) . . . 513
CS404 Research Workshop (Mandatory) . . . 517
AI367 Topics in Artificial Intelligence (Mandatory) . . . 521
AI368 Evolutionary Computing (Mandatory) . . . 527

6 Professors & Courses **531**
Taught courses by professor 531
Professor assigned per course 531

7 Equivalences with Other Curriculum Plans **535**
Equivalence from 2027 to 2018 535
Equivalence from 2018 to 2027 539

8 Laboratories **543**

List of Figures

1.1 Field of action of the Ciencia de la Computación 2

4.1 Course coding scheme. 185
4.2 Credits by area by semester 226
4.3 Distribution of courses by area considering credits. 227
4.4 Distribution of credits by course levels. 227
4.5 DAISI 228
4.6 Comparing CS-UNI and CS proposed by ACM/IEEE-CS (Curve chart). 231
4.7 Comparing CS-UNI and CS proposed by ACM/IEEE-CS (Spider chart). 231
4.8 Comparing CS-UNI and DS proposed by ACM/IEEE-CS (Curve chart). 232
4.9 Comparing CS-UNI and DS proposed by ACM/IEEE-CS (Spider chart). 232
4.10 Comparing CS-UNI and SE proposed by ACM/IEEE-CS (Curve chart). 233
4.11 Comparing CS-UNI and SE proposed by ACM/IEEE-CS (Spider chart). 233
4.12 Comparing CS-UNI and CY proposed by ACM/IEEE-CS (Curve chart). 234
4.13 Comparing CS-UNI and CY proposed by ACM/IEEE-CS (Spider chart). 234
4.14 Comparing CS-UNI and CE proposed by ACM/IEEE-CS (Curve chart). 235
4.15 Comparing CS-UNI and CE proposed by ACM/IEEE-CS (Spider chart). 235
4.16 Comparing CS-UNI and IT proposed by ACM/IEEE-CS (Curve chart). 236
4.17 Comparing CS-UNI and IT proposed by ACM/IEEE-CS (Spider chart). 236
4.18 Comparing CS-UNI and IS proposed by ACM/IEEE-CS (Curve chart). 237
4.19 Comparing CS-UNI and IS proposed by ACM/IEEE-CS (Spider chart). 237

5.1 Connection Map. BIC101 Introduction to Computing 240
5.2 Connection Map. BMA101 Linear Algebra 244
5.3 Connection Map. BMA102 Differential Calculus 246
5.4 Connection Map. BCH101 Chemistry I 249
5.5 Connection Map. BEI101 Technical and professional English I 254
5.6 Connection Map. CS100 Introduction to Computer Science 257
5.7 Connection Map. CS112 Objects-oriented programming I 268
5.8 Connection Map. CS1D1 Discrete Structures 273
5.9 Connection Map. BMA103 Integral Calculus 276
5.10 Connection Map. BFI101 Physics I 278
5.11 Connection Map. BEI102 Technical and professional English II 283
5.12 Connection Map. CS113 Objects-oriented programming II 287
5.13 Connection Map. CS2B1 Platform Based Development 293
5.14 Connection Map. AI161 Applied AI 297
5.15 Connection Map. BMA104 Advanced Differential and Integral Calculus 301
5.16 Connection Map. STA251 Probability Calculation 304
5.17 Connection Map. BEI201 Technical and professional English III 309
5.18 Connection Map. CS210 Algorithms and Data Structures 313
5.19 Connection Map. CS211 Theory of Computation 318
5.20 Connection Map. CS221 Computer Systems Architecture 323
5.21 Connection Map. CS271 Data Management 327
5.22 Connection Map. CS401 Research Methodology 331

5.23 Connection Map. MA202 Numerical Methods . . . 335
5.24 Connection Map. BEI202 Technical and professional English IV . . . 340
5.25 Connection Map. CS212 Analysis and Design of Algorithms . . . 344
5.26 Connection Map. CS261 Artificial Intelligence . . . 349
5.27 Connection Map. CS272 Databases II . . . 356
5.28 Connection Map. CS291 Software Engineering I . . . 360
5.29 Connection Map. CS2S1 Operating systems . . . 365
5.30 Connection Map. FG106 Theater . . . 370
5.31 Connection Map. BEI203 Technical and professional English V . . . 374
5.32 Connection Map. CS231 Networking and Communication . . . 378
5.33 Connection Map. CS311 Algorithms for Complex Problems . . . 383
5.34 Connection Map. CS312 Advanced Data Structures . . . 386
5.35 Connection Map. CS342 Compilers . . . 389
5.36 Connection Map. AI263 Introduction to Machine Learning . . . 392
5.37 Connection Map. FI201 Computational Physics . . . 395
5.38 Connection Map. CS251 Computer Graphics . . . 398
5.39 Connection Map. CS292 Software Engineering II . . . 401
5.40 Connection Map. CS2H1 User Experience (UX) . . . 406
5.41 Connection Map. AI264 Deep Learning . . . 411
5.42 Connection Map. FG211 Professional Ethics . . . 414
5.43 Connection Map. FG350 Leadership and Performance . . . 416
5.44 Connection Map. CS281 Computing in Society . . . 419
5.45 Connection Map. CS3I1 Computer Security . . . 425
5.46 Connection Map. CS3P1 Parallel and Distributed Computing . . . 430
5.47 Connection Map. CS402 Capstone Project I . . . 439
5.48 Connection Map. EX301 Extracurricular Activities . . . 443
5.49 Connection Map. ET201 Entrerpreneurship I . . . 445
5.50 Connection Map. AI268 Computational Vision . . . 449
5.51 Connection Map. CS391 Software Engineering III . . . 455
5.52 Connection Map. CS393 Information systems . . . 459
5.53 Connection Map. CS394 Software Architecture and DevOps for Cloud Software . . . 462
5.54 Connection Map. CS395 Software Quality . . . 468
5.55 Connection Map. CS370 Big Data . . . 475
5.56 Connection Map. CS400 Pre-professional internships . . . 478
5.57 Connection Map. CS403 Capstone Project II . . . 480
5.58 Connection Map. AI365 Advanced Generative AI Models . . . 484
5.59 Connection Map. CB309 Bioinformatics . . . 486
5.60 Connection Map. ET301 Entrerpreneurship II . . . 489
5.61 Connection Map. AI369 Robotics . . . 492
5.62 Connection Map. CS351 Topics in Computer Graphics . . . 496
5.63 Connection Map. CS353 Quantum Computing . . . 500
5.64 Connection Map. CS392 Tópicos en Ingeniería de Software . . . 504
5.65 Connection Map. CS3P3 Internet of Things . . . 511
5.66 Connection Map. CS3P2 Cloud Computing . . . 513
5.67 Connection Map. CS404 Research Workshop . . . 517
5.68 Connection Map. AI367 Topics in Artificial Intelligence . . . 521
5.69 Connection Map. AI368 Evolutionary Computing . . . 527

List of Tables

2.1 List of KUs in the Artificial Intelligence area. 12
2.2 List of KUs in the Algorithmic Foundations area. 24
2.3 List of KUs in the Architecture and Organization area. 30
2.4 List of KUs in the Data Management area. 36
2.5 List of KUs in the Foundations of Programming Languages area. 43
2.6 List of KUs in the Graphics and Interactive Techniques area. 56
2.7 List of KUs in the Human-Computer Interaction area. 60
2.8 List of KUs in the Mathematical and Statistical Foundations area. 63
2.9 List of KUs in the Networking and Communication area. 67
2.10 List of KUs in the Operating Systems area. 70
2.11 List of KUs in the Parallel and Distributed Computing area. 78
2.12 List of KUs in the Software Development Fundamentals area. 86
2.13 List of KUs in the Software Engineering area. 90
2.14 List of KUs in the Security area. 101
2.15 List of KUs in the Society, Ethics, and the Profession area. 110
2.16 List of KUs in the System Fundamentals area. 123
2.17 List of KUs in the Specialized Platform Development area. 130

3.1 List of KUs in the Calculus and Analysis area. 142
3.2 List of KUs in the Algebra and Number Theory area. 146
3.3 List of KUs in the Discrete Mathematics and Combinatorics area. 149
3.4 List of KUs in the Probability and Statistics area. 150
3.5 List of KUs in the Numerical and Scientific Analysis area. 153
3.6 List of KUs in the Mathematical Modeling and Simulation area. 156
3.7 List of KUs in the Mathematical Foundations of Data Science area. 158
3.8 List of KUs in the Quantum Information and Computing area. 160
3.9 List of KUs in the Electromagnetism area. 162
3.10 List of KUs in the Thermodynamics and Statistical Mechanics area. 163
3.11 List of KUs in the Computational and Data-Driven Physics area. 165
3.12 List of KUs in the Classical Mechanics area. 166
3.13 List of KUs in the General Chemistry area. 172
3.14 List of KUs in the Computational and Theoretical Chemistry area. 176
3.15 List of KUs in the Biochemistry and Molecular Biology area. 178
3.16 List of KUs in the Probability Theory area. 179
3.17 List of KUs in the Descriptive Statistics and Data Analysis area. 182
3.18 List of KUs in the Statistical Inference area. 182
3.19 List of KUs in the Regression and Statistical Models area. 184

4.1 Computer Science topics by course from 1st to 2nd Semester (Part 1 of 2) 193
4.2 Computer Science topics by course from 1st to 2nd Semester (Part 2 of 2) 194
4.3 Computer Science topics by course from 3rd to 4th Semester (Part 1 of 2) 195
4.4 Computer Science topics by course from 3rd to 4th Semester (Part 2 of 2) 196
4.5 Computer Science topics by course from 5th to 6th Semester (Part 1 of 3) 197
4.6 Computer Science topics by course from 5th to 6th Semester (Part 2 of 3) 199

4.7 Computer Science topics by course from 5^{th} to 6^{th} Semester (Part 3 of 3) 200
4.8 Computer Science topics by course 7^{th} Semester 201
4.9 Computer Science topics by course 8^{th} Semester (Part 1 of 3) 202
4.10 Computer Science topics by course 8^{th} Semester (Part 2 of 3) 204
4.11 Computer Science topics by course 8^{th} Semester (Part 3 of 3) 205
4.12 Computer Science topics by course from 9^{th} to 10^{th} Semester (Part 1 of 2) 206
4.13 Computer Science topics by course from 9^{th} to 10^{th} Semester (Part 2 of 2) 208
4.14 Basic Sciences for Computing topics by course from 1^{st} to 2^{nd} Semester (Part 1 of 2) . 211
4.15 Basic Sciences for Computing topics by course from 1^{st} to 2^{nd} Semester (Part 2 of 2) . 212
4.16 Basic Sciences for Computing topics by course from 3^{rd} to 4^{th} Semester (Part 1 of 2) . 213
4.17 Basic Sciences for Computing topics by course from 3^{rd} to 4^{th} Semester (Part 2 of 2) . 214
4.18 Basic Sciences for Computing topics by course from 5^{th} to 6^{th} Semester 214
4.19 Basic Sciences for Computing topics by course 7^{th} Semester 214
4.20 Basic Sciences for Computing topics by course 8^{th} Semester 215
4.21 Basic Sciences for Computing topics by course from 9^{th} to 10^{th} Semester 215
4.22 Outcomes by course 1^{st} to 2^{nd} Semester 216
4.23 Outcomes by course 3^{rd} to 4^{th} Semester 217
4.24 Outcomes by course 5^{th} to 6^{th} Semester 218
4.25 Outcomes by course 7^{th} Semester 219
4.26 Outcomes by course 8^{th} Semester 220
4.27 Outcomes by course 9^{th} to 10^{th} Semester 221
4.28 Distribution of credits by area by semester 226

Acknowledgments

In addition to the direct authors of this document, we also wish to express our gratitude to other colleagues from various universities in the country and around the world who have kindly devoted part of their time to provide us with their suggestions. Among them, we should mention:

- The entire ACM/ IEEE-CS Computing Curricula team (CC2020, CS2023, and CS2013).
- The faculty bodies of: CS-UCSP, CS-UNSA, CS-UTEC, CS-UNI.

We wish to express our special gratitude to the Board of Trustees (Patronato) of the Universidad Nacional de Ingeniería (UNI) for its support and sponsorship in the development of this Curriculum proposal, as well as for its ongoing commitment to strengthening the academic education of its students.

On the other hand, we also wish to thank the members of the community who have made various corrections that have allowed us to improve this document. Among them, we should mention: **Luis Antonio Selis Vasquez** (UNI, Peru), **Piero Castillo** (UNI, Peru), **Jairo Andre Calagua Mallqui** (UNI, Peru), **Piero Oviedo Reyes** (UNI, Peru), **Yan Cesar Huallpa Torres** (UNI, Peru), who have actively participated in the correction of this document.

This entire team assumed as a premise that the center of our effort is the academic and human education of the students, to shape them into agents of positive and disruptive change in society.

We wish to thank all of them for their contribution, which has made it possible to produce this document—unique of its kind in our country—that will serve to lay the foundations for a stronger career in this fantastic area that we have had the opportunity to study and of which we are proud to be a part: **Computing**.

Acronyms

ACM Association for Computing Machinery

AIS Association for Information Systems

CC Computer Science

CS Computer Science

IEEE-CS IEEE Computer Society

.

Chapter 1

Introduction

Computing has undergone impressive development in recent decades, becoming the engine of scientific, technological, industrial, social, economic, and cultural development, significantly transforming our daily actions.

The emergence of the computer has marked a new era in human history that was impossible to imagine several decades ago. The vast number of applications that have been developed in recent years are transforming the development of all fields of knowledge, commerce in the globalized world we live in, the way we communicate, teaching-learning processes, and even the way we entertain ourselves.

To give an idea of the relevance and importance that this discipline has reached in our country, it is enough to mention that currently approximately more than 100 Computing degree programs are offered nationally. This without considering non-university higher technical programs that are available.

All these existing programs have computing as the core of their study but do so under more than 50 different names such as: Systems Engineering, Computer Engineering, Computer and Systems Engineering, among others. Despite all of them targeting the same job market, it is at least surprising that it is not possible to find even two that share the same curriculum, even in the first year.

Commonly, during the 1970s, Computing developed within Science Faculties in most American, British, and other countries' universities. During the 1980s, computing groups in universities struggled to achieve academic legitimacy within their local context. Frequently, they transformed into Mathematics and Computing departments, until finally splitting into two separate departments of Mathematics and Computing in the 1990s. It is in this decade that a growing number of institutions recognized the pervasive influence of Computing, creating independent units as departments, schools, or institutes dedicated to this area of study, a change that has demonstrated both insight and foresight.

In Perú, an increasing number of higher education institutions have tried to follow the development of foreign universities (though not always in a very serious or successful manner), recognizing Computing as a field of study in itself, as well as its strategic importance in education, and creating departments, schools, or institutes dedicated to its study. The Facultad de Ciencias cannot be an exception to this change, in which there is already a relative delay compared to many educational institutions within and outside of Peru.

1.1 Basic definitions

The most solid worldwide reference regarding the proposal of undergraduate computing programs is the one jointly proposed by the Association for Computing Machinery (ACM), IEEE Computer Society (IEEE-CS), and the Association for Information Systems (AIS). These three organizations proposed the Computing Curricula in the document entitled: *Joint Task Force for Computing Curricula 2005, Computing Curricula 2005. Overview Report*[Shakelford et al., 2005] whose most recent version is called *Computing Curricula 2020*[ACM/IEEE-CS, 2020].

The Computer Science is a term of U.S. origin Computer Science (CS). This term is also known as informatics in the European context[1].

[1]The European term is derived from the French word *Informatique*.

According to the dictionary of the Royal Spanish Academy (http://www.rae.es), both terms are also synonyms.

At the international level, computing presents seven clearly defined profiles:

- Computer Science [ACM/IEEE-CS/AAAI Joint Task Force, 2023],
- Computer Engineering [Soldan et al., 2016],
- Information Systems [The Joint ACM/AIS IS2020 Task Force, 2020],
- Software Engineering [Díaz-Herrera and Hilburn, 2004],
- Information Technology [ACM and IEEE-CS, 2017],
- Data Science [ACM Data Science Task Force, 2021],
- Cybersecurity [ACM/IEEE-CS Joint Task Force on Cybersecurity Education, 2017].

Figure 1.1 is taken from the definition proposed in the *Computing Curricula*[Shakelford et al., 2005] in the area of Computer Science (CC) The CC covers most of the range between the upper and lower extremes, because the CC professional does not deal "solely with the hardware" that uses software or "solely with the organization" that makes use of the information that computing can provide.

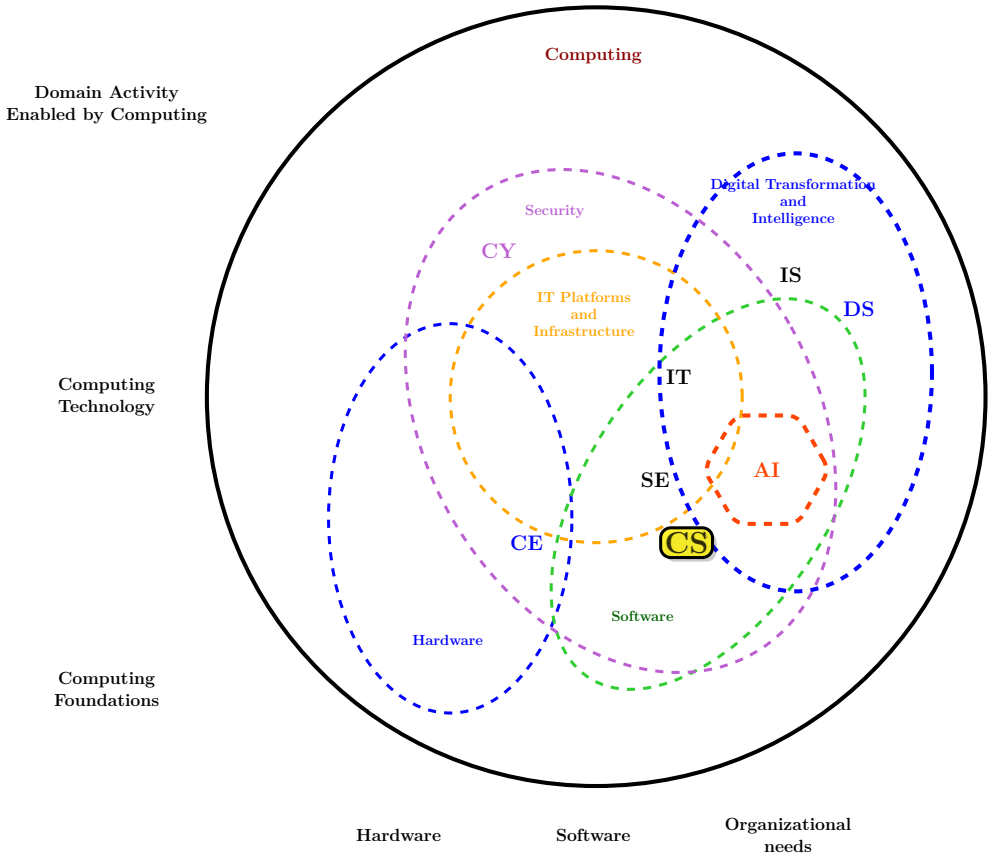

Figure 1.1: Field of action of the Ciencia de la Computación

Computer Science covers a wide range, from its theoretical and algorithmic foundations to the latest developments in robotics, computer vision, Artificial Intelligence, bioinformatics, and other exciting areas. We can think of a computer scientist's work as belonging to the following three categories:

- **Design and implementation of software**. Computer scientists undertake challenging programming tasks. They also supervise other programmers, making them aware of new approaches.
- **Devising new ways to use computers**. The progress in the areas of computer science such as networks, databases, and human-computer interaction enabled the development of the WWW and currently work is being done on the development of Grid metasystems. Furthermore, researchers are now working on making robots practical helpers that demonstrate intelligence, using databases to create new knowledge, and using computers to decipher the secrets of our DNA.

- **Developing effective ways to solve computing problems**. For example, computer scientists develop the best possible ways to store information in databases, send data across networks, and display complex images. Their theoretical foundations allow them to determine the best possible performance, and their study of algorithms helps them develop new approaches to provide better performance.

Computer Science covers the entire range from theory to programming. While other disciplines may produce graduates better prepared for specific jobs, computer science offers a broad foundation that allows its graduates to adapt to new technologies and new ideas.

The CC professional is concerned with almost everything in between these areas. Towards hardware, this professional develops software that enables the operation of devices. Towards organizational aspects, the CC professional helps information systems operate correctly in organizations. He/she generates the technology that allows other areas such as information systems to develop adequately.

The CC professional designs and develops all kinds of software, from platform infrastructure (operating systems, communication programs, etc.) to technology application (web browsers, databases, search engines, etc.). This professional creates these capabilities, but is not focused on their use. Therefore, the shaded area (fig. 1.1) for CC narrows and ends as we move towards the application and configuration of products.

1.2 Occupational Field and Job Market

Our graduate can offer their professional services in both public and private companies and institutions that require their capabilities for the offered development, including:

- Companies dedicated to the production of internationally quality software.
- Companies, institutions, and organizations that require quality software to improve their activities and/or offered services.

Our graduate can perform well in the job market without any problem because, in general, the market and occupational field demand is more oriented towards the use of tools. However, it is uncommon for professionals in this field to ask themselves: "What kind of training should I have if I want to create these tools as well as know how to use them?" Both profiles (user and creator) are quite different because it would not be possible to use something that has not yet been created. In other words, the creators of technology are the ones who create new jobs and open the possibility for others to use that technology.

Due to the research-based education, our professional must always be an innovator in their workplace. This same education allows the graduate to also consider creating their own software development company. Considering that countries like ours have a much lower cost of living than North America or Europe, an interesting possibility is exporting software, but that requires the product's quality to be at the same level as offered internationally.

This professional profile also allows our graduates to stay in our country; producing software in our country and selling it abroad is more profitable than going abroad and commercializing it there.

The occupational field for a graduate is vast and continually expanding and changing. Virtually every company or organization makes use of some type of computing services, and the good basic training of our graduates allows them to successfully meet their requirements. This graduate not only can provide solutions to existing problems but must also propose technological innovations that drive the company towards constant progress.

As the basic computerization of the country's companies advances, the need for trained individuals to solve more complex problems increases, and the curriculum we have developed aims to meet this demand in the medium and long term. The field for research and development of complex computer problems is also very broad and growing day by day worldwide.

Due to the innovative capacity of our graduate, there is a higher probability of registering patents with a high inventive level, which is especially important in our countries.

1.3 Importance of the Career in Society

One of the paths expected for a professional in the field of computing is to dedicate themselves to producing software or integrate into software-producing companies. In the field of computing, it is

common to observe that countries have Associations of Software Producers whose policies are oriented towards export. Therefore, it would not make sense to prepare our students only for the local or national market. Our graduates must be prepared to operate in the globalized world in which we live.

Our future professionals should be oriented towards creating new technology-based companies that can increase exports of Peruvian software. This new profile is aimed at generating innovative industry. If we can export competitive software, we will also be able to attract new investments. New investments would generate more well-paid jobs at a low cost compared to other types of industries. From this perspective, we can affirm that this career will be an engine that will decisively drive the country's development with a very low investment compared to other fields.

It is necessary to remember that the greatest innovation in recent commercial product versions uses technology that was known in the academic world 20 years ago or more. A clear example is databases that support spatial data and queries in recent years. However, they use data structures that have existed for decades. It is logical to think that people in the academic field do not dedicate themselves to studying the latest version of a particular software when they already knew that technology a long time ago. For the same reason, it is rare in the world to see a university have agreements with a software multinational to teach only that technology because our role is to generate that technology and not just know how to use it.

We must also not forget that the students who enter today will enter the market in approximately 5 years, and in a rapidly changing world, we cannot and should not teach them considering only the current market. Our professionals must be prepared to solve the problems that will arise in 10 or 15 years, and that is only possible through research.

1.4 Institutional Info

1.4.1 Mission

Based on the aforementioned, our mission is:

To contribute to the scientific, technological and technical development of the country forming competent professionals oriented to the creation of new science and computational technology, as engine that impels and consolidates the software industry based on scientific research and technological in innovative areas, forming, IN OUR professionals, a set of skills for solving computational problems with a social commitment.

1.4.2 Vision

- We aspire to be an accredited professional program with international standards, recognized for the quality and competitiveness of its faculty and graduates.
- We aim to be a program that stands out for the relevance and pertinence of its basic and applied research projects.
- We strive to be a program that promotes the development of the software industry in Ciencia de la Computación at an international level, incorporating its graduates into the established industry or generating new software development companies.
- We aspire to be a program that shares and disseminates knowledge with all sectors of the population and contributes to the solution of our society's strategic problems.

1.5 Accreditation information

1.5.1 Career Outcomes

Upon completion of this program, graduates will have achieved the following outcomes:

AG-C01) The Professional and the World: Analyzes and evaluates the impact of solutions to complex computing problems on the sustainable development of society.

AG-C02) Ethics: Applies ethical principles and commits to professional ethics and the standards of professional computing practice.

AG-C03) Individual and Team Work: Performs effectively as an individual and as a member or leader in diverse teams.

AG-C04) Communication: Communicates effectively in complex computing activities.

AG-C05) Project Management: Applies management principles in computing to manage projects.

AG-C06) Life-long Learning: Recognizes the importance of continuous learning and adaptation to new technologies.

AG-C07) Computing Knowledge: Applies knowledge of mathematics, science, and computing.

AG-C08) Problem Analysis: Identifies, formulates, and analyzes complex computing problems.

AG-C09) Design and Development of Solutions: Designs, implements, and evaluates solutions for complex computing problems.

AG-C10) Inquiry: Studies complex computing problems using information science methods.

AG-C11) Use of Tools: Applies modern computing tools in problem solving.

AG-C12) Applies computer science theory and software development fundamentals to produce computer-based solutions.

The competences and/or outcomes proposed in this curriculum correspond to the competences and/or outcomes proposed by Accreditation Institute for Computing, Engineering and Technology (ICACIT).

1.5.2 Educational objectives

After five years of graduating from the Ciencia de la Computación professional career, our professionals should be able to:

1. Meet and exceed the work expectations defined by the professional environment.
2. Perform as a member or leader of a team, whether specialized or multidisciplinary.
3. Propose solutions to the professional context, where one operates, based on the implementation or improvement of the state of the art in Computer Science.
4. Effectively communicate technological proposals to individuals with varying levels of knowledge and from different social backgrounds.
5. Update and adapt to new knowledge in Computer Science and diverse professional fields, either autonomously or through complementary studies.
6. Demonstrate a clear understanding of the consequences arising from technological creations related to Computer Science in aspects such as social, ethical, human, moral, legal, environmental, economic, among others.

1.5.3 Learning Outcomes

Each KU within a KA enumerates both a set of topics and the learning outcomes that students should achieve concerning the specified topics. Learning outcomes are not of equal size and do not have a uniform assignment of curricular hours; topics with the same number of hours may have very different numbers of associated learning outcomes.

Each learning outcome has an associated level of mastery. In defining the different levels, we draw from other curricular approaches, especially Bloom's taxonomy, which has been well explored within the Ciencia de la Computación. In this document, Bloom's levels were not directly applied, partly because several of them are driven by pedagogical context, which would introduce too much plurality into such a document; partly because we intend the mastery levels to be indicative and not impose theoretical restrictions on the users of this document.

We use three expected mastery levels:

Level 1 Familiarity : The student **understands** basic concepts. Answers: **"What is this?"**

Level 2 Usage : The student **applies** concepts practically. Answers: **"How to use this?"**

Level 3 Assessment : The student **evaluates and justifies** approaches. Answers: **"Why this method?"**

For example, to evaluate mastery levels, consider the notion of iteration in software development (for, while, and iterators). At the "familiarity" level, a student is expected to have a definition of the concept of iteration in software development and to know why this technique is useful.

In order to demonstrate mastery at the "Usage" level, the student must be able to properly write a program using some form of iteration.

At the "Assessment" level, in the case of iteration, a student would be required to understand multiple methods of iteration and be able to select appropriately among them for different applications.

1.6 Profiles

1.6.1 Applicant profile

The applicant seeking admission to the Escuela Profesional de Ciencia de la Computación of the Universidad Nacional de Ingeniería (UNI) must have:
Knowledge of:

1. Fundamental concepts of algebra, geometry, trigonometry and elementary mathematical analysis.
2. Basic notions of propositional logic and deductive reasoning.
3. Elementary principles of algorithms and logical sequences for problem solving.
4. Notions of probability and descriptive statistics for data analysis.

Skills for:

1. Thinking abstractly, identifying patterns, structures and relationships beyond the concrete.
2. Modeling real-world problems using symbolic and formal representations.
3. Decomposing complex problems into simpler and more manageable subproblems.
4. Learning autonomously and self-regulating, acquiring new technologies and paradigms without the need for continuous formal instruction.
5. Adapting quickly to changing technological environments and the rapid evolution of tools and languages.
6. Communicating technical ideas and solutions clearly, both orally and in writing, using appropriate terminology.
7. Maintaining concentration on analytical tasks that demand sustained effort and attention to detail.
8. Synthesizing information from multiple sources and extracting essential concepts for decision making.
9. Critically evaluating existing solutions and proposing improvements or innovative alternatives.

Attitudes of:

1. Genuine interest and curiosity about technology, programming and the functioning of computer systems.
2. Willingness for collaborative work in development teams and multidisciplinary projects.
3. Initiative to explore problems beyond the curriculum and perseverance in the face of intellectual challenges such as code debugging or optimization.
4. Growth mindset and openness to continuously learn in a field that is permanently evolving.
5. Commitment to quality, efficiency and good practices in the development of solutions.

1.6.2 Graduate profile

The graduate of the Professional Program in Computer Science is capable of:

1. Designing, developing, and implementing international-quality software solutions that address complex needs of industry, society, and research.
2. Contributing to the country's technological development through the creation of innovative, secure, and scalable software, considering global standards and industry best practices (ISO, IEEE, CMMI).

3. Transforming organizational processes and business models through the strategic application of emerging computational technologies (AI, cloud computing, cybersecurity, IoT).
4. Adapting with agility to the field's rapid evolution, thanks to training grounded in solid scientific principles and continuous self-learning capabilities.
5. Leading multidisciplinary and globally distributed teams in technological development projects, applying agile methodologies (Scrum, DevOps) and effective communication in bilingual environments.
6. Generating sustainable competitive advantages for organizations through robust system architecture, massive data analysis, and intelligent process automation.
7. Founding and growing technology-based companies (startups) with the potential to compete and innovate in global markets.
8. Accessing and excelling in top-tier graduate programs (master's and doctoral degrees) at prestigious international universities, supported by rigorous training in computer sciences.
9. Contributing to the advancement of knowledge in the field through applied research, publication in indexed conferences and journals, and development of technologies with patent potential.

1.6.3 ProfessionalProfile

The professional in Ciencia de la Computación has the following characteristics:

- Develop computational technology seeking the common good of individuals, society, and organizations.
- Transform, accelerate, and expand the boundaries of any knowledge area through innovative solutions based on the efficient use of computational technology.
- Adapt quickly to technological changes due to their education based on constant research.
- Work and lead multidisciplinary teams that carry out technological innovation projects.
- Increase the competitive advantages of any organization through the efficient use of computational technology thanks to their high abstraction capability.
- Create technology-based companies.
- Be able to pursue graduate studies with international standards in related areas.

1.6.4 Faculty profile

The professor of the Ciencia de la Computación program is a person who:

- Promotes holistic student formation through teaching, research, and extension, transmitting solid knowledge with clarity and simplicity.
- Fosters research and continuous learning so that students adapt to area advancements and seek the common good of individuals, society, and organizations.
- Pursues a PhD degree with full-time dedication, maintaining advanced academic training oriented towards international excellence.
- Publishes research in international events and high-impact indexed journals, contributing to knowledge advancement and staying updated.
- Establishes cooperation networks with international-level institutions, enriching the educational experience.
- Actively participates in student formation and integration into the academic community , promoting continuous rigor and academic excellence.

1.7 Degrees and Titles

Upon approval of the academic credits proposed in the curriculum and after fulfilling the requirements stipulated in the degree and title regulations of the program, the graduate will receive:

Grado Académico: Bachiller en Ciencia de la Computación y

Titulo Profesional: Licenciado en Ciencia de la Computación

1.8 Resources to teach

An innovative professional must be aware of the latest advances in their field. The latest advances in this area are not necessarily presented in books. We must use publications from indexed journals of worldwide circulation.

For this reason, we take as a basis the institutional subscription to ACM and IEEE-CS. It is recommended that the teacher use this material to discuss trends in all areas in class.

Computer labs must also be renewed according to the speed of this area, and this implies constant renovations to ensure that students are up to date.

The physical resources in terms of laboratories for each course are detailed in Chapter 8.

1.9 Document Organization

The remainder of this document is organized as follows: Chapter 1 defines and explains the field of action of the Ciencia de la Computación, in addition to providing a very brief explanation of the different programs related to this discipline. In the particular case of the area of computing, the proposals from ACM/IEEE-CS *Computing Curricula* [ACM/IEEE-CS, 2020].

In Chapter 2, we present the Knowledge Areas of Computer Science, indicating the topics and learning objectives of each subject,

In Chapter 3, we present the Knowledge Areas of Basic Sciences for Computing, indicating the topics and learning objectives of each subject,

Chapter 4 contains the distribution by semesters, by areas, by levels, graphical view of the curriculum, comparison with various international proposals, distribution of topics per course, as well as the distribution of skills per subject.

Chapter 5 contains detailed information for each of the courses, including the skills to which it contributes, bibliography per unit, as well as the minimum number of hours per unit.

In Chapter 6 we can observe a list of courses by professor and of professor by course.

In Chapter 7, the equivalence tables with other curricula are presented.

In Chapter 8, a suggestion of the laboratories required for teaching classes is presented, which could vary depending on the number of students.

Chapter 2

Body of knowledge of Computer Science

The 17 knowledge areas in Computer Science are:

2.1 Artificial Intelligence (AI) (Pg.. 11)
- 2.1.1 Introduction to AI: Concepts and Intelligent Behavior (Pg.. 12)
- 2.1.2 Problem Characteristics and Agent Nature (Pg.. 13)
- 2.1.3 State and Uninformed Search (Pg.. 13)
- 2.1.4 Heuristic Search and CSP (Pg.. 13)
- 2.1.5 Adversarial Search and Stochastic Optimization (Pg.. 14)
- 2.1.6 Knowledge Representation and Probabilistic Reasoning Foundations (Pg.. 14)
- 2.1.7 Independence, Markov Models and Decision Making (Pg.. 15)
- 2.1.8 Machine Learning Fundamentals (Pg.. 15)
- 2.1.9 Machine Learning Data, Representations and Evaluation (Pg.. 15)
- 2.1.10 Neural Networks and Ethics (Pg.. 16)
- 2.1.11 Machine Learning Formulation and Evaluation (Pg.. 16)
- 2.1.12 Deep Learning, RL and Ethics (Pg.. 17)
- 2.1.13 Neural Network Training and Optimization (Pg.. 17)
- 2.1.14 Regularization Techniques for Deep Networks (Pg.. 17)
- 2.1.15 Convolutional and Residual Network Architectures (Pg.. 17)
- 2.1.16 Transformer Architectures and Generative Models (Pg.. 18)
- 2.1.17 AI Applications and Societal Impact (Pg.. 18)
- 2.1.18 Additional AI Applications and Societal Impact (Pg.. 18)
- 2.1.19 Logical Representation and Reasoning (Pg.. 19)
- 2.1.20 Probabilistic Representation and Reasoning (Pg.. 19)
- 2.1.21 Planning (Pg.. 20)
- 2.1.22 Agents and Cognitive Systems (Pg.. 20)
- 2.1.23 Natural Language Processing (Pg.. 21)
- 2.1.24 Robotics (Pg.. 21)
- 2.1.25 Perception and Computer Vision (Pg.. 22)

2.2 Algorithmic Foundations (AL) (Pg.. 23)
- 2.2.1 Foundational Data Structures (Pg.. 24)
- 2.2.2 Foundational Algorithms (Pg.. 24)
- 2.2.3 Advanced Algorithms (Pg.. 25)
- 2.2.4 Algorithmic Strategies (Pg.. 25)
- 2.2.5 Complexity Analysis Framework (Pg.. 26)
- 2.2.6 Asymptotic Notation and Complexity Classes (Pg.. 26)
- 2.2.7 Complexity Analysis: Recursion, Amortization, and Tight Bounds (Pg.. 27)
- 2.2.8 Computational Complexity Theory (Pg.. 27)
- 2.2.9 Formal Languages and Automata (Pg.. 27)
- 2.2.10 Computability and Decidability (Pg.. 28)
- 2.2.11 Society, Ethics, and the Profession (Pg.. 29)

2.3 Architecture and Organization (AR) (Pg.. 29)
- 2.3.1 Digital Logic and Digital Systems (Pg.. 30)
- 2.3.2 Machine-Level Data Representation (Pg.. 30)
- 2.3.3 Assembly Level Machine Organization (Pg.. 31)
- 2.3.4 Memory Hierarchy (Pg.. 31)
- 2.3.5 Interfacing and Communication (Pg.. 32)
- 2.3.6 Functional Organization (Pg.. 32)
- 2.3.7 Performance and Energy Efficiency (Pg.. 32)
- 2.3.8 Heterogeneous Architectures (Pg.. 33)
- 2.3.9 Secure Processor Architectures (Pg.. 33)
- 2.3.10 Quantum Architectures (Pg.. 34)
- 2.3.11 Sustainability Issues (Pg.. 35)

2.4 Data Management (DM) (Pg.. 35)
- 2.4.1 The Role of Data and the Data Life Cycle (Pg.. 36)
- 2.4.2 Core Database System Concepts (Pg.. 36)
- 2.4.3 Data Modeling (Pg.. 37)
- 2.4.4 Relational Databases (Pg.. 37)
- 2.4.5 Query Construction (Pg.. 38)
- 2.4.6 Query Processing (Pg.. 38)
- 2.4.7 DBMS Internals (Pg.. 39)
- 2.4.8 NoSQL Systems (Pg.. 39)
- 2.4.9 Data Security and Privacy (Pg.. 40)
- 2.4.10 Data Analytics (Pg.. 40)
- 2.4.11 Distributed Databases/Cloud Computing (Pg.. 41)
- 2.4.12 Semi-structured and Unstructured Databases (Pg.. 41)
- 2.4.13 Society, Ethics, and the Profession (Pg.. 41)

2.5 Foundations of Programming Languages (FPL) (Pg.. 42)
- 2.5.1 Object-Oriented Programming Fundamentals (Pg.. 43)
- 2.5.2 Object-Oriented Programming: Encapsulation, Subtyping, and Reflection (Pg.. 44)
- 2.5.3 Functional Programming (Pg.. 44)
- 2.5.4 Logic Programming (Pg.. 45)
- 2.5.5 Shell Scripting (Pg.. 46)
- 2.5.6 Event-Driven and Reactive Programming (Pg.. 46)
- 2.5.7 Parallel and Distributed Computing (Pg.. 46)
- 2.5.8 Aspect-Oriented Programming (Pg.. 47)
- 2.5.9 Type Systems Fundamentals (Pg.. 47)
- 2.5.10 Type Systems: Polymorphism and Complementary Typing (Pg.. 48)
- 2.5.11 Advanced Type Theory (Pg.. 48)
- 2.5.12 Systems Execution and Memory Model (Pg.. 49)
- 2.5.13 Memory Management (Pg.. 49)
- 2.5.14 Language Translation and Execution (Pg.. 50)
- 2.5.15 Program Abstraction and Representation (Pg.. 50)
- 2.5.16 Syntax Analysis (Pg.. 51)
- 2.5.17 Compiler Semantic Analysis (Pg.. 51)
- 2.5.18 Program Analysis and Analyzers (Pg.. 52)
- 2.5.19 Code Generation (Pg.. 52)
- 2.5.20 Run-time Behavior and Systems (Pg.. 53)
- 2.5.21 Advanced Programming Constructs (Pg.. 53)
- 2.5.22 Language Pragmatics (Pg.. 54)
- 2.5.23 Formal Semantics (Pg.. 54)
- 2.5.24 Formal Development Methodologies (Pg.. 55)
- 2.5.25 Design Principles of Programming Languages (Pg.. 55)
- 2.5.26 Society, Ethics, and the Profession (Pg.. 56)

2.6 Graphics and Interactive Techniques (GIT) (Pg.. 56)

- 2.6.1 Fundamental Concepts of Graphics and Interactive Techniques (Pg.. 56)
- 2.6.2 Visualization (Pg.. 57)
- 2.6.3 Applied Rendering and Techniques (Pg.. 57)
- 2.6.4 Geometric Modeling (Pg.. 57)
- 2.6.5 Shading and Advanced Rendering (Pg.. 58)
- 2.6.6 Computer Animation (Pg.. 58)
- 2.6.7 Simulation (Pg.. 58)
- 2.6.8 Immersion (Pg.. 59)
- 2.6.9 Interaction (Pg.. 59)
- 2.6.10 Image Processing (Pg.. 59)
- 2.6.11 Tangible/Physical Computing (Pg.. 59)
- 2.6.12 Society, Ethics, and the Profession (Pg.. 60)

2.7 Human-Computer Interaction (HCI) (Pg.. 60)

- 2.7.1 Understanding the User: Individual goals and interactions with others (Pg.. 61)
- 2.7.2 Accountability and Responsibility in Design (Pg.. 61)
- 2.7.3 Accessibility and Inclusive Design (Pg.. 61)
- 2.7.4 Evaluating the Design (Pg.. 62)
- 2.7.5 System Design (Pg.. 62)
- 2.7.6 Society, Ethics, and the Profession (Pg.. 62)

2.8 Mathematical and Statistical Foundations (MSF) (Pg.. 63)

- 2.8.1 Discrete Mathematics (Pg.. 63)
- 2.8.2 Advanced Discrete Mathematics (Pg.. 64)
- 2.8.3 Probability (Pg.. 64)
- 2.8.4 Statistics (Pg.. 64)
- 2.8.5 Linear Algebra (Pg.. 65)
- 2.8.6 Calculus (Pg.. 65)

2.9 Networking and Communication (NC) (Pg.. 66)

- 2.9.1 Fundamentals of Networks and Communications (Pg.. 67)
- 2.9.2 Networked Applications (Pg.. 67)
- 2.9.3 Reliability Support (Pg.. 68)
- 2.9.4 Routing and Forwarding (Pg.. 68)
- 2.9.5 Single Hop Communication (Pg.. 68)
- 2.9.6 Network Security (Pg.. 68)
- 2.9.7 Mobility Support (Pg.. 69)
- 2.9.8 Emerging Topics (Pg.. 69)

2.10 Operating Systems (OS) (Pg.. 69)

- 2.10.1 Role and Purpose of Operating Systems (Pg.. 70)
- 2.10.2 Principles of Operating System (Pg.. 71)
- 2.10.3 Concurrency (Pg.. 71)
- 2.10.4 Protection and Safety (Pg.. 72)
- 2.10.5 Scheduling (Pg.. 73)
- 2.10.6 Process Model (Pg.. 73)
- 2.10.7 Memory Management (Pg.. 74)
- 2.10.8 Device management (Pg.. 74)
- 2.10.9 File Systems API and Implementation (Pg.. 75)
- 2.10.10 Advanced File systems (Pg.. 75)
- 2.10.11 Virtualization (Pg.. 76)
- 2.10.12 Real-time and Embedded Systems (Pg.. 76)
- 2.10.13 Fault tolerance (Pg.. 77)
- 2.10.14 Society, Ethics, and the Profession (Pg.. 77)

2.11 Parallel and Distributed Computing (PDC) (Pg.. 78)

- 2.11.1 Programs: Declarative Parallelism (Pg.. 79)
- 2.11.2 Programs: Starting Activities (Pg.. 79)
- 2.11.3 Programs: Execution Properties (Pg.. 79)
- 2.11.4 Programs: Distribution (Pg.. 79)
- 2.11.5 Programs: Implementation Mappings (Pg.. 80)
- 2.11.6 GPU Programming (Pg.. 80)
- 2.11.7 Communication (Pg.. 81)
- 2.11.8 Communication: Properties and Extensions (Pg.. 81)
- 2.11.9 Memory and Consistency (Pg.. 82)
- 2.11.10 Coordination (Pg.. 82)
- 2.11.11 Coordination: Synchronization and Atomicity (Pg.. 82)
- 2.11.12 Coordination: Advanced Properties (Pg.. 83)
- 2.11.13 Evaluation (Pg.. 83)
- 2.11.14 Evaluation: Formal Methods (Pg.. 84)
- 2.11.15 Evaluation: Testing and Measurement (Pg.. 84)
- 2.11.16 Algorithms (Pg.. 84)
- 2.11.17 Algorithms: Application Domains Survey (Pg.. 85)
- 2.11.18 Algorithms: Algorithmic Domains (Pg.. 85)

2.12 Software Development Fundamentals (SDF) (Pg.. 86)

- 2.12.1 Basic Programming Concepts and Constructs (Pg.. 87)
- 2.12.2 Programming Practices: Testing, Debugging, and Documentation (Pg.. 87)
- 2.12.3 Fundamental Data Structures (Pg.. 88)
- 2.12.4 Algorithms (Pg.. 88)
- 2.12.5 Software Development Practices (Pg.. 88)
- 2.12.6 Society, Ethics, and the Profession (Pg.. 89)

2.13 Software Engineering (SE) (Pg.. 89)

- 2.13.1 Team Collaboration and Dynamics (Pg.. 90)
- 2.13.2 Diversity, Stakeholders and Distributed Teams (Pg.. 91)
- 2.13.3 Version Control and CI/CD (Pg.. 91)
- 2.13.4 Development Environment and IDE (Pg.. 92)
- 2.13.5 Requirements Engineering (Pg.. 92)
- 2.13.6 Requirements Evolution and Estimation (Pg.. 93)
- 2.13.7 Principles and Software Architecture (Pg.. 93)
- 2.13.8 API Design and Modeling (Pg.. 93)
- 2.13.9 Software Design Quality and Evaluation (Pg.. 94)
- 2.13.10 Coding Practices (Pg.. 94)
- 2.13.11 Large-Scale Construction and Process (Pg.. 95)
- 2.13.12 Verification and validation Concepts (Pg.. 96)
- 2.13.13 Test Planning and Types (Pg.. 96)
- 2.13.14 Advanced Testing Practices (Pg.. 96)
- 2.13.15 Testing Tools and Analysis (Pg.. 97)
- 2.13.16 Performance Testing and Benchmarking (Pg.. 97)
- 2.13.17 API Compatibility and Versioning (Pg.. 98)
- 2.13.18 Refactoring Techniques (Pg.. 98)
- 2.13.19 Reliability Concepts (Pg.. 99)
- 2.13.20 Reliability Engineering (Pg.. 99)
- 2.13.21 Formal Methods (Pg.. 99)

2.14 Security (SEC) (Pg.. 100)

- 2.14.1 Security Mindset and Principles (Pg.. 101)
- 2.14.2 Access Control and Applications (Pg.. 102)
- 2.14.3 AI and Hardware Security (Pg.. 102)
- 2.14.4 Security Ethics, Law and Society (Pg.. 102)
- 2.14.5 Privacy and Security by Design (Pg.. 103)
- 2.14.6 Injection and Input Validation (Pg.. 103)
- 2.14.7 Memory Safety and Types (Pg.. 104)
- 2.14.8 Information Flow and Non-Interference (Pg.. 104)
- 2.14.9 Malware Analysis and Advanced Security (Pg.. 104)
- 2.14.10 Cryptography Fundamentals (Pg.. 105)
- 2.14.11 Cryptographic Algorithms (Pg.. 105)
- 2.14.12 Protocols, Applications and Advanced Cryptography (Pg.. 106)
- 2.14.13 Threat Analysis and Security Engineering (Pg.. 106)
- 2.14.14 Security Design and Controls Engineering (Pg.. 107)
- 2.14.15 Trusted Computing and Privacy Engineering (Pg.. 107)
- 2.14.16 Digital Forensics Techniques (Pg.. 108)
- 2.14.17 Legal Aspects and Incident Handling (Pg.. 108)
- 2.14.18 Security Governance, Management and Policy (Pg.. 109)
- 2.14.19 Risk and Data Management (Pg.. 109)

2.15 Society, Ethics, and the Profession (SEP) (Pg.. 109)

- 2.15.1 Social Impact and AI (Pg.. 110)
- 2.15.2 Digital Equity and Accessibility (Pg.. 111)
- 2.15.3 Ethical Foundations and Frameworks (Pg.. 111)
- 2.15.4 Ethical Analysis Methods and Tools (Pg.. 112)
- 2.15.5 Professional Values, Nature and Continuing Development (Pg.. 112)
- 2.15.6 Ethics Codes and Legal Accountability (Pg.. 113)
- 2.15.7 Professional Responsibility and Public Policy (Pg.. 113)
- 2.15.8 Ethical Dissent and Cultural Context (Pg.. 114)
- 2.15.9 Workplace Ethics (Pg.. 114)
- 2.15.10 Intellectual Property Foundations (Pg.. 114)
- 2.15.11 Licensing, Plagiarism and Authorship (Pg.. 115)
- 2.15.12 Digital IP Rights and Copyright (Pg.. 115)
- 2.15.13 Open Source, Piracy and DRM (Pg.. 115)
- 2.15.14 Data, Surveillance and Privacy Technology (Pg.. 115)
- 2.15.15 Privacy Philosophical Foundations and Legislation (Pg.. 116)
- 2.15.16 Civil Liberties and Expression (Pg.. 116)
- 2.15.17 Technical and Professional Communication (Pg.. 117)
- 2.15.18 Team Communication and Culture (Pg.. 117)
- 2.15.19 Environmental Footprint of Computing Systems (Pg.. 118)
- 2.15.20 Systemic Effects and Social Context (Pg.. 118)

- 2.15.21 Sustainable Design and Pervasive Computing (Pg.. 118)
- 2.15.22 Foundations and Early History of Computing (Pg.. 119)
- 2.15.23 Modern History: PC, Internet and AI (Pg.. 119)
- 2.15.24 Economic Models and Strategies (Pg.. 120)
- 2.15.25 Automation, AI and Economic Ethics (Pg.. 120)
- 2.15.26 Computer Crimes and Attacks (Pg.. 120)
- 2.15.27 Security Laws, Policies and Responsibility (Pg.. 121)
- 2.15.28 Identity and Diversity in Computing (Pg.. 121)
- 2.15.29 DEIA Historical and Cross-Cultural Context (Pg.. 122)
- 2.15.30 Inclusion and Computing Practices (Pg.. 122)
- 2.15.31 Accessible Design (Pg.. 122)

2.16 System Fundamentals (SF) (Pg.. 123)

- 2.16.1 Hardware Components and Logic (Pg.. 123)
- 2.16.2 Layers, Abstractions and Scalability (Pg.. 124)
- 2.16.3 Digital Systems and Logic (Pg.. 124)
- 2.16.4 State Machines (Pg.. 124)
- 2.16.5 Sequential and Parallel Processing (Pg.. 125)
- 2.16.6 Resource Management (Pg.. 125)
- 2.16.7 Latency, Cache and Memory Hierarchy (Pg.. 126)
- 2.16.8 Virtualization and Isolation (Pg.. 126)
- 2.16.9 Performance Metrics and Benchmarks (Pg.. 127)
- 2.16.10 Analytical Performance Models (Pg.. 127)
- 2.16.11 Events, Tools and Experimentation (Pg.. 127)
- 2.16.12 System Reliability (Pg.. 128)
- 2.16.13 System Security (Pg.. 128)
- 2.16.14 System Design (Pg.. 129)
- 2.16.15 Society, Ethics, and the Profession (Pg.. 129)

2.17 Specialized Platform Development (SPD) (Pg.. 129)

- 2.17.1 Common Aspects: Platforms, APIs and Constraints (Pg.. 130)
- 2.17.2 Common Aspects: Platform Languages and Web Programming Patterns (Pg.. 131)
- 2.17.3 Frontend, Frameworks and Web Standards (Pg.. 131)
- 2.17.4 Cloud Architecture and Data Management (Pg.. 131)
- 2.17.5 Mobile Platforms: Development Fundamentals (Pg.. 132)
- 2.17.6 Mobile Platforms: Architecture and Constraints (Pg.. 132)
- 2.17.7 Mobile Platforms: Capabilities and Asynchronous Computing (Pg.. 132)
- 2.17.8 Robot Platforms (Pg.. 132)
- 2.17.9 Embedded Characteristics and Architectures (Pg.. 133)
- 2.17.10 Systems, Programming and Sensors (Pg.. 133)
- 2.17.11 Real-Time, Memory and Verification (Pg.. 134)
- 2.17.12 Platform History, Types and Characteristics (Pg.. 134)
- 2.17.13 Sensors, Controllers and eSports Ecosystems (Pg.. 135)
- 2.17.14 Architectures and Rendering (Pg.. 135)
- 2.17.15 Algorithms, Media and Virtual Worlds (Pg.. 135)
- 2.17.16 Languages, APIs and Development Tools (Pg.. 135)
- 2.17.17 Game Engines (Pg.. 136)
- 2.17.18 Game Design Foundations (Pg.. 136)
- 2.17.19 Genres, Audiences and Social Reach (Pg.. 136)
- 2.17.20 Interactive Data and Analytics Platforms (Pg.. 137)
- 2.17.21 AI Programming and Quantum Platforms (Pg.. 137)
- 2.17.22 SPD-SEP/Mobile (Pg.. 138)
- 2.17.23 SPD-SEP/Web (Pg.. 138)
- 2.17.24 Intellectual Property in Games (Pg.. 138)
- 2.17.25 Game Interface and Accessibility (Pg.. 139)
- 2.17.26 Health, Safety and Sustainability (Pg.. 139)
- 2.17.27 Representation, Inclusion and Equality (Pg.. 139)
- 2.17.28 SPD-SEP/Robotics (Pg.. 140)
- 2.17.29 SPD-SEP/Interactive (Pg.. 140)

2.1 Artificial Intelligence (AI)

Artificial intelligence (AI) studies problems that are difficult or impractical to solve with traditional algorithmic approaches. These problems are often reminiscent of those considered to require human intelligence, and the resulting AI solution strategies typically generalize over classes of problems. AI techniques are now pervasive in computing, supporting everyday applications such as email, social media, photography, financial markets, and intelligent virtual assistants (e.g., Siri, Alexa). These techniques are also used in the design and analysis of autonomous agents that perceive their environment and interact rationally with it, such as self-driving vehicles and other robots.

Traditionally, AI has included a mix of symbolic and subsymbolic approaches. The solutions it provides rely on a broad set of general and specialized knowledge representation schemes, problem solving mechanisms, and optimization techniques. These approaches deal with perception (e.g., speech recognition, natural language understanding, computer vision), problem solving (e.g., search, planning, optimization), generation (e.g., narrative, conversation, images, models, recommendations), acting (e.g., robotics, task-automation, control), and the architectures needed to support them (e.g., single agents, multi-agent systems). Machine learning may be used within each of these aspects and can even be employed end-to-end across all of them. The study of Artificial Intelligence prepares students to determine when an AI approach is appropriate for a given problem, identify appropriate representations and reasoning mechanisms, implement them, and evaluate them with respect to both performance and their broader societal impact.

Over the past decade, the term ”artificial intelligence” has become commonplace within businesses, news articles, and everyday conversation, driven largely by a series of high-impact machine learning applications. These advances were made possible by the widespread availability of large datasets, increased computational power, and algorithmic improvements. In particular, there has been a shift from engineered representations to representations learned automatically through optimization over large datasets. The resulting advances have put such terms as ”neural networks” and ”deep learning” into everyday vernacular. Businesses now advertise AI-based solutions as value-additions to their services, so that ”artificial intelligence” is now both a technical term and a marketing buzzword.

Other disciplines, such as biology, art, architecture, and finance, increasingly use AI techniques to solve problems within their disciplines.

For the first time in our history, the broader population has access to sophisticated AI-driven tools, including tools to generate essays or poems from a prompt, artwork from a description, and fake photographs or videos depicting real people. AI technology is now in widespread use in stock trading, curating our news and social media feeds, automated evaluation of job applicants, detection of medical conditions, and influencing prison sentencing through recidivism prediction. Consequently, AI technology can have significant societal impacts and ethical considerations that must be understood and considered when developing and applying it.

Knowledge Area (KA)	CS Core	KA Core
⭐ 2.1.1 Introduction to AI: Concepts and Intelligent Behavior (Pg.. 12)	2	1
⭐ 2.1.2 Problem Characteristics and Agent Nature (Pg.. 13)	1	1
⭐ 2.1.3 State and Uninformed Search (Pg.. 13)	1	2
⭐ 2.1.4 Heuristic Search and CSP (Pg.. 13)	1	2
⭐ 2.1.5 Adversarial Search and Stochastic Optimization (Pg.. 14)	1	3
⭐ 2.1.6 Knowledge Representation and Probabilistic Reasoning Foundations (Pg.. 14)	1	1
⭐ 2.1.7 Independence, Markov Models and Decision Making (Pg.. 15)	1	1
⭐ 2.1.8 Machine Learning Fundamentals (Pg.. 15)	1	2
⭐ 2.1.9 Machine Learning Data, Representations and Evaluation (Pg.. 15)	1	2
⭐ 2.1.10 Neural Networks and Ethics (Pg.. 16)	1	1
⭐ 2.1.11 Machine Learning Formulation and Evaluation (Pg.. 16)	1	2
⭐ 2.1.12 Deep Learning, RL and Ethics (Pg.. 17)	1	2
⭐ 2.1.13 Neural Network Training and Optimization (Pg.. 17)	1	1
⭐ 2.1.14 Regularization Techniques for Deep Networks (Pg.. 17)	1	1
⭐ 2.1.15 Convolutional and Residual Network Architectures (Pg.. 17)	1	1
⭐ 2.1.16 Transformer Architectures and Generative Models (Pg.. 18)	1	1
⭐ 2.1.17 AI Applications and Societal Impact (Pg.. 18)	2	2
⭐ 2.1.18 Additional AI Applications and Societal Impact (Pg.. 18)	2	2
2.1.19 Logical Representation and Reasoning (Pg.. 19)	Elective	
2.1.20 Probabilistic Representation and Reasoning (Pg.. 19)	Elective	
2.1.21 Planning (Pg.. 20)	Elective	
2.1.22 Agents and Cognitive Systems (Pg.. 20)	Elective	
2.1.23 Natural Language Processing (Pg.. 21)	Elective	
2.1.24 Robotics (Pg.. 21)	Elective	
2.1.25 Perception and Computer Vision (Pg.. 22)	Elective	

Table 2.1: List of KUs in the Artificial Intelligence area.

2.1.1 AI/Introduction to AI: Concepts and Intelligent Behavior (CS Core: 2 hrs, KA Core: 1 hr)

Topics:

Core

- Overview of AI problems, Examples of successful recent AI applications
- Definitions of agents with examples (e.g., reactive, deliberative)
- What is intelligent behavior?: 1. The Turing test and its flaws 2. Multimodal input and output 3. Simulation of intelligent behavior 4. Rational versus non-rational reasoning
- Overview of AI Applications, growth, and impact (economic, societal, ethics)

Non Core

- Philosophical issues
- History of AI

Learning Outcomes:

Core:

1. Describe the Turing test and the "Chinese Room" thought experiment [Explain]
2. Differentiate between optimal reasoning/behavior and human-like reasoning/behavior [Evaluate]
3. Differentiate the terms: AI, machine learning, and deep learning [Explain]

2.1.2 AI/Problem Characteristics and Agent Nature (CS Core: 1 hr, KA Core: 1 hr)

Topics:
Core

- Problem characteristics: 1. Fully versus partially observable 2. Single versus multi-agent 3. Deterministic versus stochastic 4. Static versus dynamic 5. Discrete versus continuous
- Nature of agents: 1. Autonomous, semi-autonomous, mixed-initiative autonomy 2. Reflexive, goal-based, and utility-based 3. Decision making under uncertainty and with incomplete information 4. The importance of perception and environmental interactions 5. Learning-based agents 6. Embodied agents: (a) sensors, dynamics, effectors
- Practice identifying problem characteristics in example environments
- Additional depth on nature of agents with examples
- Additional depth on AI Applications, Growth, and Impact (economic, societal, ethics, security)

Learning Outcomes:
Core:

1. Enumerate the characteristics of a specific problem [Apply]

2.1.3 AI/State and Uninformed Search (CS Core: 1 hr, KA Core: 2 hrs)

Topics:
Core

- State space representation of a problem: 1. Specifying states, goals, and operators 2. Factoring states into representations (hypothesis spaces) 3. Problem solving by graph search: (a) e.g., Graphs as a space, and tree traversals as exploration of that space (b) Dynamic construction of the graph (not given upfront)
- Uninformed graph search for problem solving: 1. Breadth-first search 2. Depth-first search: (a) With iterative deepening 3. Uniform cost search
- Space and time complexities of graph search algorithms
- Bidirectional search

Non Core

- Understanding the search space: 1. Constructing search trees 2. Dynamic search spaces 3. Combinatorial explosion of search space 4. Search space topology (e.g., ridges, saddle points, local minima)

Learning Outcomes:
Core:

1. Design the state space representation for a puzzle (e.g., N-queens or 3-jug problem) [Apply]
2. Select and implement an appropriate uninformed search algorithm for a problem (e.g., tic-tac-toe), and characterize its time and space complexities [Apply]

2.1.4 AI/Heuristic Search and CSP (CS Core: 1 hr, KA Core: 2 hrs)

Topics:
Core

- Heuristic graph search for problem solving: 1. Heuristic construction and admissibility 2. Hill-climbing 3. Local minima and the search landscape: (a) Local vs global solutions 4. Greedy best-first search 5. A* search
- Beam search
- Implementation of A* search
- Constraint satisfaction

Non Core

- Variations on A* (IDA*, SMA*, RBFS)

Learning Outcomes:

Core:

1. Select and implement an appropriate informed search algorithm for a problem after designing a helpful heuristic function (e.g., a robot navigating a 2D gridworld) [Apply]
2. Evaluate whether a heuristic for a given problem is admissible/can guarantee an optimal solution [Evaluate]
3. Design and implement A*/beam search to solve a problem, and compare it against other search algorithms in terms of the solution cost, number of nodes expanded, etc [Apply]
4. Compare and contrast various heuristic searches vis-a-vis applicability to a given problem [Evaluate]
5. Model a logic or Sudoku puzzle as a constraint satisfaction problem, solve it with backtrack search, and determine how much arc consistency can reduce the search space [Apply]

2.1.5 AI/Adversarial Search and Stochastic Optimization (CS Core: 1 hr, KA Core: 3 hrs)

Topics:

Core

- Two-player adversarial games: 1. Minimax search 2. Alpha-beta pruning: (a) Ply cutoff

Non Core

- Local search
- Tabu search
- Two-player adversarial games: 1. The horizon effect 2. Opening playbooks/endgame solutions 3. What it means to ”solve” a game (e.g., checkers)
- Implementation of minimax search, beam search
- Expectimax search (MDP-solving) and chance nodes
- Stochastic search: 1. Simulated annealing 2. Genetic algorithms 3. Monte-Carlo tree search

Learning Outcomes:

Core:

1. Apply minimax search in a two-player adversarial game (e.g., connect four), using heuristic evaluation at a particular depth to compute the scores to back up [Apply]
2. Apply minimax search with alpha-beta pruning to prune search space in a two-player adversarial game (e.g., connect four) [Apply]
3. Compare and contrast genetic algorithms with classic search techniques, explaining when it is most appropriate to use a genetic algorithm to learn a model versus other forms of optimization (e.g., gradient descent) [Evaluate]

NonCore:

4. Design and implement a genetic algorithm solution to a problem [Apply]
5. Design and implement a simulated annealing schedule to avoid local minima in a problem [Apply]

2.1.6 AI/Knowledge Representation and Probabilistic Reasoning Foundations (CS Core: 1 hr, KA Core: 1 hr)

Topics:

Core

- Types of representations: 1. Symbolic, logical: (a) Creating a representation from a natural language problem statement 2. Learned subsymbolic representations 3. Graphical models (e.g., naive Bayes, Bayesian network)
- Review of probabilistic reasoning, Bayes theorem
- Bayesian reasoning: 1. Bayesian inference
- Random variables and probability distributions: 1. Axioms of probability 2. Probabilistic inference 3. Bayes' Rule (derivation) 4. Bayesian inference (more complex examples)

Learning Outcomes:

Core:

1. Given a natural language problem statement, encode it as a symbolic or logical representation [Apply]
2. Compute a probabilistic inference in a real-world problem using Bayes' theorem to determine the probability of a hypothesis given evidence [Apply]
3. Apply Bayes' rule to determine the probability of a hypothesis given evidence [Apply]
4. Compute the probability of outcomes and test whether outcomes are independent [Apply]

2.1.7 AI/Independence, Markov Models and Decision Making (CS Core: 1 hr, KA Core: 1 hr)

Topics:
Core

- Independence
- Conditional Independence
- Markov chains and Markov models
- Utility and decision making

Learning Outcomes:
Core:

1. Explain how we can make decisions under uncertainty, using concepts such as Bayes theorem and utility [Explain]

2.1.8 AI/Machine Learning Fundamentals (CS Core: 1 hr, KA Core: 2 hrs)

Topics:
Core

- Definition and examples of a broad variety of machine learning tasks: 1. Supervised learning: (a) Classification (b) Regression 2. Reinforcement learning 3. Unsupervised learning: (a) Clustering
- Fundamental ideas: 1. No free lunch theorem: no one learner can solve all problems; representational design decisions have consequences. 2. Sources of error and undecidability in machine learning
- A simple statistical-based supervised learning such as linear regression or decision trees: 1. Focus on how they work without going into mathematical or optimization details; enough to understand and use existing implementations correctly
- The overfitting problem/controlling solution complexity (regularization, pruning - intuition only): 1. The bias (underfitting) - variance (overfitting) tradeoff

Learning Outcomes:
Core:

1. Describe the differences among the three main styles of learning (supervised, reinforcement, and unsupervised) and determine which is appropriate to a particular problem domain [Explain]
2. Differentiate the terms of AI, machine learning, and deep learning [Evaluate]
3. Frame an application as a classification problem, including the available input features and output to be predicted (e.g., identifying alphabetic characters from pixel grid input) [Apply]
4. Identify overfitting in the context of a problem and learning curves and describe solutions to overfitting [Evaluate]
5. Explain how machine learning works as an optimization/search process [Explain]

2.1.9 AI/Machine Learning Data, Representations and Evaluation (CS Core: 1 hr, KA Core: 2 hrs)

Topics:
Core

- Working with Data: 1. Data preprocessing: (a) Importance and pitfalls of preprocessing choices 2. Handling missing values (imputing, flag-as-missing): (a) Implications of imputing vs flag-as-missing 3. Encoding categorical variables, encoding real-valued data 4. Normalization/standardization 5. Emphasis on real data, not textbook examples

- Representations: 1. Hypothesis spaces and complexity 2. Simple basis feature expansion, such as squaring univariate features 3. Learned feature representations
- Machine learning evaluation: 1. Separation of train, validation, and test sets 2. Performance metrics for classifiers 3. Estimation of test performance on held-out data 4. Tuning the parameters of a machine learning model with a validation set 5. Importance of understanding what a model is doing, where its pitfalls/shortcomings are, and the implications of its decisions

Learning Outcomes:

Core:

1. Apply two or more simple statistical learning algorithms to a classification task and measure the classifiers' accuracy [Apply]
2. Implement a statistical learning algorithm and the corresponding optimization process to train the classifier and obtain a prediction on new data [Apply]
3. Explain proper ML evaluation procedures, including the differences between training and testing performance, and what can go wrong with the evaluation process leading to inaccurate reporting of ML performance [Explain]
4. Compare two machine learning algorithms on a dataset, implementing the data preprocessing and evaluation methodology (e.g., metrics and handling of train/test splits) from scratch [Apply]

2.1.10 AI/Neural Networks and Ethics (CS Core: 1 hr, KA Core: 1 hr)

Topics:

Core

- Basic neural networks: 1. Fundamentals of understanding how neural networks work and their training process, without details of the calculations 2. Basic introduction to generative neural networks (e.g., large language models)
- Ethics for Machine Learning: 1. Focus on real data, real scenarios, and case studies 2. Dataset/algorithmic/evaluation bias and unintended consequences

Learning Outcomes:

Core:

1. Describe the neural network training process and resulting learned representations [Explain]

2.1.11 AI/Machine Learning Formulation and Evaluation (CS Core: 1 hr, KA Core: 2 hrs)

Topics:

Core

- Formulation of simple machine learning as an optimization problem, such as least squares linear regression or logistic regression: 1. Objective function 2. Gradient descent 3. Regularization to avoid overfitting (mathematical formulation)
- Ensembles of models: 1. Simple weighted majority combination
- Performance evaluation: 1. Other metrics for classification (e.g., error, precision, recall) 2. Performance metrics for regressors 3. Confusion matrix 4. Cross-validation: (a) Parameter tuning (grid/random search, via cross-validation)

Learning Outcomes:

Core:

1. Compare and contrast several learning techniques (e.g., decision trees, logistic regression, naive Bayes, neural networks, and belief networks), providing examples of when each strategy is superior [Compare]
2. Evaluate the performance of a simple learning system on a real-world dataset [Evaluate]

NonCore:

3. Characterize the state of the art in learning theory, including its achievements and shortcomings [Explain]
4. Explain the problem of overfitting, along with techniques for detecting and managing the problem [Explain]
5. Explain the triple tradeoff among the size of a hypothesis space, the size of the training set, and performance accuracy [Explain]

2.1.12 AI/Deep Learning, RL and Ethics (CS Core: 1 hr, KA Core: 2 hrs)

Topics:
Core

- Deep learning: 1. Deep feed-forward networks (intuition only, no mathematics) 2. Convolutional neural networks (intuition only, no mathematics) 3. Visualization of learned feature representations from deep nets 4. Other architectures (generative NN, recurrent NN, transformers, etc.)
- Overview of reinforcement learning methods
- Two or more applications of machine learning algorithms: 1. E.g., medicine and health, economics, vision, natural language, robotics, game play
- Ethics for Machine Learning: 1. Continued focus on real data, real scenarios, and case studies 2. Privacy 3. Fairness 4. Intellectual property 5. Explainability

Learning Outcomes:
Core:

1. Visualize the training progress of a neural network through learning curves in a well-established toolkit (e.g., TensorBoard) and visualize the learned features of the network [Apply]

NonCore:

2. Given a real-world application of machine learning, describe ethical issues regarding the choices of data, preprocessing steps, algorithm selection, and visualization/presentation of results [Apply]

2.1.13 AI/Neural Network Training and Optimization (CS Core: 1 hr, KA Core: 1 hr)

Topics:
Core

- Modern backpropagation: computational graphs and automatic differentiation
- Advanced optimizers: momentum, RMSProp, Adam, and learning-rate schedules

Learning Outcomes:
Core:

1. Implement a deep neural network from scratch, including its backpropagation pass [Apply]
2. Select and tune an optimizer and its learning-rate schedule for training stability [Apply]

2.1.14 AI/Regularization Techniques for Deep Networks (CS Core: 1 hr, KA Core: 1 hr)

Topics:
Core

- Dropout regularization
- Batch normalization

Learning Outcomes:
Core:

1. Diagnose overfitting in a training run using regularization diagnostics [Analyze]
2. Apply dropout and batch normalization to improve a model's generalization [Apply]

2.1.15 AI/Convolutional and Residual Network Architectures (CS Core: 1 hr, KA Core: 1 hr)

Topics:
Core

- Deep convolutional neural network architectures
- Residual connections and ResNets

Learning Outcomes:
Core:

1. Design a deep CNN architecture for a given vision task [Apply]
2. Adapt a residual architecture to a specific vision domain [Apply]

2.1.16 AI/Transformer Architectures and Generative Models (CS Core: 1 hr, KA Core: 1 hr)

Topics:
Core

- Transformers and the attention mechanism
- Generative models: variational autoencoders (VAEs) and generative adversarial networks (GANs)

Learning Outcomes:
Core:

1. Train a transformer model on a real dataset [Apply]
2. Analyze and contrast VAE and GAN generative-model architectures [Analyze]

2.1.17 AI/AI Applications and Societal Impact (CS Core: 2 hrs, KA Core: 2 hrs)

Topics:
Core

- At least one application of AI to a specific problem and field, such as medicine, health, sustainability, social media, economics, education, robotics, etc. (choose at least one for the CS Core): 1. Formulating and evaluating a specific application as an AI problem: (a) How to deal with underspecified or ill-posed problems 2. Data availability/scarcity and cleanliness: (a) Basic data cleaning and preprocessing (b) Data set bias 3. Algorithmic bias 4. Evaluation bias 5. Assessment of societal implications of the application
- Deployed deep generative models: 1. High-level overview of deep image generative models (e.g., as of 2023, DALL-E, Midjourney, Stable Diffusion, etc.), their uses, and their shortcomings/pitfalls. 2. High-level overview of large language models (e.g., as of 2023, ChatGPT, Bard, etc.), their uses, and their shortcomings/pitfalls.
- Overview of societal impact of AI: 1. Ethics 2. Fairness 3. Trust/explainability 4. Privacy and usage of training data 5. Human autonomy and oversight/regulations/legal requirements 6. Sustainability

Learning Outcomes:
Core:

1. Given a real-world application domain and problem, formulate an AI solution to it, identifying proper data/input, preprocessing, representations, AI techniques, and evaluation metrics/methodology [Apply]

2.1.18 AI/Additional AI Applications and Societal Impact (CS Core: 2 hrs, KA Core: 2 hrs)

Topics:
Core

- One or more additional applications of AI to a broad set of problems and diverse fields, such as medicine, health, sustainability, social media, economics, education, robotics, etc. (choose a different area from that chosen for the CS Core): 1. Formulating and evaluating a specific application as an AI problem: (a) How to deal with underspecified or ill-posed problems 2. Data availability/scarcity and cleanliness: (a) Basic data cleaning and preprocessing (b) Data set bias 3. Algorithmic bias 4. Evaluation bias 5. Assessment of societal implications of the application
- Additional depth on deployed deep generative models: 1. Introduction to how deep image generative models work, (e.g., as of 2023, DALL-E, Midjourney, Stable Diffusion) including discussion of attention 2. Introduction to how large language models work, (e.g., as of 2023, ChatGPT, Bard) including discussion of attention 3. Idea of foundational models, how to use them, and the benefits/issues with training them from big data
- Analysis and discussion of the societal impact of AI: 1. Ethics 2. Fairness 3. Trust/explainability 4. Privacy and usage of training data 5. Human autonomy and oversight/regulations/legal requirements 6. Sustainability

Learning Outcomes:
Core:

1. Analyze the societal impact of one or more specific real-world AI applications, identifying issues regarding ethics, fairness, bias, trust, and explainability [Evaluate]
2. Describe some of the failure modes of current deep generative models for language or images, and how this could affect their use in an application [Describe]

2.1.19 AI/Logical Representation and Reasoning

Topics:
Non Core

- Review of propositional and predicate logic
- Resolution and theorem proving (propositional logic only): 1. Forward chaining, backward chaining
- Knowledge representation issues: 1. Description logics 2. Ontology engineering
- Semantic web
- Non-monotonic reasoning (e.g., non-classical logics, default reasoning)
- Argumentation
- Reasoning about action and change (e.g., situation and event calculus)
- Temporal and spatial reasoning
- Logic programming: 1. Prolog, Answer Set Programming
- Rule-based Expert Systems
- Semantic networks
- Model-based and Case-based reasoning

Learning Outcomes:
NonCore:

1. Translate a natural language (e.g., English) sentence into a predicate logic statement [Apply]
2. Convert a logic statement into causal form [Apply]
3. Apply resolution to a set of logic statements to answer a query [Apply]
4. Compare and contrast the most common models used for structured knowledge representation, highlighting their strengths and weaknesses [Evaluate]
5. Identify the components of non-monotonic reasoning and its usefulness as a representational mechanism for belief systems [Evaluate]
6. Compare and contrast the basic techniques for representing uncertainty [Evaluate]
7. Compare and contrast the basic techniques for qualitative representation [Evaluate]
8. Apply situation and event calculus to problems of action and change [Apply]
9. Explain the distinction between temporal and spatial reasoning, and how they interrelate [Explain]
10. Explain the difference between rule-based, case-based, and model-based reasoning techniques [Explain]
11. Define the concept of a planning system and how it differs from classical search techniques [Define]
12. Describe the differences between planning as search, operator-based planning, and propositional planning, providing examples of domains where each is most applicable [Describe]
13. Explain the distinction between monotonic and non-monotonic inference [Explain]

2.1.20 AI/Probabilistic Representation and Reasoning

Topics:
Non Core

- Conditional Independence review
- Knowledge representations: 1. Bayesian Networks: (a) Exact inference and its complexity (b) Markov blankets and d-separation (c) Randomized sampling (Monte Carlo) methods (e.g., Gibbs sampling) 2. Markov Networks 3. Relational probability models 4. Hidden Markov Models

- Decision Theory: 1. Preferences and utility functions 2. Maximizing expected utility 3. Game theory

Learning Outcomes:
NonCore:

1. Compute the probability of a hypothesis given the evidence in a Bayesian network [Apply]
2. Explain how conditional independence assertions allow for greater efficiency of probabilistic systems [Explain]
3. Identify examples of knowledge representations for reasoning under uncertainty [List]
4. State the complexity of exact inference. Identify methods for approximate inference [State]
5. Design and implement at least one knowledge representation for reasoning under uncertainty [Apply]
6. Describe the complexities of temporal probabilistic reasoning [Describe]
7. Design and implement an HMM as one example of a temporal probabilistic system [Apply]
8. Describe the relationship between preferences and utility functions [Describe]
9. Explain how utility functions and probabilistic reasoning can be combined to make rational decisions [Explain]

2.1.21 AI/Planning

Topics:
Non Core

- Review of propositional and first-order logic
- Planning operators and state representations
- Total order planning
- Partial-order planning
- Plan graphs and GraphPlan
- Hierarchical planning
- Planning languages and representations: 1. PDDL
- Multi-agent planning
- MDP-based planning
- Interconnecting planning, execution, and dynamic replanning: 1. Conditional planning 2. Continuous planning 3. Probabilistic planning

Learning Outcomes:
NonCore:

1. Construct the state representation, goal, and operators for a given planning problem [Apply]
2. Encode a planning problem in PDDL and use a planner to solve it [Apply]
3. Given a set of operators, initial state, and goal state, draw the partial-order planning graph and include ordering constraints to resolve all conflicts [Apply]
4. Construct the complete planning graph for GraphPlan to solve a given problem [Apply]

2.1.22 AI/Agents and Cognitive Systems

Topics:
Non Core

- Agent architectures (e.g., reactive, layered, cognitive)
- Agent theory (including mathematical formalisms)
- Rationality, Game Theory: 1. Decision-theoretic agents 2. Markov decision processes (MDP) 3. Bandit algorithms
- Software agents, personal assistants, and information access: 1. Collaborative agents 2. Information-gathering agents 3. Believable agents (synthetic characters, modeling emotions in agents)
- Learning agents
- Cognitive systems: 1. Cognitive architectures (e.g., ACT-R, SOAR, ICARUS, FORR) 2. Capabilities (e.g., perception, decision making, prediction, knowledge maintenance) 3. Knowledge representation, organization, utilization, acquisition, and refinement 4. Applications and evaluation of cognitive systems

- Multi-agent systems: 1. Collaborating agents 2. Agent teams 3. Competitive agents (e.g., auctions, voting) 4. Swarm systems and biologically inspired models 5. Multi-agent learning
- Human-agent interaction: 1. Communication methodologies (verbal and non-verbal) 2. Practical issues 3. Applications: (a) Trading agents, supply chain management (b) Ethical issues of AI interactions with humans (c) Regulation and legal requirements of AI systems for interacting with humans

Learning Outcomes:
NonCore:

1. Characterize and contrast the standard agent architectures [Evaluate]
2. Describe the applications of agent theory to domains such as software agents, personal assistants, and believable agents, and discuss associated ethical implications [Describe]
3. Describe the primary paradigms used by learning agents [Describe]
4. Demonstrate using appropriate examples how multi-agent systems support agent interaction [Demonstrate]
5. Construct an intelligent agent using a well-established cognitive architecture (ACT-R, SOAR) for solving a specific problem [Apply]

2.1.23 AI/Natural Language Processing

Topics:
Non Core

- Deterministic and stochastic grammars
- Parsing algorithms: 1. CFGs and chart parsers (e.g., CYK) 2. Probabilistic CFGs and weighted CYK
- Representing meaning/Semantics: 1. Logic-based knowledge representations 2. Semantic roles 3. Temporal representations 4. Beliefs, desires, and intentions
- Corpus-based methods
- N-grams and HMMs
- Smoothing and backoff
- Examples of use: POS tagging and morphology
- Information retrieval: 1. Vector space model: (a) TF & IDF 2. Precision and recall
- Information extraction
- Language translation
- Text classification, categorization: 1. Bag of words model
- Deep learning for NLP: 1. RNNs 2. Transformers 3. Multi-modal embeddings (e.g., images + text) 4. Generative language models

Learning Outcomes:
NonCore:

1. Define and contrast deterministic and stochastic grammars, providing examples to show the adequacy of each [Define]
2. Simulate, apply, or implement classic and stochastic algorithms for parsing natural language [Apply]
3. Identify the challenges of representing meaning [Analyze]
4. List the advantages of using standard corpora. Identify examples of current corpora for a variety of NLP tasks [List]
5. Identify techniques for information retrieval, language translation, and text classification [Analyze]
6. Implement a TF/IDF transform, use it to extract features from a corpus, and train an off-the-shelf machine learning algorithm using those features to do text classification [Apply]

2.1.24 AI/Robotics

Topics:
Non Core

- Overview: problems and progress: 1. State-of-the-art robot systems, including their sensors and an overview of their sensor processing 2. Robot control architectures, e.g., deliberative vs reactive

control and Braitenberg vehicles 3. World modeling and world models 4. Inherent uncertainty in sensing and in control
- Sensors and effectors: 1. Sensors: e.g., LIDAR, sonar, vision, depth, stereoscopic, event cameras, microphones, haptics 2. Effectors: e.g., wheels, arms, grippers
- Coordinate frames, translation, and rotation (2D and 3D)
- Configuration space and environmental maps
- Interpreting uncertain sensor data
- Localization and mapping
- Navigation and control
- Forward and inverse kinematics
- Motion path planning and trajectory optimization
- Manipulation and grasping
- Joint control and dynamics
- Vision-based control
- Multiple-robot coordination and collaboration
- Human-robot interaction: 1. Shared workspaces 2. Human-robot teaming and physical HRI 3. Social assistive robots 4. Motion/task/goal prediction 5. Collaboration and communication (explicit vs implicit, verbal or symbolic vs non-verbal or visual) 6. Trust
- Applications and Societal, Economic, and Ethical Issues: 1. Societal, economic, right-to-work implications 2. Ethical and privacy implications of robotic applications 3. Liability in autonomous robotics 4. Autonomous weapons and ethics 5. Human oversight and control

Learning Outcomes:
NonCore:

1. List capabilities and limitations of today's state-of-the-art robot systems, including their sensors and the crucial sensor processing that informs those systems [List]
2. Integrate sensors, actuators, and software into a robot designed to undertake a specific task [Apply]
3. Program a robot to accomplish simple tasks using deliberative, reactive, and/or hybrid control architectures [Apply]
4. Implement fundamental motion planning algorithms within a robot configuration space [Apply]
5. Characterize the uncertainties associated with common robot sensors and actuators; articulate strategies for mitigating these uncertainties [Characterize]
6. List the differences among robots' representations of their external environment, including their strengths and shortcomings [List]
7. Compare and contrast at least three strategies for robot navigation within known and/or unknown environments, including their strengths and shortcomings [Compare]
8. Describe at least one approach for coordinating the actions and sensing of several robots to accomplish a single task [Describe]
9. Compare and contrast a multi-robot coordination and a human-robot collaboration approach and attribute their differences to differences between the problem settings [Compare]
10. Analyze the societal, economic, and ethical issues of a real-world robotics application [Analyze]

2.1.25 AI/Perception and Computer Vision

Topics:
Non Core

- Computer vision: 1. Image acquisition, representation, processing, and properties 2. Shape representation, object recognition, and segmentation 3. Motion analysis 4. Generative models
- Audio and speech recognition
- Touch and proprioception
- Other modalities (e.g., olfaction)
- Modularity in recognition
- Approaches to pattern recognition: 1. Classification algorithms and measures of classification quality 2. Statistical techniques 3. Deep learning techniques

Learning Outcomes:
NonCore:

1. Summarize the importance of image and object recognition in AI and indicate several significant applications of this technology [Summarize]
2. List at least three image-segmentation approaches, such as thresholding, edge-based and region-based algorithms, along with their defining characteristics, strengths, and weaknesses [List]
3. Implement 2d object recognition based on contour-based and/or region-based shape representations [Apply]
4. Distinguish the goals of sound-recognition, speech-recognition, and speaker-recognition and identify how the raw audio signal will be handled differently in each of these cases [Distinguish]
5. Provide at least two examples of a transformation of a data source from one sensory domain to another, e.g., tactile data interpreted as single-band 2d images [Generate]
6. Implement a feature-extraction algorithm on real data, e.g., an edge or corner detector for images or vectors of Fourier coefficients describing a short slice of audio signal [Apply]
7. Implement an algorithm combining features into higher-level percepts, e.g., a contour or polygon from visual primitives or phoneme hypotheses from an audio signal [Apply]
8. Implement a classification algorithm that segments input percepts into output categories and quantitatively evaluates the resulting classification [Apply]
9. Evaluate the performance of the underlying feature-extraction, relative to at least one alternative possible approach (whether implemented or not) in its contribution to the classification task (8), above [Evaluate]
10. Describe at least three classification approaches, their prerequisites for applicability, their strengths, and their shortcomings [Describe]
11. Implement and evaluate a deep learning solution to problems in computer vision, such as object or scene recognition [Apply]

2.2 Algorithmic Foundations (AL)

Algorithms and data structures are fundamental to computer science, since every theoretical computation and applied program consists of algorithms that operate on data elements possessing some underlying structure. Selecting appropriate computational solutions to real-world problems benefits from understanding the theoretical and practical capabilities and limitations of available algorithms and paradigms, including their impact on the environment and society. Moreover, this understanding provides insight into the intrinsic nature of computation, computational problems, and computational problem-solving as well as possible solution techniques independent of programming language, programming paradigm, computer hardware, or other implementation aspects.

This knowledge area focuses on the nature of computation including the concepts and skills required to design and analyze algorithms for solving real-world computational problems. It complements the implementation of algorithms and data structures found in the Software Development Foundations (SDF) knowledge area. As algorithms and data structures are essential in all advanced areas of computer science, this area provides the algorithmic foundations that every computer science graduate is expected to know. Exposure to the breadth of these foundational AL topics is designed to provide students with the basis for studying these topics in more depth, for studying additional computation and algorithm topics, and for learning advanced algorithms across a variety of CS knowledge areas and CS+X disciplines.

Knowledge Area (KA)	CS Core	KA Core
⭐ 2.2.1 Foundational Data Structures (Pg.. 24)	5	3
⭐ 2.2.2 Foundational Algorithms (Pg.. 24)	4	2
⭐ 2.2.3 Advanced Algorithms (Pg.. 25)	3	2
⭐ 2.2.4 Algorithmic Strategies (Pg.. 25)	6	
⭐ 2.2.5 Complexity Analysis Framework (Pg.. 26)	2	1
⭐ 2.2.6 Asymptotic Notation and Complexity Classes (Pg.. 26)	1	1
⭐ 2.2.7 Complexity Analysis: Recursion, Amortization, and Tight Bounds (Pg.. 27)	2	1
⭐ 2.2.8 Computational Complexity Theory (Pg.. 27)	2	1
⭐ 2.2.9 Formal Languages and Automata (Pg.. 27)	4	10
⭐ 2.2.10 Computability and Decidability (Pg.. 28)	6	14
2.2.11 Society, Ethics, and the Profession (Pg.. 29)	Elective	

Table 2.2: List of KUs in the Algorithmic Foundations area.

2.2.1 AL/Foundational Data Structures (CS Core: 5 hrs, KA Core: 3 hrs)

Topics:
Core

- Abstract Data Type (ADT) and operations on an ADT: 1. Dictionary operations (insert, delete, find)
- Arrays: 1. Numeric vs non-numeric, character strings 2. Single (vector) vs multidimensional (matrix)
- Records/Structs/Tuples and Objects
- Linked lists (for historical reasons): 1. Single vs Double and Linear vs Circular
- Stacks
- Queues and deques: 1. Heap-based priority queue
- Hash tables/maps: 1. Collision resolution and complexity (e.g., probing, chaining, rehash)
- Trees: 1. Binary, n-ary, and search trees 2. Balanced (e.g., AVL, Red-Black, Heap)
- Sets

Learning Outcomes:
Core:

1. For each ADT/Data-Structure in this unit: (a) Explain its definition, properties, representation(s), and associated ADT operations. (b) Explain step-by-step how the ADT operations associated with the data structure transform it. [Explain]
2. Explain how collision avoidance and collision resolution is handled in hash tables [Explain]
3. Explain the heap property and the use of heaps as an implementation of a priority queue [Explain]

2.2.2 AL/Foundational Algorithms (CS Core: 4 hrs, KA Core: 2 hrs)

Topics:
Core

- Graphs (e.g., [un]directed, [a]cyclic, [un]connected, and [un]weighted): 1. Graph representation: adjacency list vs matrix
- Search algorithms: 1. $O(n)$ complexity (e.g., linear/sequential array/list search) 2. $O(\log_2 n)$ complexity (e.g., binary search) 3. $O(\log_b n)$ complexity (e.g., uninformed depth/breadth-first tree search)
- Sorting algorithms (e.g., stable, unstable): 1. $O(n^2)$ complexity (e.g., insertion, selection) 2. $O(n \log n)$ complexity (e.g., quicksort, merge, timsort)
- Graph algorithms: 1. Shortest path (e.g., Dijkstra's, Floyd's) 2. Minimal spanning tree (e.g., Prim's, Kruskal's)
- Sorting algorithms: 1. $O(n \log n)$ complexity heapsort 2. Pseudo $O(n)$ complexity (e.g., bucket, counting, radix)
- Graph algorithms: 1. Transitive closure (e.g., Warshall's) 2. Topological sort

- Matching: 1. Efficient string matching (e.g., Boyer-Moore, Knuth-Morris-Pratt) 2. Longest common subsequence matching 3. Regular expression matching

Learning Outcomes:

Core:

1. For each algorithm in this unit explain step-by-step how the algorithm operates [Explain]
2. For each algorithmic approach (e.g., sorting) in this unit apply a prototypical example of the approach (e.g., merge sort) [Apply]
3. Given requirements for a problem, develop multiple solutions using various data structures and algorithms. Subsequently, evaluate the suitability, strengths, and weaknesses selecting an approach that best satisfies the requirements [Create]
4. Explain factors beyond computational efficiency that influence the choice of algorithms, such as programming time, maintainability, and the use of application-specific patterns in the input data [Explain]
5. For each of the algorithms and algorithmic approaches in the KA Core topics: (a) Explain a prototypical example of the algorithm, and (b) Explain step-by-step how the algorithm operates. [Explain]

2.2.3 AL/Advanced Algorithms (CS Core: 3 hrs, KA Core: 2 hrs)

Topics:

Non Core

- Cryptography algorithms (e.g., SHA-256)
- Parallel algorithms
- Consensus algorithms (e.g., Blockchain): 1. Proof of work vs proof of stake
- Quantum computing algorithms: 1. Oracle-based (e.g., Deutsch-Jozsa, Bernstein-Vazirani, Simon) 2. Superpolynomial speed-up via QFT (e.g., Shor's) 3. Polynomial speed-up via amplitude amplification (e.g., Grover's)
- Fast-Fourier Transform (FFT) algorithm
- Differential evolution algorithm

Learning Outcomes:

NonCore:

1. An appreciation of quantum computation and its application to certain problems [Explain]

2.2.4 AL/Algorithmic Strategies (CS Core: 6 hrs)

Topics:

Core

- Paradigms: 1. Brute-Force (e.g., linear search, selection sort, traveling salesperson, knapsack) 2. Decrease-and-Conquer: (a) By a Constant (e.g., insertion sort, topological sort) (b) By a Constant Factor (e.g., binary search) (c) By a Variable Size (e.g., Euclid's) 3. Divide-and-Conquer (e.g., binary search, quicksort, mergesort, Strassen's) 4. Greedy (e.g., Dijkstra's, Kruskal's, Knapsack) 5. Transform-and-Conquer: (a) Instance simplification (e.g., find duplicates via list presort) (b) Representation change (e.g., heapsort) (c) Problem reduction (e.g., least-common-multiple, linear programming) (d) Dynamic programming (e.g., Floyd's, Marshall, Bellman-Ford) 6. Space vs time tradeoffs (e.g., hashing)
- Handling exponential growth (e.g., heuristic A*, branch-and-bound, backtracking)
- Iteration vs recursion (e.g., factorial, tree search)
- Paradigms: 1. Approximation algorithms 2. Iterative improvement (e.g., Ford-Fulkerson, simplex) 3. Randomized/Stochastic algorithms (e.g., max-cut, balls and bins)

Non Core

- Quantum computing

Learning Outcomes:

Core:

1. For each of the paradigms in this unit: (a) Explain its definitional characteristics (b) Explain an example that demonstrates the paradigm including how this example satisfies the paradigm's characteristics. [Explain]

2. For each of the algorithms in the Foundational Data Structures, Foundational Algorithms, Advanced Algorithms unit, explain the paradigm used by the algorithm and how it exemplifies this paradigm [Explain]
3. Given an algorithm, explain the paradigm used by the algorithm and how it exemplifies this paradigm [Explain]
4. Give a real-world problem, evaluate appropriate algorithmic paradigms and algorithms from these paradigms that address the problem including evaluating the tradeoffs among the paradigms and algorithms selected [Evaluate]
5. Give examples of iterative and recursive algorithms that solve the same problem, explain the benefits and disadvantages of each approach [Explain]
6. Evaluate whether a greedy approach leads to an optimal solution [Evaluate]
7. Explain various approaches for addressing computational problems whose algorithmic solutions are exponential [Explain]

2.2.5 AL/Complexity Analysis Framework (CS Core: 2 hrs, KA Core: 1 hr)

Topics:
Core

- Complexity Analysis Framework: 1. Best, average, and worst-case performance of an algorithm 2. Empirical and relative (Order of Growth) measurements 3. Input size and primitive operations 4. Time and space efficiency
- Empirical measurements of performance
- Time and space tradeoffs in algorithms

Learning Outcomes:
Core:

1. Prepare a presentation that explains to first year students the basic concepts of algorithmic complexity including best, average, and worst-case algorithm behavior, Big-O, Omega, and Theta notations, complexity classes, time and space tradeoffs, empirical measurement, and impact on practical problems [Explain]
2. For each algorithm in the Foundational Data Structures, Foundational Algorithms, Advanced Algorithms unit, explain its runtime complexity class and why it belongs to this class [Explain]
3. Informally evaluate the foundational complexity class of simple algorithms [Evaluate]
4. Develop empirical studies to determine and validate hypotheses about the runtime complexity of various algorithms by running algorithms on input of various sizes and comparing actual performance to the theoretical analysis [Create]
5. Explain examples that illustrate time-space tradeoffs of algorithms [Explain]
6. Explain how tree balance affects the efficiency of binary search tree operations [Explain]

2.2.6 AL/Asymptotic Notation and Complexity Classes (CS Core: 1 hr, KA Core: 1 hr)

Topics:
Core

- Asymptotic complexity analysis (average and worst-case bounds): 1. Big-O, Big-Omega, and Big-Theta formal notations 2. Foundational Complexity Classes and Representative Examples/Problems: (a) $O(1)$ Constant (e.g., array access) (b) $O(\log_2 n)$ Logarithmic (e.g., binary search) (c) $O(n)$ Linear (e.g., linear search) (d) $O(n \log_2 n)$ Log Linear (e.g., mergesort) (e) $O(n^2)$ Quadratic (e.g., selection sort) (f) $O(n^c)$ Polynomial (e.g., $O(n^3)$ Gaussian elimination) (g) $O(2^n)$ Exponential (e.g., Knapsack, Satisfiability (SAT), Traveling Sales-Person (TSP), all subsets) (h) $O(n!)$ Factorial (e.g., Hamiltonian circuit, all permutations)

Learning Outcomes:
Core:

1. Using examples, explain each of the foundational complexity classes in this unit [Explain]
2. For each foundational complexity class in this unit, explain an algorithm that demonstrates the associated runtime complexity [Explain]

3. Explain to a non-technical audience the significance of tractable versus intractable algorithms using an intuitive explanation of Big-O complexity [Explain]
4. Apply Big-O notation to give upper case bounds on time/space complexity of algorithms [Apply]

2.2.7 AL/Complexity Analysis: Recursion, Amortization, and Tight Bounds (CS Core: 2 hrs, KA Core: 1 hr)

Topics:
Core
- Little-o, Little-Omega, and Little Theta notations
- Formal recursive analysis
- Amortized analysis

Learning Outcomes:
Core:
1. Given a problem to program for which there may be several algorithmic approaches, evaluate them and determine which are feasible, and select one that is optimal in implementation and run-time behavior [Evaluate]
2. Use recurrence relations to evaluate the time complexity of recursively defined algorithms [Apply]
3. Apply elementary recurrence relations using a form of the Master Theorem [Apply]

2.2.8 AL/Computational Complexity Theory (CS Core: 2 hrs, KA Core: 1 hr)

Topics:
Core
- Tractability and intractability: 1. P, NP, and NP-Complete Complexity Classes 2. NP-Complete Problems (e.g., SAT, Knapsack, TSP) 3. Reductions
- Turing Machine-based models of complexity: 1. Time complexity: (a) P, NP, NP-C, and EXP classes (b) Cook-Levin theorem 2. Space Complexity: (a) NSpace and PSpace (b) Savitch's theorem

Learning Outcomes:
Core:
1. Explain the significance of NP-Completeness [Explain]
2. Explain how NP-Hard is a lower bound and NP is an upper bound for NP-Completeness [Explain]
3. Explain examples of NP-complete problems [Explain]
4. Explain the Cook-Levin Theorem and the NP-Completeness of SAT [Explain]
5. Explain the classes P and NP [Explain]
6. Prove that a problem is NP-Complete by reducing a classic known NP-C problem to it (e.g., 3SAT and Clique) [Create]
7. Explain the P-space class and its relation to the EXP class [Explain]

2.2.9 AL/Formal Languages and Automata (CS Core: 4 hrs, KA Core: 10 hrs)

Topics:
Core
- Formal automata: 1. Finite State 2. Pushdown 3. Linear Bounded 4. Turing Machine
- Formal languages, grammars and Chomsky Hierarchy: 1. Regular (Type-3): (a) Regular Expressions 2. Context-Free (Type-2) 3. Context-Sensitive (Type-1) 4. Recursively Enumerable (Type-0)
- Relations among formal automata, languages, and grammars
- Deterministic and nondeterministic automata
- Pumping Lemma proofs: 1. Proof of Finite State/Regular-Language limitation 2. Pushdown Automata/Context-Free-Language limitation
- Equivalent models of algorithmic computation: 1. Turing Machines and Variations (e.g., multi-tape, non-deterministic) 2. Lambda Calculus 3. Mu-Recursive Functions

Learning Outcomes:
Core:

1. For each formal automaton in this unit: (a) Explain its definition comparing its characteristics with this unit's other automata (b) Using an example, explain step-by-step how the automaton operates on input including whether it accepts the associated input (c) Explain an example of inputs that can and cannot be accepted by the automaton. [Explain]
2. Given a problem, develop an appropriate automaton that addresses the problem [Create]
3. Develop a regular expression for a given regular language expressed in natural language [Create]
4. Explain the difference between regular expressions (Type-3 acceptors) and the regular expressions (Type-2 acceptors) used in programming languages [Explain]
5. For each formal model in this unit: (a) Explain its definition comparing its characteristics with the others in this unit (b) Explain example inputs that are and cannot be accepted by the language/grammar. [Explain]
6. For each formal automaton in this unit explain (compare/contrast) its deterministic and nondeterministic capabilities [Explain]
7. Apply pumping lemmas, or alternative means, to prove the limitations of Finite State and Pushdown automata [Apply]
8. Convert among equivalently powerful notations for a language, including among DFAs, NFAs, and regular expressions, and between PDAs and CFGs [Apply]

2.2.10 AL/Computability and Decidability (CS Core: 6 hrs, KA Core: 14 hrs)

Topics:
Core

- Decidability, (un)computability, and halting
- The Church-Turing thesis
- Algorithmic correctness: 1. Invariants (e.g., in iteration, recursion, tree search)
- Decidability: 1. Arithmetization and diagonalization
- Reducibility and reductions
- Time complexity based on Turing Machine
- Space complexity (e.g., Pspace, Savitch's Theorem)

Non Core

- Quantum computation: 1. Postulates of quantum mechanics: (a) State space (b) State evolution (c) State composition (d) State measurement 2. Column vector representations of qubits 3. Matrix representations of quantum operations 4. Simple quantum gates (e.g., XNOT, CNOT)

Learning Outcomes:
Core:

1. Explain a universal Turing Machine and its operation [Explain]
2. Present to an audience of co-workers and managers the impossibility of providing them a program that checks all other programs, including some seemingly simple ones, for infinite loops including an explanation of the Halting problem, why it has no algorithmic solution, and its significance for real-world algorithmic computation [Explain]
3. Explain examples of classic uncomputable problems [Explain]
4. Explain the Church-Turing Thesis and its significance for algorithmic computation [Explain]
5. Explain how (loop) invariants can be used to prove the correctness of an algorithm [Explain]
6. Apply arithmetization and diagonalization to prove the Halting Problem for Turing Machines is Undecidability [Apply]
7. Given a known undecidable language, apply a mapping reduction or computational history to prove that another language is undecidable [Apply]
8. Explain Rice's theorem and its significance [Explain]
9. Explain an example proof of a problem that is uncomputable by reducing a classic known uncomputable problem to it [Explain]
10. Explain the Primitive and General Recursive functions (zero, successor, selection, primitive recursion, composition, and Mu), their significance, and Turing Machine implementations [Explain]

11. Explain how computation is performed in Lambda Calculus (e.g., Alpha conversion and Beta reduction) [Explain]

NonCore:

12. For a quantum system give examples that explain the following postulates: (a) State Space - system state represented as a unit vector in Hilbert space (b) State Evolution - the use of unitary operators to evolve system state (c) State Composition - the use of tensor product to compose systems states (d) State Measurement - the probabilistic output of measuring a system state. [Explain]
13. Explain the operation of a quantum XNOT or CNOT gate on a quantum bit represented as a matrix and column vector, respectively [Explain]

2.2.11 AL/Society, Ethics, and the Profession

Topics:

Core

- Social, ethical, and secure algorithms
- Algorithmic fairness
- Anonymity (e.g., Differential Privacy)
- Accountability/Transparency
- Responsible algorithms
- Economic and other impacts of inefficient algorithms
- Sustainability
- Context aware computing

Learning Outcomes:

Core:

1. Develop algorithmic solutions to real-world societal problems, such as routing an ambulance to a hospital [Create]
2. Explain the impact that an algorithm may have on the environment and society when used to solve a real-world problem while considering its sustainability and that it can affect different societal groups in different ways [Explain]
3. Prepare a presentation that justifies the selection of appropriate data structures and/or algorithms to solve a given real-world problem [Explain]
4. Explain an example that articulates how differential privacy protects knowledge of an individual's data [Explain]
5. Explain the environmental impacts of design choices that relate to algorithm design [Explain]
6. Explain the tradeoffs involved in proof-of-work and proof-of-stake algorithms [Explain]

2.3 Architecture and Organization (AR)

Computing professionals spend considerable time writing efficient code to solve a particular problem in an application domain. As the shift from sequential to parallel processing occurs, a deeper understanding of the underlying computer architectures is necessary. Architecture can no longer be viewed as a black box where principles from one architecture can be applied to another. Instead, programmers should look inside the black box and use specific components to enhance system performance and energy efficiency.

The Architecture and Organization (AR) knowledge area aims to develop a deeper understanding of the hardware environments upon which almost all computing is based, and the relevant interfaces provided to higher software layers. The target hardware comprises low-end embedded system processors up to high-end enterprise multiprocessors.

The topics in this knowledge area will benefit students by enabling them to appreciate the fundamental architectural principles of modern computer systems, including the challenge of harnessing parallelism to sustain performance and energy improvements into the future. This KA will help computer science students depart from the black box approach and become more aware of the underlying computer system and the efficiencies specific architectures can achieve.

Knowledge Area (KA)	CS Core	KA Core
⭐ 2.3.1 Digital Logic and Digital Systems (Pg.. 30)		3
⭐ 2.3.2 Machine-Level Data Representation (Pg.. 30)	1	
⭐ 2.3.3 Assembly Level Machine Organization (Pg.. 31)	1	2
⭐ 2.3.4 Memory Hierarchy (Pg.. 31)	6	
⭐ 2.3.5 Interfacing and Communication (Pg.. 32)	1	
⭐ 2.3.6 Functional Organization (Pg.. 32)		2
⭐ 2.3.7 Performance and Energy Efficiency (Pg.. 32)		3
⭐ 2.3.8 Heterogeneous Architectures (Pg.. 33)		2
⭐ 2.3.9 Secure Processor Architectures (Pg.. 33)		2
⭐ 2.3.10 Quantum Architectures (Pg.. 34)		2
2.3.11 Sustainability Issues (Pg.. 35)	Elective	

Table 2.3: List of KUs in the Architecture and Organization area.

2.3.1 AR/Digital Logic and Digital Systems (KA Core: 3 hrs)

Topics:
Core

- Combinational vs sequential logic/field programmable gate arrays (FPGAs): 1. Fundamental combinational 2. Sequential logic building block
- Computer-aided design tools that process hardware and architectural representations
- High-level synthesis: 1. Register transfer notation 2. Hardware description language (e.g., Verilog/VHDL/Chisel)
- System-on-chip (SoC) design flow
- Physical constraints: 1. Gate delays 2. Fan-in and fan-out 3. Energy/power 4. Speed of light

Learning Outcomes:
Core:

1. Discuss the progression of computer technology components from vacuum tubes to VLSI, from mainframe computer architectures to the organization of warehouse-scale computers [Debate]
2. Describe parallelism and data dependencies between and within components in a modern heterogeneous computer architecture [Describe]
3. Explain the relationship between parallelism and power consumption [Explain]
4. Construct the design of basic building blocks for a computer: arithmetic-logic unit (gate-level), registers (gate-level), central processing unit (register transfer-level), and memory (register transfer-level) [Create]
5. Evaluate simple building blocks (e.g., arithmetic-logic unit, registers, movement between registers) of a simple computer design [Evaluate]
6. Analyze the timing behavior of a pipelined processor, identifying data dependency issues [Analyze]

2.3.2 AR/Machine-Level Data Representation (CS Core: 1 hr)

Topics:
Core

- Overview and history of computer architecture
- Bits, bytes, and words
- Unsigned, signed and two's complement representations
- Numeric data representation and number bases: 1. Fixed-point 2. Floating-point
- Representation of non-numeric data
- Representation of records, arrays and UTF data types

Learning Outcomes:
Core:

1. Discuss why everything in computers are data, including instructions [Debate]
2. Explain how fixed-length number representations can affect accuracy and precision [Explain]

3. Describe how negative integers are stored in sign-magnitude and two's-complement representations [Describe]
4. Discuss how different formats can represent numerical data [Debate]
5. Explain the bit-level representation of non-numeric data, such as characters, strings, records, and arrays [Explain]
6. Translate numerical data from one format to another [Translate]
7. Describe how a single adder (without overflow detection) can handle both signed (two's complement) and unsigned (binary) input without "knowing" which format a given input is using [Describe]

2.3.3 AR/Assembly Level Machine Organization (CS Core: 1 hr, KA Core: 2 hrs)

Topics:
Core

- von Neumann machine architecture
- Control unit: instruction fetch, decode, and execution
- Introduction to SIMD vs MIMD and the Flynn taxonomy
- Shared memory multiprocessors/multicore organization
- Instruction set architecture (ISA) (e.g., x86, ARM and RISC-V): 1. Fixed vs variable-width instruction sets 2. Instruction formats 3. Data manipulation, control, I/O 4. Addressing modes 5. Machine language programming 6. Assembly language programming
- Subroutine call and return mechanisms
- I/O and interrupts
- Heap, static, stack, and code segments

Learning Outcomes:
Core:

1. Discuss how the classical von Neumann functional units are implemented in embedded systems, particularly on-chip and off-chip memory [Debate]
2. Describe how instructions are executed in a classical von Neumann machine, with extensions for threads, multiprocessor synchronization, and SIMD execution [Describe]
3. Assess an example diagram with instruction-level parallelism and hazards to describe how they are managed in typical processor pipelines [Assess]
4. Discuss how instructions are represented at the machine level and in the context of a symbolic assembler [Debate]
5. Map an example of high-level language patterns into assembly/machine language notations [Map]
6. Contrast different instruction formats considering aspects such as addresses per instruction and variable-length vs fixed-length formats [Contrast]
7. Analyze a subroutine diagram to comment on how subroutine calls are handled at the assembly level [Analyze]
8. Describe basic concepts of interrupts and I/O operations [Describe]
9. Write a simple assembly language program for string/array processing and manipulation [Write]

2.3.4 AR/Memory Hierarchy (CS Core: 6 hrs)

Topics:
Core

- Memory hierarchy: the importance of temporal and spatial locality
- Main memory organization and operations
- Persistent memory (e.g., SSD, standard disks)
- Latency, cycle time, bandwidth, and interleaving
- Cache memories: 1. Address mapping 2. Block size 3. Replacement and store policy 4. Prefetching
- Multiprocessor cache coherence
- Virtual memory (hardware support)
- Fault handling and reliability

- Reliability: 1. Error coding 2. Data compression 3. Data integrity
- Processing In-Memory (PIM)

Learning Outcomes:

Core:

1. Using a memory system diagram, identify the main types of memory technology (e.g., SRAM, DRAM) and their relative cost and performance [Analyze]
2. Measure the effect of memory latency on running time [Evaluate]
3. Enumerate the functions of a system with virtual memory management [Enumerate]
4. Compute average memory access time under various cache and memory configurations and mixes of instruction and data references [Compute]

2.3.5 AR/Interfacing and Communication (CS Core: 1 hr)

Topics:

Core

- I/O fundamentals: 1. Handshaking and buffering 2. Programmed I/O 3. Interrupt-driven I/O
- Interrupt structures: vectored and prioritized, interrupt acknowledgment
- I/O devices (e.g., mouse, keyboard, display, camera, sensors, actuators)
- External storage, physical organization, and drives
- Buses fundamentals: 1. Bus protocols 2. Arbitration 3. Direct-memory access (DMA)

Learning Outcomes:

Core:

1. Analyze an interrupt control diagram to comment on how interrupts are used to implement I/O control and data transfers [Analyze]
2. Enumerate various types of buses in a computer system [Enumerate]
3. List the advantages of magnetic disks and contrast them with those of solid-state disks [List]

2.3.6 AR/Functional Organization (KA Core: 2 hrs)

Topics:

Core

- Implementation of simple datapaths, including instruction pipelining, hazard detection, and resolution (e.g., stalls, forwarding)
- Control unit: 1. Hardwired implementation 2. Microprogrammed realization
- Instruction pipelining
- Introduction to instruction-level parallelism (ILP)

Learning Outcomes:

Core:

1. Compare alternative implementation of datapaths in modern computer architectures [Compare]
2. Produce a set of control signals for adding two integers using hardwired and microprogrammed implementations [Produce]
3. Discuss instruction-level parallelism using pipelining and significant hazards that may occur [Debate]
4. Design a complete processor, including datapath and control [Design]
5. Compute the average cycles per instruction for a given processor and memory system implementation [Compute]

2.3.7 AR/Performance and Energy Efficiency (KA Core: 3 hrs)

Topics:

Core

- Performance-energy evaluation (introduction): performance, power consumption, memory, and communication costs
- Branch prediction, speculative execution, out-of-order execution, Tomasulo's algorithm
- Enhancements for vector processors and GPUs

- Hardware support for multithreading: 1. Race conditions 2. Lock implementations 3. Point-to-point synchronization 4. Barrier implementation
- Scalability
- Alternative architectures including VLIW/EPIC, accelerators, and other special purpose processors
- Dynamic voltage and frequency scaling (DVFS)
- Dark Silicon

Learning Outcomes:
Core:

1. Discuss performance and energy efficiency evaluation metrics [Debate]
2. Analyze a speculative execution diagram and write about the decisions that can be made [Analyze]
3. Create a GPU performance-watt benchmarking diagram [Create]
4. Write a multithreaded program that adds (in parallel) elements of two integer vectors [Write]
5. Recommend a set of design choices for alternative computer architectures [Recommend]
6. Enumerate key concepts associated with dynamic voltage and frequency scaling [Enumerate]
7. Measure energy savings improvement for an 8-bit integer quantization compared to a 32-bit quantization [Evaluate]

2.3.8 AR/Heterogeneous Architectures (KA Core: 2 hrs)

Topics:
Core

- SIMD and MIMD architectures (e.g., General-Purpose GPUs, TPUs, and NPUs)
- Heterogeneous memory systems: 1. Shared memory versus distributed memory 2. Volatile vs non-volatile memory 3. Coherence protocols
- Domain-Specific Architectures (DSAs): 1. Machine Learning Accelerator 2. In-networking computing 3. Embedded systems for emerging applications 4. Neuromorphic computing 5. Edge computing devices
- Packaging and integration solutions such as 3DIC and chiplets
- Machine learning in architecture design: 1. AI algorithms for workload analysis 2. Optimization of architecture configurations for performance and power efficiency

Learning Outcomes:
Core:

1. Analyze a system diagram with alternative parallel architectures, e.g., SIMD and MIMD, and identify the key differences [Analyze]
2. Discuss what memory-management issues are found in multiprocessors that are not present in uniprocessors and how these issues might be resolved [Debate]
3. Indicate the differences between memory backplane, processor memory interconnect, and remote memory via networks, their implications for access latency, and their impact on program performance [Differentiate]
4. Discuss how you would determine when to use a domain-specific accelerator instead of a general-purpose CPU [Debate]
5. Enumerate key differences in architectural design principles between a vector and scalar-based processing unit [Enumerate]
6. List the advantages and disadvantages of a PIM architecture [List]

2.3.9 AR/Secure Processor Architectures (KA Core: 2 hrs)

Topics:
Core

- Principles of Secure Hardware: 1. Security Risk Analysis, Asset Protection, and Threat Model 2. Cryptographic Acceleration with Hardware 3. Support for virtualization (e.g., OS isolation)
- Roots of trust in hardware, Physically Unclonable Functions (PUF)
- Hardware Random Number Generators

- Memory protection extensions: 1. Runtime pointer bounds checking (e.g., buffer overflow) 2. Protection at the microarchitectural level 3. Protection at the ISA level
- Trusted Execution Environment (TEE): 1. Trusted Computer Base Protections 2. Protecting virtual machines 3. Protecting containers 4. Trusted software modules (Enclaves)
- Homomorphic encryption for privacy-preserving data processing

Learning Outcomes:
Core:

1. Discuss principles of secure hardware, exploring a framework for risk analysis and asset protection [Debate]
2. Summarize how Physically Unclonable Functions (PUF) can be a unique device identifier in security applications [Summarize]
3. Distinguish a random number generator with dedicated hardware support from generators without hardware dedicated to generating entropy [Distinguish]
4. List the advantages and disadvantages of memory protection at the ISA level [List]
5. Describe key design issues of a trusted execution environment (TEE) to support virtual machines [Describe]

2.3.10 AR/Quantum Architectures (KA Core: 2 hrs)

Topics:
Core

- Principles: 1. The wave-particle duality principle 2. The uncertainty principle in the double-slit experiment 3. What is a Qubit? Superposition, interference, and measurement. Photons as qubits 4. Systems of two qubits, Entanglement, Bell states, The No-Signaling theorem
- Axioms of QM: superposition principle, measurement axiom, unitary evolution
- Single qubit gates for the circuit model of quantum computation: X, Z, H
- Two qubit gates and tensor products, working with matrices
- The No-Cloning Theorem. The Quantum Teleportation protocol
- Algorithms: 1. Simple quantum algorithms: Bernstein-Vazirani, Simon's algorithm 2. Implementing Deutsch-Josza with Mach-Zehnder Interferometers 3. Quantum factoring (Shor's Algorithm) 4. Quantum search (Grover's Algorithm)
- Implementation aspects: 1. The physical implementation of qubits 2. Classical control of a Quantum Processing Unit (QPU) 3. Error mitigation and control, NISQ and beyond 4. Measurement approaches
- Emerging Applications: 1. Post-quantum encryption 2. The Quantum Internet 3. Adiabatic quantum computation (AQC) and quantum annealing

Learning Outcomes:
Core:

1. Discuss how a quantum object produced as a particle propagates like a wave and is detected as a particle with a probability distribution corresponding to the wave [Debate]
2. Discuss the quantum-level nature that is inherently probabilistic [Debate]
3. Express your view on entanglement that can be used to create non-classical correlations, but there is no way to use quantum entanglement to send messages faster than the speed of light [Evaluate]
4. Describe quantum parallelism and the role of constructive vs destructive interference in quantum algorithms given the probabilistic nature of measurement(s) [Describe]
5. Analyze a code snippet providing the role of quantum Fourier transform (QFT) in Shor's algorithm [Analyze]
6. Write a program to implement Shor's algorithm in a simulator, highlighting the classical components and aspects of Shor's algorithm [Write]
7. Enumerate the specifics of each qubit modality (e.g., trapped ion, superconducting, silicon spin, photonic, quantum dot, neutral atom, topological, color center, electron-on-helium) [Enumerate]
8. Contrast AQC with the gate model of quantum computation and the problems each is better suited to solve [Contrast]

2.3.11 AR/Sustainability Issues

Topics:
Non Core

- Environmental impacts of implementation decisions: 1. Sustainability goals, resource consumption, and economic viability 2. Carbon footprint, hardware electronic waste 3. The energy footprint of data centers at various workloads (e.g., AI model training and use) 4. Guidelines for sustainable design standards

Learning Outcomes:
NonCore:

1. Assess the environmental impacts of a given project's deployment (e.g., the energy consumption of CPUs and GPUs, contribution to e-waste, and effect of hardware virtualization in data centers) [Assess]

2.4 Data Management (DM)

Since the mid-1970s, the study of Data Management (DM) has meant an almost exclusive study of relational database systems. Depending on institutional context, students have studied, in varying proportions, the following.

- Data modeling and database design: for example, E-R Data model, relational model, normalization theory
- Query construction: e.g., relational algebra, SQL
- Query processing: e.g., indices (B+tree, hash), algorithms (e.g., external sorting, select, project, join), query optimization (transformations, index selection)
- DBMS internals: e.g., concurrency/locking, transaction management, buffer management

Today's graduates are expected to possess DBMS user (rather than implementor) skills. These primarily include data modeling and query construction; ability to take an unorganized collection of data, organize it using a DBMS, and access/update the collection via queries.

Additionally, students need to study the following.

- The role data plays in an organization. This includes the Data Life Cycle: Creation-Processing-Review/Reporting-Retention/Retrieval-Destruction.
- The social/legal aspects of data collection: e.g., scale, data privacy, database privacy (compliance) by design, de-identification, ownership, reliability, database security, and intended and unintended applications.
- Emerging and advanced technologies that are augmenting/replacing traditional relational systems, particularly those used to support (big) data analytics, including NoSQL (e.g., JSON, XML, key-value store databases), cloud databases, MapReduce, and dataframes.
- The existing and emerging roles for those involved with data management, which include the following:
  - Product feature engineers: those who use both SQL and NoSQL operational databases.
  - Analytical engineers/data engineers: those who write analytical SQL, Python, and Scala code to build data assets for business groups.
  - Business analysts: those who build/manage data most frequently with Excel spreadsheets.
  - Data infrastructure engineers: those who implement a data management system in a variety of data applications (e.g., OLTP).
  - "Everyone" who produces or consumes data must understand the associated social, ethical, and professional issues.

One role that transcends all the above categories is that of data custodian. Previously, data were seen as a resource to be managed (Information Systems Management) just like other enterprise resources. Today, data are seen in a larger context. Data about customers can now be seen as belonging to (or in some national contexts, as owned by) those customers. There is now an accepted understanding that the safe and ethical storage, and use, of institutional data is part of being a responsible data custodian.

Furthermore, we acknowledge the tension between a curricular focus on professional preparation versus the study of a knowledge area as a scientific endeavor. This is particularly true with Data Management. For example, proving (or at least knowing) the completeness of Armstrong's Axioms is

fundamental in functional dependency theory. However, most computer science graduates will never utilize this concept during their professional careers. The same can be said for many other topics in the Data Management canon. Conversely, if our graduates can only normalize data into Boyce-Codd normal form (using an automated tool) and write SQL queries, without understanding the role that indices play in efficient query execution, we have done them and society a disservice.

To this end, the number of CS Core hours is relatively small relative to the KA Core hours. This approach is designed to allow institutions with differing contexts to customize their curricula appropriately. An institution that focuses on OLTP implementation, for example, would prioritize efficient storage and data access, while an institution that focuses on product features would prioritize programmatic access to extant databases.

However, an institution manages this tension, we wish to give voice to one of the ironies of computer science curricula. Students typically spend much of their educational life reading (and writing) data from a file or interactively, while outside of the academy the predominant data comes from databases accessed programmatically. Perhaps in the not-too-distant future students will learn programmatic database access early on and then continue this practice as they progress through their curriculum.

Finally, we understand that while the Data Management KA may be orthogonal to the SEC (Security) and SEP (Society, Ethics, and the Profession) KAs, it is also ground zero for these (and other) knowledge areas. When designing persistent data stores, the question of what should be stored must be examined from both legal and ethical perspectives. Are there privacy concerns? And just as importantly, how well protected is the data?

Knowledge Area (KA)	CS Core	KA Core
⭐ 2.4.1 The Role of Data and the Data Life Cycle (Pg.. 36)	2	
⭐ 2.4.2 Core Database System Concepts (Pg.. 36)	2	1
⭐ 2.4.3 Data Modeling (Pg.. 37)	2	3
⭐ 2.4.4 Relational Databases (Pg.. 37)	1	3
⭐ 2.4.5 Query Construction (Pg.. 38)	2	4
⭐ 2.4.6 Query Processing (Pg.. 38)		4
⭐ 2.4.7 DBMS Internals (Pg.. 39)		4
⭐ 2.4.8 NoSQL Systems (Pg.. 39)		2
⭐ 2.4.9 Data Security and Privacy (Pg.. 40)	1	2
⭐ 2.4.10 Data Analytics (Pg.. 40)		3
2.4.11 Distributed Databases/Cloud Computing (Pg.. 41)	Elective	
2.4.12 Semi-structured and Unstructured Databases (Pg.. 41)	Elective	
2.4.13 Society, Ethics, and the Profession (Pg.. 41)	Elective	

Table 2.4: List of KUs in the Data Management area.

2.4.1 DM/The Role of Data and the Data Life Cycle (CS Core: 2 hrs)

Topics:
Core
- The Data Life Cycle: Creation-Processing-Review/Reporting-Retention/Retrieval-Destruction

Learning Outcomes:
Core:
1. Identify the five stages of the Data Life Cycle [Analyze]

2.4.2 DM/Core Database System Concepts (CS Core: 2 hrs, KA Core: 1 hr)

Topics:
Core
- Purpose and advantages of database systems
- Components of database systems
- Design of core DBMS functions (e.g., query mechanisms, transaction management, buffer management, access methods)
- Database architecture, data independence, and data abstraction

- Transaction management
- Normalization
- Approaches for managing large volumes of data (e.g., NoSQL database systems, use of MapReduce)
- How to support CRUD-only applications
- Distributed databases/cloud-based systems
- Structured, semi-structured, and unstructured data
- Use of a declarative query language
- Systems supporting structured and/or stream content

Learning Outcomes:

Core:

1. Identify at least four advantages that using a database system provides [Analyze]
2. Enumerate the components of a (relational) database system [Enumerate]
3. Follow a query as it is processed by the components of a (relational) database system [Analyze]
4. Defend the value of data independence [Defend]
5. Compose a simple select-project-join query in SQL [Compose]
6. Enumerate the four properties of a correct transaction manager [Enumerate]
7. Describe the advantages for eliminating duplicate repeated data [Describe]
8. Outline how MapReduce uses parallelism to process data efficiently [Outline]
9. Evaluate the differences between structured and semi/unstructured databases [Evaluate]

2.4.3 DM/Data Modeling (CS Core: 2 hrs, KA Core: 3 hrs)

Topics:

Core

- Data modeling
- Relational data model
- Conceptual models (e.g., entity-relationship, UML diagrams)
- Semi-structured data models (expressed using DTD, XML, or JSON Schema, for example)

Non Core

- Spreadsheet models
- Object-oriented models: 1. GraphQL
- New features in SQL
- Specialized Data Modeling topics: 1. Time series data (aggregation, join) 2. Graph data (link traversal) 3. Techniques for avoiding inefficient raw data access (e.g., "avg daily price"): materialized views and special data structures (e.g., Hyperloglog, bitmap) 4. Geo-Spatial data (e.g., GIS databases)

Learning Outcomes:

Core:

1. Describe the components of the relational data model [Describe]
2. Model 1:1, 1:n, and n:m relationships using the relational data model [Use]
3. Describe the components of the E-R (or some other non-relational) data model [Describe]
4. Model a given environment using a conceptual data model [Analyze]
5. Model a given environment using the document-based or key-value store-based data model [Apply]

2.4.4 DM/Relational Databases (CS Core: 1 hr, KA Core: 3 hrs)

Topics:

Core

- Entity and referential integrity: Candidate key, superkeys
- Relational database design
- Mapping conceptual schema to a relational schema
- Physical database design: file and storage structures
- Introduction to Functional dependency theory

- Normalization Theory: 1. Decomposition of a schema; lossless-join, and dependency-preservation properties of a decomposition 2. Normal forms (BCNF) 3. Denormalization (for efficiency)

Non Core

- Functional dependency theory: 1. Closure of a set of attributes 2. Canonical Cover
- Normalization theory: 1. Multi-valued dependency (4NF) 2. Join dependency (PJNF, 5NF) 3. Representation theory

Learning Outcomes:

Core:

1. Describe the defining characteristics behind the relational data model [Describe]
2. Comment on the difference between a foreign key and a superkey [Explain]
3. Enumerate the different types of integrity constraints [Enumerate]
4. Compose a relational schema from a conceptual schema which contains 1:1, 1:n, and n:m relationships [Compose]
5. Map appropriate file structure to relations and indices [Map]
6. Describe how functional dependency theory generalizes the notion of key [Describe]
7. Defend a given decomposition as lossless and or dependency preserving [Defend]
8. Detect which normal form a given decomposition yields [Detect]
9. Comment on reasons for denormalizing a relation [Interpret]

2.4.5 DM/Query Construction (CS Core: 2 hrs, KA Core: 4 hrs)

Topics:

Core

- SQL Query Formation: 1. Interactive SQL execution 2. Programmatic execution of an SQL query
- Relational Algebra
- SQL: 1. Data definition including integrity and other constraint specifications 2. Update sublanguage

Non Core

- Relational Calculus
- QBE and 4th-generation environments
- Different ways to invoke non-procedural queries in conventional languages
- Introduction to other major query languages (e.g., XPATH, SPARQL)
- Stored procedures

Learning Outcomes:

Core:

1. Compose SQL queries that incorporate select, project, join, union, intersection, set difference, and set division [Compose]
2. Determine when a nested SQL query is correlated or not [Determine]
3. Iterate over data retrieved programmatically from a database via an SQL query [Create]
4. Define, in SQL, a relation schema, including all integrity constraints and delete/update triggers [Define]
5. Compose an SQL query to update a tuple in a relation [Compose]

2.4.6 DM/Query Processing (KA Core: 4 hrs)

Topics:

Core

- Page structures
- Index structures: 1. B+ trees 2. Hash indices: static and dynamic 3. Index creation in SQL
- File structures: 1. Heap files 2. Hash files
- Algorithms for query operators: 1. External Sorting 2. Selection 3. Projection; with and without duplicate elimination 4. Natural Joins: Nested loop, Sort-merge, Hash join 5. Analysis of algorithm efficiency
- Query transformations

- Query optimization: 1. Access paths 2. Query plan construction 3. Selectivity estimation 4. Index-only plans
- Parallel Query Processing (e.g., parallel scan, parallel join, parallel aggregation)
- Database tuning/performance: 1. Index selection 2. Impact of indices on query performance 3. Denormalization

Learning Outcomes:
Core:

1. Describe the purpose and organization of both B+ tree and hash index structures [Describe]
2. Compose an SQL command to create an index (any kind) [Compose]
3. Specify the steps for the various query operator algorithms: external sorting, projection with duplicate elimination, sort-merge join, hash-join, block nested-loop join [Analyze]
4. Derive the run-time (in I/O requests) for each of the above algorithms [Analyze]
5. Transform a query in relational algebra to its equivalent appropriate for a left-deep, pipelined execution [Redesign]
6. Compute selectivity estimates for a given selection and/or join operation [Compute]
7. Describe how to modify an index structure to facilitate an index-only operation for a given relation [Describe]
8. For a given scenario decide on which indices to support for the efficient execution of a set of queries [Assess]
9. Describe how DBMSs leverage parallelism to speed up query processing by dividing the work across multiple processors or nodes [Describe]

2.4.7 DM/DBMS Internals (KA Core: 4 hrs)

Topics:
Core

- DB Buffer Management
- Transaction Management: 1. Isolation Levels 2. ACID 3. Serializability 4. Distributed Transactions
- Concurrency Control: 1. 2-Phase Locking 2. Deadlocks handling strategies 3. Quorum-based consistency models
- Recovery Manager: 1. Relation with Buffer Manager

Non Core

- Concurrency Control: 1. Optimistic concurrency control 2. Timestamp concurrency control
- Recovery Manager: 1. Write-Ahead logging 2. ARIES recovery system (Analysis, REDO, UNDO)

Learning Outcomes:
Core:

1. Describe how a DBMS manages its Buffer Pool [Describe]
2. Describe the four properties for a correct transaction manager [Describe]
3. Outline the principle of serializability [Outline]

2.4.8 DM/NoSQL Systems (KA Core: 2 hrs)

Topics:
Core

- Why NoSQL? (e.g., Impedance mismatch between Application [CRUD] and RDBMS)
- Key-Value and Document data model

Non Core

- Storage systems (e.g., Key-Value systems, Data Lakes)
- Distribution Models (Sharding and Replication)
- Graph Databases
- Consistency Models (Update and Read, Quorum consistency, CAP theorem)
- Processing model (e.g., Map-Reduce, multi-stage map-reduce, incremental map-reduce)
- Case Studies: Cloud storage system (e.g., S3); Graph databases; "When not to use NoSQL"

Learning Outcomes:
Core:

1. Develop a use case for the use of NoSQL over RDBMS [Create]
2. Describe the defining characteristics behind Key-Value and Document-based data models [Describe]

2.4.9 DM/Data Security and Privacy (CS Core: 1 hr, KA Core: 2 hrs)

Topics:
Core

- Differences between data security and data privacy
- Protecting data and database systems from attacks, including injection attacks such as SQL injection
- Personally identifying information (PII) and its protection
- Ethical considerations in ensuring the security and privacy of data
- Need for, and different approaches to securing data at rest, in transit, and during processing
- Database auditing and its role in digital forensics
- Data inferencing and preventing attacks
- Laws and regulations governing data security and data privacy

Non Core

- Typical risk factors and prevention measures for ensuring data integrity
- Ransomware and prevention of data loss and destruction

Learning Outcomes:
Core:

1. Describe the differences in the goals for data security and data privacy [Describe]
2. Identify and mitigate risks associated with different approaches to protecting data [Analyze]
3. Describe legal and ethical considerations of end-to-end data security and privacy [Describe]
4. Develop a database auditing system given risk considerations [Create]
5. Apply several data exploration approaches to understanding unfamiliar datasets [Apply]

2.4.10 DM/Data Analytics (KA Core: 3 hrs)

Topics:
Core

- Exploratory data techniques (motivation, representation, descriptive statistics, visualizations)
- Data science lifecycle: business understanding, data understanding, data preparation, modeling, evaluation, deployment, and user acceptance
- Data mining and machine learning algorithms: e.g., classification, clustering, association, regression
- Data acquisition and governance
- Data security and privacy considerations
- Data fairness and bias
- Data visualization techniques and their use in data analytics
- Entity Resolution

Learning Outcomes:
Core:

1. Describe several data exploration approaches, including visualization, to understanding unfamiliar datasets [Describe]
2. Apply several data exploration approaches to understanding unfamiliar datasets [Apply]
3. Describe basic machine learning/data mining algorithms and when they are appropriate for use [Describe]
4. Apply several machine learning/data mining algorithms [Apply]
5. Describe legal and ethical considerations in acquiring, using, and modifying datasets [Describe]
6. Describe issues of fairness and bias in data collection and usage [Describe]

2.4.11 DM/Distributed Databases/Cloud Computing

Topics:
Non Core

- Distributed DBMS: 1. Distributed data storage 2. Distributed query processing 3. Distributed transaction model 4. Homogeneous and heterogeneous solutions 5. Client-server distributed databases
- Parallel DBMS: 1. Parallel DBMS architectures: shared memory, shared disk, shared nothing 2. Speedup and scale-up, e.g., use of the MapReduce processing model 3. Data replication and weak consistency models

Learning Outcomes:
NonCore:

1. Describe the key components of a distributed DBMS, including distributed data storage, query processing, and transaction management [Describe]
2. Analyze the tradeoffs between parallel DBMS architectures: shared memory, shared disk, and shared nothing [Analyze]
3. Describe data replication strategies and weak consistency models in distributed database systems [Describe]

2.4.12 DM/Semi-structured and Unstructured Databases

Topics:
Non Core

- Vectorized unstructured data (text, video, audio, etc.) and vector storage: 1. TF-IDF Vectorizer with ngram 2. Word2Vec 3. Array database or array data type handling
- Semi-structured databases (e.g., JSON): 1. Storage: (a) Encoding and compression of nested data types 2. Indexing: (a) Btree, skip index, Bloom filter (b) Inverted index and bitmap compression (c) Space filling curve indexing for semi-structured geo-data 3. Query processing for OLTP and OLAP use cases: (a) Insert, Select, update/delete tradeoffs (b) Case studies on Postgres/JSON, MongoDB, and Snowflake/JSON

Learning Outcomes:
NonCore:

1. Describe vectorized representations of unstructured data (e.g., TF-IDF, Word2Vec) and their storage models [Describe]
2. Apply indexing techniques (e.g., B-tree, inverted index, Bloom filter) to semi-structured data stored in JSON format [Apply]
3. Analyze query processing tradeoffs for semi-structured databases in OLTP and OLAP use cases [Analyze]

2.4.13 DM/Society, Ethics, and the Profession

Topics:
Core

- Issues related to scale
- Data privacy overall: 1. Privacy compliance by design
- Data anonymity
- Data ownership/custodianship
- Intended and unintended applications of stored data
- Reliability of data
- Provenance, data lineage, and metadata management
- Data security

Learning Outcomes:
Core:

1. Enumerate three social and three legal issues related to large data collections [Enumerate]
2. Describe the value of data privacy [Describe]
3. Identify the competing stakeholders with respect to data ownership [Analyze]

4. Enumerate three negative unintended consequences from a given (well known) data-centric application (e.g., Facebook, LastPass, Ashley Madison) [Enumerate]
5. Describe the meaning of data provenance and lineage [Describe]
6. Identify how a database might contribute to data security as well as how it may introduce insecurities [Analyze]

2.5 Foundations of Programming Languages (FPL)

The foundations of programming languages are rooted in discrete mathematics, logic, and formal languages, and provide a basis for the understanding of complex modern programming languages. Although programming languages vary according to the language paradigm and the problem domain and evolve in response to both societal needs and technological advancement, they share an underlying abstract model of computation and program development. This remains true even as processor hardware and their interface with programming tools become increasingly intertwined and progressively more complex. An understanding of the common abstractions and programming paradigms enables faster learning of programming languages.

The Foundations of Programming Languages knowledge area is concerned with articulating the underlying concepts and principles of programming languages, the formal specification of a programming language and the behavior of a program, explaining how programming languages are implemented, comparing the strengths and weaknesses of various programming paradigms, and describing how programming languages interface with entities such as operating systems and hardware. The concepts covered here are applicable to several languages and an understanding of these principles assists a learner to move readily from one language to another, as well as select a programming paradigm and language that best suits the problem at hand.

Programming languages are the medium through which programmers precisely describe concepts, formulate algorithms, and reason about solutions. Over the course of a career, a computer scientist will learn and work with many different languages, separately or together. Software developers must understand different programming models, programming features and constructs, and underlying concepts to make informed design choices among languages that support multiple complementary approaches. It would be useful to know how programming language features are defined, composed, and implemented to improve execution efficiency and long-term maintenance of developed software. Also useful is a basic knowledge of language translation, program analysis, run-time behavior, memory management and interplay of concurrent processes communicating with each other through message-passing, shared memory, and synchronization. Finally, some developers and researchers will need to design new languages, an exercise which requires greater familiarity with basic principles.

Knowledge Area (KA)	CS Core	KA Core
⭐ 2.5.1 Object-Oriented Programming Fundamentals (Pg.. 43)	2	1
⭐ 2.5.2 Object-Oriented Programming: Encapsulation, Subtyping, and Reflection (Pg.. 44)	2	1
⭐ 2.5.3 Functional Programming (Pg.. 44)	4	3
⭐ 2.5.4 Logic Programming (Pg.. 45)		2
⭐ 2.5.5 Shell Scripting (Pg.. 46)	2	
⭐ 2.5.6 Event-Driven and Reactive Programming (Pg.. 46)	2	2
⭐ 2.5.7 Parallel and Distributed Computing (Pg.. 46)	2	2
2.5.8 Aspect-Oriented Programming (Pg.. 47)	Elective	
⭐ 2.5.9 Type Systems Fundamentals (Pg.. 47)	2	2
⭐ 2.5.10 Type Systems: Polymorphism and Complementary Typing (Pg.. 48)	1	1
⭐ 2.5.11 Advanced Type Theory (Pg.. 48)	2	2
⭐ 2.5.12 Systems Execution and Memory Model (Pg.. 49)	2	
⭐ 2.5.13 Memory Management (Pg.. 49)	1	
⭐ 2.5.14 Language Translation and Execution (Pg.. 50)	2	3
⭐ 2.5.15 Program Abstraction and Representation (Pg.. 50)		3
2.5.16 Syntax Analysis (Pg.. 51)	Elective	
2.5.17 Compiler Semantic Analysis (Pg.. 51)	Elective	
2.5.18 Program Analysis and Analyzers (Pg.. 52)	Elective	
2.5.19 Code Generation (Pg.. 52)	Elective	
2.5.20 Run-time Behavior and Systems (Pg.. 53)	Elective	
2.5.21 Advanced Programming Constructs (Pg.. 53)	Elective	
2.5.22 Language Pragmatics (Pg.. 54)	Elective	
2.5.23 Formal Semantics (Pg.. 54)	Elective	
2.5.24 Formal Development Methodologies (Pg.. 55)	Elective	
2.5.25 Design Principles of Programming Languages (Pg.. 55)	Elective	
2.5.26 Society, Ethics, and the Profession (Pg.. 56)	Elective	

Table 2.5: List of KUs in the Foundations of Programming Languages area.

2.5.1 FPL/Object-Oriented Programming Fundamentals (CS Core: 2 hrs, KA Core: 1 hr)

Topics:
Core

- Imperative programming as a subset of object-oriented programming.
- Object-oriented design: 1. Decomposition into objects carrying state and having behavior. 2. Class-hierarchy design for modeling.
- Definition of classes: fields, methods, and constructors.
- Subclasses, inheritance (including multiple inheritance), and method overriding.
- Dynamic dispatch: definition of method-call.
- Exception handling.

Learning Outcomes:
Core:

1. Enumerate the differences between imperative and object-oriented programming paradigms [Enumerate]
2. Compose a class through design, implementation, and testing to meet behavioral requirements [Compose]
3. Build a simple class hierarchy utilizing subclassing that allows code to be reused for distinct subclasses [Create]
4. Predict and validate control flow in a program using dynamic dispatch [Evaluate]
5. Compare and contrast how computational solutions to a problem differ in procedural, functional, and object-oriented approaches [Compare]

6. Compare and contrast mechanisms to define and protect data elements within procedural, functional, and object-oriented approaches [Compare]

2.5.2 FPL/Object-Oriented Programming: Encapsulation, Subtyping, and Reflection (CS Core: 2 hrs, KA Core: 1 hr)

Topics:
Core

- Object-oriented idioms for encapsulation: 1. Privacy, data hiding, and visibility of class members. 2. Interfaces revealing only method signatures. 3. Abstract base classes, traits and mixins.
- Dynamic vs static properties.
- Composition vs inheritance.
- Subtyping: 1. Subtype polymorphism; implicit upcasts in typed languages. 2. Notion of behavioral replacement: subtypes acting like supertype. 3. Relationship between subtyping and inheritance.
- Collection classes, iterators, and other common library components.
- Metaprogramming and reflection.

Learning Outcomes:
Core:

1. Compare and contrast the benefits and costs/impact of using inheritance (subclasses) and composition (specifically, how to base composition on higher order functions) [Compare]
2. Explain the relationship between object-oriented inheritance (code-sharing and overriding) and subtyping (the idea of a subtype being usable in a context that expects the supertype) [Explain]
3. Use object-oriented encapsulation mechanisms such as interfaces and private members [Use]
4. Define and use iterators and other operations on aggregates, including operations that take functions as arguments, in multiple programming languages, selecting the most natural idioms for each language [Define]
5. Use collection classes and iterators effectively to solve a problem [Use]
6. Compare and contrast how computational solutions to a problem differ in procedural, functional, and object-oriented approaches [Compare]
7. Compare and contrast mechanisms to define and protect data elements within procedural, functional, and object-oriented approaches [Compare]

2.5.3 FPL/Functional Programming (CS Core: 4 hrs, KA Core: 3 hrs)

Topics:
Core

- Lambda expressions and evaluation: 1. Variable binding and scope rules. 2. Parameter-passing. 3. Nested lambda expressions and reduction order.
- Effect-free programming: 1. Function calls have no side effects, facilitating compositional reasoning. 2. Immutable variables and data copying vs reduction. 3. Use of recursion vs loops vs pipelining (map/reduce).
- Processing structured data (e.g., trees) via functions with cases for each data variant: 1. Functions defined over compound data in terms of functions applied to the constituent pieces. 2. Persistent data structures.
- Using higher-order functions (taking, returning, and storing functions).
- Metaprogramming and reflection.
- Function closures (functions using variables in the enclosing lexical environment): 1. Basic meaning and definition - creating closures at run-time by capturing the environment. 2. Canonical idioms: call-backs, arguments to iterators, reusable code via function arguments. 3. Using a closure to encapsulate data in its environment. 4. Delayed versus eager evaluation.

Non Core

- Graph reduction machine and call-by-need.
- Implementing delayed evaluation.
- Integration with logic programming paradigm using concepts such as equational logic, narrowing, residuation and semantic unification.

- Integration with other programming paradigms such as imperative and object-oriented.

Learning Outcomes:

Core:

1. Develop basic algorithms that avoid assigning to mutable states or considering reference equality [Create]
2. Develop useful functions that take and return other functions [Create]
3. Explain a simple example of lambda expression being implemented using a virtual machine, such as a SECD machine, showing storage and reclaim of the environment [Explain]
4. Correctly interpret variables and lexical scope in a program using function closures [Interpret]
5. Use functional encapsulation mechanisms such as closures and modular interfaces [Use]
6. Compare and contrast stateful vs stateless execution [Compare]
7. Define and use iterators and other operations on aggregates, including operations that take functions as arguments, in multiple programming languages, selecting the most natural idioms for each language [Define]

NonCore:

8. Illustrate graph reduction using a λ-expression using a shared subexpression [Apply]
9. Illustrate the execution of a simple nested λ-expression using an abstract machine, such as an ABC machine [Apply]
10. Illustrate narrowing, residuation, and semantic unification using simple illustrative examples [Apply]
11. Illustrate the concurrency constructs using simple programming examples of known concepts such as a buffer being read and written concurrently or sequentially [Apply]

2.5.4 FPL/Logic Programming (KA Core: 2 hrs)

Topics:

Core

- Universal vs existential quantifiers.
- First order predicate logic vs higher order logic.
- Expressing complex relations using logical connectives and simpler relations.
- Definitions of Horn clause, facts, goals and subgoals.
- Unification and unification algorithm; unification vs assertion vs expression evaluation.
- Mixing relations with functions.
- Cuts, backtracking, and non-determinism.
- Closed-world vs open-world assumptions.

Non Core

- Memory overhead of variable copying in handling iterative programs.
- Programming constructs to store partial computation and pruning search trees.
- Mixing functional programming and logic programming using concepts such as equational logic, narrowing, residuation, and semantic unification.
- Higher-order, constraint, and inductive logic programming.
- Integration with other programming paradigms such as object-oriented programming.
- Advance programming constructs such as difference-lists, creating user defined data structures, set of, etc.

Learning Outcomes:

Core:

1. Use a logic language to implement a conventional algorithm [Use]
2. Use a logic language to implement an algorithm employing implicit search using clauses, relations, and cuts [Use]
3. Use a simple illustrative example to show correspondence between First Order Predicate Logic (FOPL) and logic programs using Horn clauses [Use]
4. Use examples to illustrate the unification algorithm and its role of parameter-passing in query reduction [Use]
5. Use simple logic programs interleaving relations, functions, and recursive programming such as factorial and Fibonacci numbers and simple complex relationships between entities and illustrate execution and parameter-passing using unification and backtracking [Use]

NonCore:

6. Illustrate computation of simple programs such as Fibonacci and show overhead of recomputation, and then show how to improve execution overhead [Apply]

2.5.5 FPL/Shell Scripting (CS Core: 2 hrs)

Topics:

Core

- Error/exception handling
- Piping
- System commands: 1. Interface with operating systems
- Environment variables
- File abstraction and operators
- Data structures, such as arrays and lists.
- Regular expressions
- Programs and processes
- Workflow

Learning Outcomes:

Core:

1. Create and execute automated scripts to manage various system tasks [Create]
2. Solve various text processing problems through scripting [Solve]

2.5.6 FPL/Event-Driven and Reactive Programming (CS Core: 2 hrs, KA Core: 2 hrs)

Topics:

Core

- Procedural programming vs reactive programming: advantages of reactive programming in capturing events.
- Components of reactive programming: event-source, event signals, listeners and dispatchers, event objects, adapters, event-handlers.
- Stateless and state-transition models of event-based programming.
- Canonical uses such as GUIs, mobile devices, robots, servers.
- Using a reactive framework: 1. Defining event handlers/listeners 2. Parameterization of event senders and event arguments 3. Externally generated events and program-generated events
- Separation of model, view, and controller
- Event-driven and reactive programs as state-transition systems

Learning Outcomes:

Core:

1. Implement event handlers for use in reactive systems, such as GUIs [Implement]
2. Examine why an event-driven programming style is natural in domains where programs react to external events [Examine]
3. Define and use a reactive framework [Define]
4. Describe an interactive system in terms of a model, a view, and a controller [Describe]

2.5.7 FPL/Parallel and Distributed Computing (CS Core: 2 hrs, KA Core: 2 hrs)

Topics:

Core

- Safety and liveness: 1. Race conditions 2. Dependencies/preconditions 3. Fault models 4. Termination
- Programming models: 1. Actor models 2. Procedural and reactive models 3. Synchronous/asynchronous programming models 4. Data parallelism
- Properties: 1. Order-based properties: (a) Commutativity (b) Independence 2. Consistency-based properties: (a) Atomicity (b) Consensus
- Execution control: 1. Async await 2. Promises 3. Threads

- Communication and coordination: 1. Mutexes 2. Message-passing 3. Shared memory 4. Cobegin-coend 5. Monitors 6. Channels 7. Threads 8. Guards
- Futures
- Language support for data parallelism such as forall, loop unrolling, map/reduce
- Effect of memory-consistency models on language semantics and correct code generation
- Representational State Transfer Application Programming Interfaces (REST APIs)
- Technologies and approaches: cloud computing, high performance computing, quantum computing, ubiquitous computing
- Overheads of message-passing
- Granularity of program for efficient exploitation of concurrency
- Concurrency and other programming paradigms (e.g., functional)

Learning Outcomes:
Core:

1. Explain why programming languages do not guarantee sequential consistency in the presence of data races and what programmers must do as a result [Explain]
2. Implement correct concurrent programs using multiple programming models, such as shared memory, actors, futures, synchronization constructs, and data-parallelism primitives [Implement]
3. Use a message-passing model to analyze a communication protocol [Use]
4. Use synchronization constructions such as monitor/synchronized methods in a simple program [Use]
5. Modeling data dependency using simple programming constructs involving variables, read and write [Implement]
6. Modeling control dependency using simple constructs such as selection and iteration [Implement]
7. Explain how REST API's integrate applications and automate processes [Explain]
8. Explain benefits, constraints and challenges related to distributed and parallel computing [Explain]

2.5.8 FPL/Aspect-Oriented Programming

Topics:
Non Core

- Aspects
- Join points
- Advice: 1. Before 2. After (as finally, returning or throwing) 3. Around
- Point cuts: 1. Designators
- Weaving - static and dynamic
- Alternatives including annotations and IDEs

Learning Outcomes:
NonCore:

1. Describe the core concepts of aspect-oriented programming, including aspects, join points, advice, and point cuts [Describe]
2. Apply static and dynamic weaving strategies to modularize cross-cutting concerns in a program [Apply]
3. Analyze alternatives to aspect-oriented programming, such as annotations and IDE-based approaches, for handling cross-cutting concerns [Analyze]

2.5.9 FPL/Type Systems Fundamentals (CS Core: 2 hrs, KA Core: 2 hrs)

Topics:
Core

- A type as a set of values together with a set of operations: 1. Primitive types (e.g., numbers, Booleans) 2. Compound types built from other types (e.g., records/structs, unions, arrays, lists, functions, references using set operations)
- Association of types to variables, arguments, results, and fields
- Type safety as an aspect of program correctness

- Type safety and errors caused by using values inconsistently given their intended types
- Goals and limitations of static and dynamic typing: detecting and eliminating errors as early as possible.
- Type equivalence: structural vs name equivalence

Learning Outcomes:

Core:

1. Describe, for both a primitive and a compound type, the values that have that type [Describe]
2. Describe, for a language with a static type system, the operations that are forbidden statically, such as passing the wrong type of value to a function or method [Describe]
3. Describe examples of program errors detected by a type system [Describe]
4. Identify program properties, for multiple programming languages, that are checked statically and program properties that are checked dynamically [Analyze]
5. Describe an example program that does not type-check in a particular language and yet would have no error if run [Describe]
6. Use types and type-error messages to write and debug programs [Use]
7. Explain how typing rules define the set of operations that are legal for a type [Explain]
8. List the type rules governing the use of a particular compound type [List]
9. Explain why undecidability requires type systems to conservatively approximate program behavior [Explain]
10. Explain multiple benefits and limitations of static typing in writing, maintaining, and debugging software [Explain]

2.5.10 FPL/Type Systems: Polymorphism and Complementary Typing (CS Core: 1 hr, KA Core: 1 hr)

Topics:

Core

- Generic types (parametric polymorphism): 1. Definition and advantages of polymorphism: parametric, subtyping, overloading, and coercion 2. Comparison of monomorphic and polymorphic types 3. Comparison with ad-hoc polymorphism (overloading) and subtype polymorphism 4. Generic parameters and typing 5. Use of generic libraries such as collections 6. Comparison with ad hoc polymorphism (overloading) and subtype polymorphism 7. Prescriptive vs descriptive polymorphism 8. Implementation models of polymorphic types 9. Subtyping
- Complementary benefits of static and dynamic typing: 1. Errors early vs errors late/avoided 2. Enforce invariants during code development and code maintenance vs postpone typing decisions while prototyping and conveniently allow flexible coding patterns such as heterogeneous collections. 3. Typing rules for function, product, and sum types 4. Avoiding misuse of code vs allowing more code reuse 5. Detect incomplete programs vs allow incomplete programs to run 6. Relationship to static analysis 7. Decidability

Learning Outcomes:

Core:

1. Define and use program pieces (such as functions, classes, methods) that use generic types, including for collections [Define]
2. Discuss the differences among generics, subtyping, and overloading [Debate]

2.5.11 FPL/Advanced Type Theory (CS Core: 2 hrs, KA Core: 2 hrs)

Topics:

Non Core

- Compositional type constructors, such as product types (for aggregates), sum types (for unions), function types, quantified types, and recursive types
- Type checking
- Subtyping: 1. Subtype polymorphism; implicit upcasts in typed languages 2. Notion of behavioral replacement: subtypes acting like supertype 3. Relationship between subtyping and inheritance
- Type safety as preservation plus progress

- Type inference
- Static overloading
- Propositions as types (implication as a function, conjunction as a product, disjunction as a sum)
- Dependent types (universal quantification as dependent function, existential quantification as dependent product).

Learning Outcomes:
NonCore:

1. Define a type system precisely and compositionally [Define]
2. For various foundational type constructors, identify the values they describe and the invariants they enforce [Analyze]
3. Precisely describe the invariants preserved by a sound type system [Describe]
4. Prove type safety for a simple language in terms of preservation and progress theorems [Prove]
5. Implement a unification-based type-inference algorithm for a simple language [Implement]
6. Explain how static overloading and associated resolution algorithms influence the dynamic behavior of programs [Explain]

2.5.12 FPL/Systems Execution and Memory Model (CS Core: 2 hrs)

Topics:
Core

- Data structures for translation, execution, translation, and code mobility such as stack, heap, aliasing (sharing using pointers), indexed sequence and string
- Direct, indirect, and indexed access to memory location
- Run-time representation of data abstractions such as variables, arrays, vectors, records, pointer-based data elements such as linked-lists and trees, and objects
- Abstract low-level machine with simple instruction, stack, and heap to explain translation and execution
- Run-time layout of memory: activation record (with various pointers), static data, call-stack, heap: 1. Translating selection and iterative constructs to control-flow diagrams 2. Translating control-flow diagrams to low level abstract code 3. Implementing loops, recursion, and tail calls 4. Translating function/procedure calls and return from calls, including different parameter-passing mechanisms using an abstract machine
- Green computing.

Learning Outcomes:
Core:

1. Explain how a core language construct, such as data abstractions and control abstractions, is executed [Explain]

2.5.13 FPL/Memory Management (CS Core: 1 hr)

Topics:
Core

- Low level allocation and accessing of high-level data structures such as basic data types, n-dimensional array, vector, record, and objects
- Return from procedure as automatic deallocation mechanism for local data elements in the stack
- Manual memory management: allocating, de-allocating, and reusing heap memory
- Automated memory management: garbage collection as an automated technique using the notion of reachability

Learning Outcomes:
Core:

1. Explain how programming language implementations typically organize memory into global data, text, heap, and stack sections and how features such as recursion and memory management map to this memory model [Explain]

2. Explain why memory leaks and dangling pointer problems occur, and what can be done by a programmer to avoid/fix them [Explain]

2.5.14 FPL/Language Translation and Execution (CS Core: 2 hrs, KA Core: 3 hrs)

Topics:
Core

- Execution models for JIT (Just-In-Time), compiler, interpreter
- Use of intermediate code, e.g., bytecode
- Limitations and benefits of JIT, compiler, and interpreter
- Cross compilers/transpilers
- BNF and extended BNF representation of context-free grammar
- Parse tree using a simple sentence such as arithmetic expression or if-then-else statement
- Execution as native code or within a virtual machine
- Language translation pipeline: syntax analysis, parsing, optional type-checking, translation/code generation and optimization, linking, loading, execution
- Run-time representation of core language constructs such as objects (method tables) and functions that can be passed as parameters to and returned from functions (closures)
- Secure compiler development

Learning Outcomes:
Core:

1. Explain and understand the differences between compiled, JIT, and interpreted language implementations, including the benefits and limitations of each [Explain]
2. Differentiate syntax and parsing from semantics and evaluation [Differentiate]
3. Use BNF and extended BNF to specify the syntax of simple constructs such as if-then-else, type declaration and iterative constructs for known languages such as C++ or Python [Use]
4. Illustrate the parse tree using a simple sentence/arithmetic expression [Apply]
5. Illustrate translation of syntax diagrams to BNF/extended BNF for simple constructs such as if-then-else, type declaration, iterative constructs, etc [Apply]
6. Illustrate ambiguity in parsing using nested if-then-else/arithmetic expression and show resolution using precedence order [Apply]
7. Discuss the benefits and limitations of garbage collection, including the notion of reachability [Debate]

2.5.15 FPL/Program Abstraction and Representation (KA Core: 3 hrs)

Topics:
Core

- BNF and regular expressions
- Programs that take (other) programs as input such as interpreters, compilers, type-checkers, documentation generators
- Components of a language: 1. Definitions of alphabets, delimiters, sentences, syntax, and semantics 2. Syntax vs semantics
- Program as a set of non-ambiguous meaningful sentences
- Basic programming abstractions: constants, variables, declarations (including nested declarations), command, expression, assignment, selection, definite and indefinite iteration, iterators, function, procedure, modules, exception handling
- Mutable vs immutable variables: advantages and disadvantages of reusing existing memory location vs advantages of copying and keeping old values; storing partial computation vs recomputation
- Types of variables: static, local, nonlocal, global; need and issues with nonlocal and global variables.
- Scope rules: static vs dynamic; visibility of variables; side-effects.
- Side-effects induced by nonlocal variables, global variables and aliased variables.

Non Core

- L-values and R-values: mapping mutable variable-name to L-values; mapping immutable variable-names to R-values
- Environment vs store and their properties
- Data and control abstraction
- Mechanisms for information exchange between program units such as procedures, functions, and modules: nonlocal variables, global variables, parameter-passing, import-export between modules
- Data structures to represent code for execution, translation, or transmission.
- Low level instruction representation such as virtual machine instructions, assembly language, and binary representation
- Lambda calculus, variable binding, and variable renaming.
- Types of semantics: operational, axiomatic, denotational, behavioral; define and use abstract syntax trees; contrast with concrete syntax.

Learning Outcomes:
Core:

1. Illustrate the scope of variables and visibility using simple programs [Apply]
2. Illustrate different types of parameter-passing using simple pseudo programming language [Apply]
3. Explain side-effect using global and nonlocal variables and how to fix such programs [Explain]
4. Explain how programs that process other programs treat the other programs as their input data [Explain]
5. Describe a grammar and an abstract syntax tree for a small language [Describe]
6. Describe the benefits of having program representations other than strings of source code [Describe]
7. Implement a program to process some representation of code for some purpose, such as an interpreter, an expression optimizer, or a documentation generator [Implement]

2.5.16 FPL/Syntax Analysis

Topics:
Non Core

- Regular grammars vs context-free grammars
- Scanning and parsing based on language specifications
- Lexical analysis using regular expressions
- Tokens and their use
- Parsing strategies including top-down (e.g., recursive descent, or LL) and bottom-up (e.g., LR or GLR) techniques: 1. Lookahead tables and their application to parsing
- Language theory: 1. Chomsky hierarchy 2. Left-most/right-most derivation and ambiguity 3. Grammar transformation
- Parser error recovery mechanisms
- Generating scanners and parsers from declarative specifications

Learning Outcomes:
NonCore:

1. Use formal grammars to specify the syntax of languages [Use]
2. Illustrate the role of lookahead tables in parsing [Apply]
3. Use declarative tools to generate parsers and scanners [Use]
4. Recognize key issues in syntax definitions: ambiguity, associativity, precedence [Recognize]

2.5.17 FPL/Compiler Semantic Analysis

Topics:
Non Core

- Abstract syntax trees; contrast with concrete syntax
- Defining, traversing, and modifying high-level program representations
- Scope and binding resolution

- Static semantics: 1. Type checking. 2. Define before use 3. Annotation and extended static checking frameworks.
- L-values/R-values
- Call semantics
- Types of parameter-passing with simple illustrations and comparison: call by value, call by reference, call by value-result, call by name, call by need and their variations
- Declarative specifications such as attribute grammars and their applications in handling limited context-base grammar

Learning Outcomes:
NonCore:

1. Draw the abstract syntax tree for a small language [Diagram]
2. Implement context-sensitive, source-level static analyses such as type-checkers or resolving identifiers to identify their binding occurrences [Implement]
3. Describe semantic analyses using an attribute grammar [Describe]

2.5.18 FPL/Program Analysis and Analyzers

Topics:
Non Core

- Relevant program representations, such as basic blocks, control-flow graphs, def-use chains, and static single assignment
- Undecidability and consequences for program analysis
- Flow-insensitive analysis, such as type-checking and scalable pointer and alias analysis
- Flow-sensitive analysis, such as forward and backward dataflow analyses
- Path-sensitive analysis, such as software model checking and software verification
- Tools and frameworks for implementing analyzers
- Role of static analysis in program optimization and data dependency analysis during exploitation of concurrency
- Role of program analysis in (partial) verification and bug-finding
- Parallelization: 1. Analysis for auto-parallelization 2. Analysis for detecting concurrency bugs

Learning Outcomes:
NonCore:

1. Explain the difference between dataflow graph and control flow graph [Explain]
2. Explain why non-trivial sound program analyses must be approximate [Explain]
3. Argue why an analysis is correct (sound and terminating) [Argue]
4. Explain why potential aliasing limits sound program analysis and how alias analysis can help [Explain]
5. Use the results of a program analysis for program optimization and/or partial program correctness [Use]

2.5.19 FPL/Code Generation

Topics:
Non Core

- Instruction sets
- Control flow
- Memory management
- Procedure calls and method dispatching
- Separate compilation; linking
- Instruction selection
- Instruction scheduling (e.g., pipelining)
- Register allocation
- Code optimization as a form of program analysis
- Program generation through generative AI

Learning Outcomes:
NonCore:

1. Identify all essential steps for automatically converting source code into assembly or other low-level languages [Analyze]
2. Explain the low-level code necessary for calling functions/methods in modern languages [Explain]
3. Discuss why separate compilation requires uniform calling conventions [Debate]
4. Discuss why separate compilation limits optimization because of unknown effects of calls [Debate]
5. Discuss opportunities for optimization introduced by naive translation and approaches for achieving, optimization, such as instruction selection, instruction scheduling, register allocation, and peephole optimization [Debate]

2.5.20 FPL/Run-time Behavior and Systems

Topics:
Non Core

- Process models using stacks and heaps to allocate and deallocate activation records and recovering environments using frame pointers and return addresses during a procedure call including parameter-passing examples
- Schematics of code lookup using hash tables for methods in implementations of object-oriented programs
- Data layout for objects and activation records
- Object allocation in heap
- Implementing virtual entities and virtual methods; virtual method tables and their application
- Run-time behavior of object-oriented programs
- Compare and contrast allocation of memory during information exchange using parameter-passing and non-local variables (using chain of static links).
- Dynamic memory management approaches and techniques: malloc/free, garbage collection (mark-sweep, copying, reference counting), regions (also known as arenas or zones)
- Just-in-time compilation and dynamic recompilation
- Interface to operating system (e.g., for program initialization)
- Interoperability between programming languages including parameter-passing mechanisms and data representation: 1. Big endian, little endian 2. Data layout of composite data types such as arrays
- Other common features of virtual machines, such as class loading, threads, and security checking
- Sandboxing

Learning Outcomes:
NonCore:

1. Discuss benefits and limitations of automatic memory management [Debate]
2. Explain the use of metadata in run-time representations of objects and activation records, such as class pointers, array lengths, return addresses, and frame pointers [Explain]
3. Compare and contrast static allocation vs stack-based allocation vs heap-based allocation of data elements [Compare]
4. Explain why some data elements cannot be automatically deallocated at the end of a procedure/method call (need for garbage collection) [Explain]
5. Discuss advantages, disadvantages, and difficulties of just-in-time and dynamic recompilation [Debate]
6. Discuss the use of sandboxing in mobile code [Debate]
7. Identify the services provided by modern language run-time systems [Analyze]

2.5.21 FPL/Advanced Programming Constructs

Topics:
Non Core

- Encapsulation mechanisms
- Delayed evaluation and infinite streams

- Compare and contrast delayed evaluation vs eager evaluation
- Unification vs assertion vs expression evaluation
- Control abstractions: exception handling, continuations, monads.
- Object-oriented abstractions: multiple inheritance, mixins, traits, multimethods
- Metaprogramming: macros, generative programming, model-based development
- String manipulation via pattern-matching (regular expressions)
- Dynamic code evaluation ("eval")
- Language support for checking assertions, invariants, and pre/post-conditions
- Domain specific languages, such as database languages, data science languages, embedded computing languages, synchronous languages, hardware interface languages
- Massive parallel high performance computing models and languages

Learning Outcomes:
NonCore:

1. Use various advanced programming constructs and idioms correctly [Use]
2. Discuss how various advanced programming constructs aim to improve program structure, software quality, and programmer productivity [Debate]
3. Discuss how various advanced programming constructs interact with the definition and implementation of other language features [Debate]

2.5.22 FPL/Language Pragmatics

Topics:
Non Core

- Effect of technology needs and software requirements on programming language development and evolution
- Problem domains and programming paradigm
- Criteria for good programming language design: 1. Principles of language design such as orthogonality 2. Defining control and iteration constructs 3. Modularization of large software
- Evaluation order, precedence, and associativity
- Eager vs delayed evaluation
- Defining control and iteration constructs
- External calls and system libraries

Learning Outcomes:
NonCore:

1. Discuss the role of concepts such as orthogonality and well-chosen defaults in language design [Debate]
2. Objectively evaluate and justify language-design decisions [Evaluate]
3. Implement an example program whose result can differ under different rules for evaluation order, precedence, or associativity [Implement]
4. Illustrate uses of delayed evaluation, such as user-defined control abstractions [Apply]
5. Discuss the need for allowing calls to external calls and system libraries and the consequences for language implementation [Debate]

2.5.23 FPL/Formal Semantics

Topics:
Non Core

- Syntax vs semantics
- Approaches to semantics: axiomatic, operational, denotational, type-based
- Axiomatic semantics of abstract constructs such as assignment, selection, iteration using precondition, post-conditions, and loop invariant
- Operational semantics analysis of abstract constructs and sequence of such as assignment, expression evaluation, selection, iteration using environment and store: 1. Symbolic execution 2. Constraint checkers
- Denotational semantics: 1. Lambda Calculus.
- Proofs by induction over language semantics

- Formal definitions and proofs for type systems: 1. Propositions as types (implication as a function, conjunction as a product, disjunction as a sum) 2. Dependent types (universal quantification as dependent function, existential quantification as dependent product) 3. Parametricity

Learning Outcomes:
NonCore:

1. Construct formal semantics for a small language [Create]
2. Write a lambda-calculus program and show its evaluation to a normal form [Write]
3. Discuss the different approaches of operational, denotational, and axiomatic semantics [Debate]
4. Use induction to prove properties of all programs in a language [Use]
5. Use induction to prove properties of all programs in a language that is well-typed according to a formally defined type system [Use]
6. Use parametricity to establish the behavior of code given only its type [Use]

2.5.24 FPL/Formal Development Methodologies

Topics:
Non Core

- Formal specification languages and methodologies
- Theorem provers, proof assistants, and logics
- Constraint checkers
- Dependent types (universal quantification as dependent function, existential quantification as dependent product)
- Specification and proof discharge for fully verified software systems using pre/post conditions, refinement types, etc.
- Formal modeling and manual refinement/implementation of software systems.
- Use of symbolic testing and fuzzing in software development.
- Model checking.
- Understanding of situations where formal methods can be effectively applied and how to structure development to maximize their value.

Learning Outcomes:
NonCore:

1. Use formal modeling techniques to develop and validate architectures [Use]
2. Use proof assisted programming languages to develop fully specified and verified software artifacts [Use]
3. Use verifier and specification support in programming languages to formally validate system properties [Use]
4. Integrate symbolic validation tooling into a programming workflow [Integrate]
5. Discuss when and how formal methods can be effectively used in the development process [Debate]

2.5.25 FPL/Design Principles of Programming Languages

Topics:
Non Core

- Language design principles: 1. Simplicity 2. Security 3. Fast translation 4. Efficient object code 5. Orthogonality 6. Readability 7. Completeness 8. Implementation strategies
- Designing a language to fit a specific domain or problem
- Interoperability between programming languages
- Language portability
- Formal description of a programming language
- Green computing principles

Learning Outcomes:
NonCore:

1. Understand what constitutes good language design and apply that knowledge to evaluate a real programming language [Evaluate]

2.5.26 FPL/Society, Ethics, and the Profession

Topics:
Non Core

- Impact of English-centric programming languages
- Enhancing accessibility and inclusivity for people with disabilities - Supporting assistive technologies
- Human factors related to programming languages and usability: 1. Impact of syntax on accessibility 2. Supporting cultural differences (e.g., currency, decimals, dates) 3. Neurodiversity
- Etymology of terms such as ”class,” ”master,” and ”slave” in programming languages
- Increasing accessibility by supporting multiple languages within applications (UTF)

Learning Outcomes:
NonCore:

1. Consciously design programming languages to be inclusive and non-offensive [Design]

2.6 Graphics and Interactive Techniques (GIT)

Graphics and Interactive Techniques (GIT) encompass the principles, algorithms, and technologies involved in rendering, modeling, animating, and interacting with visual and interactive content. This knowledge area integrates concepts from computer graphics, human-computer interaction, visualization, and image processing to create immersive and effective digital experiences. With the proliferation of interactive applications in entertainment, education, scientific visualization, and user interfaces, GIT plays a crucial role in modern computing. Students studying GIT learn to design, implement, and evaluate systems that generate visual content, facilitate user interaction, and leverage emerging technologies such as virtual reality, augmented reality, and tangible computing. The area emphasizes both theoretical foundations and practical skills, preparing graduates to contribute to diverse fields including game development, simulation, data visualization, and interactive media.

Knowledge Area (KA)	CS Core	KA Core
⭐ 2.6.1 Fundamental Concepts of Graphics and Interactive Techniques (Pg.. 56)	4	3
⭐ 2.6.2 Visualization (Pg.. 57)		6
⭐ 2.6.3 Applied Rendering and Techniques (Pg.. 57)		15
⭐ 2.6.4 Geometric Modeling (Pg.. 57)		6
⭐ 2.6.5 Shading and Advanced Rendering (Pg.. 58)		6
⭐ 2.6.6 Computer Animation (Pg.. 58)		6
⭐ 2.6.7 Simulation (Pg.. 58)		6
⭐ 2.6.8 Immersion (Pg.. 59)		6
⭐ 2.6.9 Interaction (Pg.. 59)		4
⭐ 2.6.10 Image Processing (Pg.. 59)		6
⭐ 2.6.11 Tangible/Physical Computing (Pg.. 59)		6
2.6.12 Society, Ethics, and the Profession (Pg.. 60)	Elective	

Table 2.6: List of KUs in the Graphics and Interactive Techniques area.

2.6.1 GIT/Fundamental Concepts of Graphics and Interactive Techniques (CS Core: 4 hrs, KA Core: 3 hrs)

Topics:
Core

- Overview of computer graphics pipeline and interactive systems
- Basic rendering concepts: rasterization, ray tracing, shading models
- Color theory and color spaces
- Coordinate systems and transformations
- Basic interaction techniques: input devices, event handling

- Advanced rendering concepts: global illumination, physically based rendering
- Graphics hardware and acceleration techniques
- Human perception and visual cognition

Learning Outcomes:

Core:

1. Explain the basic stages of the computer graphics pipeline [Explain]
2. Describe basic rendering techniques such as rasterization and ray tracing [Describe]
3. Apply color theory to design visually effective graphics [Apply]
4. Implement coordinate transformations for 2D and 3D graphics [Implement]
5. Design basic interactive applications using common input devices and event handling [Design]
6. Compare advanced rendering techniques such as global illumination and physically based rendering [Compare]
7. Analyze the role of graphics hardware in accelerating rendering tasks [Analyze]
8. Explain how human perception influences graphics and interaction design [Explain]

2.6.2 GIT/Visualization (KA Core: 6 hrs)

Topics:

Core

- Principles of data visualization: perception, cognition, and effective encoding
- Types of visualizations: scientific, information, geographical
- Visualization techniques for multivariate, temporal, and hierarchical data
- Interactive visualization and exploration
- Evaluation of visualization effectiveness

Learning Outcomes:

Core:

1. Apply principles of perception and cognition to design effective visualizations [Apply]
2. Select appropriate visualization techniques for different data types and tasks [Evaluate]
3. Implement interactive visualizations for data exploration [Implement]
4. Evaluate the effectiveness of a visualization using empirical methods [Evaluate]

2.6.3 GIT/Applied Rendering and Techniques (KA Core: 15 hrs)

Topics:

Core

- Rasterization algorithms: scan conversion, antialiasing
- Ray tracing: acceleration structures, sampling, shading
- Texture mapping and filtering
- Shading models: Lambertian, Phong, BRDF
- Advanced techniques: shadow mapping, environment mapping, deferred rendering

Learning Outcomes:

Core:

1. Implement rasterization algorithms for primitive rendering [Implement]
2. Implement basic ray tracing with acceleration structures [Implement]
3. Apply texture mapping and filtering techniques to enhance visual realism [Apply]
4. Compare different shading models and their applications [Compare]
5. Implement advanced rendering techniques such as shadow mapping or deferred rendering [Implement]

2.6.4 GIT/Geometric Modeling (KA Core: 6 hrs)

Topics:

Core

- Geometric representations: polygons, curves, surfaces, implicit forms
- Modeling techniques: splines, subdivision surfaces, constructive solid geometry
- Procedural modeling and generation

- Geometric algorithms: intersection, distance, decomposition

Learning Outcomes:

Core:

1. Describe different geometric representations and their trade-offs [Describe]
2. Implement modeling techniques such as spline curves or subdivision surfaces [Implement]
3. Create procedural models using algorithms or rules [Create]
4. Apply geometric algorithms to solve problems like intersection testing or mesh processing [Apply]

2.6.5 GIT/Shading and Advanced Rendering (KA Core: 6 hrs)

Topics:

Core

- Advanced shading: physically based shading, subsurface scattering
- Global illumination: radiosity, photon mapping, path tracing
- Real-time rendering techniques: level of detail, culling, occlusion
- GPU programming and shaders

Learning Outcomes:

Core:

1. Implement advanced shading models for realistic materials [Implement]
2. Compare global illumination algorithms and their trade-offs [Compare]
3. Optimize real-time rendering using techniques like level of detail and culling [Create]
4. Develop GPU shaders for custom rendering effects [Create]

2.6.6 GIT/Computer Animation (KA Core: 6 hrs)

Topics:

Core

- Animation principles: timing, spacing, anticipation, follow-through
- Keyframe animation and interpolation
- Procedural animation: particle systems, flocking, physics-based
- Character animation: skeletal animation, skinning, inverse kinematics
- Motion capture and retargeting

Learning Outcomes:

Core:

1. Apply animation principles to create believable motion [Apply]
2. Implement keyframe animation with interpolation techniques [Implement]
3. Create procedural animations such as particle systems or physics-based simulations [Create]
4. Implement character animation using skeletal systems and skinning [Implement]
5. Process motion capture data for animation retargeting [Implement]

2.6.7 GIT/Simulation (KA Core: 6 hrs)

Topics:

Core

- Physics-based simulation: rigid bodies, fluids, cloth
- Numerical methods for simulation: integration, collision detection
- Real-time simulation constraints and optimizations
- Applications in games, visual effects, and scientific visualization

Learning Outcomes:

Core:

1. Implement physics-based simulations for rigid bodies or fluids [Implement]
2. Apply numerical integration methods for dynamic simulations [Apply]
3. Optimize simulations for real-time performance [Create]
4. Develop simulations for specific applications such as games or scientific visualization [Create]

2.6.8 GIT/Immersion (KA Core: 6 hrs)

Topics:
Core
- Virtual reality (VR) and augmented reality (AR) concepts
- Immersive display technologies: head-mounted displays, CAVEs
- Tracking and input for immersive environments
- Presence and comfort in immersive systems

Learning Outcomes:
Core:
1. Explain the core concepts and differences between VR and AR [Explain]
2. Evaluate immersive display technologies and their applications [Evaluate]
3. Implement tracking and input solutions for immersive environments [Implement]
4. Design immersive experiences that maximize presence and minimize discomfort [Design]

2.6.9 GIT/Interaction (KA Core: 4 hrs)

Topics:
Core
- Interaction paradigms: direct manipulation, gesture-based, tangible
- User interface design for graphics applications
- Multimodal interaction: combining visual, auditory, haptic feedback
- Evaluation of interaction techniques: usability, user experience

Learning Outcomes:
Core:
1. Compare different interaction paradigms and their suitability for various tasks [Compare]
2. Design user interfaces for graphics-intensive applications [Design]
3. Implement multimodal interaction systems incorporating multiple feedback channels [Implement]
4. Conduct usability evaluations of interaction techniques [Evaluate]

2.6.10 GIT/Image Processing (KA Core: 6 hrs)

Topics:
Core
- Image representation and formats
- Filtering and enhancement: convolution, frequency domain
- Image segmentation and feature detection
- Morphological operations
- Applications in graphics, vision, and multimedia

Learning Outcomes:
Core:
1. Manipulate images using different representations and formats [Process]
2. Apply filtering techniques for image enhancement and analysis [Apply]
3. Implement image segmentation algorithms for feature extraction [Implement]
4. Use morphological operations for image processing tasks [Use]
5. Develop applications that integrate image processing with graphics or vision systems [Create]

2.6.11 GIT/Tangible/Physical Computing (KA Core: 6 hrs)

Topics:
Core
- Tangible interfaces and physical computing
- Sensors and actuators for interactive systems
- Embedded systems for interactive art and design
- Interactive installations and environments

Learning Outcomes:
Core:

1. Design tangible interfaces that bridge physical and digital interactions [Design]
2. Integrate sensors and actuators to create responsive systems [Integrate]
3. Develop embedded systems for interactive art or design projects [Create]
4. Create interactive installations that engage users in physical spaces [Create]

2.6.12 GIT/Society, Ethics, and the Profession

Topics:
Core

- Ethical considerations in graphics and interaction: representation, accessibility, privacy
- Social impact of immersive technologies and visualization
- Professional practice in graphics and interactive industries

Learning Outcomes:
Core:

1. Analyze ethical issues related to representation, accessibility, and privacy in graphics and interaction [Analyze]
2. Discuss the social implications of immersive technologies and data visualization [Debate]
3. Describe professional practices and standards in graphics and interactive industries [Describe]

2.7 Human-Computer Interaction (HCI)

Human-Computer Interaction (HCI) is a multidisciplinary field that focuses on the design, evaluation, and implementation of interactive computing systems for human use, and the study of major phenomena surrounding them. HCI integrates knowledge and methods from computer science, psychology, design, sociology, and other fields to create technologies that are useful, usable, and satisfying for people. The core goal of HCI is to improve interactions between users and computers by making systems more usable, accessible, and efficient while considering human capabilities and limitations.

In today's world where computing technology permeates every aspect of daily life-from smartphones and websites to medical devices and smart home systems-HCI principles are essential for creating systems that people can effectively and safely use. HCI emphasizes user-centered design processes, iterative prototyping, and empirical evaluation to ensure that systems meet real user needs. The field addresses diverse topics including usability, user experience (UX), accessibility, interaction design, user research methods, and the social implications of technology.

HCI prepares students to design, implement, and evaluate interactive systems that are not only functional but also consider human factors, ethical implications, and societal impact. Graduates with HCI expertise work in roles such as UX designers, interaction designers, usability engineers, and product managers across industries including software development, healthcare, education, entertainment, and public policy.

Knowledge Area (KA)	CS Core	KA Core
⭐ 2.7.1 Understanding the User: Individual goals and interactions with others (Pg.. 61)	2	5
⭐ 2.7.2 Accountability and Responsibility in Design (Pg.. 61)	2	2
⭐ 2.7.3 Accessibility and Inclusive Design (Pg.. 61)	2	2
⭐ 2.7.4 Evaluating the Design (Pg.. 62)	1	2
⭐ 2.7.5 System Design (Pg.. 62)	1	5
2.7.6 Society, Ethics, and the Profession (Pg.. 62)	Elective	

Table 2.7: List of KUs in the Human-Computer Interaction area.

2.7.1 HCI/Understanding the User: Individual goals and interactions with others (CS Core: 2 hrs, KA Core: 5 hrs)

Topics:
Core
- Human capabilities and limitations: perception, cognition, motor skills
- User characteristics: demographics, expertise, cultural background
- User goals, tasks, and contexts of use
- Social and organizational aspects of interaction
- Cognitive models and theories relevant to HCI
- Emotional and affective aspects of interaction
- User research methods: interviews, surveys, observations

Learning Outcomes:
Core:
1. Describe human capabilities and limitations that affect interaction design [Describe]
2. Analyze how user characteristics influence interaction with computing systems [Analyze]
3. Identify user goals and tasks in specific contexts of use [Analyze]
4. Explain social and organizational factors that impact technology adoption and use [Explain]
5. Apply cognitive models to predict and explain user behavior [Apply]
6. Consider emotional and affective factors in interaction design [Analyze]
7. Conduct user research using appropriate methods such as interviews or observations [Evaluate]

2.7.2 HCI/Accountability and Responsibility in Design (CS Core: 2 hrs, KA Core: 2 hrs)

Topics:
Core
- Ethical considerations in HCI: fairness, transparency, accountability
- Design for values and value-sensitive design
- Responsible innovation and design ethics
- Algorithmic accountability and bias in interactive systems
- Professional responsibility in HCI practice
- Legal and regulatory considerations for interactive systems

Learning Outcomes:
Core:
1. Identify ethical issues in HCI design and implementation [Analyze]
2. Apply value-sensitive design approaches to technology development [Apply]
3. Discuss responsible innovation practices in HCI [Debate]
4. Analyze algorithmic accountability and bias in interactive systems [Analyze]
5. Describe professional responsibilities in HCI practice [Describe]
6. Explain legal and regulatory considerations relevant to interactive systems [Explain]

2.7.3 HCI/Accessibility and Inclusive Design (CS Core: 2 hrs, KA Core: 2 hrs)

Topics:
Core
- Accessibility principles and guidelines
- Disabilities and assistive technologies
- Inclusive design and universal design principles
- Evaluation methods for accessibility
- Designing for diverse abilities and contexts
- Accessibility standards and compliance

Learning Outcomes:
Core:
1. Apply accessibility principles to interactive system design [Apply]
2. Describe how different disabilities affect interaction with technology [Describe]

3. Design inclusive systems using universal design principles [Design]
4. Conduct accessibility evaluations of interactive systems [Evaluate]
5. Create designs that accommodate diverse abilities and usage contexts [Create]
6. Ensure compliance with accessibility standards and regulations [Evaluate]

2.7.4 HCI/Evaluating the Design (CS Core: 1 hr, KA Core: 2 hrs)

Topics:
Core

- Usability principles and heuristics
- Evaluation methods: heuristic evaluation, cognitive walkthrough
- User testing: planning, conducting, analyzing
- Advanced evaluation methods: controlled experiments, field studies
- UX metrics and measurement
- Analytical evaluation methods: modeling and simulation

Learning Outcomes:
Core:

1. Apply usability principles and heuristics to evaluate designs [Apply]
2. Conduct heuristic evaluations and cognitive walkthroughs [Evaluate]
3. Plan, conduct, and analyze user testing sessions [Plan]
4. Design and conduct controlled experiments for HCI research [Design]
5. Measure user experience using appropriate metrics [Evaluate]
6. Use analytical methods to model and predict user performance [Use]

2.7.5 HCI/System Design (CS Core: 1 hr, KA Core: 5 hrs)

Topics:
Core

- Design process: user-centered design, iterative design
- Interaction design: concepts, principles, patterns
- Prototyping techniques: low-fidelity, high-fidelity, interactive
- Information architecture and navigation design
- Advanced interaction techniques: gesture, voice, multimodal
- Design for specific domains: mobile, web, ubiquitous computing
- Collaborative and social computing systems

Learning Outcomes:
Core:

1. Apply user-centered design processes to develop interactive systems [Apply]
2. Design interactions using established principles and patterns [Design]
3. Create prototypes at different fidelity levels for design exploration [Create]
4. Structure information and navigation for effective user interaction [Structure]
5. Implement advanced interaction techniques such as gesture or voice interfaces [Implement]
6. Design interactive systems for specific domains and platforms [Design]
7. Design collaborative systems that support group interaction [Design]

2.7.6 HCI/Society, Ethics, and the Profession

Topics:
Core

- Social impact of interactive technologies
- Ethical issues in HCI research and practice
- Professional practice in HCI
- Cultural and cross-cultural considerations in HCI
- Sustainability and HCI
- Future trends and challenges in HCI

Learning Outcomes:
Core:

1. Analyze the social impact of interactive technologies on individuals and society [Analyze]
2. Address ethical issues in HCI research and design projects [Evaluate]
3. Describe professional practices and career paths in HCI [Describe]
4. Design systems that account for cultural differences and contexts [Design]
5. Incorporate sustainability considerations into HCI design [Integrate]
6. Discuss emerging trends and future challenges in HCI [Debate]

2.8 Mathematical and Statistical Foundations (MSF)

The mathematical and statistical foundations of computer science provide the theoretical underpinnings for the discipline. They include discrete mathematics, probability, statistics, linear algebra, and calculus. These areas are essential for understanding algorithms, data structures, machine learning, graphics, cryptography, and many other topics in computer science.

Knowledge Area (KA)	CS Core	KA Core
⭐ 2.8.1 Discrete Mathematics (Pg.. 63)	15	6
⭐ 2.8.2 Advanced Discrete Mathematics (Pg.. 64)	15	6
⭐ 2.8.3 Probability (Pg.. 64)	11	29
⭐ 2.8.4 Statistics (Pg.. 64)	10	30
⭐ 2.8.5 Linear Algebra (Pg.. 65)	5	35
⭐ 2.8.6 Calculus (Pg.. 65)		40

Table 2.8: List of KUs in the Mathematical and Statistical Foundations area.

2.8.1 MSF/Discrete Mathematics (CS Core: 15 hrs, KA Core: 6 hrs)

Topics:
Core

- Sets: operations, cardinality, power sets, Cartesian products
- Relations: properties, equivalence relations, partial orders
- Functions: injective, surjective, bijective, composition, inverse
- Logic: propositional and predicate logic, quantifiers, proofs
- Graphs: definitions, paths, cycles, trees, connectivity
- Counting: permutations, combinations, binomial theorem
- Recurrence relations: solving simple recurrences

Learning Outcomes:
Core:

1. Define basic set operations and relations [Explain]
2. Apply set operations to solve problems [Apply]
3. Define properties of relations [Explain]
4. Determine if a relation is an equivalence relation [Apply]
5. Define types of functions [Explain]
6. Compose functions and find inverses [Apply]
7. Explain propositional and predicate logic [Explain]
8. Construct simple logical proofs [Apply]
9. Define basic graph concepts [Explain]
10. Solve basic graph problems [Apply]
11. Explain permutations and combinations [Explain]
12. Apply counting techniques to solve problems [Apply]
13. Define recurrence relations [Explain]
14. Solve simple recurrence relations [Apply]

2.8.2 MSF/Advanced Discrete Mathematics (CS Core: 15 hrs, KA Core: 6 hrs)

Topics:
Core

- Advanced set theory: infinite sets, Cantor's diagonal argument
- Advanced relations: closures, lattices
- Advanced functions: recursive functions, computability
- Advanced logic: proof systems, completeness, decidability
- Advanced graph theory: planarity, coloring, networks
- Advanced counting: generating functions, inclusion-exclusion
- Advanced recurrence: generating functions, asymptotic behavior

Learning Outcomes:
Core:

1. Explain concepts of infinite sets and cardinality [Explain]
2. Apply closure operations to relations [Apply]
3. Analyze advanced graph properties [Evaluate]
4. Develop generating functions for counting problems [Create]

2.8.3 MSF/Probability (CS Core: 11 hrs, KA Core: 29 hrs)

Topics:
Core

- Basic probability: sample spaces, events, axioms
- Conditional probability, independence, Bayes' theorem
- Random variables: discrete and continuous, distributions
- Expectation, variance, standard deviation
- Common distributions: uniform, binomial, normal
- Joint distributions, covariance, correlation
- Limit theorems: law of large numbers, central limit theorem
- Stochastic processes: Markov chains
- Information theory: entropy, mutual information

Learning Outcomes:
Core:

1. Define basic probability concepts [Explain]
2. Calculate probabilities of events [Apply]
3. Define conditional probability and independence [Explain]
4. Apply Bayes' theorem to update probabilities [Apply]
5. Define random variables and distributions [Explain]
6. Compute expectation and variance [Apply]
7. Describe common probability distributions [Explain]
8. Apply common distributions to model real-world phenomena [Apply]
9. Explain joint distributions and dependence [Explain]
10. Compute covariance and correlation [Apply]
11. Analyze the implications of limit theorems [Evaluate]
12. Model simple systems using Markov chains [Create]

2.8.4 MSF/Statistics (CS Core: 10 hrs, KA Core: 30 hrs)

Topics:
Core

- Descriptive statistics: measures of center and spread
- Data visualization: histograms, box plots, scatter plots
- Sampling distributions: sampling variability, central limit theorem
- Estimation: point estimates, confidence intervals
- Hypothesis testing: null and alternative hypotheses, p-values
- Regression: simple and multiple linear regression
- Analysis of variance (ANOVA)

- Nonparametric methods
- Experimental design: randomization, control, blinding

Learning Outcomes:

Core:

1. Define descriptive statistics measures [Explain]
2. Compute descriptive statistics for a dataset [Apply]
3. Explain common data visualization techniques [Explain]
4. Create appropriate visualizations for given data [Apply]
5. Define sampling distributions and variability [Explain]
6. Apply the central limit theorem to approximate distributions [Apply]
7. Define point estimates and confidence intervals [Explain]
8. Compute confidence intervals for means and proportions [Apply]
9. Define hypothesis testing concepts [Explain]
10. Conduct a simple hypothesis test and interpret p-values [Apply]
11. Explain simple and multiple linear regression [Explain]
12. Fit a regression model and interpret coefficients [Apply]
13. Analyze variance using ANOVA [Evaluate]
14. Design a simple experiment with appropriate controls [Create]

2.8.5 MSF/Linear Algebra (CS Core: 5 hrs, KA Core: 35 hrs)

Topics:

Core

- Vectors: operations, dot product, cross product
- Matrices: operations, inverses, determinants
- Systems of linear equations: Gaussian elimination, matrix form
- Vector spaces: subspaces, basis, dimension
- Linear transformations: matrix representation, kernel, image
- Eigenvalues and eigenvectors: diagonalization
- Orthogonality: orthonormal bases, Gram-Schmidt
- Singular value decomposition (SVD)
- Applications: least squares, computer graphics, machine learning

Learning Outcomes:

Core:

1. Define vector operations and properties [Explain]
2. Compute vector operations and products [Apply]
3. Define matrix operations and properties [Explain]
4. Compute matrix operations, inverses, and determinants [Apply]
5. Explain systems of linear equations and solution methods [Explain]
6. Solve systems of linear equations using elimination [Apply]
7. Define vector spaces, subspaces, basis, and dimension [Explain]
8. Determine if a set of vectors forms a basis [Apply]
9. Define linear transformations and their representations [Explain]
10. Find the kernel and image of a linear transformation [Apply]
11. Explain eigenvalues and eigenvectors [Explain]
12. Diagonalize a matrix when possible [Apply]
13. Analyze orthogonality and orthonormal bases [Evaluate]
14. Apply singular value decomposition to a matrix [Create]

2.8.6 MSF/Calculus (KA Core: 40 hrs)

Topics:

Core

- Limits and continuity
- Derivatives: definition, rules, applications
- Integrals: definite and indefinite, fundamental theorem

- Series: convergence, Taylor series
- Multivariable calculus: partial derivatives, gradients
- Optimization: local and global extrema, Lagrange multipliers
- Differential equations: basic solution techniques
- Numerical methods: integration, differentiation

Learning Outcomes:
Core:

1. Define limits and continuity [Explain]
2. Compute limits of functions [Apply]
3. Define derivatives and differentiation rules [Explain]
4. Compute derivatives of functions [Apply]
5. Define integrals and the fundamental theorem of calculus [Explain]
6. Compute definite and indefinite integrals [Apply]
7. Define series and convergence [Explain]
8. Compute Taylor series expansions [Apply]
9. Explain partial derivatives and gradients [Explain]
10. Compute partial derivatives and gradients [Apply]
11. Analyze optimization problems using calculus [Evaluate]
12. Solve simple differential equations [Create]

2.9 Networking and Communication (NC)

Networking and communication play a central role in interconnected computer systems that are transforming the daily lives of billions of people. The public internet provides connectivity for networked applications that serve ever-increasing numbers of individuals and organizations around the world. Complementing the public sector, major proprietary networks leverage their global footprints to support cost-effective distributed computing, storage, and content delivery. Advances in satellite networks expand connectivity to rural areas. Device-to-device communication underlies the emerging Internet of Things.

This knowledge area deals with key concepts in networking and communication, as well as their representative instantiations in the internet and other computer networks. Besides the basic principles of switching and layering, the area at its core provides knowledge on naming, addressing, reliability, error control, flow control, congestion control, domain hierarchy, routing, forwarding, modulation, encoding, framing, and access control. The area also covers knowledge units in network security and mobility, such as security threats, countermeasures, device-to-device communication, and multi-hop wireless networking. In addition to the fundamental principles, the area includes their specific realization of the Internet as well as hands-on skills in the implementation of networking and communication concepts. Finally, the area comprises emerging topics such as network virtualization and quantum networking.

As the main learning outcome, learners develop a thorough understanding of the role and operation of networking and communication in networked computer systems. They learn how network structure and communication protocols affect the behavior of distributed applications. The area can be used to educate not only key principles but also their specific instantiations in the internet and equip the student with hands-on implementation skills. While computer-system, networking, and communication technologies are advancing at a fast pace, the gained fundamental knowledge enables the student to readily apply the concepts in new technological settings.

Knowledge Area (KA)	CS Core	KA Core
⭐ 2.9.1 Fundamentals of Networks and Communications (Pg.. 67)	3	
⭐ 2.9.2 Networked Applications (Pg.. 67)	4	
⭐ 2.9.3 Reliability Support (Pg.. 68)		6
⭐ 2.9.4 Routing and Forwarding (Pg.. 68)		4
⭐ 2.9.5 Single Hop Communication (Pg.. 68)		3
⭐ 2.9.6 Network Security (Pg.. 68)		3
⭐ 2.9.7 Mobility Support (Pg.. 69)		4
⭐ 2.9.8 Emerging Topics (Pg.. 69)		4

Table 2.9: List of KUs in the Networking and Communication area.

2.9.1 NC/Fundamentals of Networks and Communications (CS Core: 3 hrs)

Topics:
Core

- Importance of networking in contemporary computing, and associated challenges. See also: Digital Equity and Accessibility, Social Impact and AI, Civil Liberties and Expression, Data, Surveillance and Privacy Technology, Privacy Philosophical Foundations and Legislation
- Organization of the internet (e.g., users, Internet Service Providers, autonomous systems, content providers, content delivery networks)
- Switching techniques (e.g., circuit and packet)
- Layers and their roles (application, transport, network, datalink, and physical)
- Layering principles (e.g., encapsulation and hourglass model). See also: Digital Systems and Logic, Sequential and Parallel Processing, State Machines
- Network elements (e.g., routers, switches, hubs, access points, and hosts)
- Basic queueing concepts (e.g., relationship with latency, congestion, service levels, etc.)

Learning Outcomes:
Core:

1. Articulate the organization of the internet [Articulate]
2. List and define the appropriate network terminology [List]
3. Describe the layered structure of a typical networked architecture [Describe]
4. Identify the different types of complexity in a network (edges, core, etc.) [Analyze]

2.9.2 NC/Networked Applications (CS Core: 4 hrs)

Topics:
Core

- Naming and address schemes (e.g., DNS, and Uniform Resource Identifiers)
- Distributed application paradigms (e.g., client/server, peer-to-peer, cloud, edge, and fog). See also: Communication, Coordination
- Diversity of networked application demands (e.g., latency, bandwidth, and loss tolerance). See also: Communication, Sustainable Design and Pervasive Computing, Environmental Footprint of Computing Systems, Systemic Effects and Social Context, Digital Equity and Accessibility, Social Impact and AI
- Coverage of application-layer protocols (e.g., HTTP)
- Interactions with TCP, UDP, and Socket APIs. See also: Programs: Distribution, Programs: Execution Properties, GPU Programming, Programs: Implementation Mappings, Programs: Declarative Parallelism, Programs: Starting Activities

Learning Outcomes:
Core:

1. Define the principles of naming, addressing, resource location [Define]
2. Analyze the needs of specific networked application demands [Analyze]
3. Describe the details of one application layer protocol [Describe]
4. Implement a simple client-server socket-based application [Implement]

2.9.3 NC/Reliability Support (KA Core: 6 hrs)

Topics:
Core

- Unreliable delivery (e.g., UDP)
- Principles of reliability (e.g., delivery without loss, duplication, or out of order). See also: System Reliability
- Error control (e.g., retransmission, error correction)
- Flow control (e.g., stop and wait, window based)
- Congestion control (e.g., implicit and explicit congestion notification)
- TCP and performance issues (e.g., Tahoe, Reno, Vegas, Cubic)

Learning Outcomes:
Core:

1. Describe the operation of reliable delivery protocols [Describe]
2. List the factors that affect the performance of reliable delivery protocols [List]
3. Describe some TCP reliability design issues [Describe]
4. Design and implement a simple reliable protocol [Design]

2.9.4 NC/Routing and Forwarding (KA Core: 4 hrs)

Topics:
Core

- Routing paradigms and hierarchy (e.g., intra/inter domain, centralized and decentralized, source routing, virtual circuits, QoS)
- Forwarding methods (e.g., forwarding tables and matching algorithms)
- IP and Scalability issues (e.g., NAT, CIDR, BGP, different versions of IP)

Learning Outcomes:
Core:

1. Describe various routing paradigms and hierarchies [Describe]
2. Describe how packets are forwarded in an IP network [Describe]
3. Describe how the Internet tackles scalability challenges [Describe]

2.9.5 NC/Single Hop Communication (KA Core: 3 hrs)

Topics:
Core

- Introduction to modulation, bandwidth, and communication media
- Encoding and Framing
- Medium Access Control (MAC) (e.g., random access and scheduled access)
- Ethernet and WiFi
- Switching (e.g., spanning trees, VLANS).
- Local Area Network Topologies (e.g., data center, campus networks).

Learning Outcomes:
Core:

1. Describe some basic aspects of modulation, bandwidth, and communication media [Describe]
2. Describe in detail a MAC protocol [Describe]
3. Demonstrate understanding of encoding and framing solution tradeoffs [Demonstrate]
4. Describe details of the implementation of Ethernet [Describe]
5. Describe how switching works [Describe]
6. Describe one kind of a LAN topology [Describe]

2.9.6 NC/Network Security (KA Core: 3 hrs)

Topics:
Core

- General intro about security (Threats, vulnerabilities, and countermeasures). See also: Computer Crimes and Attacks, Security Laws, Policies and Responsibility, AI and Hardware Security,

Access Control and Applications, Security Mindset and Principles, Security Design and Controls Engineering, Threat Analysis and Security Engineering, Trusted Computing and Privacy Engineering
- Network specific threats and attack types (e.g., denial of service, spoofing, sniffing and traffic redirection, attacker-in-the-middle, message integrity attacks, routing attacks, ransomware, and traffic analysis). See also: AI and Hardware Security, Access Control and Applications, Security Mindset and Principles, Security Design and Controls Engineering, Threat Analysis and Security Engineering, Trusted Computing and Privacy Engineering
- Countermeasures: 1. Cryptography (e.g. SSL, TLS, symmetric/asymmetric) 2. Architectures for secure networks (e.g., secure channels, secure routing protocols, secure DNS, VPNs, DMZ, Zero Trust Network Access, hyper network security, anonymous communication protocols, isolation) 3. Network monitoring, intrusion detection, firewalls, spoofing and DoS protection, honeypots, tracebacks, BGP Sec, RPKI

Learning Outcomes:

Core:

1. Describe some of the threat models of network security [Describe]
2. Describe specific network-based countermeasures [Describe]
3. Analyze various aspects of network security from a case study [Analyze]

2.9.7 NC/Mobility Support (KA Core: 4 hrs)

Topics:

Core

- Principles of cellular communication (e.g., 4G, 5G)
- Principles of Wireless LANs (mainly 802.11)
- Device to device communication (e.g., IoT communication)
- Multi-hop wireless networks (e.g., ad hoc networks, opportunistic, delay tolerant)

Learning Outcomes:

Core:

1. Describe some aspects of cellular communication such as registration [Describe]
2. Describe how 802.11 supports mobile users [Describe]
3. Describe practical uses of device-to-device communication, as well as multihop [Describe]
4. Describe one type of mobile network such as ad hoc [Describe]

2.9.8 NC/Emerging Topics (KA Core: 4 hrs)

Topics:

Core

- Middleboxes (e.g., advances in usage of AI, intent-based networking, filtering, deep packet inspection, load balancing, NAT, CDN)
- Network Virtualization (e.g., SDN, Data Center Networks)
- Quantum Networking (e.g., Intro to the domain, teleportation, security, Quantum Internet)
- Satellite, mmWave, Visible Light

Learning Outcomes:

Core:

1. Describe the value of advances in middleboxes in networks [Describe]
2. Describe the importance of Software Defined Networks [Describe]
3. Describe some of the added value achieved by using Quantum Networking [Describe]

2.10 Operating Systems (OS)

The operating system is a collection of services needed to safely interface the hardware with applications. Core topics focus on the mechanisms and policies needed to virtualize computation, memory, and Input/Output (I/O). Overarching themes that are reused at many levels in computer systems are well illustrated in operating systems (e.g., polling vs interrupts, caching, flexibility vs costs, scheduling approaches to processes, page replacement, etc.). The Operating Systems knowledge area contains

the key underlying concepts for other knowledge areas - trust boundaries, concurrency, persistence, and safe extensibility.

Knowledge Area (KA)	CS Core	KA Core
⭐ 2.10.1 Role and Purpose of Operating Systems (Pg.. 70)	2	
⭐ 2.10.2 Principles of Operating System (Pg.. 71)	2	
⭐ 2.10.3 Concurrency (Pg.. 71)	2	1
⭐ 2.10.4 Protection and Safety (Pg.. 72)	2	1
⭐ 2.10.5 Scheduling (Pg.. 73)		2
⭐ 2.10.6 Process Model (Pg.. 73)		2
⭐ 2.10.7 Memory Management (Pg.. 74)		1
⭐ 2.10.8 Device management (Pg.. 74)		1
⭐ 2.10.9 File Systems API and Implementation (Pg.. 75)		2
⭐ 2.10.10 Advanced File systems (Pg.. 75)		1
⭐ 2.10.11 Virtualization (Pg.. 76)		1
⭐ 2.10.12 Real-time and Embedded Systems (Pg.. 76)		1
⭐ 2.10.13 Fault tolerance (Pg.. 77)		1
2.10.14 Society, Ethics, and the Profession (Pg.. 77)	Elective	

Table 2.10: List of KUs in the Operating Systems area.

2.10.1 OS/Role and Purpose of Operating Systems (CS Core: 2 hrs)

Topics:
Core

- Operating systems mediate between general purpose hardware and application-specific software.
- Universal operating system functions (e.g., process, user and device interfaces, persistence of data)
- Extended and/or specialized operating system functions (e.g., embedded systems, server types such as file, web, multimedia, boot loaders and boot security)
- Design issues (e.g., efficiency, robustness, flexibility, portability, security, compatibility, power, safety, tradeoffs between error checking and performance, flexibility and performance, and security and performance). See also: Security Design and Controls Engineering, Threat Analysis and Security Engineering, Trusted Computing and Privacy Engineering
- Influences of security, networking, multimedia, parallel and distributed computing
- Overarching concern of security/protection: Neglecting to consider security at every layer creates an opportunity to inappropriately access resources. 1. Unauthorized access to files on an unencrypted drive can be achieved by moving the media to another computer. 2. Operating systems enforced security can be defeated by infiltrating the boot layer before the operating system is loaded. 3. Process isolation can be subverted by inadequate authorization checking at API boundaries. 4. Vulnerabilities in system firmware can provide attack vectors that bypass the operating system entirely. 5. Improper isolation of virtual machine memory, computing, and hardware can expose the host system to attacks from guest systems. 6. The operating system may need to mitigate exploitation of hardware and firmware vulnerabilities, leading to potential performance reductions (e.g., Spectre and Meltdown mitigations).
- Exposure of operating systems functions in shells and systems programming. See also: Shell Scripting

Learning Outcomes:
Core:

1. Understand the objectives and functions of modern operating systems [Explain]
2. Evaluate the design issues in different usage scenarios (e.g., real time OS, mobile, server) [Evaluate]
3. Understand the functions of a contemporary operating system with respect to convenience, efficiency, and the ability to evolve [Explain]

4. Understand how evolution and stability are desirable and mutually antagonistic in operating systems function [Explain]

2.10.2 OS/Principles of Operating System (CS Core: 2 hrs)

Topics:
Core

- Operating system software design and approaches (e.g., monolithic, layered, modular, microkernel, unikernel)
- Abstractions, processes, and resources
- Concept of system calls and links to application program interfaces (e.g., Win32, Java, Posix). See also: Assembly Level Machine Organization
- The evolution of the link between hardware architecture and the operating system functions
- Protection of resources means protecting some machine instructions/functions. See also: Assembly Level Machine Organization 1. Applications cannot arbitrarily access memory locations or file storage device addresses. 2. Protection of coprocessors and network devices
- Leveraging interrupts from hardware level: service routines and implementations. See also: Assembly Level Machine Organization 1. Timer interrupts for implementing time slices 2. I/O interrupts for putting blocking threads to sleep without polling
- Concept of user/system state and protection, transition to kernel mode using system calls. See also: Assembly Level Machine Organization
- Mechanism for invoking system calls, the corresponding mode and context switch and return from interrupt. See also: Assembly Level Machine Organization
- Performance costs of context switches and associated cache flushes when performing process switches in Spectre-mitigated environments.

Learning Outcomes:
Core:

1. Understand how the application of software design approaches to operating systems design/implementation (e.g., layered, modular, etc.) affects the robustness and maintainability of an operating system [Explain]
2. Categorize system calls by purpose [Categorize]
3. Understand dynamics of invoking a system call (e.g., passing parameters, mode change) [Explain]
4. Evaluate whether a function can be implemented in the application layer or can only be accomplished by system calls [Evaluate]
5. Apply OS techniques for isolation, protection, and throughput across OS functions (e.g., starvation similarities in process scheduling, disk request scheduling, semaphores, etc.) and beyond [Apply]
6. Understand how the separation into kernel and user mode affects safety and performance [Explain]
7. Understand the advantages and disadvantages of using interrupt processing in enabling multiprogramming [Explain]
8. Analyze potential vectors of attack via the operating systems and the security features designed to guard against them [Analyze]

2.10.3 OS/Concurrency (CS Core: 2 hrs, KA Core: 1 hr)

Topics:
Core

- Thread abstraction relative to concurrency
- Race conditions, critical regions (role of interrupts, if needed). See also: Programs: Distribution, Programs: Execution Properties, GPU Programming, Programs: Implementation Mappings, Programs: Declarative Parallelism, Programs: Starting Activities
- Deadlocks and starvation. See also: Coordination
- Multiprocessor issues (spin-locks, reentrancy).
- Multiprocess concurrency vs multithreading

- Thread creation, states, structures. See also: Digital Systems and Logic, Sequential and Parallel Processing, State Machines
- Thread APIs
- Deadlocks and starvation (necessary conditions/mitigations). See also: Coordination
- Implementing thread safe code (semaphores, mutex locks, condition variables). See also: Performance and Energy Efficiency, Events, Tools and Experimentation, Performance Metrics and Benchmarks, Analytical Performance Models, Evaluation
- Race conditions in shared memory. See also: Coordination

Non Core

- Managing atomic access to OS objects (e.g., big kernel lock vs many small locks vs lockless data structures like lists)

Learning Outcomes:
Core:

1. Understand the advantages and disadvantages of concurrency as inseparable functions within the operating system framework [Explain]
2. Understand how architecture level implementation results in concurrency problems including race conditions [Explain]
3. Understand concurrency issues in multiprocessor systems [Explain]
4. Understand the range of mechanisms that can be employed at the operating system level to realize concurrent systems and describe the benefits of each [Explain]
5. Understand techniques for achieving synchronization in an operating system (e.g., describe how a semaphore can be implemented using OS primitives) including intra-concurrency control and use of hardware atomics [Explain]
6. Accurately analyze code to identify race conditions and appropriate solutions for addressing race conditions [Analyze]

2.10.4 OS/Protection and Safety (CS Core: 2 hrs, KA Core: 1 hr)

Topics:
Core

- Overview of operating system security mechanisms. See also: AI and Hardware Security, Access Control and Applications, Security Mindset and Principles
- Attacks and antagonism (scheduling, etc.). See also: AI and Hardware Security, Access Control and Applications, Security Mindset and Principles
- Review of major vulnerabilities in real operating systems. See also: AI and Hardware Security, Access Control and Applications, Security Mindset and Principles
- Operating systems mitigation strategies such as backups. See also: System Reliability
- Policy/mechanism separation. See also: Risk and Data Management, Security Governance, Management and Policy
- Security methods and devices. See also: AI and Hardware Security, Access Control and Applications, Security Mindset and Principles 1. Rings of protection (history from Multics to virtualized x86) 2. x86_64 rings -1 and -2 (hypervisor and ME/PSP)
- Protection, access control, and authentication. See also: AI and Hardware Security, Access Control and Applications, Security Mindset and Principles, Protocols, Applications and Advanced Cryptography, Cryptographic Algorithms, Cryptography Fundamentals

Learning Outcomes:
Core:

1. Understand the requirement for protection and security mechanisms in operating systems [Explain]
2. List and describe the attack vectors that leverage OS vulnerabilities [List]
3. Understand the mechanisms available in an OS to control access to resources [Explain]
4. Summarize the features and limitations of an operating system that impact protection and security [Summarize]

2.10.5 OS/Scheduling (KA Core: 2 hrs)

Topics:
Core

- Preemptive and non-preemptive scheduling
- Schedulers and policies (e.g., first come, first serve, shortest job first, priority, round robin, multilevel). See also: Resource Management
- Concepts of Symmetric Multi-Processor (SMP) scheduling and cache coherence. See also: Memory Hierarchy
- Timers (e.g., building many timers out of finite hardware timers). See also: Assembly Level Machine Organization
- Fairness and starvation

Non Core

- Subtopics of operating systems such as energy-aware scheduling and real-time scheduling. See also: Performance and Energy Efficiency, Embedded Characteristics and Architectures, Real-Time, Memory and Verification, Systems, Programming and Sensors, Mobile Platforms: Capabilities and Asynchronous Computing, Mobile Platforms: Architecture and Constraints, Mobile Platforms: Development Fundamentals
- Cooperative scheduling, such as Linux futexes and userland scheduling.

Learning Outcomes:
Core:

1. Compare and contrast the common algorithms used for both preemptive and non-preemptive scheduling of tasks in operating systems, such as priority, performance comparison, and fair-share schemes [Compare]
2. Explain the relationships between scheduling algorithms and application domains [Explain]
3. Explain the distinctions among types of processor scheduler such as short-term, medium-term, long-term, and I/O [Explain]
4. Evaluate a problem or solution to determine appropriateness for asymmetric and/or symmetric multiprocessing [Evaluate]
5. Evaluate a problem or solution to determine appropriateness as a process vs threads [Evaluate]
6. List some contexts benefitting from preemption and deadline scheduling [List]

NonCore:

7. Explain the ways that the logic embodied in scheduling algorithms are applicable to other operating systems mechanisms, such as first come first serve or priority to disk I/O, network scheduling, project scheduling, and problems beyond computing [Explain]

2.10.6 OS/Process Model (KA Core: 2 hrs)

Topics:
Core

- Processes and threads relative to virtualization protected memory, process state, memory isolation, etc.
- Memory footprint/segmentation (e.g., stack, heap, etc.). See also: Assembly Level Machine Organization
- Creating and loading executables, shared libraries, and dynamic linking. See also: Language Translation and Execution
- Dispatching and context switching. See also: Assembly Level Machine Organization
- Interprocess communication (e.g., shared memory, message passing, signals, environment variables). See also: Communication

Learning Outcomes:
Core:

1. Understand how processes and threads use concurrency features to virtualize control [Explain]
2. Understand reasons for using interrupts, dispatching, and context switching to support concurrency and virtualization in an operating system [Explain]
3. Understand the different states that a task may pass through, and the data structures needed to support the management of many tasks [Explain]

4. Understand the different ways of allocating memory to tasks, citing the relative merits of each [Explain]
5. Apply the appropriate interprocess communication mechanism for a specific purpose in a programmed software artifact [Apply]

2.10.7 OS/Memory Management (KA Core: 1 hr)

Topics:
Core

- Review of physical memory, address translation and memory management hardware See also: Memory Hierarchy, Discrete Mathematics
- Impact of memory hierarchy including cache concept, cache lookup, and per-CPU caching on operating system mechanisms and policy See also: Memory Hierarchy, Latency, Cache and Memory Hierarchy, Virtualization and Isolation
- Logical and physical addressing, address space virtualization. See also: Memory Hierarchy, Discrete Mathematics
- Concepts of paging, page replacement, thrashing and allocation of pages and frames
- Allocation/deallocation/storage techniques (algorithms and data structure) performance and flexibility 1. Arenas, slab allocators, free lists, size classes, heterogeneously sized pages (huge pages)
- Memory caching and cache coherence and the effect of flushing the cache to avoid speculative execution vulnerabilities. See also: Functional Organization, Memory Hierarchy, Latency, Cache and Memory Hierarchy, Virtualization and Isolation
- Security mechanisms and concepts in memory management including sandboxing, protection, isolation, and relevant vectors of attack. See also: AI and Hardware Security, Access Control and Applications, Security Mindset and Principles

Non Core

- Virtual memory: leveraging virtual memory hardware for OS services and efficiency

Learning Outcomes:
Core:

1. Explain memory hierarchy and cost-performance tradeoffs [Explain]
2. Summarize the principles of virtual memory as applied to caching and paging [Summarize]
3. Evaluate the tradeoffs in terms of memory size (main memory, cache memory, auxiliary memory) and processor speed [Evaluate]
4. Describe the reason for and use of cache memory (performance and proximity, how caches complicate isolation and virtual machine abstraction) [Describe]
5. Code/Develop efficient programs that consider the effects of page replacement and frame allocation on the performance of a process and the system in which it executes [Create]

NonCore:

6. Explain how hardware is utilized for efficient virtualization [Explain]

2.10.8 OS/Device management (KA Core: 1 hr)

Topics:
Core

- Buffering strategies. See also: Interfacing and Communication
- Direct Memory Access (DMA) and polled I/O, Memory-mapped I/O. See also: Interfacing and Communication 1. DMA communication protocols (e.g., ring buffers etc.)
- Historical and contextual - Persistent storage device management (e.g., magnetic, Solid State Device (SSD)). See also: Foundations and Early History of Computing, Modern History: PC, Internet and AI

Non Core

- Device interface abstractions, hardware abstraction layer
- Device driver purpose, abstraction, implementation, and testing challenges
- High-level fault tolerance in device communication

Learning Outcomes:
Core:

1. Explain architecture level device control implementation and link relevant operating system mechanisms and policy (e.g., buffering strategies, direct memory access) [Explain]
2. Explain OS device management layers and the architecture (e.g., device controller, device driver, device abstraction) [Explain]
3. Explain the relationship between the physical hardware and the virtual devices maintained by the operating system [Explain]
4. Explain I/O data buffering and describe strategies for implementing it [Explain]
5. Describe the advantages and disadvantages of direct memory access and discuss the circumstances in which its use is warranted [Describe]

NonCore:

6. Describe the complexity and best practices for the creation of device drivers [Describe]

2.10.9 OS/File Systems API and Implementation (KA Core: 2 hrs)

Topics:
Core

- Concept of a file including data, metadata, operations, and access-mode
- File system mounting
- File access control
- File sharing
- Basic file allocation methods, including linked allocation table
- File system structures comprising file allocation including various directory structures and methods for uniquely identifying files (e.g., name, identified or metadata storage location)
- Allocation/deallocation/storage techniques (algorithms and data structure) impact on performance and flexibility (i.e., internal and external fragmentation and compaction)
- Free space management such as using bit tables vs linking
- Implementation of directories to segment and track file location

Learning Outcomes:
Core:

1. Explain the choices to be made in designing file systems [Explain]
2. Evaluate different approaches to file organization, recognizing the strengths and weaknesses of each [Evaluate]
3. Apply software constructs appropriately given knowledge of the file system implementation [Apply]

2.10.10 OS/Advanced File systems (KA Core: 1 hr)

Topics:
Core

- File systems: partitioning, mount/unmount, virtual file systems
- In-depth implementation techniques
- Memory-mapped files. See also: Interfacing and Communication
- Special-purpose file systems
- Naming, searching, access, backups
- Journaling and log-structured file systems. See also: System Reliability

Non Core

- Distributed file systems
- Encrypted file systems
- Fault tolerance

Learning Outcomes:
Core:

1. Explain how hardware developments have led to changes in the priorities for the design and the management of file systems [Explain]
2. Map file abstractions to a list of relevant devices and interfaces [Map]

3. Identify and categorize different mount types [Analyze]
4. Explain specific file systems requirements and the specialized file systems features that meet those requirements [Explain]
5. Explain the use of journaling and how log-structured file systems enhance fault tolerance [Explain]

NonCore:

6. Explain purpose and complexity of distributed file systems [Explain]
7. List examples of distributed file systems protocols [List]
8. Explain mechanisms in file systems to improve fault tolerance [Explain]

2.10.11 OS/Virtualization (KA Core: 1 hr)

Topics:

Core

- Using virtualization and isolation to achieve protection and predictable performance. See also: Latency, Cache and Memory Hierarchy, Virtualization and Isolation
- Advanced paging and virtual memory. See also: Latency, Cache and Memory Hierarchy, Virtualization and Isolation
- Virtual file systems and virtual devices.
- Containers and their comparison to virtual machines.
- Thrashing (e.g., Popek and Goldberg requirements for recursively virtualizable systems).

Non Core

- Types of virtualizations (including hardware/software, OS, server, service, network). See also: Latency, Cache and Memory Hierarchy, Virtualization and Isolation
- Portable virtualization; emulation vs isolation. See also: Latency, Cache and Memory Hierarchy, Virtualization and Isolation
- Cost of virtualization. See also: Latency, Cache and Memory Hierarchy, Virtualization and Isolation
- Virtual machines and container escapes, dangers from a security perspective. See also: Security Design and Controls Engineering, Threat Analysis and Security Engineering, Trusted Computing and Privacy Engineering
- Hypervisors- hardware virtual machine extensions, hosts with kernel support, QEMU KVM

Learning Outcomes:

Core:

1. Explain how hardware architecture provides support and efficiencies for virtualization [Explain]
2. Explain the difference between emulation and isolation [Explain]
3. Evaluate virtualization tradeoffs [Evaluate]

NonCore:

4. Explain hypervisors and the need for them in conjunction with different types of hypervisors [Explain]

2.10.12 OS/Real-time and Embedded Systems (KA Core: 1 hr)

Topics:

Core

- Process and task scheduling.
- Deadlines and real-time issues. See also: Embedded Characteristics and Architectures, Real-Time, Memory and Verification, Systems, Programming and Sensors
- Low-latency vs ”soft real-time” vs ”hard real time”. See also: Embedded Characteristics and Architectures, Real-Time, Memory and Verification, Systems, Programming and Sensors, Event-Driven and Reactive Programming

Non Core

- Memory/disk management requirements in a real-time environment.
- Failures, risks, and recovery.
- Special concerns in real-time systems (safety).

Learning Outcomes:
Core:
1. Explain what makes a system a real-time system [Explain]
2. Explain latency and its sources in software systems and its characteristics [Explain]
3. Explain special concerns that real-time systems present, including risk, and how these concerns are addressed [Explain]

NonCore:
4. Explain specific real time operating systems features and mechanisms [Explain]

2.10.13 OS/Fault tolerance (KA Core: 1 hr)

Topics:
Core
- Reliable and available systems. See also: System Reliability
- Software and hardware approaches to address tolerance (RAID). See also: System Reliability

Non Core
- Spatial and temporal redundancy. See also: System Reliability
- Methods used to implement fault tolerance. See also: System Reliability
- Error identification and correction mechanisms, checksums of volatile memory in RAM. See also: Memory Hierarchy
- File system consistency check and recovery.
- Journaling and log-structured file systems. See also: System Reliability
- Use-cases for fault-tolerance (databases, safety-critical). See also: System Reliability
- Examples of OS mechanisms for detection, recovery, restart to implement fault tolerance, use of these techniques for the OS's own services. See also: System Reliability

Learning Outcomes:
Core:
1. Explain how operating systems can facilitate fault tolerance, reliability, and availability [Explain]
2. Explain the range of methods for implementing fault tolerance in an operating system [Explain]
3. Explain how an operating system can continue functioning after a fault occurs [Explain]
4. Explain the performance and flexibility tradeoffs that impact using fault tolerance [Explain]

NonCore:
5. Describe operating systems fault tolerance issues and mechanisms in detail [Describe]

2.10.14 OS/Society, Ethics, and the Profession

Topics:
Core
- Open source in operating systems. See also: Open Source, Piracy and DRM, Digital IP Rights and Copyright, Intellectual Property Foundations, Licensing, Plagiarism and Authorship 1. Identification of vulnerabilities in open-source kernels, 2. Open-source guest operating systems, 3. Open-source host operating systems, and 4. Changes in monetization (paid vs free upgrades).
- End-of-life issues with sunsetting operating systems. Example concept: Privacy implications of using proprietary operating systems/operating environments, including telemetry, automated scanning of personal data, built-in advertising, and automatic cloud integration.

Learning Outcomes:
Core:
1. Explain advantages and disadvantages of finding and addressing bugs in open-source kernels [Explain]
2. Contextualize history and positive and negative impact of Linux as an open-source product [Contextualize]
3. List complications with reliance on operating systems past end-of-life [List]
4. Understand differences in finding and addressing bugs for various operating systems payment models [Explain]

2.11 Parallel and Distributed Computing (PDC)

Parallel and distributed programming arranges, coordinates, and controls multiple computations occurring at the same time across different places. The ubiquity of parallelism and distribution are inevitable consequences of increasing numbers of gates in processors, processors in computers, and computers everywhere that may be used to improve performance compared to sequential programs, while also coping with the intrinsic interconnectedness of the world, and the possibility that some components or connections fail or behave maliciously. Parallel and distributed programming removes the restrictions of sequential programming that require computational steps to occur in a serial order in a single place, revealing further distinctions, techniques, and analyses applying at each layer of computing systems.

In most conventional usage, "parallel" programming focuses on establishing and coordinating multiple activities that may occur at the same time, "distributed" programming focuses on establishing and coordinating activities that may occur in different places, and "concurrent" programming focuses on interactions of ongoing activities with each other and the environment. However, all three terms may apply in most contexts. Parallelism generally implies some form of distribution because multiple activities occurring without sequential ordering constraints happen in multiple physical places (unless they rely on context-switching or quantum effects). Conversely, actions in different places need not bear any specific sequential ordering with respect to each other in the absence of communication constraints.

Parallel, distributed, and concurrent programming techniques form the core of High Performance Computing (HPC), distributed systems, and increasingly, nearly every computing application. The PDC knowledge area has evolved from a diverse set of advanced topics into a central body of knowledge and practice, permeating almost every other aspect of computing. Growth of the field has occurred irregularly across different subfields of computing, sometimes with different goals, terminology, and practices, masking the considerable overlap of basic ideas and skills that are the primary focus of this knowledge area. Nearly every problem with a sequential solution also admits parallel and/or distributed solutions; additional problems and solutions arise only in the context of concurrency. Nearly every application domain of parallel and distributed computing is a well-developed area of study and/or engineering too large to enumerate.

Knowledge Area (KA)	CS Core	KA Core
⭐ 2.11.1 Programs: Declarative Parallelism (Pg.. 79)	1	1
⭐ 2.11.2 Programs: Starting Activities (Pg.. 79)	1	1
⭐ 2.11.3 Programs: Execution Properties (Pg.. 79)	1	1
⭐ 2.11.4 Programs: Distribution (Pg.. 79)	1	1
⭐ 2.11.5 Programs: Implementation Mappings (Pg.. 80)	1	1
2.11.6 GPU Programming (Pg.. 80)	Elective	
⭐ 2.11.7 Communication (Pg.. 81)	2	4
⭐ 2.11.8 Communication: Properties and Extensions (Pg.. 81)	1	1
⭐ 2.11.9 Memory and Consistency (Pg.. 82)	1	1
⭐ 2.11.10 Coordination (Pg.. 82)	1	3
⭐ 2.11.11 Coordination: Synchronization and Atomicity (Pg.. 82)	1	2
⭐ 2.11.12 Coordination: Advanced Properties (Pg.. 83)	1	2
⭐ 2.11.13 Evaluation (Pg.. 83)	1	3
⭐ 2.11.14 Evaluation: Formal Methods (Pg.. 84)	1	1
⭐ 2.11.15 Evaluation: Testing and Measurement (Pg.. 84)	1	1
⭐ 2.11.16 Algorithms (Pg.. 84)	1	3
⭐ 2.11.17 Algorithms: Application Domains Survey (Pg.. 85)	1	3
⭐ 2.11.18 Algorithms: Algorithmic Domains (Pg.. 85)	1	3

Table 2.11: List of KUs in the Parallel and Distributed Computing area.

2.11.1 PDC/Programs: Declarative Parallelism (CS Core: 1 hr, KA Core: 1 hr)

Topics:
Core

- Parallelism 1. Declarative parallelism: Determining which actions can, or must not, be performed in parallel, at the level of instructions, functions, closures, compound actions, sessions, tasks, and services is the core idea underlying PDC algorithms; failing to do so is the primary source of errors. See also: Algorithms 2. Defining order: for example, using "happens-before" relationships or series-parallel directed acyclic graphs (DAGs) representing programs. 3. Independence: determining when order does not matter, in terms of commutativity, dependencies, preconditions. 4. Guaranteeing order between otherwise parallel actions when necessary, including locks, safe publication; and enforcing communication - sending a message happens before receiving it; and relaxing it when not necessary.

Learning Outcomes:
Core:

1. Graphically represent (as a Directed Acyclic Graph - DAG) how to parallelize a compound numeric expression; for example, $a = (b + c) * (d + e)$ [Design]
2. Explain why consistency and fault tolerance concepts do not arise in purely sequential programs [Explain]

2.11.2 PDC/Programs: Starting Activities (CS Core: 1 hr, KA Core: 1 hr)

Topics:
Core

- Starting activities 1. Options allowing actions to be performed (eventually) at locations range from hardwired to configuration scripts; also establishing communication and resource management; these are expressed differently in languages and contexts, typically relying on automated provisioning and management by platforms. See also: Resource Management 2. Procedural: Allowing multiple actions to start at a given program point; for example, starting new threads, possibly delimiting their scope or organizing them into hierarchical groups. 3. Reactive: Enabling upon event occurrence by installing an event handler, with less control over when actions start or finish, and may apply even on single processors. 4. Dependent: Enabling upon completion of others; for example, sequencing sets of parallel actions. See also: Coordination. 5. Granularity: The execution cost of action bodies should outweigh the overhead of organizing them.

Learning Outcomes:
Core:

1. Write a service that creates a thread (or other form of procedural activation) to return a requested web page to each new client [Write]

2.11.3 PDC/Programs: Execution Properties (CS Core: 1 hr, KA Core: 1 hr)

Topics:
Core

- Execution Properties 1. Non-deterministic execution of actions without order. 2. Consistency: Ensuring agreement among parties on values and predicates when necessary to avoid races, maintain safety and atomicity, or reach consensus. 3. Fault tolerance: Handling failures in parts or communication, including (Byzantine) misbehavior due to unreliable parties and protocols, when necessary to maintain progress or availability. See also: System Reliability. 4. Trade-offs are a focus of evaluation. See also: Evaluation.

Learning Outcomes:
Core:

1. Write a function that counts events, such as network packet receptions, efficiently [Write]

2.11.4 PDC/Programs: Distribution (CS Core: 1 hr, KA Core: 1 hr)

Topics:
Core

- Distribution 1. Defining locations as devices that execute actions, including hardware components, remote hosts; may also include external uncontrolled devices, hosts, and users. See also: Interfacing and Communication 2. A device may time-share or emulate multiple parallel actions using fewer processors through scheduling and virtualization. See also: Scheduling 3. Naming or identifying locations (e.g., device IDs) and actions as parts (e.g., thread IDs). 4. Activities across locations may communicate through media. See also: Communication

Learning Outcomes:
Core:

1. Write a filter/map/reduce program in multiple styles [Write]

2.11.5 PDC/Programs: Implementation Mappings (CS Core: 1 hr, KA Core: 1 hr)

Topics:
Core

- One or more of the following mappings and mechanisms across layered systems: 1. Instruction-level and data parallelism in the CPU. See also: Functional Organization. 2. SIMD and heterogeneous data parallelism. See also: Heterogeneous Architectures. 3. Scheduled concurrency on multi-core, tasks, actors. See also: Scheduling. 4. Clusters, clouds; elastic provisioning. See also: Common Aspects: Platforms, APIs and Constraints, Common Aspects: Platform Languages and Web Programming Patterns. 5. Networked distributed systems. See also: Networked Applications. 6. Emerging technologies such as quantum computing and molecular computing.

Learning Outcomes:
Core:

1. Explain the trade-offs between different mapping strategies in terms of performance, cost, and implementation complexity [Explain]

2.11.6 PDC/GPU Programming

Topics:
Non Core

- General-Purpose GPU Computing (GPGPU) 1. GPU architecture overview: streaming multiprocessors (SMs), CUDA cores, and warp execution model. 2. SIMT (Single Instruction, Multiple Threads) execution model and its differences from CPU SIMD. 3. Host-device interaction: data transfer between CPU (host) and GPU (device) memory via PCIe. 4. Use cases: scientific computing, machine learning, image processing, and large-scale simulation.
- CUDA Programming Model 1. CUDA execution hierarchy: grids, thread blocks, and individual threads; mapping to GPU hardware. 2. Kernel definition and launch syntax; configuration parameters (grid and block dimensions). 3. Thread indexing: threadIdx, blockIdx, blockDim, gridDim; computing global indices in 1D, 2D, and 3D. 4. Warp divergence: performance implications of branching within a warp. 5. Intra-block synchronization primitive: __syncthreads().
- CUDA Memory Hierarchy 1. Global memory: large, high-latency, accessible by all threads; coalesced access patterns for performance. 2. Shared memory: low-latency on-chip memory shared within a thread block; bank conflicts. 3. Registers and local memory: per-thread private storage. 4. Constant and texture memory: read-only caches optimized for specific access patterns. 5. Unified Memory: simplified programming model with automatic host-device data migration.

Learning Outcomes:
NonCore:

1. Write a CUDA kernel that performs a data-parallel operation (e.g., vector addition, matrix multiplication) and correctly configure grid and block dimensions [Write]
2. Analyze the impact of global memory access patterns (coalesced vs. strided) and shared memory usage on GPU kernel performance [Analyze]
3. Design a parallel algorithm using the CUDA programming model, including thread hierarchy, memory allocation strategy, and host-device data transfers [Design]

2.11.7 PDC/Communication (CS Core: 2 hrs, KA Core: 4 hrs)

Topics:
Core

- Media 1. Varieties: channels (message passing or I/O), shared memory, heterogeneous, data stores 2. Reliance on the availability and nature of underlying hardware, connectivity, and protocols; language support, emulation. See also: Interfacing and Communication
- Channels 1. Explicit (usually named) party-to-party communication media 2. APIs: Sockets, architectural, language-based, and toolkit constructs, such as Message Passing Interface (MPI), and layered constructs such as Remote Procedure Call (RPC). See also: Fundamentals of Networks and Communications 3. I/O channel APIs
- Memory 1. Shared memory architectures in which parties directly communicate only with memory at given addresses, with extensions to heterogeneous memory supporting multiple memory stores with explicit data transfer across them; for example, GPU local and shared memory, Direct Memory Access (DMA) 2. Memory hierarchies: Multiple layers of sharing domains, scopes, and caches; locality: latency, false-sharing 3. Consistency properties: Bitwise atomicity limits, coherence, local ordering
- Data Stores 1. Cooperatively maintained data structures implementing maps and related ADTs 2. Varieties: Owned, shared, sharded, replicated, immutable, versioned

Learning Outcomes:
Core:

1. Explain the similarities and differences among: (1) Party A sends a message on channel X with contents 1 received by party B (2) A sets shared variable X to 1, read by B (3) A sets "X=1" in a distributed shared map accessed by B [Explain]
2. Write a program that distributes different segments of a data set to multiple workers, and collects results (for the simplest example, summing segments of an array) [Write]
3. Write a parallel program that requests data from multiple sites and summarizes them using some form of reduction [Write]
4. Compare the performance of buffered versus unbuffered versions of a producer-consumer program [Compare]

2.11.8 PDC/Communication: Properties and Extensions (CS Core: 1 hr, KA Core: 1 hr)

Topics:
Core

- One or more of the following properties and extensions 1. Topologies: Unicast, Multicast, Mailboxes, Switches; Routing via hardware and software interconnection networks 2. Media concurrency properties: Ordering, consistency, idempotency, overlapping communication with computation 3. Media performance: Latency, bandwidth (throughput) contention (congestion), responsiveness (liveness), reliability (error and drop rates), protocol-based progress (acks, timeouts, mediation) 4. Media security properties: integrity, privacy, authentication, authorization. See also: Information Flow and Non-Interference, Injection and Input Validation, Memory Safety and Types, Malware Analysis and Advanced Security 5. Data formats: Marshaling, validation, encryption, compression 6. Channel policies: Endpoints, sessions, buffering, saturation response (waiting vs dropping), rate control 7. Multiplexing and demultiplexing many relatively slow I/O devices or parties; completion-based and scheduler-based techniques; async-await, select and polling APIs 8. Formalization and analysis of channel communication; for example, CSP 9. Applications of queuing theory to model and predict performance.

Learning Outcomes:
Core:

1. Determine whether a given communication scheme provides sufficient security properties for a given usage [Determine]
2. Give an example of a scenario in which blocking message sends can deadlock [Create]
3. Describe at least one design technique for avoiding liveness failures in programs using multiple locks [Describe]

2.11.9 PDC/Memory and Consistency (CS Core: 1 hr, KA Core: 1 hr)

Topics:
Core

- Memory models, data consistency, and fault tolerance 1. Memory models: sequential and release/acquire consistency 2. Memory management; including reclamation of shared data; reference counts and alternatives 3. Bulk data placement and transfer; reducing message traffic and improving locality; overlapping data transfer and computation; impact of data layout such as array-of-structs vs struct-of-arrays 4. Emulating shared memory: distributed shared memory, Remote Direct Memory Access (RDMA) 5. Data store consistency: Atomicity, linearizability, transactionality, coherence, causal ordering, conflict resolution, eventual consistency, blockchains 6. Faults, partitioning, and partial failures; voting; protocols such as Paxos and Raft. 7. Design tradeoffs among consistency, availability, partition (fault) tolerance; impossibility of meeting all at once 8. Security and trust: Byzantine failures, proof of work and alternatives

Learning Outcomes:
Core:

1. Give an example of an ordering of accesses among concurrent activities (e.g., program with a data race) that is not sequentially consistent [Create]
2. Write a program that illustrates memory-access or message reordering [Write]
3. Describe the relative merits of optimistic versus conservative concurrency control under different rates of contention among updates [Describe]
4. Give an example of a scenario in which an attempted optimistic update may never complete [Create]
5. Modify a concurrent system to use a more scalable, reliable, or available data store [Create]
6. Using an existing platform supporting replicated data stores, write a program that maintains a key-value mapping even when one or more hosts fail [Write]

2.11.10 PDC/Coordination (CS Core: 1 hr, KA Core: 3 hrs)

Topics:
Core

- Dependencies 1. Initiation or progress of one activity may be dependent on other activities, so as to avoid race conditions, ensure termination, or meet other requirements 2. Ensuring progress by avoiding dependency cycles, using monotonic conditions, removing inessential dependencies
- Control constructs and design patterns 1. Completion-based: Barriers, joins, including termination control 2. Data-enabled: Queues, producer-consumer designs 3. Condition-based: Polling, retrying, backoffs, helping, suspension, signaling, timeouts 4. Reactive: Enabling and triggering continuations

Learning Outcomes:
Core:

1. Show how to ensure that a program correctly terminates when all of a set of concurrent tasks have completed [Design]
2. Write a function that efficiently counts events such as sensor inputs or networking packet receptions [Write]
3. Write a filter/map/reduce program in multiple styles [Write]
4. Write a program in which the termination of one set of parallel actions is followed by another [Write]
5. Write a service that creates a thread (or other procedural form of activation) to return a requested web page to each new client [Write]

2.11.11 PDC/Coordination: Synchronization and Atomicity (CS Core: 1 hr, KA Core: 2 hrs)

Topics:
Core

- Atomicity 1. Atomic instructions, enforced local access orderings 2. Locks and mutual exclusion; lock granularity 3. Using locks in a specific language; maintaining liveness without introducing

races 4. Deadlock avoidance: Ordering, coarsening, randomized retries; backoffs, encapsulation via lock managers 5. Common errors: Failing to lock or unlock when necessary, holding locks while invoking unknown operations 6. Avoiding locks: replication, read-only, ownership, and non-blocking constructions

Learning Outcomes:

Core:

1. Show how to avoid or repair a race error in a given program [Analyze]

2.11.12 PDC/Coordination: Advanced Properties (CS Core: 1 hr, KA Core: 2 hrs)

Topics:

Core

- One or more of the following properties and extensions 1. Progress properties including lock-free, wait-free, fairness, priority scheduling, interactions with consistency, reliability 2. Performance with respect to contention, granularity, convoying, scaling 3. Non-blocking data structures and algorithms 4. Ownership and resource control 5. Lock variants and alternatives: sequence locks, read-write locks; Read-Copy-Update (RCU), reentrancy; tickets; controlling spinning versus blocking 6. Transaction-based control: Optimistic and conservative 7. Distributed locking: reliability 8. Alternatives to barriers: Clocks; counters, virtual clocks; dataflow and continuations; futures and RPC; consensus-based, gathering results with reducers and collectors 9. Speculation, selection, cancellation; observability and security consequences 10. Resource control using semaphores and condition variables 11. Control flow: Scheduling computations, series-parallel loops with (possibly elected) leaders, pipelines and streams, nested parallelism 12. Exceptions and failures. Handlers, detection, timeouts, fault tolerance, voting

Learning Outcomes:

Core:

1. Write a program that speculatively searches for a solution by multiple activities, terminating others when one is found [Write]
2. Write a program in which a numerical exception (such as divide by zero) in one activity causes termination of others [Write]
3. Write a program for multiple parties to agree upon the current time of day; discuss its limitations compared to protocols such as network transfer protocol (NTP) [Write]

2.11.13 PDC/Evaluation (CS Core: 1 hr, KA Core: 3 hrs)

Topics:

Core

- Safety and liveness requirements in terms of temporal logic constructs to express ”always” and ”eventually” See also: Parallel and Distributed Computing
- Identifying, testing for, and repairing violations, including common forms of errors such as failure to ensure necessary ordering (race errors), atomicity (including check-then-act errors), and termination (livelock)
- Performance requirements metrics for throughput, responsiveness, latency, availability, energy consumption, scalability, resource usage, communication costs, waiting and rate control, fairness; service level agreements. See also: Latency, Cache and Memory Hierarchy, Virtualization and Isolation
- Performance impact of design and implementation choices, including granularity, overhead, consensus costs, and energy consumption. See also: Sustainable Design and Pervasive Computing, Environmental Footprint of Computing Systems, Systemic Effects and Social Context
- Estimating scalability limitations, for example using Amdahl’s Law or Universal Scalability Law. See also: Events, Tools and Experimentation, Performance Metrics and Benchmarks, Analytical Performance Models

Learning Outcomes:

Core:

1. Revise a specification to enable parallelism and distribution without violating other essential properties or features [Redesign]

2. Explain how concurrent notions of safety and liveness extend their sequential counterparts [Explain]
3. Specify a set of invariants that must hold at each bulk-parallel step of a computation [Analyze]
4. Write a test program that can reveal a data race error; for example, missing an update when two activities both try to increment a variable [Write]
5. In a given context, explain the extent to which introducing parallelism in an otherwise sequential program would be expected to improve throughput and/or reduce latency, and how it may impact energy efficiency [Explain]
6. Show how scaling and efficiency change for sample problems without and with the assumption of problem size changing with the number of processors; further explain whether and how scalability would change under relaxations of sequential dependencies [Design]

2.11.14 PDC/Evaluation: Formal Methods (CS Core: 1 hr, KA Core: 1 hr)

Topics:
Core

- Formal verification and analysis methods 1. Extensions to formal sequential requirements such as linearizability 2. Protocol, session, and transactional specifications 3. Use of tools such as Unified Modelling Language (UML), Temporal Logic of Actions (TLA), program logics 4. Security analysis: safety and liveness in the presence of hostile or buggy behaviors by other parties; required properties of communication mechanisms (for example lack of cross-layer leakage), input screening, rate limiting. See also: AI and Hardware Security, Access Control and Applications, Security Mindset and Principles 5. Static analysis applied to correctness, throughput, latency, resources, energy. See also: Sustainable Design and Pervasive Computing, Environmental Footprint of Computing Systems, Systemic Effects and Social Context 6. Directed Acyclic Graph (DAG) model analysis of algorithmic efficiency (work, span, critical paths)

Learning Outcomes:
Core:

1. Specify and measure behavior when a service is requested by unexpectedly many clients [Analyze]
2. Identify and repair a performance problem due to sequential bottlenecks [Analyze]
3. Empirically compare throughput of two implementations of a common design (perhaps using an existing test harness framework) [Compare]

2.11.15 PDC/Evaluation: Testing and Measurement (CS Core: 1 hr, KA Core: 1 hr)

Topics:
Core

- Testing tools and measurement techniques 1. Testing and debugging; tools such as race detectors, fuzzers, lock dependency checkers, unit/stress/torture tests, visualizations, continuous integration, continuous deployment, and test generators 2. Measuring and comparing throughput, overhead, waiting, contention, communication, data movement, locality, resource usage, behavior in the presence of excessive numbers of events, clients, or threads. See also: Events, Tools and Experimentation, Performance Metrics and Benchmarks, Analytical Performance Models 3. Application domain specific analyses and evaluation techniques

Learning Outcomes:
Core:

1. Identify and repair a performance problem due to communication or data latency [Analyze]
2. Identify and repair a performance problem due to resource management overhead [Analyze]
3. Identify and repair a reliability or availability problem [Analyze]

2.11.16 PDC/Algorithms (CS Core: 1 hr, KA Core: 3 hrs)

Topics:
Core

- Expressing and implementing algorithms in given languages and frameworks, to initiate activities (for example threads), use shared memory constructs, and channel, socket, and/or remote procedure call APIs. See also: Parallel and Distributed Computing. 1. Data parallel examples including map/reduce. 2. Using channel, socket, and/or RPC APIs in a given language, with program control for sending (usually procedural) vs receiving. (usually reactive or RPC-based). 3. Using locks, barriers, and/or synchronizers to maintain liveness without introducing races.

Learning Outcomes:

Core:

1. Implement a parallel/distributed component based on a known algorithm [Implement]
2. Write a data-parallel program that for example computes the average of an array of numbers [Write]
3. Write a producer-consumer program in which one component generates numbers, and another computes their average. Measure speedups when the numbers are small scalars versus large multi-precision values [Write]

2.11.17 PDC/Algorithms: Application Domains Survey (CS Core: 1 hr, KA Core: 3 hrs)

Topics:

Core

- Survey of common application domains across multicore, reactive, data parallel, cluster, cloud, open distributed systems, and frameworks (with reference to the following table). 1. Multicore: Typical Execution agents: Threads. Typical Communication mechanisms: Shared memory, Atomics, locks. Typical Algorithmic domains: Resource management, data processing. Typical Engineering goals: Throughput, latency, energy. 2. Reactive: Typical Execution agents: Handlers, threads. Typical Communication mechanisms: I/O Channels. Typical Algorithmic domains: Services, real-time. Typical Engineering goals: Latency. 3. Data parallel: Typical Execution agents: GPU, SIMD, accelerators, hybrid. Typical Communication mechanisms: Heterogeneous memory. Typical Algorithmic domains: Linear algebra, graphics, data analysis. Typical Engineering goals: Throughput, energy. 4. Cluster: Typical Execution agents: Managed hosts. Typical Communication mechanisms: Sockets, channels. Typical Algorithmic domains: Simulation, data analysis. Typical Engineering goals: Throughput. 5. Cloud: Typical Execution agents: Provisioned hosts. Typical Communication mechanisms: Service APIs. Typical Algorithmic domains: Web applications. Typical Engineering goals: Scalability. 6. Open distributed: Typical Execution agents: Autonomous hosts. Typical Communication mechanisms: Sockets, Data stores. Typical Algorithmic domains: Fault tolerant data stores and services. Typical Engineering goals: Reliability.

Learning Outcomes:

Core:

1. Extend an event-driven sequential program by establishing a new activity in an event handler (for example a new thread in a GUI action handler) [Design]
2. Improve the performance of a sequential component by introducing parallelism and/or distribution [Create]
3. Choose among different parallel/distributed designs for components of a given system [Assess]

2.11.18 PDC/Algorithms: Algorithmic Domains (CS Core: 1 hr, KA Core: 3 hrs)

Topics:

Core

- One of more of the following algorithmic domains. See also: Algorithmic Strategies: 1. Linear algebra: Vector and matrix operations, numerical precision/stability, applications in data analytics and machine learning. 2. Data processing: sorting, searching and retrieval, concurrent data structures. 3. Graphs, search, and combinatorics: Marking, edge-parallelization, bounding, speculation, network-based analytics. 4. Modeling and simulation: differential equations; randomization, N-body problems, genetic algorithms. 5. Computational logic: satisfiability (SAT), concurrent logic programming. 6. Graphics and computational geometry: Transforms,

rendering, ray-tracing. 7. Resource management: Allocating, placing, recycling and scheduling processors, memory, channels, and hosts; exclusive vs shared resources; static, dynamic and elastic algorithms; Real-time constraints; Batching, prioritization, partitioning; decentralization via work-stealing and related techniques. 8. Services: Implementing web APIs, electronic currency, transaction systems, multiplayer games.

Learning Outcomes:

Core:

1. Design, implement, analyze, and evaluate a component or application for X operating in a given context, where X is in one of the listed domains, for example, a genetic algorithm for factory floor design [Design]
2. Critique the design and implementation of an existing component or application, or one developed by classmates [Critique]
3. Compare the performance and energy efficiency of multiple implementations of a similar design, for example, multicore versus clustered versus GPU [Compare]

2.12 Software Development Fundamentals (SDF)

Fluency in the process of software development is fundamental to the study of computer science. To use computers to solve problems most effectively, students must be competent at reading and writing programs. Beyond programming skills, however, they must be able to select and use appropriate data structures and algorithms and use modern development and testing tools.

The SDF knowledge area brings together fundamental concepts and skills related to software development, focusing on concepts and skills that should be taught early in a computer science program, typically in the first year. This includes fundamental programming concepts and their effective use in writing programs, use of fundamental data structures which may be provided by the programming language, basics of programming practices for writing good quality programs, reading, and understanding programs, and some understanding of the impact of algorithms on the performance of the programs. The 43 hours of material in this knowledge area may be augmented with core material from other knowledge areas as students progress to mid- and upper-level courses.

This knowledge area assumes a contemporary programming language with built-in support for common data types including associative data types like dictionaries/maps as the vehicle for introducing students to programming (e.g., Python, Java). However, this is not to discourage the use of older or lower-level languages for SDF - the knowledge units below can be suitably adapted for the actual language used.

The emergence of generative AI and Large Language Models (LLMs), which can generate programs for many programming tasks, will undoubtedly affect the programming profession and consequently the teaching of many CS topics. However, to be able to effectively use generative AI in programming tasks, a programmer must have a good understanding of programs, and hence must still learn the foundations of programming and develop basic programming skills - which is the aim of SDF. Consequently, we feel that the desired outcomes for SDF should remain the same, though different instructors may now give more emphasis to program understanding, documenting, specifications, analysis, and testing. (This is like teaching students multiplication, addition, etc. even though calculators can be used to do them).

Knowledge Area (KA)	CS Core	KA Core
⭐ 2.12.1 Basic Programming Concepts and Constructs (Pg.. 87)	10	
⭐ 2.12.2 Programming Practices: Testing, Debugging, and Documentation (Pg.. 87)	10	
⭐ 2.12.3 Fundamental Data Structures (Pg.. 88)	12	
⭐ 2.12.4 Algorithms (Pg.. 88)	6	
⭐ 2.12.5 Software Development Practices (Pg.. 88)	5	
2.12.6 Society, Ethics, and the Profession (Pg.. 89)	Elective	

Table 2.12: List of KUs in the Software Development Fundamentals area.

2.12.1 SDF/Basic Programming Concepts and Constructs (CS Core: 10 hrs)

Topics:
Core

- Basic concepts such as variables, primitive data types, expressions, and their evaluation
- How imperative programs work: state and state transitions on execution of statements, flow of control
- Basic constructs such as assignment statements, conditional and iterative statements, basic I/O
- Key modularity constructs such as functions (and methods and classes, if supported in the language) and related concepts like parameter passing, scope, abstraction, data encapsulation Object-Oriented Programming Fundamentals, Object-Oriented Programming: Encapsulation, Subtyping, and Reflection
- Input and output using files and APIs
- Structured data types available in the chosen programming language like sequences (e.g., arrays, lists), associative containers (e.g., dictionaries, maps), others (e.g., sets, tuples) and when and how to use them Foundational Data Structures, Foundational Algorithms, Advanced Algorithms

Learning Outcomes:
Core:

1. Develop programs that use the fundamental programming constructs: assignment and expressions, basic I/O, conditional and iterative statements [Create]
2. Develop programs using functions with parameter passing [Create]
3. Develop programs that effectively use the different structured data types provided in the language like arrays/lists, dictionaries, and sets [Create]
4. Develop programs that use file I/O to provide data persistence across multiple executions [Create]
5. Develop programs that use APIs to access or update data (e.g., from the web) [Create]
6. Develop programs that create simple classes and instantiate objects of those classes (if supported by the language) [Create]
7. Trace the flow of control during the execution of a program [Trace]
8. Use appropriate terminology to identify elements of a program (e.g., identifier, operator, operand) [Use]

2.12.2 SDF/Programming Practices: Testing, Debugging, and Documentation (CS Core: 10 hrs)

Topics:
Core

- Libraries and frameworks provided by the language (when/where applicable)
- Recursion
- Dealing with runtime errors in programs (e.g., exception handling).
- Basic concepts of programming errors, testing, and debugging Coding Practices, Large-Scale Construction and Process, Information Flow and Non-Interference, Injection and Input Validation, Memory Safety and Types, Malware Analysis and Advanced Security
- Documenting/commenting code at the program and module level. Coding Practices, Large-Scale Construction and Process
- Develop a security mindset. AI and Hardware Security, Access Control and Applications, Security Mindset and Principles

Learning Outcomes:
Core:

1. Develop programs that use language-provided libraries and frameworks (where applicable) [Create]
2. Explain the concept of recursion and identify when and how to use it effectively [Explain]
3. Develop recursive functions [Create]
4. Develop programs that can handle runtime errors [Create]

5. Read a given program and explain what it does [Explain]
6. Write comments for a program or a module specifying what it does [Write]

2.12.3 SDF/Fundamental Data Structures (CS Core: 12 hrs)

Topics:
Core

- Standard abstract data types such as lists, stacks, queues, sets, and maps/dictionaries, including operations on them.
- Selecting and using appropriate data structures.
- Performance implications of choice of data structure(s).
- Strings and string processing.

Learning Outcomes:
Core:

1. Write programs that use each of the key abstract data types provided in the language (e.g., arrays, tuples/records/structs, lists, stacks, queues, and associative data types like sets, dictionaries/maps) [Write]
2. Select the appropriate data structure for a given problem [Evaluate]
3. Explain how the performance of a program may change when using different data structures or operations [Explain]
4. Write programs that work with text by using string processing capabilities provided by the language [Write]

2.12.4 SDF/Algorithms (CS Core: 6 hrs)

Topics:
Core

- Concept of algorithm and notion of algorithm efficiency
- Some common algorithms (e.g., sorting, searching, tree traversal, graph traversal)
- Impact of algorithms on time-space efficiency of programs

Learning Outcomes:
Core:

1. Explain the role of algorithms for writing programs [Explain]
2. Demonstrate how a problem may be solved by different algorithms, each with different properties [Demonstrate]
3. Explain some common algorithms (e.g., sorting, searching, tree traversal, graph traversal) [Explain]
4. Explain the impact on space/time performance of some algorithms [Explain]

2.12.5 SDF/Software Development Practices (CS Core: 5 hrs)

Topics:
Core

- Basic testing, including test case design
- Use of a general-purpose IDE, including its debugger
- Programming style that improves readability
- Specifying functionality of a module in a natural language.

Learning Outcomes:
Core:

1. Develop tests for modules and apply a variety of strategies to design test cases [Create]
2. Explain some limitations of testing programs [Explain]
3. Build, execute, and debug programs using a modern IDE and associated tools such as visual debuggers [Analyze]
4. Apply basic programming style guidelines to aid readability of programs such as comments, indentation, proper naming of variables, etc [Apply]
5. Write specifications of a module as module comment describing its functionality [Write]

2.12.6 SDF/Society, Ethics, and the Profession

Topics:
Core

- Intellectual property rights of programmers for programs they develop.
- Plagiarism and academic integrity.
- Responsibility and liability of programmers regarding code they develop for solutions. AI and Hardware Security, Access Control and Applications, Security Mindset and Principles
- Basic professional work ethics of programmers.

Learning Outcomes:
Core:

1. Explain/understand some of the intellectual property issues relating to programs [Explain]
2. Explain/understand when code developed by others can be used and proper ways of disclosing their use [Explain]
3. Explain/understand the responsibility of programmers when developing code for an overall solution (which may be developed by a team) [Explain]
4. Explain/understand one or more codes of conduct applicable to programmers [Explain]

2.13 Software Engineering (SE)

As far back as the early 1970s, British computer scientist Brian Randell allegedly said, "Software engineering is the multi-person construction of multi-version programs." This is an essential insight: while programming is the skill that governs our ability to write a program, software engineering is distinct in two dimensions: time and people.

First, a software engineering project is a team endeavor; being a solitary programming expert is insufficient. Skilled software engineers must demonstrate expertise in communication and collaboration. Programming may be an individual activity, but software engineering is a collaborative one, deeply tied to issues of professionalism, teamwork, and communication.

Second, a software engineering project is usually "multi-version." It has an expected lifespan; it needs to function properly for months, years, or decades. Features may be added or removed to meet product requirements. The engineering team itself will likely change. The technological context will shift, as our computing platforms evolve, programming languages change, dependencies upgrade, etc. This exposure to matters of time and change is novel when compared to a programming project: it isn't enough to build a thing that works, instead it must work and stay working. Many of the most challenging topics in tech share "time will lead to change" as a root cause: backward compatibility, version skew, dependency management, schema changes, protocol evolution.

Software engineering presents a particularly difficult challenge for learning in an academic setting. Given that the major differences between programming and software engineering are time and teamwork, it is hard to generate lessons that *require* successful teamwork and that faithfully present the challenges of time. Additionally, some topics in software engineering will be more authentic and more relevant if our learners experience collaborative and long-term software engineering projects *in vivo* rather than in the classroom. Regardless of whether that happens as an internship, involvement in an open-source project, or full-time engineering role, a month of full-time hands-on experience has more available hours than the average software engineering course.

Thus, a software engineering curriculum must focus on concepts needed by most new-graduate hires, and that either are novel for those who are trained primarily as programmers, or that are abstract concepts that may not get explicitly stated/shared on the job. Such topics include, but are not limited to:

- Testing
- Teamwork, collaboration
- Communication
- Design
- Maintenance and evolution
- Software engineering tools

All evidence suggests that the role of software in our society will continue to grow for the foreseeable future. Additionally, the era of "two programmers in a garage" seems to have drawn to a close. Most

important software these days is a team effort, building on existing code and leveraging existing functionality. The study of software engineering skills is a deeply important counterpoint to the everyday experience of computing students - we must impress on them the reality that few software projects are managed by writing from scratch as a solo endeavor. Communication, teamwork, planning, testing, and tooling are far more important as our students move on from the classroom and make their mark on the wider world.

Although most CS graduates will go on to an industry position that requires this material, the CS Core topics presented here are of value regardless of whether graduates go on to industry or academia.

Knowledge Area (KA)	CS Core	KA Core
⭐ 2.13.1 Team Collaboration and Dynamics (Pg.. 90)	2	2
⭐ 2.13.2 Diversity, Stakeholders and Distributed Teams (Pg.. 91)	1	1
⭐ 2.13.3 Version Control and CI/CD (Pg.. 91)	1	2
⭐ 2.13.4 Development Environment and IDE (Pg.. 92)	1	2
⭐ 2.13.5 Requirements Engineering (Pg.. 92)		2
⭐ 2.13.6 Requirements Evolution and Estimation (Pg.. 93)		1
⭐ 2.13.7 Principles and Software Architecture (Pg.. 93)	1	2
⭐ 2.13.8 API Design and Modeling (Pg.. 93)	1	1
⭐ 2.13.9 Software Design Quality and Evaluation (Pg.. 94)	1	2
⭐ 2.13.10 Coding Practices (Pg.. 94)	1	2
⭐ 2.13.11 Large-Scale Construction and Process (Pg.. 95)	1	2
⭐ 2.13.12 Verification and validation Concepts (Pg.. 96)	1	1
⭐ 2.13.13 Test Planning and Types (Pg.. 96)	1	1
⭐ 2.13.14 Advanced Testing Practices (Pg.. 96)	1	1
⭐ 2.13.15 Testing Tools and Analysis (Pg.. 97)	1	1
⭐ 2.13.16 Performance Testing and Benchmarking (Pg.. 97)	1	1
⭐ 2.13.17 API Compatibility and Versioning (Pg.. 98)		2
⭐ 2.13.18 Refactoring Techniques (Pg.. 98)		1
⭐ 2.13.19 Reliability Concepts (Pg.. 99)		2
⭐ 2.13.20 Reliability Engineering (Pg.. 99)		1
2.13.21 Formal Methods (Pg.. 99)	Elective	

Table 2.13: List of KUs in the Software Engineering area.

2.13.1 SE/Team Collaboration and Dynamics (CS Core: 2 hrs, KA Core: 2 hrs)

Topics:

Core

- Effective communication, including oral and written, as well as formal (email, docs, comments, presentations) and informal (team chat, meetings). Technical and Professional Communication, Team Communication and Culture
- Common causes of team conflict, and approaches for conflict resolution.
- Cooperative programming: 1. Pair programming or Swarming 2. Code review 3. Collaboration through version control
- Roles and responsibilities in a software team: Ethics Codes and Legal Accountability, Professional Responsibility and Public Policy, Professional Values, Nature and Continuing Development, Ethical Dissent and Cultural Context, Workplace Ethics 1. Advantages of teamwork 2. Risks and complexity of such collaboration
- Team processes - responsibilities for tasks, effort estimation, meeting structure, work schedule

Learning Outcomes:

Core:

1. Follow effective team communication practices [Analyze]
2. Articulate the sources of, hazards of, and potential benefits of team conflict - especially focusing on the value of disagreeing about ideas or proposals without insulting people [Articulate]
3. Facilitate a conflict-resolution and problem-solving strategy in a team setting [Analyze]

4. Collaborate effectively in cooperative development/programming [Analyze]
5. Propose and delegate necessary roles and responsibilities in a software development team [Propose]
6. Compose and follow an agenda for a team meeting [Compose]
7. Facilitate through involvement in a team project, the central elements of team building, establishing healthy team culture, and team management including creating and executing a team work plan [Create]

2.13.2 SE/Diversity, Stakeholders and Distributed Teams (CS Core: 1 hr, KA Core: 1 hr)

Topics:
Core

- Importance of team diversity and inclusivity. Technical and Professional Communication, Team Communication and Culture
- Interfacing with stakeholders, as a team: 1. Management & other non-technical teams 2. Customers 3. Users
- Risks associated with physical, distributed, hybrid, and virtual teams - including communication, perception, structure, points of failure, mitigation, and recovery, etc.

Learning Outcomes:
Core:

1. Promote the importance of and benefits that diversity and inclusivity brings to a software development team [Defend]
2. Reference, as a team, the importance of, and strategies to interface with stakeholders outside the team on both technical and non-technical levels [Reference]
3. Enumerate the risks associated with physical, distributed, hybrid, and virtual teams and possible points of failure and how to mitigate against and recover/learn from failures [Enumerate]

2.13.3 SE/Version Control and CI/CD (CS Core: 1 hr, KA Core: 2 hrs)

Topics:
Core

- Software configuration management and version control: Software Development Practices 1. Configuration in version control, reproducible builds/configuration. 2. Version control branching strategies. Development branches vs release branches. Trunk-based development. 3. Merging/rebasing strategies, when relevant.
- Release management.
- Testing tools including static and dynamic analysis tools. Software Development Practices, Information Flow and Non-Interference, Injection and Input Validation, Memory Safety and Types, Malware Analysis and Advanced Security
- Software process automation: 1. Build systems - the value of fast, hermetic, reproducible builds, compare/contrast approaches to building a project. 2. Continuous Integration (CI) - the use of automation and automated tests to do preliminary validation that the current head/trunk revision builds and passes (basic) tests. 3. Continuous Deployment (CD) - the use of automation to automatically release every change that passes the automated tests to the production environment, ensuring frequent and reliable deliveries. 4. Dependency management - updating external/upstream dependencies, package management, SemVer.
- Collaborative review workflow as a quality gate 1. Mandatory pull-request review before merging 2. Code ownership via CODEOWNERS-style rules 3. Conventional commit message conventions 4. Branch-protection rules enforcing checks before merge
- Dependency and supply-chain vulnerability scanning integrated into CI pipelines

Learning Outcomes:
Core:

1. Describe the difference between centralized and distributed software configuration management [Describe]
2. Describe how version control can be used to help manage software release management [Describe]

3. Identify configuration items and use a source code control tool in a small team-based project [Analyze]
4. Describe how available static and dynamic test tools can be integrated into the software development environment [Describe]
5. Understand the use of CI/CD systems as a ground-truth for the state of the team's shared code (build and test success) [Explain]
6. Apply a pull-request review workflow with code-ownership rules and branch protection to enforce quality gates on a team project [Apply]
7. Use a dependency/supply-chain vulnerability scanner as part of a CI pipeline to detect vulnerable dependencies [Use]

2.13.4 SE/Development Environment and IDE (CS Core: 1 hr, KA Core: 2 hrs)

Topics:
Core

- Design and communication tools (docs, diagrams, common forms of design diagrams like UML).
- Tool integration concepts and mechanisms. Software Development Practices
- Use of modern IDE facilities - debugging, refactoring, searching/indexing, ML-powered code assistants, etc. Software Development Practices

Learning Outcomes:
Core:

1. Describe the issues that are important in selecting a set of tools for the development of a specific software system, including tools for requirements tracking, design modeling, implementation, build automation, and testing [Describe]
2. Demonstrate the capability to use software tools in support of the development of a software product of medium size [Demonstrate]

2.13.5 SE/Requirements Engineering (KA Core: 2 hrs)

Topics:
Core

- Describe functional requirements using, for example, use cases or user stories. 1. Using at least one method of documenting and structuring functional requirements. 2. Understanding how the method supports design and implementation. 3. Strengths and weaknesses of using a specific approach.
- Properties of requirements including consistency, validity, completeness, and feasibility.
- Requirements elicitation. 1. Sources of requirements, for example, users, administrators, or support personnel. 2. Methods of requirement gathering, for example, surveys, interviews, or behavioral analysis.
- Non-functional requirements, for example, security, usability, or performance, also called as Quality Attributes. Sustainable Design and Pervasive Computing, Environmental Footprint of Computing Systems, Systemic Effects and Social Context
- Risk identification and management, including ethical considerations surrounding the proposed product. Ethics Codes and Legal Accountability, Professional Responsibility and Public Policy, Professional Values, Nature and Continuing Development, Ethical Dissent and Cultural Context, Workplace Ethics
- Communicating and/or formalizing requirement specifications.

Learning Outcomes:
Core:

1. Compare different methods of eliciting requirements along multiple axes [Compare]
2. Identify differences between two methods of describing functional requirements (e.g., customer interviews, user studies) and the situations where each would be preferred [Analyze]
3. Identify which behaviors are required, allowed, or barred from a given set of requirements and a list of candidate behaviors [Analyze]
4. Collect a set of requirements for a simple software system [Analyze]

5. Identify areas of a software system that need to be changed, given a description of the system and a set of new requirements to be implemented [Analyze]
6. Identify the functional and non-functional requirements in a set of requirements [Analyze]

2.13.6 SE/Requirements Evolution and Estimation (KA Core: 1 hr)

Topics:
Non Core

- Prototyping a tool for both eliciting and validating/confirming requirements.
- Product evolution: when requirements change, how to understand what effect that has and what changes need to be made.
- Effort estimation: 1. Learning techniques for better estimating the effort required to complete a task; 2. Practicing estimation and comparing it to how long tasks take; 3. Effort estimation is quite difficult, so students are likely to be way off in many cases, but seeing the process play out with their own work is valuable.

Learning Outcomes:
NonCore:

1. Create a prototype of a software system to validate a set of requirements - building a mock-up, MVP, etc [Create]
2. Estimate the time to complete a set of tasks, then compare estimates to the actual time taken [Estimate]
3. Determine an implementation sequence for a set of tasks, adhering to dependencies between them, with a goal to retire risk as early as possible [Determine]
4. Write a requirement specification for a simple software system [Write]

2.13.7 SE/Principles and Software Architecture (CS Core: 1 hr, KA Core: 2 hrs)

Topics:
Core

- System design principles. System Reliability 1. Levels of abstraction (e.g., architectural design and detailed design) 2. Separation of concerns 3. Information hiding 4. Coupling and cohesion
- Software architecture. System Reliability 1. Design paradigms (a) Top-down functional decomposition/layered design (b) Data-oriented architecture (c) Object-oriented analysis and design (d) Event-driven design 2. Standard architectures (e.g., client-server and microservice architectures including REST discussions, n-layer, pipes-and-filters, Model View Controller) 3. Identifying component boundaries and dependencies
- Programming in the large vs programming in the small. System Reliability
- Code smells and other indications of code quality, distinct from correctness. Security Design and Controls Engineering, Threat Analysis and Security Engineering, Trusted Computing and Privacy Engineering

Learning Outcomes:
Core:

1. Identify the standard software architecture of a given high-level design [Analyze]
2. Select and use an appropriate design paradigm to design a simple software system and explain how system design principles have been applied in this design [Evaluate]
3. Adapt a flawed system design to better follow principles such as separation of concerns or information hiding [Create]
4. Identify the dependencies among a set of software components in an architectural design [Analyze]

2.13.8 SE/API Design and Modeling (CS Core: 1 hr, KA Core: 1 hr)

Topics:
Core

- API design principles 1. Consistency (a) Consistent APIs are easier to learn and less error-prone (b) Consistency is both internal (between different portions of the API) and external (following

common API patterns) 2. Composability 3. Documenting contracts (a) API operations should describe their effect on the system, but not generally their implementation (b) Preconditions, postconditions, and invariants 4. Expandability 5. Error reporting (a) Errors should be clear, predictable, and actionable (b) Input that does not match the contract should produce an error (c) Errors that can be reliably managed without reporting should be managed
- Identifying and codifying data invariants and time invariants
- Structural and behavioral models of software designs

Learning Outcomes:

Core:

1. Design an API for a single component of a large software system, including identifying and documenting each operation's invariants, contract, and error conditions [Design]
2. Evaluate an API description in terms of consistency, composability, and expandability [Evaluate]
3. Expand an existing design to include a new piece of functionality [Create]

2.13.9 SE/Software Design Quality and Evaluation (CS Core: 1 hr, KA Core: 2 hrs)

Topics:

Core

- Data design Data Modeling 1. Data structures 2. Storage systems
- Requirement traceability 1. Understanding which requirements are satisfied by a design

Non Core

- Design modeling, for instance with class diagrams, entity relationship diagrams, or sequence diagrams
- Measurement and analysis of design quality
- Principles of secure design and coding Security Design and Controls Engineering, Threat Analysis and Security Engineering, Trusted Computing and Privacy Engineering 1. Principle of least privilege 2. Principle of fail-safe defaults 3. Principle of psychological acceptability
- Evaluating design tradeoffs (e.g., efficiency vs reliability, security vs usability)
- ISO/IEC 25010 quality characteristics model and cost of poor quality 1. Quality characteristics: functional suitability, performance efficiency, compatibility, usability, reliability, security, maintainability, portability 2. Cost of poor quality and technical debt as a business/economic concern, not just a technical one

Learning Outcomes:

Core:

1. Design a set of data structures to implement a provided API surface [Design]
2. Identify which requirements are satisfied by a provided software design [Analyze]

NonCore:

3. Translate a natural language software design into class diagrams [Translate]
4. Adapt a flawed system design to better follow the principles of least privilege and fail-safe defaults [Create]
5. Contrast two software designs across different qualities, such as efficiency or usability [Contrast]
6. Evaluate a software system against the ISO/IEC 25010 quality characteristics and estimate the business cost of poor quality/technical debt [Evaluate]

2.13.10 SE/Coding Practices (CS Core: 1 hr, KA Core: 2 hrs)

Topics:

Core

- Practical small-scale testing Software Development Practices 1. Unit testing 2. Test-driven development - This is particularly valuable for students psychologically, as it is far easier to engage constructively with the challenge of identifying challenging inputs for a given API (edge cases, corner cases) a priori. If they implement first, the instinct is often to avoid trying to crash their new creation, while a test-first approach gives them the intellectual satisfaction of spotting the problem cases and then watching as more tests pass during the development process.

- Documentation Software Development Practices 1. Interface documentation - describe interface requirements, potentially including (formal or informal) contracts, pre and post conditions, invariants. 2. Implementation documentation should focus on tricky and non-obvious pieces of code, whether because the code is using advanced language features, or the behavior of the code is complex. (Do not add comments that re-state common/obvious operations and simple language features.) (a) Clarify dataflow, computation, etc., focusing on what the code is. (b) Identify subtle/tricky pieces of code and refactor to be self-explanatory if possible or provide appropriate comments to clarify.
- Coding style Software Development Practices 1. Style guides 2. Commenting 3. Naming
- "Best Practices" for coding: techniques, idioms/patterns, mechanisms for building quality programs Software Development Practices, Information Flow and Non-Interference, Injection and Input Validation, Memory Safety and Types, Malware Analysis and Advanced Security 1. Defensive coding practices 2. Secure coding practices and principles 3. Using exception handling mechanisms to make programs more robust, fault-tolerant
- Debugging Software Development Practices
- Logging
- Use of libraries and frameworks developed by others Software Development Practices

Learning Outcomes:

Core:

1. Write appropriate unit tests for a small component (several functions, a single type, etc.) [Write]
2. Write appropriate interface and (if needed) implementation comments for a small component [Write]
3. Describe techniques, coding idioms and mechanisms for implementing designs to achieve desired properties such as reliability, efficiency, and robustness [Describe]
4. Write robust code using exception handling mechanisms [Write]
5. Describe secure coding and defensive coding practices [Describe]
6. Select and use a defined coding standard in a small software project [Use]
7. Compare and contrast integration strategies including top-down, bottom-up, and sandwich integration [Compare]
8. Describe the process of analyzing and implementing changes to code base developed for a specific project [Describe]
9. Describe the process of analyzing and implementing changes to a large existing code base [Describe]

2.13.11 SE/Large-Scale Construction and Process (CS Core: 1 hr, KA Core: 2 hrs)

Topics:

Non Core

- Larger-scale testing 1. Test doubles (stubs, mocks, fakes) 2. Dependency injection
- Work sequencing, including dependency identification, milestones, and risk retirement 1. Dependency identification: Identifying the dependencies between different tasks 2. Milestones: A collection of tasks that serve as a marker of progress when completed. Ideally, the milestone encompasses a useful unit of functionality. 3. Risk retirement: Identifying what elements of a project are risky and prioritizing completing tasks that address those risks.
- Potential security problems in programs Information Flow and Non-Interference, Injection and Input Validation, Memory Safety and Types, Malware Analysis and Advanced Security 1. Buffer and other types of overflows 2. Race conditions 3. Improper initialization, including choice of privileges 4. Input validation
- Documentation (autogenerated)
- Development context: "green field" vs existing code base 1. Change impact analysis 2. Change actualization
- Release management
- DevOps practices 1. Definition of Done as a shared quality bar for completed work 2. Blameless postmortems for learning from incidents/failures 3. Team quality culture: shared ownership, feedback, and continuous learning

Learning Outcomes:

NonCore:

1. Rewrite a simple program to remove common vulnerabilities, such as buffer overflows, integer overflows and race conditions [Apply]
2. Write a software component that performs some non-trivial task and is resilient to input and run-time errors [Write]

2.13.12 SE/Verification and validation Concepts (CS Core: 1 hr, KA Core: 1 hr)

Topics:
Core

- Verification and validation concepts 1. Verification: Are we building the thing right? 2. Validation: Did we build the right thing?
- Why testing matters: Does the component remain functional as the code evolves?
- Testing objectives 1. Usability 2. Reliability 3. Conformance to specification 4. Performance 5. Security

Learning Outcomes:
Core:

1. Explain why testing is important [Explain]
2. Distinguish between program validation and verification [Distinguish]
3. Describe different objectives of testing [Describe]

2.13.13 SE/Test Planning and Types (CS Core: 1 hr, KA Core: 1 hr)

Topics:
Core

- Test kinds 1. Unit 2. Integration 3. Validation 4. System
- Stylistic differences between tests and production code: DAMP vs DRY - more duplication is warranted in test code.
- Test planning and generation 1. Test case generation, from formal models, specifications, etc. 2. Test coverage (a) Test matrices (b) Code coverage - how much of the code is tested? (c) Environment coverage - how many hardware architectures, operating systems, browsers, etc. are tested? 3. Test data and inputs

Learning Outcomes:
Core:

1. Compare and contrast the different types and levels of testing (regression, unit, integration, systems, and acceptance) [Compare]
2. Describe techniques for creating a test plan and generating test cases [Describe]
3. Create a test plan for a medium-size code segment which includes a test matrix and generation of test data and inputs [Create]
4. Implement a test plan for a medium-size code segment [Implement]

2.13.14 SE/Advanced Testing Practices (CS Core: 1 hr, KA Core: 1 hr)

Topics:
Core

- Test development 1. Test-driven development 2. Object oriented testing, mocking, and dependency injection 3. Opaque-box (previously, black-box) and transparent-box (previously, white-box) testing techniques 4. Test tooling, including code coverage, static analysis, and fuzzing
- Verification and validation in the development cycle 1. Code reviews 2. Test automation, including automation of tooling 3. Pre-commit and post-commit testing 4. Tradeoffs between test coverage and throughput/latency of testing 5. Defect tracking and prioritization: reproducibility of reported defects
- Domain specific verification and validation challenges 1. Performance testing and benchmarking 2. Asynchrony, parallelism, and concurrency 3. Safety-critical 4. Numeric

Learning Outcomes:
Core:

1. Identify the fundamental principles of test-driven development methods and explain the role of automated testing in these methods [Analyze]
2. Discuss issues involving the testing of object-oriented software [Debate]
3. Describe mocking and dependency injection and their application [Describe]
4. Undertake, as part of a team activity, a code review of a medium-size code segment [Evaluate]
5. Describe the role that tools can play in the validation of software [Describe]
6. Automate the testing in a small software project [Design]
7. Explain the roles, pros, and cons of pre-commit and post-commit testing [Explain]
8. Discuss the tradeoffs between test coverage and test throughput/latency and how this can impact verification [Debate]
9. Use a defect tracking tool to manage software defects in a small software project [Use]
10. Discuss the limitations of testing in certain domains [Debate]

2.13.15 SE/Testing Tools and Analysis (CS Core: 1 hr, KA Core: 1 hr)

Topics:
Non Core

- Verification and validation tooling and automation 1. Static analysis 2. Code coverage 3. Fuzzing 4. Dynamic analysis and fault containment (sanitizers, etc.) 5. Fault logging and fault tracking
- Test planning and generation 1. Fault estimation and testing termination including defect seeding 2. Use of random and pseudo random numbers in testing
- Testing asynchronous, parallel, and concurrent systems
- Verification and validation of non-code artifacts (documentation, training materials)
- Behavior-driven development (BDD) 1. Expressing acceptance criteria as Gherkin Given/When/Then scenarios 2. Executing BDD scenarios as automated end-to-end tests
- Mutation testing and static code-quality analysis 1. Mutation testing to assess test-suite effectiveness 2. Static detection of code smells and excessive cyclomatic complexity

Learning Outcomes:
NonCore:

1. Describe and compare different tools for verification and validation [Describe]
2. Automate the use of different tools in a small software project [Design]
3. Explain how and when random numbers should be used in testing [Explain]
4. Describe approaches for fault estimation [Describe]
5. Estimate the number of faults in a small software application based on fault density and fault seeding [Estimate]
6. Describe techniques and issues with testing asynchronous, concurrent, and parallel software [Describe]
7. Create a test plan for a medium-size code segment which contains asynchronous, concurrent, and/or parallel code, including a test matrix and generation of test data and inputs [Create]
8. Describe techniques for the verification and validation of non-code artifacts [Describe]
9. Write Gherkin Given/When/Then scenarios for a feature's acceptance criteria and execute them as automated end-to-end tests [Write]
10. Analyze a test suite's mutation score and refactor tests to kill surviving mutants [Analyze]

2.13.16 SE/Performance Testing and Benchmarking (CS Core: 1 hr, KA Core: 1 hr)

Topics:
Non Core

- Performance testing and benchmarking 1. Throughput and latency 2. Degradation under load (stress testing, FIFO vs LIFO handling of requests) 3. Speedup and scaling (a) Amdahl's law (b) Gustafson's law (c) Soft and weak scaling 4. Identifying and measuring figures of merits 5. Common performance bottlenecks (a) Compute-bound (b) Memory-bandwidth bound

(c) Latency-bound 6. Statistical methods and best practices for benchmarking (a) Estimation of uncertainty (b) Confidence intervals 7. Analysis and presentation (graphs, etc.) 8. Timing techniques

Learning Outcomes:

NonCore:

1. Describe throughput and latency and provide examples of each [Describe]
2. Explain speedup and the different forms of scaling and how they are computed [Explain]
3. Describe common performance bottlenecks [Describe]
4. Describe statistical methods and best practices for benchmarking software [Describe]
5. Explain techniques for and challenges with measuring time when constructing a benchmark [Explain]
6. Identify the figures of merit, construct and run a benchmark, and statistically analyze and visualize the results for a small software project [Analyze]

2.13.17 SE/API Compatibility and Versioning (KA Core: 2 hrs)

Topics:

Core

- Hyrum's Law/The Law of Implicit Interfaces
- Backward compatibility 1. Compatibility is not a property of a single entity, it's a property of a relationship. 2. Backward compatibility needs to be evaluated in terms of provider + consumer(s) or with a well-specified model of what forms of compatibility a provider aspires to/promises.
- Versioning 1. Semantic Versioning (SemVer) 2. Trunk-based development

Learning Outcomes:

Core:

1. Identify both explicit and implicit behavior of an interface and identify potential risks from Hyrum's Law [Analyze]
2. Identify changes that can be broadly considered "backward compatible," potentially with explicit statements about what usage is or is not supported [Analyze]
3. Evaluate whether a proposed change is sufficiently safe given the versioning methodology in use for a given project [Evaluate]

2.13.18 SE/Refactoring Techniques (KA Core: 1 hr)

Topics:

Core

- Refactoring 1. Standard refactoring patterns (rename, inline, outline, etc.) 2. Use of refactoring tools in IDE 3. Application of static-analysis tools (to identify code in need of refactoring, generate changes, etc.) 4. Value of refactoring as a remedy for technical debt

Non Core

- "Large Scale" Refactoring - techniques when a refactoring change is too large to commit safely (large projects), or when it is impossible to synchronize change between provider + all consumers (multiple repositories, consumers with private code). 1. Express both old and new APIs so that they can co-exist. 2. Minimize the size of behavior changes. 3. Why these techniques are required, (e.g., "API consumers I can see" vs "consumers I can't see").

Learning Outcomes:

Core:

1. Consider inputs from static analysis tools and/or Software Design principles to identify code in need of refactoring [Analyze]
2. Refactor the implementation of an interface to improve design, clarity, etc. with minimal/zero impact on existing users [Create]

NonCore:

3. Plan a complex multi-step refactoring to change default behavior of an API safely [Plan]

2.13.19 SE/Reliability Concepts (KA Core: 2 hrs)

Topics:
Core

- Concept of reliability as probability of failure or mean time between failures, and faults as cause of failures
- Identifying reliability requirements for different kinds of software
- Software failures caused by defects/bugs, and so for high reliability the goal is to have minimum defects - by injecting fewer defects (better training, education, planning), and by removing most of the injected defects (testing, code review, etc.)
- Software reliability, system reliability and failure behavior
- Defect injection and removal cycle, and different approaches for defect removal
- Compare the "error budget" approach to reliability with the "error-free" approach and identify domains where each is relevant.

Learning Outcomes:
Core:

1. Describe how to determine the level of reliability required by a software system [Describe]
2. Explain the problems that exist in achieving very high levels of reliability [Explain]
3. Understand approaches to minimizing faults that can be applied at each stage of the software lifecycle [Explain]

2.13.20 SE/Reliability Engineering (KA Core: 1 hr)

Topics:
Non Core

- Software reliability models
- Software fault tolerance techniques and models 1. Contextual differences in fault tolerance (e.g., crashing a flight critical system is strongly avoided, crashing a data processing system before corrupt data is written to storage is highly valuable)
- Software reliability engineering practices - including reviews, testing, practical model checking
- Identification of dependent and independent failure domains, and their impact on system reliability
- Measurement-based analysis of software reliability - telemetry, monitoring and alerting, dashboards, release qualification metrics, etc.

Learning Outcomes:
NonCore:

1. Demonstrate the ability to apply multiple methods to develop reliability estimates for a software system [Demonstrate]
2. Identify methods that will lead to the realization of a software architecture that achieves a specified level of reliability [Analyze]
3. Identify ways to apply redundancy to achieve fault tolerance [Analyze]
4. Identify single-point-of-failure (SPF) dependencies in a system design [Analyze]

2.13.21 SE/Formal Methods

Topics:
Non Core

- Formal specification of interfaces 1. Specification of pre- and post- conditions 2. Formal languages for writing and analyzing pre- and post-conditions.
- Problem areas well served by formal methods 1. Lock-free programming, data races 2. Asynchronous and distributed systems, deadlock, livelock, etc.
- Comparison to other tools and techniques for defect detection 1. Testing 2. Fuzzing
- Formal approaches to software modeling and analysis 1. Model checkers 2. Model finders

Learning Outcomes:
NonCore:

1. Describe the role formal specification and analysis techniques can play in the development of complex software and compare their use as validation and verification techniques with testing [Describe]
2. Apply formal specification and analysis techniques to software designs and programs with low complexity [Apply]
3. Explain the potential benefits and drawbacks of using formal specification languages [Explain]

2.14 Security (SEC)

Computing supports nearly every facet of modern critical infrastructure: transportation, communication, healthcare, education, energy generation and distribution, to name a few. With rampant attacks on and breaches of this infrastructure, computer science graduates have an important role in designing, implementing, and operating software systems that are robust, safe, and secure.

The Security (SEC) knowledge area focuses on developing a *security mindset* into the overall ethos of computer science graduates so that security is embedded in all their work products. Computer science students need to learn about system vulnerabilities and understand threats against computer systems. The *Security* title choice was intentional to serve as a one-word umbrella term for this knowledge area, which also includes concepts to support privacy, cryptography, secure systems, secure data, and secure code.

The SEC knowledge area relies on shared concepts pervasive in all the other areas of CS2023. It identifies seven crosscutting concepts of cybersecurity: *confidentiality*, *integrity*, *availability*, *risk assessment*, *systems thinking*, *adversarial thinking*, and *human-centered thinking*. The seventh concept, *human-centered thinking*, is additional to the six crosscutting concepts originally defined in the Cybersecurity Curricula 2017 (CSEC2017). This addition reinforces to students that humans are also a link in the overall chain of security. Principles of protecting systems include security-by-design, privacy-by-design, defense-in-depth, and zero-trust.

Another concept is the notion of assurance, which is an attestation that security mechanisms need to comply with the security policies that have been defined for data, processes, and systems. Assurance is tied in with the concepts of verification and validation. Considerations of data privacy and security are shared with the DM (technical aspects) and SEP knowledge areas.

The SEC knowledge area thus sits atop several of the other CS2023 knowledge areas, while including additional concepts that are not present in those knowledge areas. CS2023 treats security as a crucial component of the skillset of any CS graduate, and the hours needed for security preparation come from all the other 16 CS2023 knowledge areas.

Knowledge Area (KA)	CS Core	KA Core
⭐ 2.14.1 Security Mindset and Principles (Pg.. 101)	1	3
⭐ 2.14.2 Access Control and Applications (Pg.. 102)	1	3
⭐ 2.14.3 AI and Hardware Security (Pg.. 102)	1	2
⭐ 2.14.4 Security Ethics, Law and Society (Pg.. 102)	1	2
⭐ 2.14.5 Privacy and Security by Design (Pg.. 103)	1	1
⭐ 2.14.6 Injection and Input Validation (Pg.. 103)	1	2
⭐ 2.14.7 Memory Safety and Types (Pg.. 104)	1	1
⭐ 2.14.8 Information Flow and Non-Interference (Pg.. 104)	1	1
⭐ 2.14.9 Malware Analysis and Advanced Security (Pg.. 104)	1	3
⭐ 2.14.10 Cryptography Fundamentals (Pg.. 105)	1	2
⭐ 2.14.11 Cryptographic Algorithms (Pg.. 105)	1	2
⭐ 2.14.12 Protocols, Applications and Advanced Cryptography (Pg.. 106)	1	2
⭐ 2.14.13 Threat Analysis and Security Engineering (Pg.. 106)	1	4
⭐ 2.14.14 Security Design and Controls Engineering (Pg.. 107)	1	3
⭐ 2.14.15 Trusted Computing and Privacy Engineering (Pg.. 107)	1	3
⭐ 2.14.16 Digital Forensics Techniques (Pg.. 108)		4
⭐ 2.14.17 Legal Aspects and Incident Handling (Pg.. 108)		3
⭐ 2.14.18 Security Governance, Management and Policy (Pg.. 109)		2
⭐ 2.14.19 Risk and Data Management (Pg.. 109)		1

Table 2.14: List of KUs in the Security area.

2.14.1 SEC/Security Mindset and Principles (CS Core: 1 hr, KA Core: 3 hrs)

This knowledge unit introduces fundamental security concepts, crosscutting principles, and the development of a security mindset. It covers confidentiality, integrity, availability, risk assessment, systems thinking, adversarial thinking, and human-centered thinking. It also includes basic authentication, authorization, vulnerabilities, threats, attack surfaces, denial of service, protection principles, and the impact of AI on security and privacy.

Topics:

Core

- Developing a security mindset incorporating crosscutting concepts: confidentiality, integrity, availability, risk assessment, systems thinking, adversarial thinking, human-centered thinking.
- Vulnerabilities, threats, attack surfaces, and attack vectors Protection and Safety.
- Denial of Service (DoS) and Distributed Denial of Service (DDoS) Protection and Safety.
- Principles and practices of protection, e.g., least privilege, open design, fail-safe defaults, defense in depth, and zero trust; and how they can be implemented Principles of Operating System, Protection and Safety, Coding Practices, Large-Scale Construction and Process, Computer Crimes and Attacks, Security Laws, Policies and Responsibility.
- Optimization considerations between security, privacy, performance, and other design goals System Design, Software Development Practices, Performance Testing and Benchmarking, Testing Tools and Analysis, Verification and validation Concepts, Test Planning and Types, Advanced Testing Practices.
- Concepts of trust and trustworthiness.

Learning Outcomes:

Core:

1. Evaluate a system for possible attacks that can be launched by an adversary [Evaluate]
2. Design and develop approaches to protect a system from a set of identified threats [Design]
3. Describe how harm to user privacy can be avoided [Describe]
4. Develop a system that incorporates various principles of security and privacy [Create]

2.14.2 SEC/Access Control and Applications (CS Core: 1 hr, KA Core: 3 hrs)

Topics:
Core

- Basic concepts of authentication and authorization/access control.
- Access control models (e.g., discretionary, mandatory, role-based, and attribute-based).
- Security controls.
- Applications of a security mindset: web, cloud, and mobile devices System Design, Common Aspects: Platforms, APIs and Constraints, Common Aspects: Platform Languages and Web Programming Patterns.
- Protecting embedded and cyber-physical systems Embedded Characteristics and Architectures, Real-Time, Memory and Verification, Systems, Programming and Sensors.
- Principles of usable security and human-centered computing System Design, Computer Crimes and Attacks, Security Laws, Policies and Responsibility.

Learning Outcomes:
Core:

1. Compare the different access control models in terms of functionality and performance [Compare]

2.14.3 SEC/AI and Hardware Security (CS Core: 1 hr, KA Core: 2 hrs)

Topics:
Core

- Impact of AI on security and privacy: using AI to bolster defenses as well as address increased adversarial capabilities due to AI AI Applications and Societal Impact, Additional AI Applications and Societal Impact, Society, Ethics, and the Profession, System Design.
- Security and trust in AI/machine learning systems, e.g., fit for purpose, ethical operating boundaries, authoritative knowledge sources, verified training data, repeatable system evaluation tests, system attestation, independent validation/certification; unintended consequences from: adverse effect AI Applications and Societal Impact, Additional AI Applications and Societal Impact, Machine Learning Formulation and Evaluation, Deep Learning, RL and Ethics, Introduction to AI: Concepts and Intelligent Behavior, Problem Characteristics and Agent Nature, Machine Learning Fundamentals, Machine Learning Data, Representations and Evaluation, Neural Networks and Ethics, Computer Crimes and Attacks, Security Laws, Policies and Responsibility.
- Security risks in building and operating AI/machine learning systems (e.g., algorithm bias, knowledge corpus bias, training corpus bias, copyright violation) AI Applications and Societal Impact, Additional AI Applications and Societal Impact, Machine Learning Formulation and Evaluation, Deep Learning, RL and Ethics, Introduction to AI: Concepts and Intelligent Behavior, Problem Characteristics and Agent Nature, Machine Learning Fundamentals, Machine Learning Data, Representations and Evaluation, Neural Networks and Ethics.
- Hardware considerations in security, e.g., principles of secure hardware, secure processor architectures, cryptographic acceleration, compartmentalization, software-hardware interaction Assembly Level Machine Organization, Machine-Level Data Representation, Role and Purpose of Operating Systems.

Learning Outcomes:
Core:

1. Show how an adversary could use machine learning algorithms to reduce the security of a system [Analyze]
2. Show how a developer could improve the security of a system using machine learning algorithms [Analyze]
3. Describe hardware (especially CPU) vulnerabilities that can impact software [Describe]

2.14.4 SEC/Security Ethics, Law and Society (CS Core: 1 hr, KA Core: 2 hrs)

This unit covers societal, ethical, legal, and professional aspects of security and privacy, including security-by-design, privacy-by-design, ethical hacking, and the implications of emerging technologies.

Topics:
Core

- Societal impacts on breakdowns in security and privacy Digital Equity and Accessibility, Social Impact and AI, Civil Liberties and Expression, Data, Surveillance and Privacy Technology, Privacy Philosophical Foundations and Legislation, Computer Crimes and Attacks, Security Laws, Policies and Responsibility.
- Applicability of laws and regulations on security and privacy Computer Crimes and Attacks, Security Laws, Policies and Responsibility.
- Professional ethical considerations when designing secure systems and maintaining privacy; ethical hacking Civil Liberties and Expression, Data, Surveillance and Privacy Technology, Privacy Philosophical Foundations and Legislation, Ethics Codes and Legal Accountability, Professional Responsibility and Public Policy, Professional Values, Nature and Continuing Development, Ethical Dissent and Cultural Context, Workplace Ethics, Computer Crimes and Attacks, Security Laws, Policies and Responsibility.
- Security and privacy implications of malicious AI/machine learning actors, e.g., identifying deep fakes Machine Learning Formulation and Evaluation, Deep Learning, RL and Ethics, Introduction to AI: Concepts and Intelligent Behavior, Problem Characteristics and Agent Nature, Machine Learning Fundamentals, Machine Learning Data, Representations and Evaluation, Neural Networks and Ethics, Civil Liberties and Expression, Data, Surveillance and Privacy Technology, Privacy Philosophical Foundations and Legislation, Computer Crimes and Attacks, Security Laws, Policies and Responsibility.
- Societal impacts of Internet of Things (IoT) devices and other emerging technologies on security and privacy Civil Liberties and Expression, Data, Surveillance and Privacy Technology, Privacy Philosophical Foundations and Legislation, Computer Crimes and Attacks, Security Laws, Policies and Responsibility.

Learning Outcomes:
Core:

1. Calculate the impact of a breakdown in security of a given system [Calculate]
2. Construct a system that conforms to security laws [Create]
3. Apply a set of privacy regulations to design a system that protects privacy [Apply]
4. Evaluate the legal ramifications of a system not corresponding to applicable laws and regulations [Evaluate]

2.14.5 SEC/Privacy and Security by Design (CS Core: 1 hr, KA Core: 1 hr)

Topics:
Core

- Principles and practices of privacy Computer Crimes and Attacks, Security Laws, Policies and Responsibility.
- Security by design System Design, System Security.
- Privacy by design and privacy engineering Civil Liberties and Expression, Data, Surveillance and Privacy Technology, Privacy Philosophical Foundations and Legislation, Computer Crimes and Attacks, Security Laws, Policies and Responsibility.

Learning Outcomes:
Core:

1. Construct a system that is designed to avoid harm to user privacy [Create]

2.14.6 SEC/Injection and Input Validation (CS Core: 1 hr, KA Core: 2 hrs)

This unit covers common software vulnerabilities, secure coding practices, defensive programming, and tools for static and dynamic analysis to prevent and mitigate security flaws.
Topics:
Core

- Common vulnerabilities and weaknesses.
- SQL injection and other injection attacks.
- Cross-site scripting techniques and mitigations.

- Input validation and data sanitization Protection and Safety, Basic Programming Concepts and Constructs, Programming Practices: Testing, Debugging, and Documentation, Performance Testing and Benchmarking, Testing Tools and Analysis, Verification and validation Concepts, Test Planning and Types, Advanced Testing Practices.

Learning Outcomes:

Core:

1. Identify underlying problems in given examples of an enumeration of common weaknesses and explain how they can be circumvented [Analyze]
2. Apply input validation and data sanitization techniques to enhance security of a program [Apply]
3. Evaluate and prevent SQL injections attacks on a database application [Evaluate]
4. Evaluate and prevent cross-site scripting attacks against a website [Evaluate]
5. Construct a program that tests for all input handling errors [Create]

2.14.7 SEC/Memory Safety and Types (CS Core: 1 hr, KA Core: 1 hr)

Topics:

Core

- Type safety and type-safe languages Advanced Type Theory, Systems Execution and Memory Model, Type Systems Fundamentals, Type Systems: Polymorphism and Complementary Typing, Protection and Safety, Basic Programming Concepts and Constructs, Programming Practices: Testing, Debugging, and Documentation, Performance Testing and Benchmarking, Testing Tools and Analysis, Verification and validation Concepts, Test Planning and Types, Advanced Testing Practices.
- Buffer overflows, stack smashing, and integer overflows Assembly Level Machine Organization, Systems Execution and Memory Model, Protection and Safety.
- Security issues due to race conditions Parallel and Distributed Computing, Evaluation.

Learning Outcomes:

Core:

1. Describe how the selection of a programming language can impact the security of the system being constructed [Describe]
2. Rewrite a program in a type-safe language (e.g., Java or Rust) originally written in an unsafe programming language (e.g., C/C++) [Redesign]
3. Evaluate a program for possible buffer overflow attacks and rewrite to prevent such attacks [Evaluate]
4. Evaluate a set of related programs for possible race conditions and prevent an adversary from exploiting them [Evaluate]

2.14.8 SEC/Information Flow and Non-Interference (CS Core: 1 hr, KA Core: 1 hr)

Topics:

Core

- Principles of noninterference and nondeducibility.
- Preventing information flow attacks.

Learning Outcomes:

Core:

1. Construct a system that is protected from unauthorized information flows [Create]

2.14.9 SEC/Malware Analysis and Advanced Security (CS Core: 1 hr, KA Core: 3 hrs)

Topics:

Core

- Offensive security techniques as a defense.
- AI-assisted malware detection techniques.
- Ransomware: creation, prevention, and mitigation.

- Secure use of third-party components Coding Practices, Large-Scale Construction and Process, Performance Testing and Benchmarking, Testing Tools and Analysis, Verification and validation Concepts, Test Planning and Types, Advanced Testing Practices.
- Malware: varieties, creation, reverse engineering, and defense against them Language Translation and Execution, Systems Execution and Memory Model.
- Assurance: testing (including fuzzing and penetration testing), verification, and validation Protection and Safety, Basic Programming Concepts and Constructs, Programming Practices: Testing, Debugging, and Documentation, Coding Practices, Large-Scale Construction and Process, Performance Testing and Benchmarking, Testing Tools and Analysis, Verification and validation Concepts, Test Planning and Types, Advanced Testing Practices.
- Static and dynamic analyses Program Analysis and Analyzers, Protection and Safety, Evaluation, Performance Testing and Benchmarking, Testing Tools and Analysis, Verification and validation Concepts, Test Planning and Types, Advanced Testing Practices.
- Secure compilers and secure code generation Language Translation and Execution, Run-time Behavior and Systems.

Learning Outcomes:
Core:

1. Describe different kinds of malicious software [Describe]
2. Explain the risks of misusing interfaces with third-party code and how to correctly use third-party code [Explain]
3. Discuss the need to update software to fix security vulnerabilities and the lifecycle management of the fix [Debate]
4. Apply static and dynamic tools to identify programming faults [Apply]
5. Evaluate a system for the existence of malware and remove it [Evaluate]
6. Implement preventive techniques to reduce the occurrence of ransomware [Implement]

2.14.10 SEC/Cryptography Fundamentals (CS Core: 1 hr, KA Core: 2 hrs)

This unit introduces cryptographic principles, algorithms, protocols, and their applications in securing data and communications, including symmetric/asymmetric encryption, hashing, digital signatures, and emerging topics like quantum and post-quantum cryptography.
Topics:
Core

- Differences between algorithmic, applied, and mathematical views of cryptography.
- Mathematical preliminaries: modular arithmetic, Euclidean algorithm, probabilistic independence, linear algebra basics, number theory, finite fields, complexity, asymptotic analysis Discrete Mathematics, Linear Algebra.
- Basic cryptography: symmetric key and public key cryptography Foundational Data Structures, Foundational Algorithms, Advanced Algorithms, Discrete Mathematics.
- Classical cryptosystems, such as shift, substitution, transposition ciphers, code books, and machines Discrete Mathematics.
- Kerckhoff's principle and use of vetted libraries Coding Practices, Large-Scale Construction and Process.

Learning Outcomes:
Core:

1. Explain the role of cryptography in supporting security and privacy [Explain]
2. Discuss the risks of inventing one's own cryptographic methods [Debate]
3. Discuss the importance of prime numbers in cryptography and explain their use in cryptographic algorithms [Debate]
4. Implement and cryptanalyze classical ciphers [Implement]

2.14.11 SEC/Cryptographic Algorithms (CS Core: 1 hr, KA Core: 2 hrs)

Topics:
Core

- Basic cryptographic building blocks, including symmetric encryption, asymmetric encryption, hashing, and message authentication Discrete Mathematics.
- Additional mathematics: primality, factoring, and elliptic curve cryptography Discrete Mathematics.
- Private-key cryptosystems: substitution-permutation networks, linear cryptanalysis, differential cryptanalysis, DES, and AES Discrete Mathematics, Network Security.
- Public-key cryptosystems: Diffie-Hellman and RSA Discrete Mathematics.
- Data integrity and authentication: hashing, and digital signatures Data Security and Privacy, Discrete Mathematics.
- Attacker capabilities: chosen-message attack (for signatures), birthday attacks, side channel attacks, and fault injection attacks Network Security.

Learning Outcomes:
Core:

1. Describe how crypto keys can be managed securely [Describe]
2. Compare the space and time performance of a given set of cryptographic methods [Compare]
3. Discuss how modern private-key cryptosystems work and ways to cryptanalyze them [Debate]
4. Discuss how modern public-key cryptosystems work and ways to cryptanalyze them [Debate]
5. Compare different cryptographic algorithms in terms of security [Compare]
6. Explain key exchange protocols and show approaches to reduce their failure [Explain]

2.14.12 SEC/Protocols, Applications and Advanced Cryptography (CS Core: 1 hr, KA Core: 2 hrs)

Topics:
Core

- Usage of cryptography in real-world applications, e.g., electronic cash, secure channels between clients and servers, secure electronic mail, entity authentication, device pairing, steganography, and voting systems Image Processing, Network Security.
- Cryptographic protocols: challenge-response authentication, zero-knowledge protocols, commitment, oblivious transfer, secure two- or multi-party computation, hash functions, secret sharing, and applications Discrete Mathematics.
- Quantum cryptography; Post Quantum/Quantum resistant cryptography Foundational Data Structures, Foundational Algorithms, Advanced Algorithms, Discrete Mathematics.
- Blockchain and cryptocurrencies Discrete Mathematics, Communication.

Learning Outcomes:
Core:

1. Describe real-world applications of cryptographic primitives and protocols [Describe]
2. Discuss how quantum cryptography works and the impact of quantum computing on cryptographic algorithms [Debate]

2.14.13 SEC/Threat Analysis and Security Engineering (CS Core: 1 hr, KA Core: 4 hrs)

This unit covers systematic approaches to security engineering, including threat modeling, risk analysis, security design, attack domains, and trustworthy computing.
Topics:
Core

- Security engineering goals: building systems that remain dependable despite errors, accidents, or malicious adversaries Coding Practices, Large-Scale Construction and Process, Performance Testing and Benchmarking, Testing Tools and Analysis, Verification and validation Concepts, Test Planning and Types, Advanced Testing Practices, Computer Crimes and Attacks, Security Laws, Policies and Responsibility.
- Problem analysis and situational analysis to address system security Performance Testing and Benchmarking, Testing Tools and Analysis, Verification and validation Concepts, Test Planning and Types, Advanced Testing Practices.

- Security analysis, covering security requirements analysis; security controls analysis; threat analysis; and vulnerability analysis Program Analysis and Analyzers, Evaluation.
- Security attack domains and attack surfaces, e.g., communications and networking, hardware, physical, social engineering, software, and supply chain Network Security.
- Security attack modes, techniques, and tactics, e.g., authentication abuse; brute force; buffer manipulation; code injection; content insertion; denial of service; eavesdropping; function bypass; impersonation; integrity attack; interception; phishing; protocol analysis; privilege abuse; spoofing; and traffic injection Network Security, Protection and Safety, Performance Testing and Benchmarking, Testing Tools and Analysis, Verification and validation Concepts, Test Planning and Types, Advanced Testing Practices.

Learning Outcomes:
Core:

1. Create a threat model for a system or system design [Create]
2. Apply situational analysis to develop secure solutions under a specified scenario [Apply]
3. Identify and mitigate security vulnerabilities and weaknesses in a system [Analyze]

2.14.14 SEC/Security Design and Controls Engineering (CS Core: 1 hr, KA Core: 3 hrs)

Topics:
Core

- Engineering tradeoff analysis based on time, cost, risk tolerance, risk acceptance, return on investment, and so on Evaluation, Performance Testing and Benchmarking, Testing Tools and Analysis, Verification and validation Concepts, Test Planning and Types, Advanced Testing Practices.
- Security design and engineering, including functional requirements, security subsystems, information protection, security testing, security assessment, and evaluation Evaluation, Performance Testing and Benchmarking, Requirements Engineering, Requirements Evolution and Estimation, Testing Tools and Analysis, Verification and validation Concepts, Test Planning and Types, Advanced Testing Practices.
- Design and development of cyber-physical systems.

Learning Outcomes:
Core:

1. Evaluate a given scenario for tradeoff analysis for system performance, risk assessment, and costs [Evaluate]
2. Design a set of technical security controls, countermeasures, and information protections to meet the security requirements and security objectives for a system [Design]
3. Evaluate the effectiveness of security functions, technical controls, and componentry for a system [Evaluate]

2.14.15 SEC/Trusted Computing and Privacy Engineering (CS Core: 1 hr, KA Core: 3 hrs)

Topics:
Core

- Privacy engineering goals: building systems that design, implement, and deploy privacy features and controls Civil Liberties and Expression, Data, Surveillance and Privacy Technology, Privacy Philosophical Foundations and Legislation.
- Attestation of software products with respect to their specification and adaptiveness Performance Testing and Benchmarking, Requirements Engineering, Requirements Evolution and Estimation, Testing Tools and Analysis, Verification and validation Concepts, Test Planning and Types, Advanced Testing Practices.
- Considerations for trustworthy computing, e.g., tamper resistant packaging, trusted boot, trusted kernel, hardware root of trust, software signing and verification, hardware-based cryptography, virtualization, and containers Coding Practices, Large-Scale Construction and Process, Per-

formance Testing and Benchmarking, Testing Tools and Analysis, Verification and validation Concepts, Test Planning and Types, Advanced Testing Practices.

Learning Outcomes:

Core:

1. Evaluate and predict emergent behavior in areas such as Data Science, AI, and Machine Learning [Evaluate]

2.14.16 SEC/Digital Forensics Techniques (KA Core: 4 hrs)

This unit covers principles, methodologies, and tools for digital forensics, including evidence collection, analysis, incident response, and legal considerations.

Topics:

Core

- Basic principles and methodologies for digital forensics.
- System design for forensics.
- Forensics in different situations: operating systems, file systems, application forensics, web forensics, network forensics, mobile device forensics, use of database auditing Network Security.
- Attacks on forensics and preventing such attacks.

Learning Outcomes:

Core:

1. Explain what a digital investigation is and how it can be implemented Computer Crimes and Attacks, Security Laws, Policies and Responsibility [Explain]
2. Design and implement software to support forensics [Design]
3. Extract data from a hard drive to comply with the law Computer Crimes and Attacks, Security Laws, Policies and Responsibility [Extract]
4. Recover data based on a given search term from an imaged system [Analyze]
5. Reconstruct data and events from an application history, or a web artifact, or a cloud database, or a mobile device. Mobile Platforms: Capabilities and Asynchronous Computing, Mobile Platforms: Architecture and Constraints, Mobile Platforms: Development Fundamentals, Cloud Architecture and Data Management, Frontend, Frameworks and Web Standards [Apply]
6. Capture and analyze network traffic. Network Security [Analyze]
7. Develop approaches to address the challenges associated with mobile device forensics [Create]
8. Apply forensics tools to investigate security breaches [Apply]
9. Identify and mitigate anti-forensic methods [Analyze]

2.14.17 SEC/Legal Aspects and Incident Handling (KA Core: 3 hrs)

Topics:

Core

- Incident handling processes.
- Rules of evidence - general concepts and differences between jurisdictions Computer Crimes and Attacks, Security Laws, Policies and Responsibility.
- Legal issues: digital evidence protection and management, chains of custody, reporting, serving as an expert witness Computer Crimes and Attacks, Security Laws, Policies and Responsibility.

Learning Outcomes:

Core:

1. Describe legal requirements for using seized data and its usage. Computer Crimes and Attacks, Security Laws, Policies and Responsibility [Describe]
2. Describe and implement an end-to-end chain of custody from initial digital evidence seizure to evidence disposal. Civil Liberties and Expression, Data, Surveillance and Privacy Technology, Privacy Philosophical Foundations and Legislation, Computer Crimes and Attacks, Security Laws, Policies and Responsibility [Describe]
3. Discuss a person's professional responsibilities and liabilities when testifying as a forensics expert Ethics Codes and Legal Accountability, Professional Responsibility and Public Policy, Professional Values, Nature and Continuing Development, Ethical Dissent and Cultural Context,

Workplace Ethics [Debate]

2.14.18 SEC/Security Governance, Management and Policy (KA Core: 2 hrs)

This unit covers organizational security governance, risk management, policy development, data lifecycle management, and protection of critical assets.

Topics:

Core

- Protecting critical assets from threats.
- Security governance: organizational objectives and general risk assessment.
- Security management: achieve and maintain appropriate levels of confidentiality, integrity, availability, accountability, authenticity, and reliability Performance Testing and Benchmarking, Testing Tools and Analysis, Verification and validation Concepts, Test Planning and Types, Advanced Testing Practices.
- Security policy: organizational policies, issue-specific policies, system-specific policies.

Learning Outcomes:

Core:

1. Describe critical assets and how they can be protected [Describe]
2. Differentiate between security governance, management, and controls, giving examples of each [Differentiate]
3. Describe a technical control and implement it to mitigate specific threats [Describe]

2.14.19 SEC/Risk and Data Management (KA Core: 1 hr)

Topics:

Core

- Approaches to identifying and mitigating risks to computing infrastructure.
- Data lifecycle management policies: data collection, backups, and retention; cloud storage and services; breach disclosure Data Security and Privacy.

Learning Outcomes:

Core:

1. Identify and assess risk of programs and database applications causing breaches [Analyze]
2. Design and implement appropriate backup strategies conforming to a given policy [Design]
3. Discuss a breach disclosure policy based on legal requirements and implement the policy [Debate]
4. Identify the risks and benefits of outsourcing to the cloud [Analyze]

2.15 Society, Ethics, and the Profession (SEP)

The ACM Code of Ethics and Professional Conduct states: "Computing professionals' actions change the world. To act responsibly, they should reflect upon the wider impacts of their work, consistently supporting the public good." The IEEE Code of Ethics starts by recognizing "the importance of our technologies in affecting the quality of life throughout the world." The AAAI Code of Professional Ethics and Conduct begins with "Computing professionals, and in particular, AI professionals' actions change the world. To act responsibly, they should reflect upon the wider impacts of their work, consistently supporting the public good."

While technical issues dominate the computing curriculum, they do not constitute a complete educational program in the broader context. It is more evident today than ever that students must also be exposed to the larger societal context of computing to develop an understanding of the critical and relevant social, ethical, legal, and professional issues and responsibilities at hand. This need to incorporate the study of these non-technical issues into the ACM curriculum was formally recognized in 1991.

Computer science educators may opt to deliver the material in this knowledge area within the contexts of traditional technical and theoretical courses, in dedicated courses, and as part of capstone, project, and professional practice courses. The material in this knowledge area is best covered through a combination of all the above.

The ACM Code of Ethics and Professional Conduct, the IEEE Code of Ethics, and the AAAI Code of Professional Ethics and Conduct provide guidance that serve as the basis for the conduct of all computing professionals in their work. The ACM Code emphasizes that ethical reasoning is not an algorithm to be followed, and computer professionals are expected to consider how their work impacts the public good as the primary consideration.

A computer science student must not graduate without understanding how society and ethics influence the computing profession. Nor should it be possible to complete a computer science degree without learning how computing professionals influence society, the ethical considerations involved in shaping that impact, and the student-turned-graduate's role in these relationships, as both computing professionals and members of society.

Knowledge Area (KA)	CS Core	KA Core
⭐ 2.15.1 Social Impact and AI (Pg.. 110)	2	1
⭐ 2.15.2 Digital Equity and Accessibility (Pg.. 111)	2	1
⭐ 2.15.3 Ethical Foundations and Frameworks (Pg.. 111)	2	1
⭐ 2.15.4 Ethical Analysis Methods and Tools (Pg.. 112)	1	1
⭐ 2.15.5 Professional Values, Nature and Continuing Development (Pg.. 112)	1	1
⭐ 2.15.6 Ethics Codes and Legal Accountability (Pg.. 113)	1	1
⭐ 2.15.7 Professional Responsibility and Public Policy (Pg.. 113)	1	1
⭐ 2.15.8 Ethical Dissent and Cultural Context (Pg.. 114)	1	1
⭐ 2.15.9 Workplace Ethics (Pg.. 114)	1	1
⭐ 2.15.10 Intellectual Property Foundations (Pg.. 114)	1	1
⭐ 2.15.11 Licensing, Plagiarism and Authorship (Pg.. 115)	1	1
⭐ 2.15.12 Digital IP Rights and Copyright (Pg.. 115)	1	1
⭐ 2.15.13 Open Source, Piracy and DRM (Pg.. 115)	1	1
⭐ 2.15.14 Data, Surveillance and Privacy Technology (Pg.. 115)	1	1
⭐ 2.15.15 Privacy Philosophical Foundations and Legislation (Pg.. 116)	1	1
⭐ 2.15.16 Civil Liberties and Expression (Pg.. 116)	1	1
⭐ 2.15.17 Technical and Professional Communication (Pg.. 117)	1	1
⭐ 2.15.18 Team Communication and Culture (Pg.. 117)	1	1
⭐ 2.15.19 Environmental Footprint of Computing Systems (Pg.. 118)	1	1
⭐ 2.15.20 Systemic Effects and Social Context (Pg.. 118)	1	1
⭐ 2.15.21 Sustainable Design and Pervasive Computing (Pg.. 118)	1	1
⭐ 2.15.22 Foundations and Early History of Computing (Pg.. 119)	1	1
⭐ 2.15.23 Modern History: PC, Internet and AI (Pg.. 119)	1	1
⭐ 2.15.24 Economic Models and Strategies (Pg.. 120)		1
⭐ 2.15.25 Automation, AI and Economic Ethics (Pg.. 120)		1
⭐ 2.15.26 Computer Crimes and Attacks (Pg.. 120)	1	1
⭐ 2.15.27 Security Laws, Policies and Responsibility (Pg.. 121)	1	1
⭐ 2.15.28 Identity and Diversity in Computing (Pg.. 121)	1	1
⭐ 2.15.29 DEIA Historical and Cross-Cultural Context (Pg.. 122)	1	1
⭐ 2.15.30 Inclusion and Computing Practices (Pg.. 122)	1	1
⭐ 2.15.31 Accessible Design (Pg.. 122)	1	1

Table 2.15: List of KUs in the Society, Ethics, and the Profession area.

2.15.1 SEP/Social Impact and AI (CS Core: 2 hrs, KA Core: 1 hr)

Social Context provides the foundation for all other knowledge units in SEP, particularly Professional Ethics. It covers the social implications of computing in a hyper-networked world, the impact of computing applications on individual well-being and safety, the consequences of involving computing technologies in civic life, and how deficits in diversity and accessibility in computing affect society.

Topics:

Core

- Social implications (e.g., political and cultural ideologies) in a hyper-networked world where the capabilities and impact of social media, artificial intelligence, and computing in general are rapidly evolving.
- Impact of computing applications (e.g., social media, artificial intelligence applications) on individual well-being, and safety of all kinds (e.g., physical, emotional, economic).
- Consequences of involving computing technologies, particularly artificial intelligence, biometric technologies, and algorithmic decision-making systems, in civic life (e.g., facial recognition technology, biometric tags, resource distribution algorithms, policing software) and how human agency and oversight is crucial.
- Growth and control of the internet, data, computing, and artificial intelligence.

Learning Outcomes:

Core:

1. Describe the different ways that computer technology (networks, mobile computing, artificial intelligence) mediates social interaction at the personal and collective levels [Describe]
2. Interpret the social context of a given design and its implementation [Interpret]
3. Analyze the efficacy of a given design and implementation using empirical data [Analyze]
4. Understand the implications of technology use (e.g., social media) for different identities, cultures, and communities [Explain]
5. Describe the internet's role in facilitating communication between citizens, governments, and each other [Describe]
6. Analyze the effects of reliance on computing in the implementation of democracy (e.g., delivery of social services, electronic voting) [Analyze]
7. Describe the impact of a lack of appropriate representation of people from historically minoritized populations in the computing profession (e.g., industry culture, product diversity) [Describe]

2.15.2 SEP/Digital Equity and Accessibility (CS Core: 2 hrs, KA Core: 1 hr)

Topics:

Core

- How deficits in diversity and accessibility in computing affect society and what steps can be taken to improve equity in computing.
- Often referred to as the digital divide, differences in access to digital technology resources and its resulting ramifications for gender, class, ethnicity, geography, and/or developing countries, including consideration of responsibility to those who might be less wealthy, under threat, or who would struggle to have their voices heard.
- Accessibility issues, including legal requirements such as Web Content Accessibility Guidelines (www.w3.org/TR/WCAG21).
- Context-aware computing.

Learning Outcomes:

Core:

1. Identify developers' assumptions and values embedded in hardware and software design, especially as they pertain to usability for diverse populations including under-served and those with disabilities [Analyze]
2. Discuss the implications of context awareness in ubiquitous computing systems [Debate]
3. Express how access to the internet and computing technologies affect different societies [Explain]
4. Identify why/how internet access can be viewed as a human right [Analyze]

2.15.3 SEP/Ethical Foundations and Frameworks (CS Core: 2 hrs, KA Core: 1 hr)

Ethical theories and principles are the foundations of ethical analysis because they are the viewpoints which can provide guidance along the pathway to a decision. Each theory emphasizes different assumptions and methods for determining the ethicality of a given action. It is important for students to recognize that decisions in different contexts may require different ethical theories (including combinations) to arrive at ethically acceptable outcomes, and what constitutes 'acceptable' depends on a variety of factors such as cultural context. Applying methods for ethical analysis requires both an understanding of the underlying principles and assumptions guiding a given tool and an awareness of

the social context for that decision. Traditional ethical frameworks as provided by western philosophy can be useful, but they are not all-inclusive. Effort must be taken to include decolonial, indigenous, and historically marginalized ethical perspectives whenever possible.

Topics:

Core

- Avoiding fallacies and misrepresentation in argumentation.
- Ethical theories and decision-making (philosophical and social frameworks, e.g. [1]).
- Recognition of the role culture plays in our understanding, adoption, design, and use of computing technology.
- Why ethics is important in computing, and how ethics is similar to, and different from, laws and social norms.
- Introduction to ethical frameworks (e.g., consequentialism such as utilitarianism, non-consequentialism such as duty, rights, or justice, agent-centered such as virtue or feminism, contractarianism, ethics of care) and their use for analyzing an ethical dilemma.

Learning Outcomes:

Core:

1. Describe how a given cultural context impacts decision making [Describe]
2. Express the use of example and analogy in ethical argument [Analyze]
3. Analyze (and avoid) basic logical fallacies in an argument [Analyze]
4. Analyze an argument to identify premises and conclusion [Analyze]
5. Evaluate how and why ethics is so important in computing and how it relates to cultural norms, values, and law [Evaluate]
6. Justify a decision made on ethical grounds [Justify]
7. Discuss the advantages and disadvantages of traditional ethical frameworks [Debate]
8. Analyze ethical dilemmas related to the creation and use of technology from multiple perspectives using ethical frameworks [Analyze]

2.15.4 SEP/Ethical Analysis Methods and Tools (CS Core: 1 hr, KA Core: 1 hr)

Topics:

Core

- Professional checklists.
- Evaluation rubrics.
- Stakeholder analysis.
- Standpoint theory.

Learning Outcomes:

Core:

1. Distinguish all stakeholder positions in relation to their cultural context in a given situation [Distinguish]
2. Analyze the potential for introducing or perpetuating ethical debt (deferred consideration of ethical impacts or implications) in technical decisions [Analyze]

2.15.5 SEP/Professional Values, Nature and Continuing Development (CS Core: 1 hr, KA Core: 1 hr)

Computing professionals have responsibilities beyond their technical skills. This unit explores the community values and legal frameworks that guide the profession, the nature of professionalism (including care, discipline, and mentoring), and the professional's responsibility to remain current in a constantly evolving field.

Topics:

Core

- Community values and the laws by which we live.
- The nature of being a professional including care, attention, discipline, fiduciary responsibility, and mentoring.
- Keeping up to date as a computing professional in terms of familiarity, tools, skills, legal and professional frameworks as well as the ability and responsibility to self-assess and progress in

the computing field.

Learning Outcomes:

Core:

1. Identify ethical issues that arise in software design, development practices, and software deployment [Analyze]
2. Discuss how to address ethical issues in specific situations [Debate]
3. Describe the mechanisms that typically exist for a professional to keep up to date in ethical matters [Describe]

2.15.6 SEP/Ethics Codes and Legal Accountability (CS Core: 1 hr, KA Core: 1 hr)

Professional ethics codes from organizations such as ACM, IEEE, and AAAI define standards of conduct and responsibility in computing. This unit examines professional certification, relevant ethics codes, and the legal and ethical dimensions of software accountability, including correctness, reliability, and security.

Topics:

Core

- Professional certification, codes of ethics, conduct, and practice, such as the ACM, IEEE, AAAI, and other international societies.
- Accountability, responsibility, and liability (e.g., software correctness, reliability and safety, warranty, negligence, strict liability, ethical approaches to security vulnerability disclosures) including whether a product/service should be built, not just doing so because it is technically possible.

Learning Outcomes:

Core:

1. Express the ethical responsibility of ensuring software correctness, reliability and safety including from where this responsibility arises (e.g., ACM/IEEE/AAAI Codes of Ethics, laws and regulations, organizational policies) [Analyze]
2. Describe the strengths and weaknesses of relevant professional codes as expressions of being a professional and guides to decision-making [Describe]

2.15.7 SEP/Professional Responsibility and Public Policy (CS Core: 1 hr, KA Core: 1 hr)

Computing professionals operate within complex social, legal, and cultural frameworks. This unit examines theories that explain how humans create and use technology—including decolonial and social justice approaches—and explores professionals' responsibilities to recognize, report, and maintain awareness of the ethical consequences of their work, including their participation in public policy.

Topics:

Core

- Introduction to theories describing the human creation and use of technology including instrumentalism, sociology of technological systems, disability justice, neutrality thesis, pragmatism, and decolonial models, including developing and using technology to right wrongs and do good.
- Strategies for recognizing and reporting designs, systems, software, and professional conduct (or their outcomes) that may violate law or professional codes of ethics.
- The role of the computing professional and professional societies in public policy.
- Maintaining awareness of consequences.

Learning Outcomes:

Core:

1. Analyze a global computing issue, observing the role of professionals and government officials in managing this problem [Analyze]
2. Describe ways in which professionals and professional organizations may contribute to public policy [Describe]
3. Describe the consequences of inappropriate professional behavior [Describe]

2.15.8 SEP/Ethical Dissent and Cultural Context (CS Core: 1 hr, KA Core: 1 hr)

Ethical dissent is a critical component of responsible professional conduct. This unit addresses whistleblowing as an accountability mechanism, and explores how regional cultural context shapes understanding and response to ethical dilemmas in the computing profession.

Topics:

Core

- Ethical dissent and whistleblowing.
- The relationship between regional culture and ethical dilemmas.

Learning Outcomes:

Core:

1. Be familiar with whistleblowing and have access to the knowledge to guide one through an incident [Assess]
2. Identify examples of how regional culture interplays with ethical dilemmas [Analyze]

2.15.9 SEP/Workplace Ethics (CS Core: 1 hr, KA Core: 1 hr)

Topics:

Core

- Dealing with harassment and discrimination.
- Forms of professional credentialing.
- Ergonomics and healthy computing environments.
- Time-to-market and cost considerations versus quality professional standards.

Learning Outcomes:

Core:

1. Describe forms of harassment and discrimination and avenues of assistance [Describe]
2. Assess various forms of professional credentialing [Assess]
3. State the relationship between ergonomics in computing environments and people's health [State]
4. Describe issues associated with industries' push to focus on time-to-market versus enforcing quality professional standards [Describe]

2.15.10 SEP/Intellectual Property Foundations (CS Core: 1 hr, KA Core: 1 hr)

Intellectual property refers to a range of intangible rights of ownership in any product of human intellect, such as a software program. Laws, which vary by locality, provide different methods for protecting these rights of ownership based on their type. Ideally, intellectual property laws balance the interests of creators and users of the property. There are four types of intellectual property rights relevant to software: patents, copyrights, trade secrets, and trademarks. Moreover, property rights are often protected by user licenses. Each affords a different type of legal protection.

Topics:

Core

- Intellectual property rights.
- Legal foundations for intellectual property protection.
- Philosophical foundations of intellectual property.
- Forms of intellectual property (e.g., copyrights, patents, trade secrets, trademarks) and the rights they protect.

Learning Outcomes:

Core:

1. Defend legal and ethical uses of copyrighted materials [Defend]
2. Discuss the philosophical bases of intellectual property in an appropriate context (e.g., country) [Debate]
3. Contrast the protections and obligations of copyright, patent, trade secret, and trademarks [Contrast]
4. Describe the rationale for the legal protection of intellectual property in the appropriate context (e.g., country) [Describe]

2.15.11 SEP/Licensing, Plagiarism and Authorship (CS Core: 1 hr, KA Core: 1 hr)

Topics:
Core
- Common software licenses (e.g., MIT, GPL and its variants, Apache, Mozilla, Creative Commons).
- Plagiarism and authorship.

Learning Outcomes:
Core:
1. Select an appropriate software license for a given project [Analyze]
2. Interpret the intent and implementation of software licensing [Interpret]
3. Analyze the ethical issues inherent in various plagiarism detection mechanisms [Analyze]
4. Identify multiple forms of plagiarism beyond verbatim copying of text or software (e.g., intentional paraphrasing, authorship misrepresentation, and improper attribution) [Analyze]

2.15.12 SEP/Digital IP Rights and Copyright (CS Core: 1 hr, KA Core: 1 hr)

Topics:
Core
- Intangible digital intellectual property (IDIP).
- Limitations on copyright protections, including fair use and the first sale doctrine.
- Intellectual property laws and treaties that impact the enforcement of copyrights.

Learning Outcomes:
Core:
1. Describe and critique legislation and precedent aimed at digital copyright infringements [Describe]
2. Identify contemporary examples of intangible digital intellectual property [Analyze]
3. Discuss whether a use of copyrighted material is likely to be fair use [Debate]
4. Analyze the use of copyrighted work under the concepts of fair use and the first sale doctrine [Analyze]
5. Discuss the nature of anti-circumvention laws in the context of copyright protection [Debate]

2.15.13 SEP/Open Source, Piracy and DRM (CS Core: 1 hr, KA Core: 1 hr)

Topics:
Core
- Software piracy and technical methods for enforcing intellectual property rights, such as digital rights management and closed source software as a trade secret.
- Moral and legal foundations of the open-source movement.
- Systems that use others' data (e.g., large language models).

Learning Outcomes:
Core:
1. Distinguish the conflicting issues involved in securing software patents [Distinguish]
2. Identify the goals of the open-source movement and its impact on fields beyond computing, such as the right-to-repair movement [Analyze]
3. Summarize the global nature of software piracy [Summarize]
4. Criticize the use of technical measures of digital rights management (e.g., encryption, watermarking, copy restrictions, and region lockouts) from multiple stakeholder perspectives [Criticize]

2.15.14 SEP/Data, Surveillance and Privacy Technology (CS Core: 1 hr, KA Core: 1 hr)

The massive collection of data in transactional systems, surveillance platforms, cloud computing, and artificial intelligence poses fundamental challenges for privacy. This unit examines the implications of these systems, conceptions of anonymity and pseudonymity, and technological solutions for protecting

privacy, such as end-to-end encryption and differential privacy.

Topics:

Core

- Privacy implications of widespread data collection including but not limited to transactional databases, data warehouses, surveillance systems, cloud computing, and artificial intelligence.
- Conceptions of anonymity, pseudonymity, and identity.
- Technology-based solutions for privacy protection (e.g., end-to-end encryption and differential privacy).

Learning Outcomes:

Core:

1. Evaluate solutions to privacy threats in transactional databases and data warehouses [Evaluate]
2. Describe the role of data collection in the implementation of pervasive surveillance systems (e.g., RFID, face recognition, toll collection, mobile computing) [Describe]
3. Distinguish the concepts and goals of anonymity and pseudonymity [Distinguish]
4. Describe the ramifications of technology-based privacy protections, including differential privacy and end-to-end encryption [Describe]

2.15.15 SEP/Privacy Philosophical Foundations and Legislation (CS Core: 1 hr, KA Core: 1 hr)

The right to privacy has deep philosophical and legal roots that vary by cultural and legal context. This unit examines philosophical and legal conceptions of privacy, international legal frameworks such as GDPR, and sector-specific legislation such as HIPAA and the EU AI Act, critically evaluating their intent, value, and implementation.

Topics:

Core

- Philosophical and legal conceptions of the nature of privacy including the right to privacy.
- Legal foundations of privacy protection in relevant jurisdictions (e.g., GDPR in the EU).
- Privacy legislation in areas of practice (e.g., HIPAA in the US, AI Act in the EU).

Learning Outcomes:

Core:

1. Discuss the philosophical basis for the legal protection of personal privacy in an appropriate context (e.g., country) [Debate]
2. Critique the intent, potential value, and implementation of various forms of privacy legislation and principles beyond what the law requires [Critique]

2.15.16 SEP/Civil Liberties and Expression (CS Core: 1 hr, KA Core: 1 hr)

Topics:

Core

- Civil liberties, privacy rights, and cultural differences.
- Basic Principles of human-subjects research and principles beyond what the law requires (e.g., Belmont Report, UN Universal Declaration on Human Rights and how this relates to technology).
- Freedom of expression and its limitations.
- User-generated content, content moderation, and liability.

Learning Outcomes:

Core:

1. Identify cultural differences regarding the nature and necessity of privacy and other civil liberties [Analyze]
2. Identify strategies to enable appropriate freedom of expression [Analyze]

2.15.17 SEP/Technical and Professional Communication (CS Core: 1 hr, KA Core: 1 hr)

Computing is an inherently collaborative and social discipline making communication an essential aspect of the profession. Much but not all of this communication occurs in a professional setting where communication styles, expectations, and norms differ from other contexts where similar technology might be used. Both professional and informal communication conveys information to various audiences who may have different goals and needs for that information. Good communication is also necessary for transparency and trustworthiness. It is also important to note that computing professionals are not just communicators but are also listeners who must be able to hear and thoughtfully make use of feedback received from various stakeholders. Effective communication skills are not something one 'just knows' - they are developed and can be learned. Communication skills are best taught in context throughout the undergraduate curriculum.

Topics:

Core

- Oral, written, and electronic team and group communication.
- Technical communication materials (e.g., source code, and documentation, tutorials, reference materials, API documentation).
- Communicating with different stakeholders such as customers, leadership, or the public.
- Tradeoffs in competing factors that affect communication channels and choices.

Learning Outcomes:

Core:

1. Understand the importance of writing concise and accurate technical documents following well-defined standards for format and for including appropriate tables, figures, and references [Produce]
2. Analyze written technical documentation for technical accuracy, concision, lack of ambiguity, and awareness of audience [Analyze]
3. Compose and deliver an audience-aware, accessible, and organized formal presentation [Compose]
4. Identify and describe qualities of effective communication (e.g., virtual, face-to-face, intragroup, shared documents) [Analyze]
5. Choose an appropriate way to communicate delicate ethical concerns [Assess]

2.15.18 SEP/Team Communication and Culture (CS Core: 1 hr, KA Core: 1 hr)

Topics:

Core

- Team collaboration (including tools) and conflict resolution.
- Accessibility and inclusivity requirements for addressing professional audiences.
- Cultural competence in communication including considering the impact of difference in natural language.
- Communicating to solve problems or make recommendations in the workplace, such as raising ethical concerns or addressing accessibility issues.

Learning Outcomes:

Core:

1. Plan interactions (e.g., virtual, face-to-face, shared documents) with others in ways that invite inclusive participation, model respectful consideration of others' contributions, and explicitly value diversity of ideas [Plan]
2. Understand how to communicate effectively and appropriately as a member of a team including conflict resolution techniques [Apply]
3. Discuss ways to influence performance and results in diverse and cross-cultural teams [Debate]
4. Assess personal strengths and weaknesses to work remotely as part of a team drawing from diverse backgrounds and experiences [Assess]

2.15.19 SEP/Environmental Footprint of Computing Systems (CS Core: 1 hr, KA Core: 1 hr)

The lifecycle of hardware and software carries significant environmental costs. This unit examines the environmental and social impacts of computing systems—from e-waste and carbon footprint to energy consumption in data centers and AI models—evaluating their effect on design decisions and global society.

Topics:

Core

- Environmental, social, and cultural impacts of implementation decisions (e.g., sustainability goals, algorithmic bias/outcomes, economic viability, and resource consumption).
- Local/regional/global social and environmental impacts of computing systems and their use (e.g., carbon footprints, resource usage, e-waste) due to hardware (e.g., e-waste, data centers, rare element and resource utilization, recycling) and software (e.g., cloud-based services, blockchain, AI model training and use). This includes everyday use of hardware (cheap hardware replaced frequently) and software (web-browsing, email, and other services with hidden/remote computational demands).

Learning Outcomes:

Core:

1. Assess the environmental impacts of a given project's deployment (e.g., energy consumption, contribution to e-waste, impact of manufacturing) [Assess]
2. Describe global social and environmental impacts of computer use and disposal [Describe]
3. Describe the environmental impacts of design choices within the field of computing that relate to algorithm design, operating system design, networking design, database design, etc [Describe]
4. Assess computing applications in respect to environmental issues (e.g., energy, pollution, resource usage, recycling and reuse, food management and production) [Assess]

2.15.20 SEP/Systemic Effects and Social Context (CS Core: 1 hr, KA Core: 1 hr)

Complex computing technologies such as generative AI, social media, and remote work generate systemic effects that are difficult to predict. This unit analyzes how software sustainability depends on surrounding social systems—users, organizations, markets, and policies—and how technological designs in turn transform those systems.

Topics:

Core

- Systemic effects of complex computing technologies and phenomena (e.g., generative AI, data centers, social media, offshoring, remote work).
- How the sustainability of software systems is interdependent with social systems, including the knowledge and skills of its users, organizational processes and policies, and its societal context (e.g., market forces, government policies).

Learning Outcomes:

Core:

1. List the sustainable effects of modern practices and activities (e.g., remote work, e-commerce, cryptocurrencies, AI models, data centers) [List]
2. Analyze the social and environmental impacts of new system designs [Analyze]

2.15.21 SEP/Sustainable Design and Pervasive Computing (CS Core: 1 hr, KA Core: 1 hr)

Topics:

Core

- Guidelines for sustainable design standards.
- Pervasive computing: Information processing that has been integrated into everyday objects and activities, such as smart energy systems, social networking, and feedback systems to promote sustainable behavior, transportation, environmental monitoring, citizen science and activism.

Learning Outcomes:
Core:

1. Identify ways to be a sustainable practitioner in a specific area or with a specific project [Analyze]
2. Design guidelines for sustainable IT design or deployment in areas such as smart energy systems, social networking, transportation, agriculture, supply-chain systems, environmental monitoring, and citizen activism [Design]

2.15.22 SEP/Foundations and Early History of Computing (CS Core: 1 hr, KA Core: 1 hr)

History is important because it provides a mechanism for understanding why our computing systems operate the way they do, the societal contexts in which current approaches arose, and how those continue to echo through the discipline today. Not only does computing affect society but vice-versa, resulting in a complex socio-technical context that is constantly changing, requiring the perspective of history to put the present, as well as possible futures, into appropriate perspective. It also informs decisions based on successes and failures of the past including harm done and how to not repeat them. The history of computing is often taught in context with foundational concepts, such as system fundamentals and software development fundamentals. A focus should be placed on those who, due to marginalization, have not historically featured as prominently as they should.
Topics:
Core

- The history of computing: hardware, software, and human/organizational.
- The role of history in the present including within different social contexts, and the relevance of this history on the future.
- Age I (Pre-digital): Ancient analog computing (Stonehenge, Antikythera mechanism, Salisbury Cathedral clock, etc.), human-calculated number tables, Euclid, Lovelace, Babbage, Gödel, Church, Turing, pre-electronic (electro-mechanical and mechanical) hardware.
- Age II (Early modern computing): ENIAC, UNIVAC, Bombes (Bletchley Park and codebreakers), computer companies (e.g., IBM), mainframes, etc.

Learning Outcomes:
Core:

1. Understand the relevance and impact of computing history on recent events, present context, and possible future outcomes, from more than one cultural perspective [Assess]
2. Discuss how perspectives held today have been shaped by history, and that alternative perspectives exist (e.g., fears of AI replacing human workers vs AI augmenting human work, various views on Moore's Law) [Debate]
3. Identify formative and consequential trends in the history of the computing field [Analyze]
4. Identify the contributions of pioneering individuals or organizations (research labs, computer companies, government offices) in the computing field [Analyze]

2.15.23 SEP/Modern History: PC, Internet and AI (CS Core: 1 hr, KA Core: 1 hr)

Topics:
Core

- Age III (PC era): PCs, modern computer hardware and software, Moore's Law.
- Age IV (Internet): Networking, internet architecture, browsers and their evolution, standards, born-on-the-internet companies, and services (e.g., Google, Amazon, Microsoft, etc.), distributed computing.
- Age V (Mobile & Cloud): Mobile computing and smartphones, cloud computing and models thereof (e.g., SaaS), remote servers, security and privacy, social media.
- Age VI (AI): Decision making systems, recommender systems, generative AI and other machine learning driven tools and technologies.

Learning Outcomes:
Core:

1. Discuss the historical context for important moments in history of computing, such as the move from vacuum tubes to transistors (TRADIC), early seminal operating systems (e.g., OS 360), Xerox PARC and the first Apple computer with a GUI, the creation of specific programming language paradigms, the first computer virus, the creation of the internet, the creation of the WWW, the dot com bubble, Y2K, the introduction of smartphones, etc [Debate]
2. Compare daily life before and after the advent of milestone developments (e.g., personal computers or the internet) [Compare]

2.15.24 SEP/Economic Models and Strategies (KA Core: 1 hr)

The economies of computing are important to those who develop and provide computing resources and services to others as well as society in general. They impact users of these resources and services, both professional and non-professional. Computing professionals have a duty to know the impact of these topics on their own roles and activities and how choices made will impact users and society.

Topics:

Core

- Economic models: regulated and unregulated, monopolies, network effects, and open market; knowledge and attention economies.
- Pricing and deployment strategies: planned obsolescence, subscriptions, freemium, software licensing, open-source, free software, adware.
- Impacts of differences in access to computing resources, and the effect of skilled labor supply and demand on the quality of computing products.

Learning Outcomes:

Core:

1. Summarize the social effects of economic models (e.g., the knowledge and attention economies) [Summarize]
2. Describe the differences and similarities of competing strategies (e.g., subscription vs freemium vs free) [Describe]
3. Discuss examples of digital divides [Debate]

2.15.25 SEP/Automation, AI and Economic Ethics (KA Core: 1 hr)

Topics:

Core

- Automation, AI, and their effects on job markets, developers, and users.
- Ethical concerns surrounding the attention economy and other economies of computing (e.g. informed consent, data collection, use of verbose legalese in user agreements).

Learning Outcomes:

Core:

1. Understand the effects of automation and AI on society [Assess]
2. Understand the ethical implications of computing economies that rely on attention and data [Assess]

2.15.26 SEP/Computer Crimes and Attacks (CS Core: 1 hr, KA Core: 1 hr)

While security policies, laws and computer crimes are important topics, it is essential they are viewed with the foundation of other social and professional knowledge units, such as Intellectual Property, Privacy and Civil Liberties, Social Context, and Professional Ethics. Computers, the internet, and artificial intelligence, perhaps more than any other technologies, have transformed society over the past 75 years. At the same time, these technologies have contributed to unprecedented threats to privacy; new categories of computer crime and antisocial behavior; major disruptions to organizations; and the large-scale concentration of risk in information systems.

Topics:

Core

- Computer crimes, legal redress for computer criminals and impact on victims and society.
- Social engineering, computing-enabled fraud, identity theft and recovery from these.

- Cyber terrorism, criminal hacking, and hacktivism.
- Malware, viruses, worms.
- Attacks on critical infrastructure such as electrical grids and pipelines.

Learning Outcomes:
Core:

1. List classic examples of computer crimes and social engineering incidents with societal impact [List]
2. Identify issues with laws that apply to computer crimes [Analyze]
3. Describe the motivation and ramifications of cyber terrorism, data theft, hacktivism, ransomware, and other attacks [Describe]
4. Examine the ethical and legal issues surrounding the misuse of access and various breaches of security [Examine]

2.15.27 SEP/Security Laws, Policies and Responsibility (CS Core: 1 hr, KA Core: 1 hr)

Topics:
Core

- Non-technical fundamentals of security (e.g., human engineering, policy, confidentiality).
- Benefits and challenges of existing and proposed computer crime laws.
- Security policies and the challenges of change and compliance.
- Responsibility for security throughout the computing life cycle.
- International and local laws and how they intersect.

Learning Outcomes:
Core:

1. Discuss the professional's role in security and the tradeoffs and challenges involved [Debate]
2. Investigate measures that can be taken by both individuals and organizations including governments to prevent or mitigate the undesirable effects of computer crimes [Investigate]
3. Design a company-wide security policy, which includes procedures for managing passwords and employee monitoring [Design]
4. Understand how legislation from one region may affect activities in another (e.g., how EU GDPR applies globally, when EU persons are involved) [Assess]

2.15.28 SEP/Identity and Diversity in Computing (CS Core: 1 hr, KA Core: 1 hr)

Identity—including race, gender, background, and ability—shapes both the experience of computing professionals and the products they create. This unit examines how identity interacts with computing technologies and environments, and the demonstrable benefits of diverse development teams, as well as the costs of a lack of diversity.

Topics:
Core

- How identity impacts and is impacted by computing technologies and environments (academic and professional).
- The benefits of diverse development teams and the impacts of teams that are not diverse.

Learning Outcomes:
Core:

1. Define and distinguish equity, equality, diversity, inclusion, and accessibility [Define]
2. Identify examples of the benefits that diverse teams can bring to software products, and how a lack of diversity has costs [Analyze]
3. Assess the impact of power and privilege in the computing profession as it relates to culture, industry, products, and society [Assess]

2.15.29 SEP/DEIA Historical and Cross-Cultural Context (CS Core: 1 hr, KA Core: 1 hr)

Current inequalities in computing have deep historical roots in systemic social mechanisms. This unit examines historical marginalization in computing, global infrastructure challenges, and cross-cultural differences in DEIA, exploring how systemic changes can improve equity in both familiar and unfamiliar contexts.

Topics:

Core

- Historic marginalization due to systemic social mechanisms, technological supremacy and global infrastructure challenges to diversity, equity, inclusion, and accessibility.
- Cross-cultural differences in, and needs for, diversity, equity, inclusion, and accessibility.

Learning Outcomes:

Core:

1. Compare the demographics of your institution's computer science and STEM majors to the overall institutional demographics. If they differ, identify factors that contribute to inequitable access, engagement, and achievement in computer science among marginalized groups. If they do not, assess why not [Compare]
2. Develop examples of systemic changes that could positively address diversity, equity, inclusion, and accessibility in a familiar context (i.e., in an introductory computing course) and an unfamiliar context and when these might be different, or the same [Create]
3. Compare the demographics of your institution to the overall community demographics. If they differ, identify factors that contribute to inequitable access, engagement, and achievement among marginalized groups. If they do not, assess why not [Compare]

2.15.30 SEP/Inclusion and Computing Practices (CS Core: 1 hr, KA Core: 1 hr)

Topics:

Core

- Inclusive language and charged terminology, and why their use matters.
- Inclusive behaviors and why they matter.
- How computing professionals can influence and impact diversity, equity, inclusion and accessibility, including but not only through the software they create.
- Experts and their practices that reflect the identities of the classroom and the world through practical DEIA principles.

Learning Outcomes:

Core:

1. Identify language, practices, and behaviors that may make someone feel included in a workplace and/or a team, and why is it relevant. Avoid charged terminology - see Words Matter (www.acm.org/diversity-inclusion/words-matter) - this includes identifying and accommodating users who are often excluded without thought and not considered at all [Analyze]
2. Demonstrate collegiality and respect when working with team members who do not share your identity. It is not enough to merely assign team projects. Faculty should prepare students for teamwork and monitor, mentor, and assess the effectiveness of their student teams throughout a project [Demonstrate]
3. Identify developers' assumptions and values embedded in hardware and software design, especially as they pertain to usability by diverse populations [Analyze]
4. Analyze the work of experts who reflect the identities of the classroom and the world [Analyze]

2.15.31 SEP/Accessible Design (CS Core: 1 hr, KA Core: 1 hr)

Topics:

Core

- Designing and developing technology with accessibility in mind.

Learning Outcomes:

Core:

1. Evaluate the accessibility of your classroom or lab. Evaluate the accessibility of your webpage. (See www.w3.org/WAI and www.w3.org/TR/WCAG21) [Evaluate]

2.16 System Fundamentals (SF)

A computer system is a set of hardware and software infrastructures upon which applications are constructed. Computer systems have become a pillar of people's daily life. As such, it is essential for students to learn knowledge about computer systems, grasp the skills to use and design these systems, and understand the fundamental rationale and principles in computer systems. It could equip students with the necessary competence for a career related to computer science.

In the curriculum of computer science, the study of computer systems typically spans multiple knowledge areas, including, but not limited to, operating systems, parallel and distributed systems, communications networks, computer architecture and organization, and software engineering. The System Fundamentals knowledge area, as suggested by its name, focuses on the fundamental concepts and design principles in computer systems that are shared by these courses within their respective cores. The goal of this knowledge area is to present an integrative view of these fundamental concepts and design principles in a unified albeit simplified fashion, providing a common foundation for the different specialized mechanisms and policies appropriate to the specific domain area. The fundamental concepts in this knowledge area include an overview of computer systems, basic concepts such as state and state transition, resource allocation and scheduling, and so on. Moreover, this knowledge area introduces basic design principles to improve the reliability, availability, efficiency, and security of computer systems.

Knowledge Area (KA)	CS Core	KA Core
⭐ 2.16.1 Hardware Components and Logic (Pg.. 123)	2	
⭐ 2.16.2 Layers, Abstractions and Scalability (Pg.. 124)	2	
⭐ 2.16.3 Digital Systems and Logic (Pg.. 124)	2	
⭐ 2.16.4 State Machines (Pg.. 124)	2	
⭐ 2.16.5 Sequential and Parallel Processing (Pg.. 125)	2	
⭐ 2.16.6 Resource Management (Pg.. 125)	1	1
⭐ 2.16.7 Latency, Cache and Memory Hierarchy (Pg.. 126)	2	2
⭐ 2.16.8 Virtualization and Isolation (Pg.. 126)	1	1
⭐ 2.16.9 Performance Metrics and Benchmarks (Pg.. 127)	1	1
⭐ 2.16.10 Analytical Performance Models (Pg.. 127)	1	1
⭐ 2.16.11 Events, Tools and Experimentation (Pg.. 127)	1	1
⭐ 2.16.12 System Reliability (Pg.. 128)	2	1
⭐ 2.16.13 System Security (Pg.. 128)	2	1
⭐ 2.16.14 System Design (Pg.. 129)	2	1
2.16.15 Society, Ethics, and the Profession (Pg.. 129)	Elective	

Table 2.16: List of KUs in the System Fundamentals area.

2.16.1 SF/Hardware Components and Logic (CS Core: 2 hrs)

Introduces the building blocks and basic components of a computer, hardware as a computational paradigm, and the concept of pipelining.

Topics:

Core

- Basic building blocks and components of a computer (gates, flip-flops, registers, interconnections; datapath + control + memory).
- Hardware as a computational paradigm: Fundamental logic building blocks; Logic expressions, minimization, sum of product forms Digital Logic and Digital Systems.
- Basic concept of pipelining, overlapped processing stages Functional Organization.

Learning Outcomes:

Core:

1. Describe the basic building blocks of computers and their role in the historical development of computer architecture [Describe]
2. Design a simple logic circuit using the fundamental building blocks of logic design to solve a simple problem (e.g., adder) [Design]

2.16.2 SF/Layers, Abstractions and Scalability (CS Core: 2 hrs)

Covers programming abstractions, the distinction between application and operating system services, the concept of scaling, and the layered organization of computer systems.

Topics:

Core

- Programming abstractions, interfaces, use of libraries Programs: Distribution, Programs: Execution Properties, GPU Programming, Programs: Implementation Mappings, Programs: Declarative Parallelism, Programs: Starting Activities.
- Distinction and interaction between application and OS services, remote procedure call Role and Purpose of Operating Systems.
- Basic concept of scaling: performance vs problem size.

Learning Outcomes:

Core:

1. Describe how computing systems are constructed of layers upon layers, based on separation of concerns, with well-defined interfaces, hiding details of low layers from the higher layers [Describe]
2. Describe that hardware, OS, VM, and application are additional layers of interpretation/processing [Describe]
3. Describe the mechanisms of how errors are detected, signaled back, and handled through the layers [Describe]
4. Construct a simple program (e.g., a TCP client/server) using methods of layering, error detection and recovery, and reflection of error status across layers [Create]
5. Identify bugs in a layered program by using tools for program tracing, single stepping, and debugging [Analyze]
6. Understand the concept of strong vs weak scaling, i.e., how performance is affected by the scale of the problem vs the scale of resources to solve the problem. This can be motivated by simple, real-world examples [Assess]

2.16.3 SF/Digital Systems and Logic (CS Core: 2 hrs)

Covers digital vs. analog systems, logic gates, Boolean expressions, clocks, and combinational and sequential logic, as the foundation for understanding computer systems operation.

Topics:

Core

- Digital vs Analog/Discrete vs Continuous Systems.
- Simple logic gates, logical expressions, Boolean logic simplification.
- Clocks, State, Sequencing.
- Combinational Logic, Sequential Logic, Registers, Memories Digital Logic and Digital Systems.

Learning Outcomes:

Core:

1. Describe the differences between digital and analog systems, and between discrete and continuous systems. Can give real-world examples of these systems [Describe]

2.16.4 SF/State Machines (CS Core: 2 hrs)

Introduces the concept of state and state transitions, finite state machines (FSMs), and their application to computers and network protocols as examples of state machines.

Topics:

Core

- State and state transition (e.g., starting state, final state, life cycle of states) Formal Languages and Automata.
- Finite state machines (e.g., NFA, DFA) Formal Languages and Automata.
- Computers and Network Protocols as examples of State Machines Fundamentals of Networks and Communications.

Learning Outcomes:

Core:

1. Describe computations as a system characterized by a known set of configurations with transitions from one unique configuration (state) to another (state) [Describe]
2. Describe the distinction between systems whose output is only a function of their input (stateless) and those with memory/history (stateful) [Describe]
3. Develop state machine descriptions for simple problem statement solutions (e.g., traffic light sequencing, pattern recognizers) [Create]
4. Describe a computer as a state machine that interprets machine instructions [Describe]
5. Explain how a program or network protocol can also be expressed as a state machine and that alternative representations for the same computation can exist [Explain]
6. Derive the time-series behavior of a state machine from its state machine representation (e.g., TCP connection management state machine) [Analyze]

2.16.5 SF/Sequential and Parallel Processing (CS Core: 2 hrs)

Covers sequential and parallel processing, including single-thread and multi-thread models at the application level, and the evaluation of performance of parallel program versions.

Topics:

Core

- Sequential vs parallel processing. Concurrency, Programs: Distribution, Programs: Execution Properties, GPU Programming, Programs: Implementation Mappings, Programs: Declarative Parallelism, Programs: Starting Activities.
- Application-level sequential processing: single thread Concurrency, Programs: Distribution, Programs: Execution Properties, GPU Programming, Programs: Implementation Mappings, Programs: Declarative Parallelism, Programs: Starting Activities.
- Simple application-level parallel processing: request level (web services/client-server/distributed), single thread per server, multiple threads with multiple servers, pipelining Concurrency, Programs: Distribution, Programs: Execution Properties, GPU Programming, Programs: Implementation Mappings, Programs: Declarative Parallelism, Programs: Starting Activities.

Learning Outcomes:

Core:

1. Write a simple sequential problem and a simple parallel version of the same program [Write]
2. Evaluate the performance of simple sequential and parallel versions of a program with different problem sizes and be able to describe the speed-ups achieved [Evaluate]
3. Describe on an execution timeline how parallelism events and operations can take place simultaneously (i.e., at the same time). Explain how work can be performed in less elapsed time if this can be exploited [Describe]

2.16.6 SF/Resource Management (CS Core: 1 hr, KA Core: 1 hr)

This knowledge unit covers different types of resources in computer systems and common resource allocation and scheduling algorithms.

Topics:

Core

- Different types of resources (e.g., processor share, memory, disk, net bandwidth) Fundamentals of Networks and Communications, File Systems API and Implementation, Memory Management, Scheduling.
- Common resource allocation/scheduling algorithms (e.g., first-come-first-serve, priority-based scheduling, fair scheduling, and preemptive scheduling) Scheduling.
- Advantages and disadvantages of common scheduling algorithms Scheduling.

Learning Outcomes:
Core:

1. Define how finite computer resources (e.g., processor share, memory, storage, and network bandwidth) are managed by their careful allocation to existing entities [Define]
2. Describe how common resource allocation/scheduling algorithms work [Describe]
3. Develop common scheduling algorithms and evaluate their performances [Create]
4. Describe the pros and cons of common scheduling algorithms [Describe]

2.16.7 SF/Latency, Cache and Memory Hierarchy (CS Core: 2 hrs, KA Core: 2 hrs)

Covers latencies in computer systems, spatial and temporal locality in caches, cache coherency, and the processor memory hierarchy with its average memory access time formula.
Topics:
Core

- Latencies in computer systems: 1. Speed of light and computers (one foot per nanosecond vs one GHz clocks) Functional Organization 2. Memory vs disk latencies vs across-the-network memory Memory Hierarchy, Performance and Energy Efficiency
- Caches and the effects of spatial and temporal locality on performance in processors and systems Memory Hierarchy, Performance and Energy Efficiency, Memory Management.
- Caches and cache coherency in databases, operating systems, distributed systems, and computer architecture Memory Hierarchy, DBMS Internals, Memory Management.
- Introduction to the processor memory hierarchy Memory Hierarchy, Performance and Energy Efficiency.
- The formula for average memory access time Memory Hierarchy.

Learning Outcomes:
Core:

1. Describe the breakdown of the latency of computer systems in terms of memory, disk, and network [Describe]
2. Explain the importance of locality in determining system performance [Explain]
3. Calculate average memory access time and describe the tradeoffs in memory hierarchy performance in terms of capacity, miss/hit rate, and access time [Calculate]

2.16.8 SF/Virtualization and Isolation (CS Core: 1 hr, KA Core: 1 hr)

Covers the rationale for isolation and virtualization for protection and predictable performance, levels of indirection via virtual memory, and methods for implementing virtual memory and virtual machines.
Topics:
Core

- Rationale of virtualization and isolation: protection and predictable performance Virtualization.
- Levels of indirection, illustrated by virtual memory for managing physical memory resources Virtualization.
- Methods for implementing virtual memory and virtual machines Virtualization.

Learning Outcomes:
Core:

1. Explain why it is important to isolate and protect the execution of individual programs and environments that share common underlying resources [Explain]
2. Describe how the concept of indirection can create the illusion of a dedicated machine and its resources even when physically shared among multiple programs and environments [Describe]
3. Evaluate the performance of two application instances running on separate virtual machines and determine the effect of performance isolation [Evaluate]

2.16.9 SF/Performance Metrics and Benchmarks (CS Core: 1 hr, KA Core: 1 hr)

Covers performance figures of merit, representative workloads and benchmarks, and collection and analysis methods for the quantitative evaluation of computer system performance.

Topics:

Core

- Performance figures of merit Performance and Energy Efficiency, Evaluation.
- Workloads and representative benchmarks, and methods of collecting and analyzing performance figures of merit Performance and Energy Efficiency, Evaluation.

Learning Outcomes:

Core:

1. Explain how the components of system architecture contribute to improving its performance [Explain]
2. Explain the circumstances in which a given figure of a system performance metric is useful [Explain]
3. Explain the usage and inadequacies of benchmarks as a measure of system performance [Explain]

2.16.10 SF/Analytical Performance Models (CS Core: 1 hr, KA Core: 1 hr)

Covers the CPI equation as a design tool, Amdahl's Law and its limitations, order of magnitude analysis (Big O), critical path analysis, and analytical tools for quantitative evaluation.

Topics:

Core

- CPI (Cycles per Instruction) equation as a tool for understanding tradeoffs in the design of instruction sets, processor pipelines, and memory system organizations Performance and Energy Efficiency, Evaluation.
- Amdahl's Law: the part of the computation that cannot be sped up limits the effect of the parts that can Performance and Energy Efficiency, Evaluation.
- Order of magnitude analysis (Big O notation) Asymptotic Notation and Complexity Classes, Complexity Analysis: Recursion, Amortization, and Tight Bounds, Complexity Analysis Framework, Computational Complexity Theory.
- Analysis of slow and fast paths of a system Functional Organization.
- Analytical tools to guide quantitative evaluation.

Learning Outcomes:

Core:

1. Describe Amdahl's law and discuss its limitations [Describe]
2. Apply limit studies or simple calculations to produce order-of-magnitude estimates for a given performance metric in a given context [Apply]

2.16.11 SF/Events, Tools and Experimentation (CS Core: 1 hr, KA Core: 1 hr)

Covers events and their effect on performance (stalls, cache misses, page faults), understanding layered systems as evaluation platforms, microbenchmark pitfalls, and tools and experiments for measuring system performance.

Topics:

Core

- Events on their effect on performance (e.g., instruction stalls, cache misses, page faults) Functional Organization, Memory Management.
- Understanding layered systems, workloads, and platforms, their implications for performance, and the challenges they represent for evaluation.
- Microbenchmark pitfalls.

Learning Outcomes:

Core:

1. Apply software tools to profile and measure program performance [Apply]
2. Design and conduct a performance-oriented experiment of a common system (e.g., an OS and Spark) [Design]

3. Design a performance experiment on a layered system to determine the effect of a system parameter on system performance [Design]

2.16.12 SF/System Reliability (CS Core: 2 hrs, KA Core: 1 hr)

This knowledge unit covers the distinction between bugs, faults, and failures; reliability vs availability; and reliability through redundancy including check and retry, redundant encoding, and duplication.

Topics:

Core

- Distinction between bugs, faults, and failures Coordination, Reliability Concepts, Reliability Engineering.
- Reliability vs availability.
- Reliability through redundancy: 1. check and retry Reliability Support, Fault tolerance 2. redundant encoding (error correction codes, CRC, FEC, RAID) Memory Hierarchy, Distributed Databases/Cloud Computing, Reliability Support 3. duplication/mirroring/replicas Distributed Databases/Cloud Computing
- Other approaches to reliability (e.g., journaling) Reliability Support, Fault tolerance, Reliability Concepts, Reliability Engineering.

Learning Outcomes:

Core:

1. Explain the distinction between program errors, system errors, and hardware faults (e.g., corrupted memory) and exceptions (e.g., attempt to divide by zero) [Explain]
2. Articulate the distinction between detecting, handling, and recovering from faults and the methods for their implementation [Articulate]
3. Describe the role of error correction codes in providing error checking and correction techniques in memories, storage, and networks [Describe]
4. Apply simple algorithms for exploiting redundant information for the purposes of data correction [Apply]
5. Compare different error detection and correction methods for their data overhead, implementation complexity, and relative execution time for encoding, detecting, and correcting errors [Compare]

2.16.13 SF/System Security (CS Core: 2 hrs, KA Core: 1 hr)

This knowledge unit covers common system security issues and countermeasures, including cryptography and security architecture, as well as representative countermeasure systems such as intrusion detection systems and antivirus systems.

Topics:

Core

- Common system security issues (e.g., viruses, denial-of-service attacks, and eavesdropping) Network Security, Protection and Safety, Security Design and Controls Engineering, Threat Analysis and Security Engineering, Trusted Computing and Privacy Engineering, AI and Hardware Security, Access Control and Applications, Security Mindset and Principles.
- Countermeasures Network Security, Principles of Operating System, Protection and Safety: 1. Cryptography Protocols, Applications and Advanced Cryptography, Cryptographic Algorithms, Cryptography Fundamentals 2. Security architecture Security Design and Controls Engineering, Threat Analysis and Security Engineering, Trusted Computing and Privacy Engineering
- Representative countermeasure systems: 1. Intrusion detection systems, firewalls Network Security 2. Antivirus systems

Learning Outcomes:

Core:

1. Describe some common system security issues and give examples [Describe]
2. Describe some countermeasures against system security issues [Describe]
3. Describe representative countermeasure systems [Describe]

2.16.14 SF/System Design (CS Core: 2 hrs, KA Core: 1 hr)

This knowledge unit covers common criteria of system design (e.g., liveness, safety, robustness, scalability, and security) and the designs of representative systems.

Topics:

Core

- Common criteria of system design (e.g., liveness, safety, robustness, scalability, and security) Evaluation.
- Designs of representative systems (e.g., Apache web server, Spark, and Linux).

Learning Outcomes:

Core:

1. Describe common criteria of system design [Describe]
2. Given the functionality requirements of a system and its key design criteria, provide a high-level design of this system [Generate]
3. Describe the design of some representative systems [Describe]

2.16.15 SF/Society, Ethics, and the Profession

This knowledge unit covers intellectual property rights of computer systems, common software licenses, and computer crimes.

Topics:

Core

- Intellectual property rights of computer systems Open Source, Piracy and DRM, Digital IP Rights and Copyright, Intellectual Property Foundations, Licensing, Plagiarism and Authorship.
- Common software licenses Open Source, Piracy and DRM, Digital IP Rights and Copyright, Intellectual Property Foundations, Licensing, Plagiarism and Authorship.
- Computer crimes Computer Crimes and Attacks, Security Laws, Policies and Responsibility.

Learning Outcomes:

Core:

1. Describe the intellectual property rights of computer systems [Describe]
2. List representative software licenses and compare their differences [List]
3. List representative computer crimes [List]

2.17 Specialized Platform Development (SPD)

The Specialized Platform Development (SPD) knowledge area refers to attributes involving the creation of software targeting non-traditional hardware platforms. Developing for each specialized platform, for example, robots, mobile systems, web-based systems, and embedded systems, typically involves unique considerations.

Societal and industry needs have created a high demand for developers on specialized platforms, such as mobile applications, web platforms, robotic platforms, and embedded systems. Some unique professional abilities relevant to this knowledge area include the following:

- Creating applications that provide a consistent user experience across various devices, screen sizes, and operating systems.
- Developing application programming interfaces (APIs) to support the functionality of each specialized platform.
- Managing challenges related to resource constraints such as computation, memory, storage, and networking and communication.
- Applying cross-cutting concerns such as optimization, security, better development practices, etc.

Knowledge Area (KA)	CS Core	KA Core
⭐ 2.17.1 Common Aspects: Platforms, APIs and Constraints (Pg.. 130)	3	2
⭐ 2.17.2 Common Aspects: Platform Languages and Web Programming Patterns (Pg.. 131)	2	1
⭐ 2.17.3 Frontend, Frameworks and Web Standards (Pg.. 131)		3
⭐ 2.17.4 Cloud Architecture and Data Management (Pg.. 131)		3
⭐ 2.17.5 Mobile Platforms: Development Fundamentals (Pg.. 132)		2
⭐ 2.17.6 Mobile Platforms: Architecture and Constraints (Pg.. 132)		2
⭐ 2.17.7 Mobile Platforms: Capabilities and Asynchronous Computing (Pg.. 132)		1
⭐ 2.17.8 Robot Platforms (Pg.. 132)		4
⭐ 2.17.9 Embedded Characteristics and Architectures (Pg.. 133)		1
⭐ 2.17.10 Systems, Programming and Sensors (Pg.. 133)		2
⭐ 2.17.11 Real-Time, Memory and Verification (Pg.. 134)		3
⭐ 2.17.12 Platform History, Types and Characteristics (Pg.. 134)		1
⭐ 2.17.13 Sensors, Controllers and eSports Ecosystems (Pg.. 135)		1
⭐ 2.17.14 Architectures and Rendering (Pg.. 135)		1
⭐ 2.17.15 Algorithms, Media and Virtual Worlds (Pg.. 135)		1
⭐ 2.17.16 Languages, APIs and Development Tools (Pg.. 135)		1
⭐ 2.17.17 Game Engines (Pg.. 136)		1
⭐ 2.17.18 Game Design Foundations (Pg.. 136)		1
⭐ 2.17.19 Genres, Audiences and Social Reach (Pg.. 136)		1
⭐ 2.17.20 Interactive Data and Analytics Platforms (Pg.. 137)		2
⭐ 2.17.21 AI Programming and Quantum Platforms (Pg.. 137)		1
2.17.22 SPD-SEP/Mobile (Pg.. 138)	Elective	
2.17.23 SPD-SEP/Web (Pg.. 138)	Elective	
2.17.24 Intellectual Property in Games (Pg.. 138)	Elective	
2.17.25 Game Interface and Accessibility (Pg.. 139)	Elective	
2.17.26 Health, Safety and Sustainability (Pg.. 139)	Elective	
2.17.27 Representation, Inclusion and Equality (Pg.. 139)	Elective	
2.17.28 SPD-SEP/Robotics (Pg.. 140)	Elective	
2.17.29 SPD-SEP/Interactive (Pg.. 140)	Elective	

Table 2.17: List of KUs in the Specialized Platform Development area.

2.17.1 SPD/Common Aspects: Platforms, APIs and Constraints (CS Core: 3 hrs, KA Core: 2 hrs)

Topics:
Core

- Overview of development platforms (i.e., web, mobile, game, robotics, embedded, and interactive). 1. Input/sensors/control devices/haptic devices 2. Resource constraints (a) Computational (b) Data storage (c) Memory (d) Communication 3. Requirements - security, uptime availability, fault tolerance Reliability Concepts, Reliability Engineering, Security Design and Controls Engineering, Threat Analysis and Security Engineering, Trusted Computing and Privacy Engineering 4. Output/actuators/haptic devices
- Programming via platform-specific Application Programming Interface (API) vs traditional application construction
- Programming under platform constraints and requirements (e.g., available development tools, development, security considerations) AI and Hardware Security, Access Control and Applications, Security Mindset and Principles

Learning Outcomes:
Core:

1. List the constraints of mobile programming [List]

2.17.2 SPD/Common Aspects: Platform Languages and Web Programming Patterns (CS Core: 2 hrs, KA Core: 1 hr)

Topics:
Core

- Overview of platform Languages (e.g., Python, Swift, Lua, Kotlin)
- Techniques for learning and mastering a platform-specific programming language

Learning Outcomes:
Core:

1. List the characteristics of scripting languages [List]
2. Describe the three-tier model of web programming [Describe]
3. Describe how the state is maintained in web programming [Describe]

2.17.3 SPD/Frontend, Frameworks and Web Standards (KA Core: 3 hrs)

Covers web programming languages, frameworks, the SaaS model, web standards (DOM, accessibility), and security and privacy considerations.
Topics:
Core

- Web programming languages (e.g., HTML5, JavaScript, Typescript, CSS)
- Web platforms, frameworks, or meta-frameworks 1. Cloud services 2. API, Web Components
- Software as a Service (SaaS)
- Web standards such as document object model, accessibility Accessibility and Inclusive Design
- Security and Privacy Considerations Computer Crimes and Attacks, Security Laws, Policies and Responsibility

Learning Outcomes:
Core:

1. Compare and contrast web programming with general-purpose programming [Compare]
2. Describe the differences between Software-as-a-Service (SaaS) and traditional software products [Describe]
3. Discuss how web standards impact software development [Debate]
4. Evaluate an existing web application against current web standards [Evaluate]

2.17.4 SPD/Cloud Architecture and Data Management (KA Core: 3 hrs)

Covers requirements analysis for web applications, cloud computing services, data management, architectures (microservices, serverless, event-driven), and storage solutions.
Topics:
Non Core

- Analyzing requirements for web applications
- Computing services NoSQL Systems 1. Cloud Hosting 2. Scalability (e.g., Autoscaling, Clusters) 3. Cost estimation for services
- Data management Core Database System Concepts 1. Data residency: where the data is located and what paths can be taken to access it 2. Data integrity: guaranteeing data is accessible and that data is deleted when required
- Architecture 1. Monoliths vs Microservices 2. Micro-frontends 3. Event-Driven vs RESTful architectures: advantages and disadvantages 4. Serverless, cloud computing on demand
- Storage solutions NoSQL Systems, Relational Databases 1. Relational Databases 2. NoSQL databases

Learning Outcomes:
Core:

1. Design and implement a web-based application using a microservice architecture design [Design]
2. Describe the constraints, such as hosting, services, and scalability, related to web platforms [Describe]

2.17.5 SPD/Mobile Platforms: Development Fundamentals (KA Core: 2 hrs)

Topics:
Core

- Development with 1. Mobile programming languages 2. Mobile programming environments
- Mobile platform constraints 1. User interface design Understanding the User: Individual goals and interactions with others 2. Security
- Access 1. Accessing data through APIs Query Construction 2. Designing API endpoints for mobile apps: pitfalls and design considerations 3. Network and web interfaces Data Modeling, Fundamentals of Networks and Communications

Learning Outcomes:
Core:

1. Compare mobile programming with general-purpose programming [Compare]
2. Evaluate the pros and cons of native and cross-platform mobile application development [Evaluate]

2.17.6 SPD/Mobile Platforms: Architecture and Constraints (KA Core: 2 hrs)

Topics:
Non Core

- Development 1. Native versus cross-platform development 2. Software design/architecture patterns for mobile applications API Design and Modeling, Principles and Software Architecture, Software Design Quality and Evaluation
- Mobile platform constraints 1. Responsive user interface design Accessibility and Inclusive Design 2. Heterogeneity and mobility of devices 3. Differences in user experiences (e.g., between mobile and web-based applications) 4. Power and performance tradeoff
- Specification and testing Software Development Practices, Performance Testing and Benchmarking, Testing Tools and Analysis, Verification and validation Concepts, Test Planning and Types, Advanced Testing Practices

Learning Outcomes:
NonCore:

1. Design the architecture of a mobile application applying appropriate software design patterns, justifying the choice between native and cross-platform development [Design]
2. Evaluate the constraints of a given mobile platform (responsive UI, device heterogeneity, energy-performance tradeoffs) when specifying and testing an application [Evaluate]

2.17.7 SPD/Mobile Platforms: Capabilities and Asynchronous Computing (KA Core: 1 hr)

Topics:
Non Core

- Mobile computing affordances 1. Location-aware applications 2. Sensor-driven computing (e.g., gyroscope, accelerometer, health data from a watch) 3. Telephony and instant messaging 4. Augmented reality Immersion
- Asynchronous computing Algorithms 1. Difference from traditional synchronous programming 2. Handling success via callbacks 3. Handling errors asynchronously 4. Testing asynchronous code and typical problems in testing

Learning Outcomes:
Core:

1. Develop a location-aware mobile application with data API integration [Create]
2. Build a sensor-driven mobile application capable of logging data on a remote server [Create]
3. Create a communication app incorporating telephony and instant messaging [Create]

2.17.8 SPD/Robot Platforms (KA Core: 4 hrs)

Topics:
Core

- Types of robotic platforms and devices Robotics
- Sensors, embedded computation, and effectors (actuators) Tangible/Physical Computing
- Robot-specific languages and libraries Robotics
- Robotic software architecture (e.g., using the Robot Operating System (ROS))
- Robotic platform constraints and design considerations Robotics
- Interconnections with physical or simulated systems Simulation, Tangible/Physical Computing
- Robotic Algorithms Robotics, Computer Animation 1. Forward kinematics 2. Inverse kinematics 3. Dynamics 4. Navigation and path planning 5. Grasping and manipulation
- Safety and interaction considerations Digital Equity and Accessibility, Social Impact and AI, Ethics Codes and Legal Accountability, Professional Responsibility and Public Policy, Professional Values, Nature and Continuing Development, Ethical Dissent and Cultural Context, Workplace Ethics

Learning Outcomes:

Core:

1. Design and implement an application on a given robotic platform [Design]
2. Integrate an Arduino-based robot kit and program it to navigate a maze [Integrate]
3. Compare robot-specific languages and techniques with those used for general-purpose software development [Compare]
4. Explain the rationale behind the design of the robotic platform and its interconnections with physical or simulated systems [Explain]
5. Given a high-level application, design a robot software architecture using ROS specifying all components and interconnections (ROS topics) to accomplish that application [Design]
6. Discuss the constraints a given robotic platform imposes on developers [Debate]

2.17.9 SPD/Embedded Characteristics and Architectures (KA Core: 1 hr)

Covers the unique characteristics of embedded systems compared to other systems, APIs for custom architectures (GPU, FPGA), and the evaluation of architectures according to requirements.

Topics:

Core

- Introduction to the unique characteristics of embedded systems 1. Real-time vs soft real-time and non-real-time systems 2. Resource constraints, such as memory profiles and deadlines Memory Hierarchy
- API for custom architectures 1. GPU technology Heterogeneous Architectures, Shading and Advanced Rendering 2. Field Programmable Gate Arrays (FPGA) Digital Logic and Digital Systems 3. Cross-platform systems

Learning Outcomes:

Core:

1. Describe the unique characteristics of embedded systems versus other systems [Describe]
2. Identify different types of embedded architectures [Analyze]
3. Evaluate which architecture is best for a given set of requirements [Evaluate]

2.17.10 SPD/Systems, Programming and Sensors (KA Core: 2 hrs)

Covers embedded systems with microcontrollers, interrupts and energy efficiency, embedded programming, and interfacing with sensors and actuators.

Topics:

Core

- Embedded Systems 1. Microcontrollers 2. Interrupts and feedback 3. Interrupt handlers in high-level languages Hardware Components and Logic, Layers, Abstractions and Scalability 4. Hard and soft interrupts and trap-exits Principles of Operating System 5. Interacting with hardware, actuators, and sensors 6. Energy efficiency 7. Loosely timed coding and synchronization 8. Software adapters
- Embedded programming
- Sensors and actuators

Learning Outcomes:
Core:
1. Interface with sensors/actuators [Program]
2. Debug a problem with an existing embedded platform [Analyze]
3. Design and develop software to interact with and control hardware [Design]

2.17.11 SPD/Real-Time, Memory and Verification (KA Core: 3 hrs)

Covers hard and soft real-time systems, real-time and memory resource management, safety considerations, analysis and verification, and embedded application design.
Topics:
Core
- Hard real-time systems vs soft real-time systems Real-time and Embedded Systems 1. Timeliness 2. Time synchronization/scheduling 3. Prioritization 4. Latency 5. Compute jitter
- Real-time resource management
- Memory management 1. Mapping programming construct (variable) to a memory location Memory Hierarchy 2. Shared memory Memory Management 3. Manual memory management. 4. Garbage collection Language Translation and Execution
- Safety considerations and safety analysis Digital Equity and Accessibility, Social Impact and AI, Ethics Codes and Legal Accountability, Professional Responsibility and Public Policy, Professional Values, Nature and Continuing Development, Ethical Dissent and Cultural Context, Workplace Ethics
- Analysis and verification
- Application design

Learning Outcomes:
Core:
1. Design and implement a small embedded system for a given platform (e.g., a smart alarm clock or a drone) [Design]
2. Design methods for real-time systems [Design]
3. Evaluate real-time scheduling and schedulability analysis [Evaluate]
4. Evaluate formal specification and verification of timing constraints and properties [Evaluate]

2.17.12 SPD/Platform History, Types and Characteristics (KA Core: 1 hr)

Covers the historical evolution of game platforms, typical platform types, and their technical characteristics and constraints.
Topics:
Core
- Evolution of Game Platforms (e.g., Brown Box to Metaverse and beyond; improvement in computing architectures (CPU and GPU); platform convergence and mobility) Digital Logic and Digital Systems
- Typical Game Platforms (e.g., Personal Computer; Home Console; Handheld Console; Arcade Machine; Interactive Television; Mobile Phone; Tablet; Integrated Head-Mounted Display; Immersive Installations and Simulators; Internet of Things enabled Devices; CAVE Systems; Web Browsers; Cloud-based Streaming Systems)
- Characteristics and Constraints of Different Game Platforms (e.g., features: local storage, internetworking, peripherals; run-time performance: GPU/CPU frequency, number of cores; chipsets: physics processing units, vector co-processors; expansion bandwidth (PCIe); network throughput (Ethernet); memory types and capacities (DDR/GDDR); maximum stack depth; power consumption; thermal design; Endian)

Learning Outcomes:
Core:
1. Describe the key constraints a specific game platform will likely impose on developers [Describe]
2. Compare the interfaces of different game platforms, highlighting their respective implications for human-computer interaction [Compare]

2.17.13 SPD/Sensors, Controllers and eSports Ecosystems (KA Core: 1 hr)

Covers typical sensors, controllers and actuators in game platforms, and eSports ecosystems including events, streamed media and professional play.

Topics:

Core

- Typical Sensors, Controllers, and Actuators (e.g., distinctive control system designs - peripherals (mouse, keypad, joystick), game controllers, wearables, interactive surfaces; electronics and bespoke hardware; computer vision, inside-out tracking, and outside-in tracking; IoT-enabled electronics and I/O) Interaction
- eSports Ecosystems (e.g., evolution of gameplay across platforms; games and eSports; game events such as LAN/arcade tournaments and international events such as the Olympic eSports Series; streamed media and spectatorship; multimedia technologies and broadcast management; professional play; data and machine learning for coaching and training)

Learning Outcomes:

Core:

1. Explain how eSports are streamed to large audiences over the internet [Explain]

2.17.14 SPD/Architectures and Rendering (KA Core: 1 hr)

Covers CPU and GPU architectures for games and pipelines for graphical rendering and physical simulation.

Topics:

Core

- CPU and GPU Architectures (e.g., Flynn's taxonomy; parallelization; instruction sets; standard components - graphics compute array, graphics memory controller, video graphics array basic input/output system; bus interface; power management unit; video processing unit; display interface) Heterogeneous Architectures
- Pipelines for Graphical Rendering and Physical Simulation (e.g., tile-based, immediate-mode) Applied Rendering and Techniques

Learning Outcomes:

Core:

1. Recall the characteristics of common general-purpose graphics processing architectures [Recall]
2. Identify the key stages of the immediate-mode rendering pipeline [Analyze]

2.17.15 SPD/Algorithms, Media and Virtual Worlds (KA Core: 1 hr)

Covers common algorithms, data structures and mathematical functions in games, and media representations in virtual worlds.

Topics:

Core

- Common Contexts for Algorithms, Data Structures, and Mathematical Functions (e.g., game loops; spatial partitioning, viewport culling, and level of detail; collision detection and resolution; physical simulation; behavior for intelligent agents; procedural content generation) Foundational Data Structures, Foundational Algorithms, Advanced Algorithms, Discrete Mathematics
- Media Representations in Virtual Worlds (e.g., I/O and computation techniques for virtual worlds: audio; music; sprites; models and textures; text; dialogue; multimedia (e.g., olfaction, tactile)) Fundamental Concepts of Graphics and Interactive Techniques

Learning Outcomes:

Core:

1. Translate complex mathematical functions into performant source code [Translate]
2. Optimize a function for processing collision detection in a simulated environment [Create]

2.17.16 SPD/Languages, APIs and Development Tools (KA Core: 1 hr)

Covers programming and shading languages, graphics APIs and libraries, and development tools and environments for games.

Topics:
Core

- Game Development Tools and Techniques 1. Programming Languages (e.g., C++; C#; Lua; Python; JavaScript) 2. Shader Languages (e.g., HLSL, GLSL; Shader Graph) 3. Graphics Libraries and APIs (e.g., DirectX; SDL; OpenGL; Metal; Vulkan; WebGL). Applied Rendering and Techniques, System Design 4. Common Development Tools and Environments (e.g., IDEs; Debuggers; Profilers; Version Control Systems including those handling binary assets; Development Kits and Production/Consumer Kits; Emulators) Software Development Practices, Development Environment and IDE, Version Control and CI/CD

Learning Outcomes:
Core:

1. Use an industry-standard graphics API to render a 3D model in a virtual scene [Use]
2. Modify a shader to change a visual effect according to stated requirements [Create]
3. Recommend an appropriate set of development tools and techniques for implementing a game of a particular genre for a given platform [Recommend]

2.17.17 SPD/Game Engines (KA Core: 1 hr)

Covers open-source game engines and production techniques, quality assurance, optimization, porting and internationalization.

Topics:
Core

- Game Engines 1. Open Game Engines (e.g., Unreal; Unity; Godot; CryEngine; Phyre; Source 2; Pygame and Ren'Py; Phaser; Twine; Spring RTS) 2. Techniques (e.g., Ideation, Prototyping, Iterative Design and Implementation, Compiling Executable Builds, Development Operations and Quality Assurance - Play Testing and Technical Testing, Profiling; Optimization, Porting; Internationalization and Localization, Networking) Performance and Energy Efficiency, Requirements Engineering, Requirements Evolution and Estimation

Learning Outcomes:
Core:

1. Implement a game for a particular platform according to the specification [Implement]
2. Assess a game's run-time and memory performance using an industry-standard tool and development environment [Assess]
3. Discuss the key challenges in making a digital game that is cross-platform compatible [Debate]

2.17.18 SPD/Game Design Foundations (KA Core: 1 hr)

Covers game design vocabulary and models, design thinking and user-centred experience design.

Topics:
Core

- Game Design Vocabulary (e.g., game definitions; mechanics-dynamics-aesthetics model; industry terminology; experience design; models of experience and emotion)
- Design Thinking and User-Centered Experience Design (e.g., methods of designing games; iteration, incrementing, and the double-diamond; phases of pre- and post-production; quality assurance, including alpha and beta testing; stakeholder and customer involvement; community management) API Design and Modeling, Principles and Software Architecture, Software Design Quality and Evaluation

Learning Outcomes:
Core:

1. Express how game developers can enhance the accessibility of a game interface [Demonstrate]

2.17.19 SPD/Genres, Audiences and Social Reach (KA Core: 1 hr)

Covers game genres, audiences and player taxonomies, and the proliferation of game technologies to domains beyond entertainment.

Topics:
Core
- Game Genres (e.g., adventure; walking simulator; first-person shooter; real-time strategy; multiplayer online battle arena (MOBA); role-playing game (rpg))
- Audiences and Player Taxonomies (e.g., people who play games; diversity and broadening participation; pleasures, player types, and preferences; Bartle, Yee) Understanding the User: Individual goals and interactions with others
- Proliferation of Digital Game Technologies to Domains Beyond Entertainment (e.g., Education and Training; Serious Games; Virtual Production; eSports; Gamification; Immersive Experience Design; Creative Industry Practice; Artistic Practice; Procedural Rhetoric) AI Applications and Societal Impact, Additional AI Applications and Societal Impact

Learning Outcomes:
Core:
1. Create novel forms of gameplay using frontier game platforms [Create]

2.17.20 SPD/Interactive Data and Analytics Platforms (KA Core: 2 hrs)

Covers interactive data analysis platforms such as Jupyter notebooks, Google Colab, R, cloud SQL platforms, and design tools with low-latency feedback loops.
Topics:
Non Core
- Computational Notebook Platforms (e.g., Jupyter notebooks; Google Colab; R; SPSS; Observable)
- Cloud SQL and Data Analysis Platforms (e.g., BigQuery) Query Construction 1. Apache Spark 2. Data Visualizations Visualization
- Interactive Presentations Backed by Data
- Design Tools Requiring Low-Latency Feedback Loops 1. Rendering tools 2. Graphic design tools

Learning Outcomes:
NonCore:
1. Analyze large datasets interactively [Analyze]
2. Create a backing track for a musical performance, such as live coding [Create]
3. Create compelling computational notebooks that construct a narrative for a given journalistic goal/story [Create]
4. Implement interactive code that uses a dataset and generates exploratory graphics [Implement]
5. Contextualize the attributes of different data analysis styles, such as interactive vs engineered pipeline [Contextualize]
6. Write a program using a notebook computing platform (e.g., searching, sorting, or graph manipulation) [Write]

2.17.21 SPD/AI Programming and Quantum Platforms (KA Core: 1 hr)

Covers prompt programming with large language models (LLMs) and generative AI, and quantum platforms for programming quantum logic operators and available quantum services.
Topics:
Non Core
- Prompt programming 1. Generative AI (e.g., OpenAI's ChatGPT, OpenAI's Codex, GitHub's Copilot) and LLMs are accessed/interacted
- Quantum Platforms Quantum Architectures 1. Program quantum logic operators in quantum machines. 2. Use API for available quantum services 3. Signal analysis/Fourier analysis/Signal processing (for music composition, audio/RF analysis) Image Processing

Learning Outcomes:
NonCore:
1. Create a program that performs a task using LLM systems [Create]
2. Contrast a program developed by an AI platform and by a human [Contrast]
3. Implement a system that interacts with a human without using a screen [Implement]

4. Demonstrate a quantum gate outcome using a quantum platform [Demonstrate]

2.17.22 SPD/SPD-SEP/Mobile

Topics:
Non Core
- Privacy and data protection
- Accessibility in mobile design
- Security and cybersecurity
- Social impacts of mobile technology
- Ethical use of AI and algorithms

Learning Outcomes:
NonCore:
1. Understand and uphold ethical responsibilities for safeguarding user privacy and data protection in mobile applications [Assess]
2. Design mobile applications with accessibility in mind, ensuring effective use by people with disabilities [Design]
3. Demonstrate proficiency in secure coding practices to mitigate risks associated with various security threats in mobile development [Demonstrate]
4. Analyze the broader social impacts of mobile technology, including its influence on communication patterns, relationships, and mental health [Analyze]
5. Comprehend the ethical considerations of using AI in mobile applications, ensuring unbiased and fair algorithms [Explain]

2.17.23 SPD/SPD-SEP/Web

Topics:
Non Core
- Privacy concerns with mobile apps
- Designing for inclusivity and accessibility
- Ethical use of AI in mobile apps
- Sustainable app development and server hosting
- Avoiding spam or intrusive notifications
- Addressing cyberbullying and harassment
- Promoting positive online communities
- Monetization and advertising
- Ethical use of gamification

Learning Outcomes:
NonCore:
1. Understand how mobile computing impacts communications and the flow of information within society [Analyze]
2. Design mobile apps that have made daily tasks easier/faster [Design]
3. Recognize how the ubiquity of mobile computing has affected work-life balance [Recognize]
4. Understand how mobile computing impacts health monitoring and healthcare services [Analyze]
5. Define how mobile apps are used to educate about and help achieve UN sustainability goals [Define]

2.17.24 SPD/Intellectual Property in Games

Covers intellectual property rights in the creative industries, including copyright, trademarks, licensing and enforcement.
Topics:
Non Core
- Intellectual Property Rights in Creative Industries 1. Intellectual Property Ownership: copyright, trademark; design right, patent, trade secret, civil versus criminal law; international

agreements; procedural content generation and the implications of generative artificial intelligence 2. Licensing: Usage and fair usage exceptions; open-source license agreements; proprietary and bespoke licensing; enforcement

Learning Outcomes:

NonCore:

1. Discuss how creators can protect their intellectual property [Debate]
2. Compare options for open-source licensing of a game development tool [Compare]

2.17.25 SPD/Game Interface and Accessibility

Covers the usability and accessibility of game interfaces, including heuristic evaluation methods and design for players with disabilities.

Topics:

Non Core

- Fair Access to Play 1. Game Interface Usability: user requirements, affordances, ergonomic design, user research, experience measurement, and heuristic evaluation methods for games 2. Game Interface Accessibility: forms of impairment and disability; means to facilitate game access; universal design; legislated requirements for game platforms; compliance evaluation; challenging game mechanics and access

Learning Outcomes:

NonCore:

1. Identify common pitfalls in game interfaces that exclude players with impaired or non-functional vision [Analyze]
2. Describe how heuristic evaluation can be used to identify usability problems in game interfaces [Describe]
3. Recommend changes to a specific game interface to improve access to players who are deaf or whose hearing is otherwise impaired [Recommend]

2.17.26 SPD/Health, Safety and Sustainability

Covers game-related injuries and their mitigation, risk assessment for events and manufacturing, player mental health, and the supply chain and sustainability of platform hardware.

Topics:

Non Core

- Game-Related Health and Safety 1. Injuries in Play: ways of mitigating common upper body injuries, such as repetitive strain injury; exercise psychology and physiotherapy in eSports 2. Risk Assessment for Events and Manufacturing: control of substances hazardous to health (COSHH); fire safety; electrical and electronics safety; risk assessment for games and game events; risk assessment for manufacturing 3. Mental Health: motivation to play; gamification and gameful design; game psychology - internet gaming disorder
- Platform Hardware Supply Chain and Sustainability 1. Platform Lifecycle: platform composition - materials, assembly; mineral excavation and processing; power usage; recycling; planned obsolescence. 2. Modern Slavery: supply chains; forced labor and civil rights; working conditions; detection and remission; certification bodies and charitable endeavors.

Learning Outcomes:

NonCore:

1. Explain why upper body injuries are common in eSports [Explain]
2. Modify a policy for a LAN party event to include mitigations that lower the risk of fire [Design]
3. Design a gamification strategy to motivate serious play for an awareness-raising game [Design]
4. Assess the appropriateness of two manufacturers for producing a new game console [Assess]
5. Discuss whether games are addictive [Debate]

2.17.27 SPD/Representation, Inclusion and Equality

Covers representation in games media and industry, inclusion of diverse audiences, equality and social mobility in the industry, and the use of games to advocate for social causes.

Topics:
Non Core
- Representation in the Media and Industry 1. Inclusion: identity and identification; inclusion of a broad range of characters for diverse audiences; media representation and its effects; media literacy; content analysis; stereotyping; sexualization 2. Equality: histories and controversies, such as gamergate, quality of life in the industry, professional discourse and conduct in business contexts, pathways to game development careers, social mobility, the experience of developers from different backgrounds and identities, gender, and technology

Learning Outcomes:
NonCore:
1. Discuss how to reform characters and dialogues in a scene to reduce stereotype threat [Debate]
2. Illustrate how the portrayal of race in a game can influence the risk of social exclusion in the associated online community around the game [Apply]
3. Analyze the role of company hiring policies and advocacy on social mobility [Analyze]
4. Suggest how the portrayal of women in video games influences how players perceive members of those groups [Evaluate]
5. Create a video game that successfully advocates for climate science [Create]

2.17.28 SPD/SPD-SEP/Robotics

Topics:
Non Core
- Fairness, transparency, and accountability in robotic algorithms
- Mitigating biases in robot decision-making
- Public safety in shared spaces with robots
- Compliance with data protection laws
- Patient consent and trust in medical robots

Learning Outcomes:
NonCore:
1. Identify instances of bias in robotic algorithms and propose strategies to mitigate them [Analyze]
2. Evaluate and critique robotic systems for ethical and fairness considerations, suggesting improvements where necessary [Evaluate]
3. Analyze real-world examples of biases in robot decision-making and develop strategies to reduce bias in robotic systems [Analyze]
4. Assess the potential risks associated with robots in public spaces and propose safety measures to mitigate those risks [Assess]
5. Evaluate the impact of patient consent and trust on the effectiveness of medical robot deployments in healthcare contexts [Evaluate]

2.17.29 SPD/SPD-SEP/Interactive

Topics:
Non Core
- Ethical guidelines when using AI models to assist in journalism and content creation
- Accountability for AI-generated outputs
- Behavior among prompt programmers and AI developers
- Trust with the public when using AI models

Learning Outcomes:
NonCore:
1. Indicate a framework for accountability in AI model deployment, including clear documentation and attribution [Evaluate]
2. Discuss ethical codes of conduct and professional standards relevant to prompt programming and AI development [Debate]
3. Create communication plans and materials to educate the public about AI capabilities, limitations, and ethical safeguards [Create]

Chapter 3

Body of knowledge of Basic Sciences for Computing

The 19 knowledge areas in Basic Sciences for Computing are:

3.1 Calculus and Analysis (CAN) (Pg.. 142)
- 3.1.1 Line Integrals and Vector Fields (Pg.. 142)
- 3.1.2 Multiple Integrals (Pg.. 143)
- 3.1.3 Multivariable Transformations (Pg.. 143)
- 3.1.4 Surface Integrals and Fundamental Theorems (Pg.. 143)
- 3.1.5 Vector Functions of a Real Variable (Pg.. 144)
- 3.1.6 Parametric Equations and Polar Coordinates (Pg.. 144)
- 3.1.7 Limits and Continuity (Pg.. 144)
- 3.1.8 Differential Calculus (Pg.. 144)
- 3.1.9 Antiderivatives and Integration Techniques (Pg.. 145)
- 3.1.10 Improper Integrals (Pg.. 145)
- 3.1.11 Sequences, Series, and Power Series (Pg.. 145)
- 3.1.12 Multivariable Differential Calculus (Pg.. 145)
- 3.1.13 Harmonic Analysis and Fourier Theory (Pg.. 146)

3.2 Algebra and Number Theory (ANT) (Pg.. 146)
- 3.2.1 Matrices and Systems of Linear Equations (Pg.. 146)
- 3.2.2 Vector Spaces and Linear Transformations (Pg.. 147)
- 3.2.3 Eigenvalues, Inner Product Spaces, and Quadratic Forms (Pg.. 147)
- 3.2.4 Group Theory (Pg.. 148)
- 3.2.5 Rings, Fields, and Galois Theory (Pg.. 148)
- 3.2.6 Cryptography and Coding Theory (Pg.. 148)

3.3 Discrete Mathematics and Combinatorics (DMC) (Pg.. 149)
- 3.3.1 Enumerative Combinatorics (Pg.. 149)
- 3.3.2 Graph Theory (Pg.. 149)
- 3.3.3 Discrete Optimization (Pg.. 149)
- 3.3.4 Mathematical Logic and Set Theory (Pg.. 150)

3.4 Probability and Statistics (PST) (Pg.. 150)
- 3.4.1 Probability Axioms and Distributions (Pg.. 150)
- 3.4.2 Limit Theorems (Pg.. 151)
- 3.4.3 Stochastic Processes (Pg.. 151)
- 3.4.4 Estimation Theory (Pg.. 151)
- 3.4.5 Bayesian Inference (Pg.. 152)
- 3.4.6 Time Series Analysis (Pg.. 152)

3.5 Numerical and Scientific Analysis (NSA) (Pg.. 152)
- 3.5.1 Approximation Theory and Interpolation (Pg.. 153)
- 3.5.2 Numerical Integration and Quadrature (Pg.. 153)
- 3.5.3 Error Analysis and Floating-Point Arithmetic (Pg.. 154)
- 3.5.4 Root-Finding for Nonlinear Equations (Pg.. 154)
- 3.5.5 Numerical Linear Algebra (Pg.. 154)
- 3.5.6 Numerical Methods for Differential Equations (Pg.. 155)
- 3.5.7 Optimization Algorithms (Pg.. 155)
- 3.5.8 Finite Element Method (Pg.. 155)
- 3.5.9 Parallel and High-Performance Computing (Pg.. 156)

3.6 Mathematical Modeling and Simulation (MMS) (Pg.. 156)
- 3.6.1 First-Order Ordinary Differential Equations (Pg.. 156)
- 3.6.2 Higher-Order Differential Equations and Systems (Pg.. 157)
- 3.6.3 Dynamical Systems and Chaos (Pg.. 157)
- 3.6.4 Mathematical Control Theory (Pg.. 157)
- 3.6.5 Stochastic Differential Equations (Pg.. 158)

3.7 Mathematical Foundations of Data Science (FDS) (Pg.. 158)
- 3.7.1 Statistical Learning Theory (Pg.. 158)
- 3.7.2 Optimization for Machine Learning (Pg.. 159)
- 3.7.3 Information Theory (Pg.. 159)
- 3.7.4 Dimensionality Reduction (Pg.. 159)
- 3.7.5 Mathematical Foundations of Deep Learning (Pg.. 160)

3.8 Quantum Information and Computing (QIC) (Pg.. 160)
- 3.8.1 Quantum Mechanics Foundations for Computing (Pg.. 160)
- 3.8.2 Quantum Circuits and Gates (Pg.. 161)
- 3.8.3 Quantum Algorithms (Pg.. 161)
- 3.8.4 Entanglement and Bell Inequalities (Pg.. 162)

3.9 Electromagnetism (ELM) (Pg.. 162)
- 3.9.1 Electrostatics: Electric Field and Gauss's Law (Pg.. 162)
- 3.9.2 Electric Current, Circuits, and Kirchhoff's Laws (Pg.. 163)
- 3.9.3 Maxwell's Equations and Gauge Transformations (Pg.. 163)

3.10 Thermodynamics and Statistical Mechanics (TSM) (Pg.. 163)
- 3.10.1 Laws of Thermodynamics and State Functions (Pg.. 164)
- 3.10.2 Kinetic Theory of Gases (Pg.. 164)
- 3.10.3 Heat Engines, Entropy, and the Second Law (Pg.. 164)

3.11 Computational and Data-Driven Physics (CDP) (Pg.. 165)
- 3.11.1 Numerical Methods for Differential Equations in Physics (Pg.. 165)
- 3.11.2 Monte Carlo Methods and Simulations (Pg.. 165)
- 3.11.3 Machine Learning for Physical Discovery (Pg.. 166)

3.12 Classical Mechanics (CME) (Pg.. 166)
- 3.12.1 Rotational Motion and Angular Momentum (Pg.. 166)
- 3.12.2 Systems of Particles and Linear Momentum (Pg.. 167)
- 3.12.3 Kinematics and Dynamics of Particles (Pg.. 168)
- 3.12.4 Work, Energy, and Conservation Laws (Pg.. 168)
- 3.12.5 Rigid Body Dynamics (Pg.. 168)
- 3.12.6 Oscillations and Waves (Pg.. 169)
- 3.12.7 Lagrangian and Hamiltonian Mechanics (Pg.. 169)
- 3.12.8 Elasticity (Pg.. 170)
- 3.12.9 Fluids: Hydrostatics, Hydrodynamics, and Viscosity (Pg.. 170)
- 3.12.10 Vibrations and Waves (Pg.. 170)
- 3.12.11 Transport Phenomena (Pg.. 171)

3.13 General Chemistry (GCH) (Pg.. 171)
- 3.13.1 Atomic Structure (Pg.. 172)
- 3.13.2 Chemical Bonding (Pg.. 172)
- 3.13.3 Matter and Energy (Pg.. 173)
- 3.13.4 States of Matter (Pg.. 173)
- 3.13.5 Stoichiometry (Pg.. 173)
- 3.13.6 Environmental and Water Chemistry (Pg.. 174)
- 3.13.7 Polymers and Geosynthetics (Pg.. 174)
- 3.13.8 Chemical Equilibrium (Pg.. 174)
- 3.13.9 Ionic Equilibrium and Salt Hydrolysis (Pg.. 175)
- 3.13.10 Salts, Solubility, and Precipitation Equilibria (Pg.. 175)
- 3.13.11 Electrochemistry and Electrolysis (Pg.. 175)
- 3.13.12 Cement Chemistry (Pg.. 176)

3.14 Computational and Theoretical Chemistry (CTC) (Pg.. 176)
- 3.14.1 Molecular Modeling and Simulation (Pg.. 176)
- 3.14.2 Machine Learning in Chemical Discovery (Pg.. 177)
- 3.14.3 Computational Drug Design (Pg.. 177)

3.15 Biochemistry and Molecular Biology (BMB) (Pg.. 177)
- 3.15.1 Structure and Function of Biomolecules (Pg.. 178)
- 3.15.2 Molecular Genetics and Enzymology (Pg.. 178)
- 3.15.3 Bioinformatics and Computational Biology (Pg.. 179)

3.16 Probability Theory (PRO) (Pg.. 179)
- 3.16.1 Sample Spaces and Events (Pg.. 179)
- 3.16.2 Probability Axioms and Properties (Pg.. 180)
- 3.16.3 Conditional Probability and Independence (Pg.. 180)
- 3.16.4 Discrete Random Variables and Distributions (Pg.. 180)
- 3.16.5 Continuous Random Variables and Distributions (Pg.. 181)
- 3.16.6 Joint, Marginal, and Conditional Distributions (Pg.. 181)
- 3.16.7 Limit Theorems: LLN and CLT (Pg.. 181)

3.17 Descriptive Statistics and Data Analysis (DSC) (Pg.. 182)
- 3.17.1 Numerical Summaries: Location, Spread, and Shape (Pg.. 182)

3.18 Statistical Inference (INF) (Pg.. 182)
- 3.18.1 Point Estimation (Pg.. 182)
- 3.18.2 Interval Estimation and Confidence Intervals (Pg.. 183)
- 3.18.3 Hypothesis Testing Foundations (Pg.. 183)
- 3.18.4 Bayesian Inference (Pg.. 183)

3.19 Regression and Statistical Models (REG) (Pg.. 184)
- 3.19.1 Simple Linear Regression (Pg.. 184)

3.1 Calculus and Analysis (CAN)

This area covers the fundamental study of change, limits, and integration. It spans from single-variable differential and integral calculus to multivariable analysis and harmonic methods, providing the essential analytical toolkit for algorithm analysis, signal processing, and machine learning.

Knowledge Area (KA)	CS Core	KA Core
⭐ 3.1.1 Line Integrals and Vector Fields (Pg.. 142)	1	1
⭐ 3.1.2 Multiple Integrals (Pg.. 143)	1	1
⭐ 3.1.3 Multivariable Transformations (Pg.. 143)	1	1
⭐ 3.1.4 Surface Integrals and Fundamental Theorems (Pg.. 143)	1	1
⭐ 3.1.5 Vector Functions of a Real Variable (Pg.. 144)	1	1
⭐ 3.1.6 Parametric Equations and Polar Coordinates (Pg.. 144)	1	1
⭐ 3.1.7 Limits and Continuity (Pg.. 144)	1	1
⭐ 3.1.8 Differential Calculus (Pg.. 144)	1	1
⭐ 3.1.9 Antiderivatives and Integration Techniques (Pg.. 145)	1	1
⭐ 3.1.10 Improper Integrals (Pg.. 145)	1	1
⭐ 3.1.11 Sequences, Series, and Power Series (Pg.. 145)	1	1
⭐ 3.1.12 Multivariable Differential Calculus (Pg.. 145)	1	1
⭐ 3.1.13 Harmonic Analysis and Fourier Theory (Pg.. 146)	1	1

Table 3.1: List of KUs in the Calculus and Analysis area.

3.1.1 CAN/Line Integrals and Vector Fields (CS Core: 1 hr, KA Core: 1 hr)

Scalar and vector line integrals, conservative fields, potential functions, path independence, and Green's Theorem.

Topics:

Core
- Vector fields: definition, visualization, divergence, and curl
- Line integrals of scalar functions: arc length and mass
- Line integrals of vector fields: work and circulation
- Conservative fields, potential functions, and path independence
- Green's Theorem and its applications in the plane

Learning Outcomes:

Core:
1. Compute scalar and vector line integrals over oriented curves [Usage]

2. Determine whether a vector field is conservative and construct its potential function [Usage]
3. Apply Green's Theorem to evaluate line integrals via double integrals [Assessment]

3.1.2 CAN/Multiple Integrals (CS Core: 1 hr, KA Core: 1 hr)

Integration of functions of several variables: double and triple integrals in Cartesian, polar, cylindrical, and spherical coordinates, with applications.

Topics:

Core

- Double integrals over rectangles and general regions; iterated integrals (Fubini's theorem)
- Double integrals in polar coordinates and change-of-variables formula
- Triple integrals in Cartesian coordinates and applications to volume and mass
- Triple integrals in cylindrical and spherical coordinates
- Applications: area, volume, centers of mass, and moments of inertia

Learning Outcomes:

Core:

1. Evaluate double and triple integrals over general regions using iterated integrals [Usage]
2. Apply coordinate changes in polar, cylindrical, and spherical systems to simplify multiple integrals [Usage]
3. Compute areas, volumes, and centers of mass using multiple integrals [Assessment]

3.1.3 CAN/Multivariable Transformations (CS Core: 1 hr, KA Core: 1 hr)

Vector functions of vector variables: differentiability, the Jacobian matrix, change of variables, and the inverse and implicit function theorems.

Topics:

Core

- Transformations from $\mathbb{R}^n$ to $\mathbb{R}^m$ and their differentiability
- Jacobian matrix and Jacobian determinant
- Change of variables in polar, cylindrical, and spherical coordinates
- Inverse function theorem
- Implicit function theorem in multivariable contexts

Learning Outcomes:

Core:

1. Describe transformations between coordinate systems using the Jacobian matrix [Familiarity]
2. Compute the Jacobian and apply change of variables to evaluate multiple integrals [Usage]
3. Apply the inverse and implicit function theorems to analyze local invertibility [Assessment]

3.1.4 CAN/Surface Integrals and Fundamental Theorems (CS Core: 1 hr, KA Core: 1 hr)

Parametric surfaces, scalar and flux surface integrals, curl, divergence, and the fundamental theorems of Stokes and Gauss.

Topics:

Core

- Parametric surfaces, tangent plane, and surface area
- Surface integrals of scalar functions
- Flux integrals: surface integrals of vector fields
- Stokes' Theorem: relation between line integrals and surface integrals
- Divergence Theorem (Gauss): relation between flux and volume integral

Learning Outcomes:

Core:

1. Parametrize surfaces and evaluate scalar and flux surface integrals [Usage]
2. Apply Stokes' and Divergence Theorems to reduce complex integrals [Assessment]
3. Relate local differential properties (curl, divergence) to global behavior via integral theorems [Assessment]

3.1.5 CAN/Vector Functions of a Real Variable (CS Core: 1 hr, KA Core: 1 hr)

Space curves as vector functions of a real variable, including differential geometry of curves: Frenet frame, curvature, and torsion.

Topics:

Core

- Definition, domain, and range of vector functions of a real variable
- Limits, continuity, and derivatives of vector functions
- Arc length and natural parametrization of a curve
- Tangent, normal, and binormal vectors; moving Frenet frame
- Curvature, torsion, and Frenet-Serret formulas

Learning Outcomes:

Core:

1. Identify limits, continuity, and derivatives of vector functions of a real variable [Familiarity]
2. Compute tangent, normal, and binormal vectors and reparametrize a curve by arc length [Usage]
3. Calculate curvature and torsion using the Frenet-Serret formulas [Assessment]

3.1.6 CAN/Parametric Equations and Polar Coordinates (CS Core: 1 hr, KA Core: 1 hr)

Parametric curves and polar coordinates, used to describe trajectories and geometries in kinematics and engineering design.

Topics:

Core

- Parametric curves and their calculus, applied to trajectories in kinematics
- Polar coordinates and their conversion to Cartesian coordinates

Learning Outcomes:

Core:

1. Calculate the velocity and acceleration of a trajectory described parametrically [Usage]
2. Convert between polar and Cartesian coordinates when modeling engineering geometries [Usage]

3.1.7 CAN/Limits and Continuity (CS Core: 1 hr, KA Core: 1 hr)

Rigorous study of limits and continuity of single-variable functions, including the epsilon-delta definition, limit laws, and fundamental theorems.

Topics:

Core

- Epsilon-delta definition of a limit and formal proof techniques
- Limit laws, one-sided limits, and limits at infinity
- Continuity: types, properties, and the Intermediate Value Theorem
- L'Hôpital's rule and indeterminate forms
- Limits of sequences: convergence, Squeeze Theorem, and Bolzano-Weierstrass

Learning Outcomes:

Core:

1. State the epsilon-delta definition of a limit and explain its geometric meaning [Familiarity]
2. Evaluate limits using limit laws, L'Hôpital's rule, and asymptotic analysis [Usage]
3. Prove continuity or discontinuity of a function and apply the Intermediate Value Theorem [Assessment]

3.1.8 CAN/Differential Calculus (CS Core: 1 hr, KA Core: 1 hr)

Derivatives of single-variable functions: definition, rules, higher-order derivatives, and applications to optimization and algorithm analysis.

Topics:

Core

- Derivative as a limit; geometric and physical interpretations; differentiability
- Product, quotient, and chain rules; implicit differentiation; related rates

- Rolle's theorem, Mean Value Theorem, and monotonicity criteria
- Higher-order derivatives, Taylor polynomials, and remainder estimation
- Extrema: first and second derivative tests, curve sketching, and applied optimization

Learning Outcomes:

Core:

1. Interpret the derivative as a rate of change and as the slope of the tangent line [Familiarity]
2. Apply differentiation rules and implicit differentiation to compute derivatives of composite and implicit functions [Usage]
3. Solve optimization problems and sketch curves using the Mean Value Theorem and derivative tests [Assessment]

3.1.9 CAN/Antiderivatives and Integration Techniques (CS Core: 1 hr, KA Core: 1 hr)

Topics:

Core

- The Fundamental Theorem of Calculus and the antiderivative
- Integration techniques: substitution, parts, partial fractions, and trigonometric substitution

Learning Outcomes:

Core:

1. State the Fundamental Theorem of Calculus and explain the relationship between differentiation and integration [Familiarity]
2. Compute definite and indefinite integrals using standard techniques including substitution and integration by parts [Usage]

3.1.10 CAN/Improper Integrals (CS Core: 1 hr, KA Core: 1 hr)

Topics:

Core

- Improper integrals: convergence criteria and comparison tests

Learning Outcomes:

Core:

1. Determine convergence of improper integrals using comparison and other convergence tests [Assessment]

3.1.11 CAN/Sequences, Series, and Power Series (CS Core: 1 hr, KA Core: 1 hr)

Topics:

Core

- Sequences and series: convergence tests (ratio, root, comparison, integral)
- Power series, radius of convergence, Taylor and Maclaurin series

Learning Outcomes:

Core:

1. Determine convergence of power series and construct Taylor series for standard functions [Assessment]

3.1.12 CAN/Multivariable Differential Calculus (CS Core: 1 hr, KA Core: 1 hr)

Differential calculus of scalar functions of several variables: limits, partial derivatives, gradient, tangent plane, optimization, and Taylor's theorem.

Topics:

Core

- Limits and continuity of functions of several variables in $\mathbb{R}^n$
- Partial derivatives, directional derivatives, and the gradient vector
- Tangent plane, total differential, and linear approximation
- Extrema of multivariable functions and Lagrange multipliers

- Taylor's theorem for functions of several variables

Learning Outcomes:

Core:

1. Evaluate limits and determine continuity of functions of several variables [Familiarity]
2. Compute gradients, directional derivatives, and tangent planes for multivariable functions [Usage]
3. Apply Lagrange multipliers to solve constrained optimization problems in computing and data science [Usage]
4. Expand functions of several variables using Taylor's theorem and estimate error bounds [Assessment]

3.1.13 CAN/Harmonic Analysis and Fourier Theory (CS Core: 1 hr, KA Core: 1 hr)

Fourier series and transforms with applications to signal processing, compression, and spectral analysis in computing.

Topics:

Core

- Fourier series: convergence, Parseval's theorem, and L^2 completeness
- The Fourier transform: properties, inversion, and convolution
- Plancherel theorem and tempered distributions (Schwartz space)
- Wavelet theory: multiresolution analysis and the discrete wavelet transform
- Applications to signal processing, PDE solving, and data compression

Learning Outcomes:

Core:

1. Decompose periodic functions into Fourier series and assess pointwise and L^2 convergence [Familiarity]
2. Apply the Fourier transform to solve linear systems and compute convolutions in digital signal processing [Usage]
3. Analyze the decay rate of Fourier coefficients in relation to the smoothness of a function [Assessment]

3.2 Algebra and Number Theory (ANT)

This area covers the algebraic structures essential for computing: linear algebra for machine learning and graphics, abstract algebra for cryptography and coding theory, and the number-theoretic foundations of modern security protocols.

Knowledge Area (KA)	CS Core	KA Core
⭐ 3.2.1 Matrices and Systems of Linear Equations (Pg.. 146)	1	1
⭐ 3.2.2 Vector Spaces and Linear Transformations (Pg.. 147)	1	1
⭐ 3.2.3 Eigenvalues, Inner Product Spaces, and Quadratic Forms (Pg.. 147)	1	1
⭐ 3.2.4 Group Theory (Pg.. 148)	1	1
⭐ 3.2.5 Rings, Fields, and Galois Theory (Pg.. 148)	1	1
⭐ 3.2.6 Cryptography and Coding Theory (Pg.. 148)	1	1

Table 3.2: List of KUs in the Algebra and Number Theory area.

3.2.1 ANT/Matrices and Systems of Linear Equations (CS Core: 1 hr, KA Core: 1 hr)

Matrix algebra, systems of linear equations and their solution methods, determinants, and matrix decompositions, with applications to data science and computer graphics.

Topics:

Core

- Matrix algebra: addition, scalar multiplication, matrix multiplication, transpose, types of matrices, and the inverse of a matrix
- Solving systems of linear equations via Gaussian elimination and Gauss-Jordan; row echelon form, rank, and consistency of systems
- Determinants: definition, properties, Cramer's rule, and geometric interpretation
- Matrix decompositions: LU, QR, and singular value decomposition (SVD)

Learning Outcomes:
Core:

1. Identify types of matrices and perform basic matrix algebra operations, including computing the inverse of a matrix [Familiarity]
2. Solve systems of linear equations using Gaussian elimination or Gauss-Jordan, determining the rank and consistency of the system [Usage]
3. Apply Cramer's rule and the LU, QR, and SVD decompositions to solve linear systems and factor matrices [Assessment]

3.2.2 ANT/Vector Spaces and Linear Transformations (CS Core: 1 hr, KA Core: 1 hr)

Vector spaces, subspaces, bases, and linear maps between vector spaces, with applications to machine learning.
Topics:
Core

- Vector spaces and subspaces; bases, dimension, and linear independence
- Linear maps, kernel and image, rank-nullity theorem, and matrix representations

Learning Outcomes:
Core:

1. Identify whether a set with given operations forms a vector space and determine a basis [Familiarity]
2. Determine the kernel, image, and matrix representation of a linear map, verifying the rank-nullity theorem [Usage]
3. Apply linear transformations to model and solve problems involving change of basis and composition of maps [Assessment]

3.2.3 ANT/Eigenvalues, Inner Product Spaces, and Quadratic Forms (CS Core: 1 hr, KA Core: 1 hr)

Eigenvalues, diagonalization, inner product spaces, and quadratic forms, with applications to machine learning and data science.
Topics:
Core

- Eigenvalues and eigenvectors; diagonalization and the spectral theorem
- Inner product spaces, orthogonality, Gram-Schmidt process, and least squares
- Quadratic forms and definiteness: classification as positive, negative, or indefinite via eigenvalues

Learning Outcomes:
Core:

1. Identify the eigenvalues and eigenvectors of a matrix and determine whether it is diagonalizable [Familiarity]
2. Compute the diagonalization of matrices via the spectral theorem and apply the Gram-Schmidt process to obtain orthogonal bases [Usage]
3. Apply least squares methods and the classification of quadratic forms via eigenvalues to solve approximation and optimization problems [Assessment]

3.2.4 ANT/Group Theory (CS Core: 1 hr, KA Core: 1 hr)

Groups, subgroups, homomorphisms, quotient groups, and the classification of finite groups, with applications to symmetry in computing and cryptography.

Topics:

Core

- Groups, subgroups, cyclic groups, and order of elements
- Cosets, Lagrange's theorem, and normal subgroups
- Group homomorphisms, isomorphisms, and the isomorphism theorems
- Group actions, orbits, stabilizers, and Burnside's lemma

Learning Outcomes:

Core:

1. State Lagrange's theorem and explain the relationship between subgroups and cosets [Familiarity]
2. Apply the isomorphism theorems to analyze homomorphic images and quotient groups [Usage]
3. Analyze symmetry groups arising in computing contexts such as permutation groups over data structures [Assessment]

3.2.5 ANT/Rings, Fields, and Galois Theory (CS Core: 1 hr, KA Core: 1 hr)

Rings, ideals, polynomial rings, field extensions, and finite fields foundational for coding theory and cryptography.

Topics:

Core

- Rings, ideals, quotient rings, and the ring isomorphism theorems
- Polynomial rings: irreducibility, factorization, and unique factorization domains
- Field extensions: algebraic and transcendental elements, degree, and splitting fields
- Finite fields: structure, existence, and applications in coding theory

Learning Outcomes:

Core:

1. Classify rings as integral domains, PIDs, or UFDs and justify the classification [Familiarity]
2. Construct and analyze finite fields used in AES and other cryptographic algorithms [Usage]
3. Apply field extension theory to analyze error-correcting codes over finite fields [Assessment]

3.2.6 ANT/Cryptography and Coding Theory (CS Core: 1 hr, KA Core: 1 hr)

Mathematical foundations of modern cryptography and error-correcting codes, including RSA, elliptic curves, and linear codes.

Topics:

Core

- Modular arithmetic, the RSA cryptosystem, and discrete logarithm problem
- Elliptic curves over finite fields and elliptic curve cryptography (ECC)
- Linear codes: generator and parity-check matrices, Hamming distance, and error correction
- Cyclic codes: polynomial representation, BCH codes, and Reed-Solomon codes

Non Core

- Post-quantum cryptography: lattice-based schemes and the Learning With Errors (LWE) problem

Learning Outcomes:

Core:

1. Explain the mathematical hardness assumptions underlying RSA and elliptic curve cryptography [Familiarity]
2. Construct a linear code, compute its parameters, and determine its error-correction capability [Usage]

NonCore:

3. Analyze the security assumptions of lattice-based cryptographic schemes in the post-quantum context [Assessment]

3.3 Discrete Mathematics and Combinatorics (DMC)

This area covers the mathematics of finite and countable structures, including combinatorial enumeration, graph theory, discrete optimization, and the logical foundations of computer science. It is core to algorithm design, formal languages, type systems, and compiler theory.

Knowledge Area (KA)	CS Core	KA Core
⭐ 3.3.1 Enumerative Combinatorics (Pg.. 149)	1	1
⭐ 3.3.2 Graph Theory (Pg.. 149)	1	1
⭐ 3.3.3 Discrete Optimization (Pg.. 149)	1	1
⭐ 3.3.4 Mathematical Logic and Set Theory (Pg.. 150)	1	1

Table 3.3: List of KUs in the Discrete Mathematics and Combinatorics area.

3.3.1 DMC/Enumerative Combinatorics (CS Core: 1 hr, KA Core: 1 hr)

Systematic counting of discrete structures using bijections, recursions, generating functions, and the principle of inclusion-exclusion.

Topics:

Core

- Basic counting principles: permutations, combinations, and the binomial theorem
- Principle of inclusion-exclusion and applications to derangements and Euler's totient function
- Ordinary and exponential generating functions; solving recurrences
- Integer partitions, Young tableaux, and the hook-length formula

Learning Outcomes:

Core:

1. Apply basic counting principles and inclusion-exclusion to enumeration problems [Familiarity]
2. Construct ordinary and exponential generating functions to enumerate combinatorial structures [Usage]
3. Derive closed-form formulas for combinatorial sequences using bijections and recurrences [Assessment]

3.3.2 DMC/Graph Theory (CS Core: 1 hr, KA Core: 1 hr)

Structure and properties of graphs: connectivity, colorings, matchings, planarity, and spectral graph theory, with applications to networks and algorithms.

Topics:

Core

- Graphs: definitions, isomorphism, degree sequences, trees, and spanning trees
- Connectivity, Menger's theorem, and network flows (max-flow min-cut)
- Matchings (Hall's theorem), graph colorings, and the chromatic polynomial
- Planar graphs, Euler's formula, Kuratowski's theorem, and the four-color theorem
- Adjacency and Laplacian matrices, eigenvalues, and expander graphs

Learning Outcomes:

Core:

1. Identify structural properties of graphs (connectivity, planarity, bipartiteness) and apply Euler's formula [Familiarity]
2. Apply Hall's marriage theorem and network flow algorithms to matching and routing problems [Usage]
3. Analyze a graph's spectrum to bound its chromatic number and connectivity properties [Assessment]

3.3.3 DMC/Discrete Optimization (CS Core: 1 hr, KA Core: 1 hr)

Integer programming, network flows, matroids, matching algorithms, and the theory of NP-completeness, foundational for algorithm design in computing.

Topics:
Core
- Integer linear programming: branch and bound, cutting planes, and totally unimodular matrices
- Network flow algorithms: shortest paths, max-flow, min-cost flow
- Matroid theory: independent sets, greedy algorithm, and matroid intersection
- Complexity theory: P vs. NP, NP-completeness, and reductions
- Approximation algorithms and the polynomial-time approximation scheme (PTAS)

Learning Outcomes:
Core:
1. Explain the relationship between linear programming relaxations and integer programming bounds [Familiarity]
2. Apply network flow and matroid algorithms to solve combinatorial optimization problems [Usage]
3. Prove NP-completeness of a combinatorial problem via polynomial reduction [Assessment]

3.3.4 DMC/Mathematical Logic and Set Theory (CS Core: 1 hr, KA Core: 1 hr)

Propositional and first-order logic, proof systems, Gödel's theorems, and computability, forming the logical foundation of programming languages and formal verification.
Topics:
Core
- Propositional logic: syntax, semantics, satisfiability, and resolution
- First-order logic: models, soundness, and the Gödel completeness theorem
- Gödel's incompleteness theorems and their implications for formal systems
- Axiomatic set theory (ZFC): ordinals, cardinals, and the axiom of choice
- Computability: Turing machines, decidability, and the halting problem

Learning Outcomes:
Core:
1. State Gödel's completeness and first incompleteness theorems and explain their significance for computing [Familiarity]
2. Construct formal proofs in first-order logic and determine the satisfiability of formulas [Usage]
3. Analyze the undecidability of problems by reduction to the halting problem [Assessment]

3.4 Probability and Statistics (PST)

This area covers the mathematical theory of uncertainty and inference, from probability axioms and stochastic processes to Bayesian inference and statistical learning, which are essential foundations for AI, data science, and machine learning.

Knowledge Area (KA)	CS Core	KA Core
⭐ 3.4.1 Probability Axioms and Distributions (Pg.. 150)	1	1
⭐ 3.4.2 Limit Theorems (Pg.. 151)	1	1
⭐ 3.4.3 Stochastic Processes (Pg.. 151)	1	1
⭐ 3.4.4 Estimation Theory (Pg.. 151)	1	1
⭐ 3.4.5 Bayesian Inference (Pg.. 152)	1	1
⭐ 3.4.6 Time Series Analysis (Pg.. 152)	1	1

Table 3.4: List of KUs in the Probability and Statistics area.

3.4.1 PST/Probability Axioms and Distributions (CS Core: 1 hr, KA Core: 1 hr)

Kolmogorov's axiomatic foundation of probability, random variables, distributions, and expectations, with applications in algorithm analysis and AI.
Topics:
Core

- Kolmogorov axioms, sigma-algebras, and probability spaces
- Random variables: discrete and continuous distributions, CDF, PDF, and PMF
- Expectation, variance, moments, and moment generating functions
- Joint, marginal, and conditional distributions; independence and covariance
- Standard families: Binomial, Poisson, Normal, Exponential, Gamma, and Beta

Learning Outcomes:

Core:

1. State Kolmogorov's axioms and construct a probability space for a standard experiment [Familiarity]
2. Compute expectations, variances, and moment generating functions for standard distributions [Usage]
3. Determine joint and conditional distributions and assess independence of random variables [Assessment]

3.4.2 PST/Limit Theorems (CS Core: 1 hr, KA Core: 1 hr)

Laws of large numbers, central limit theorem, modes of convergence, and large deviations, foundational for algorithm analysis and statistical inference.

Topics:

Core

- Modes of convergence: almost sure, in probability, in L^p, and in distribution
- Weak and strong laws of large numbers and their applications
- Central Limit Theorem: statement, proof via characteristic functions, and Berry-Esseen bound
- Characteristic functions and the continuity theorem (Lévy)

Learning Outcomes:

Core:

1. Distinguish between modes of convergence of random variables and provide counterexamples [Familiarity]
2. Apply the Central Limit Theorem to approximate distributions of sums and means [Usage]
3. Prove the strong law of large numbers and estimate large deviation probabilities [Assessment]

3.4.3 PST/Stochastic Processes (CS Core: 1 hr, KA Core: 1 hr)

Markov chains, Poisson processes, Brownian motion, martingales, and stochastic calculus, with applications to queuing, simulation, and reinforcement learning.

Topics:

Core

- Discrete-time Markov chains: classification of states, stationary distributions, and ergodic theorem
- Poisson processes: definition, properties, and generalizations (compound, inhomogeneous)
- Brownian motion: definition, path properties, and the reflection principle
- Martingales, stopping times, optional sampling theorem, and Doob's inequalities

Learning Outcomes:

Core:

1. Classify states of a Markov chain and determine its stationary distribution [Familiarity]
2. Apply martingale theory and the optional sampling theorem to solve stopping time problems [Usage]
3. Model a queueing system using Poisson arrival processes and analyze its steady-state distribution [Assessment]

3.4.4 PST/Estimation Theory (CS Core: 1 hr, KA Core: 1 hr)

Point and interval estimation: sufficiency, maximum likelihood, and asymptotic theory, used throughout statistical machine learning and data science.

Topics:

Core

- Sufficient statistics, the factorization theorem, and completeness
- Unbiased estimators, the Cramér-Rao lower bound, and UMVUE
- Maximum likelihood estimation: properties, consistency, and asymptotic normality
- Confidence intervals: pivotal quantities, likelihood ratio, and bootstrap methods
- Asymptotic theory: delta method, M-estimators, and the Fisher information matrix

Learning Outcomes:

Core:

1. Explain the Cramér-Rao bound and identify when a UMVUE exists [Familiarity]
2. Derive maximum likelihood estimators and compute their asymptotic variance [Usage]
3. Construct confidence intervals using pivotal quantities and bootstrap resampling [Assessment]

3.4.5 PST/Bayesian Inference (CS Core: 1 hr, KA Core: 1 hr)

Bayesian paradigm: prior and posterior distributions, conjugate families, hierarchical models, and MCMC methods, central to probabilistic machine learning and AI.

Topics:

Core

- Bayes' theorem in continuous settings; prior and posterior distributions
- Conjugate prior families: Beta-Binomial, Dirichlet-Multinomial, Normal-Normal
- Hierarchical Bayesian models and empirical Bayes
- Markov chain Monte Carlo (MCMC): Metropolis-Hastings and Gibbs sampling
- Variational inference and the evidence lower bound (ELBO)

Learning Outcomes:

Core:

1. Apply Bayes' theorem to update a prior with observed data and obtain a posterior distribution [Familiarity]
2. Construct a hierarchical Bayesian model and derive its joint and conditional distributions [Usage]
3. Implement Gibbs sampling or Metropolis-Hastings to approximate a posterior distribution [Assessment]

3.4.6 PST/Time Series Analysis (CS Core: 1 hr, KA Core: 1 hr)

Stationary processes, ARMA models, spectral analysis, state-space models, and forecasting.

Topics:

Core

- Stationarity, autocovariance function, and the autocorrelation function (ACF)
- ARMA models: identification, estimation (Yule-Walker, MLE), and diagnostics
- Spectral density, the periodogram, and Wiener-Khinchin theorem
- State-space models and the Kalman filter
- ARIMA models, seasonal adjustment, and multi-step forecasting

Learning Outcomes:

Core:

1. Identify stationarity and determine the order of an ARMA model from ACF and PACF plots [Familiarity]
2. Fit ARIMA models to time series data, validate residuals, and produce forecasts [Usage]
3. Apply the Kalman filter to estimate hidden states in a linear Gaussian state-space model [Assessment]

3.5 Numerical and Scientific Analysis (NSA)

This area covers numerical algorithms for computation: error analysis, linear algebra solvers, numerical integration, differential equations, and optimization methods essential for implementing reliable computational systems and scientific software.

Knowledge Area (KA)	CS Core	KA Core
⭐ 3.5.1 Approximation Theory and Interpolation (Pg.. 153)	1	1
⭐ 3.5.2 Numerical Integration and Quadrature (Pg.. 153)	1	1
⭐ 3.5.3 Error Analysis and Floating-Point Arithmetic (Pg.. 154)	1	1
⭐ 3.5.4 Root-Finding for Nonlinear Equations (Pg.. 154)	1	1
⭐ 3.5.5 Numerical Linear Algebra (Pg.. 154)	1	1
⭐ 3.5.6 Numerical Methods for Differential Equations (Pg.. 155)	1	1
⭐ 3.5.7 Optimization Algorithms (Pg.. 155)	1	1
⭐ 3.5.8 Finite Element Method (Pg.. 155)	1	1
⭐ 3.5.9 Parallel and High-Performance Computing (Pg.. 156)	1	1

Table 3.5: List of KUs in the Numerical and Scientific Analysis area.

3.5.1 NSA/Approximation Theory and Interpolation (CS Core: 1 hr, KA Core: 1 hr)

Polynomial and spline interpolation, best approximation, Chebyshev polynomials, and rational approximation.

Topics:

Core

- Polynomial interpolation: Lagrange, Newton divided differences, and Runge's phenomenon
- Spline interpolation: cubic splines, B-splines, and piecewise polynomial methods
- Best approximation in normed spaces: Chebyshev (minimax) and least squares
- Chebyshev polynomials: properties, orthogonality, and spectral convergence
- Trigonometric approximation and the fast Fourier transform (FFT)

Learning Outcomes:

Core:

1. Explain Runge's phenomenon and justify the choice of Chebyshev nodes to mitigate it [Familiarity]
2. Construct cubic spline and Chebyshev interpolants for given data and estimate the interpolation error [Usage]
3. Apply the FFT to efficiently compute trigonometric approximations of a sampled function [Assessment]

3.5.2 NSA/Numerical Integration and Quadrature (CS Core: 1 hr, KA Core: 1 hr)

Quadrature rules, adaptive integration, Gaussian quadrature, and multi-dimensional integration.

Topics:

Core

- Newton-Cotes rules: trapezoidal, Simpson's, and composite rules; error analysis
- Adaptive quadrature and automatic error control
- Gaussian quadrature: optimal nodes and weights, orthogonal polynomial connection
- Numerical treatment of improper and singular integrals
- Monte Carlo integration and quasi-Monte Carlo methods for high-dimensional integrals

Learning Outcomes:

Core:

1. Derive the error formula for composite Simpson's rule and identify its order of accuracy [Familiarity]
2. Select and apply an appropriate quadrature rule (Gaussian, adaptive) based on integrand regularity [Usage]
3. Apply Monte Carlo integration to estimate high-dimensional integrals and quantify the statistical error [Assessment]

3.5.3 NSA/Error Analysis and Floating-Point Arithmetic (CS Core: 1 hr, KA Core: 1 hr)

Sources of numerical error, IEEE 754 floating-point arithmetic, rounding, cancellation, and forward and backward error analysis, critical for reliable software engineering.

Topics:

Core

- IEEE 754 floating-point representation: machine epsilon, overflow, underflow, and rounding modes
- Rounding errors, catastrophic cancellation, and loss of significance
- Forward and backward error analysis; condition number of a problem
- Numerical stability: stable vs. unstable algorithms; Wilkinson's backward error analysis
- Error propagation in arithmetic operations and function evaluation

Learning Outcomes:

Core:

1. Explain the IEEE 754 floating-point standard and identify the sources of rounding and cancellation errors [Familiarity]
2. Estimate the condition number of a problem and predict the accuracy of a numerical result [Usage]
3. Analyze an algorithm for numerical stability using forward and backward error bounds [Assessment]

3.5.4 NSA/Root-Finding for Nonlinear Equations (CS Core: 1 hr, KA Core: 1 hr)

Iterative methods for approximating roots of nonlinear equations and systems of nonlinear equations, the basis of solvers used in optimization and machine learning.

Topics:

Core

- Bisection method: interval bracketing, stopping criteria, and error bound
- Newton-Raphson method: derivation, quadratic convergence, and failure cases
- Secant method as an alternative to Newton's method without an explicit derivative
- Order of convergence and efficiency comparison among iterative methods
- Extension of Newton's method to systems of nonlinear equations via the Jacobian matrix, the basis of solvers used in machine learning

Learning Outcomes:

Core:

1. Compare the convergence guarantees of the bisection, Newton-Raphson, and secant methods [Familiarity]
2. Implement the Newton-Raphson method to approximate roots of a nonlinear function [Usage]
3. Analyze the order of convergence of an iterative method and justify its choice based on computational cost [Assessment]

3.5.5 NSA/Numerical Linear Algebra (CS Core: 1 hr, KA Core: 1 hr)

Algorithms for solving linear systems, least squares, eigenvalue problems, and matrix factorizations, applied in machine learning and scientific computing.

Topics:

Core

- Gaussian elimination with partial pivoting, LU factorization, and complexity analysis
- QR factorization via Householder and Gram-Schmidt; least squares problems
- Eigenvalue algorithms: power iteration, QR algorithm, and the Lanczos method
- Singular value decomposition: computation, truncation, and applications (PCA, pseudoinverse)
- Iterative methods for large sparse systems: CG, GMRES, and preconditioning

Learning Outcomes:

Core:

1. Compare direct and iterative solvers and select the appropriate method for a given matrix structure [Familiarity]
2. Apply QR factorization and SVD to solve least squares problems and compute low-rank approximations [Usage]
3. Analyze the numerical stability and computational cost of eigenvalue algorithms [Assessment]

3.5.6 NSA/Numerical Methods for Differential Equations (CS Core: 1 hr, KA Core: 1 hr)

Finite difference and Runge-Kutta methods for ODEs, stability analysis, and finite difference methods for PDEs, used in scientific simulation and physics-based computing.

Topics:

Core

- Runge-Kutta methods: Euler, RK4, and embedded methods for error control
- Linear multistep methods: Adams-Bashforth, Adams-Moulton, and BDF methods
- Stability analysis: zero-stability, absolute stability regions, and stiff ODEs
- Finite difference methods for parabolic and elliptic PDEs: stability and convergence

Learning Outcomes:

Core:

1. Compare explicit and implicit ODE solvers and explain their relative stability for stiff problems [Familiarity]
2. Implement a Runge-Kutta method with adaptive step control and apply it to a system of ODEs [Usage]
3. Analyze the stability and convergence of a finite difference scheme for a parabolic PDE [Assessment]

3.5.7 NSA/Optimization Algorithms (CS Core: 1 hr, KA Core: 1 hr)

Gradient methods, Newton and quasi-Newton methods, constrained optimization, and convex optimization algorithms foundational for training machine learning models.

Topics:

Core

- Gradient descent and line search methods: Armijo-Wolfe conditions and convergence rates
- Newton's method and quasi-Newton methods (BFGS, L-BFGS)
- Constrained optimization: KKT conditions, penalty methods, and sequential quadratic programming
- Convex optimization algorithms: interior-point methods and ADMM
- Stochastic gradient descent (SGD), variance reduction, and Adam optimizer

Learning Outcomes:

Core:

1. Explain the convergence guarantees of gradient descent for smooth convex functions [Familiarity]
2. Apply Newton and quasi-Newton methods to unconstrained optimization problems [Usage]
3. Formulate a constrained optimization problem as a KKT system and apply an interior-point method [Assessment]

3.5.8 NSA/Finite Element Method (CS Core: 1 hr, KA Core: 1 hr)

Variational formulation, Galerkin methods, finite element spaces, error estimates, and applications to elliptic PDEs.

Topics:

Core

- Weak (variational) formulation of boundary value problems and Sobolev spaces
- Galerkin method: finite-dimensional approximation spaces and the stiffness matrix
- Triangular and quadrilateral finite element spaces: Lagrange elements and conformity
- A priori error estimates: Céa's lemma and interpolation error bounds

Non Core

- A posteriori error estimates and adaptive mesh refinement

Learning Outcomes:

Core:

1. Derive the weak formulation of an elliptic BVP and show its equivalence to the strong form [Familiarity]
2. Assemble the global stiffness matrix and load vector for a linear finite element discretization [Usage]
3. Estimate the H^1 error of a finite element solution using Céa's lemma and interpolation theory [Assessment]

3.5.9 NSA/Parallel and High-Performance Computing (CS Core: 1 hr, KA Core: 1 hr)

Parallel architectures, performance models, parallel algorithms for linear algebra, and GPU computing.

Topics:

Core

- Parallel architectures: shared memory (OpenMP), distributed memory (MPI), and GPU (CUDA)
- Performance models: Amdahl's law, roofline model, and communication complexity
- Parallel algorithms for dense and sparse linear algebra (ScaLAPACK, PETSc)
- Domain decomposition and parallel PDE solvers

Non Core

- Automatic differentiation (forward and reverse mode) and its role in scientific ML

Learning Outcomes:

Core:

1. Explain Amdahl's law and identify the bottlenecks limiting parallel speedup in a given algorithm [Familiarity]
2. Implement a parallel linear algebra routine using MPI or OpenMP and measure parallel efficiency [Usage]

NonCore:

3. Apply automatic differentiation to compute gradients of complex scientific functions for optimization [Assessment]

3.6 Mathematical Modeling and Simulation (MMS)

This area covers mathematical tools for describing and simulating computational and physical systems, spanning ordinary differential equations, dynamical systems, control theory, and stochastic differential equations, with direct applications to simulation engineering, robotics, and AI systems.

Knowledge Area (KA)	CS Core	KA Core
⭐ 3.6.1 First-Order Ordinary Differential Equations (Pg.. 156)	1	1
⭐ 3.6.2 Higher-Order Differential Equations and Systems (Pg.. 157)	1	1
⭐ 3.6.3 Dynamical Systems and Chaos (Pg.. 157)	1	1
⭐ 3.6.4 Mathematical Control Theory (Pg.. 157)	1	1
⭐ 3.6.5 Stochastic Differential Equations (Pg.. 158)	1	1

Table 3.6: List of KUs in the Mathematical Modeling and Simulation area.

3.6.1 MMS/First-Order Ordinary Differential Equations (CS Core: 1 hr, KA Core: 1 hr)

Theory and solution methods for first-order ODEs and classification of equation types.

Topics:

Core

- First-order ODEs: separable, linear, exact equations, and integrating factors

Learning Outcomes:
Core:

1. Classify ODEs by order, linearity, and type, and select an appropriate solution method [Familiarity]

3.6.2 MMS/Higher-Order Differential Equations and Systems (CS Core: 1 hr, KA Core: 1 hr)

Theory and solution methods for higher-order ODEs and linear systems: characteristic equations, eigenvalue systems, existence and uniqueness theorems, and Laplace transform methods used in system modeling.
Topics:
Core

- Higher-order linear ODEs: characteristic equations, variation of parameters, and undetermined coefficients
- Systems of linear ODEs: matrix exponential, eigenvalue methods, and phase portraits
- Existence and uniqueness theorems (Picard-Lindelöf) and continuation of solutions
- Laplace transform methods for solving ODEs with discontinuous forcing

Learning Outcomes:
Core:

1. Solve linear systems of ODEs using eigenvalue methods and sketch their phase portraits [Usage]
2. Apply the Laplace transform to solve ODEs with piecewise and impulsive forcing functions [Assessment]

3.6.3 MMS/Dynamical Systems and Chaos (CS Core: 1 hr, KA Core: 1 hr)

Phase plane analysis, stability, bifurcations, limit cycles, and chaos in continuous and discrete systems, relevant to network dynamics, biological computing models, and agent systems.
Topics:
Core

- Phase portraits of 2D systems: equilibria, stability, and limit cycles
- Lyapunov stability theory and the linearization (Hartman-Grobman) theorem
- Bifurcation theory: saddle-node, pitchfork, Hopf, and global bifurcations
- Chaos: Lorenz system, sensitivity to initial conditions, and Lyapunov exponents

Learning Outcomes:
Core:

1. Sketch phase portraits of 2D autonomous systems and classify equilibria by stability [Familiarity]
2. Apply Lyapunov's method and the Hartman-Grobman theorem to assess stability of equilibria [Usage]
3. Identify bifurcation types in parameterized systems and trace the evolution of invariant sets [Assessment]

3.6.4 MMS/Mathematical Control Theory (CS Core: 1 hr, KA Core: 1 hr)

State-space models, controllability and observability, feedback design, Lyapunov stability, and optimal control, with applications to robotics, embedded systems, and autonomous agents.
Topics:
Core

- State-space representation of linear systems: transfer functions and realizations
- Controllability and observability: rank conditions and Kalman decomposition
- Feedback stabilization: pole placement, LQR design, and the separation principle
- Lyapunov stability for nonlinear systems and input-output stability
- Optimal control: the Pontryagin Maximum Principle and Hamilton-Jacobi-Bellman equation

Learning Outcomes:
Core:

1. Determine the controllability and observability of a linear system from its state-space matrices [Familiarity]
2. Design a state-feedback controller using pole placement or LQR optimization [Usage]
3. Apply the Pontryagin Maximum Principle to solve a basic optimal control problem [Assessment]

3.6.5 MMS/Stochastic Differential Equations (CS Core: 1 hr, KA Core: 1 hr)

Itô calculus, SDEs driven by Brownian motion, the Fokker-Planck equation, and numerical simulation of SDEs, foundational for probabilistic ML and diffusion models.

Topics:

Core

- Itô integral, quadratic variation, and Itô's formula
- Existence and uniqueness of solutions to SDEs under Lipschitz conditions
- Fokker-Planck equation: derivation and connection to SDEs
- Feynman-Kac formula and connections to parabolic PDEs
- Euler-Maruyama and Milstein schemes for numerical simulation of SDEs

Learning Outcomes:

Core:

1. State Itô's formula and apply it to compute the differential of a function of a stochastic process [Familiarity]
2. Derive the Fokker-Planck equation from a given SDE and interpret it as a probability flow [Usage]
3. Implement the Euler-Maruyama scheme to simulate trajectories of a stochastic differential equation [Assessment]

3.7 Mathematical Foundations of Data Science (FDS)

This area covers the mathematical theory underlying modern data science and machine learning: statistical learning theory, optimization, information theory, dimensionality reduction, and the mathematics of deep learning, all essential for Computing students across AI, data science, and software engineering.

Knowledge Area (KA)	CS Core	KA Core
⭐ 3.7.1 Statistical Learning Theory (Pg.. 158)	1	1
⭐ 3.7.2 Optimization for Machine Learning (Pg.. 159)	1	1
⭐ 3.7.3 Information Theory (Pg.. 159)	1	1
⭐ 3.7.4 Dimensionality Reduction (Pg.. 159)	1	1
⭐ 3.7.5 Mathematical Foundations of Deep Learning (Pg.. 160)	1	1

Table 3.7: List of KUs in the Mathematical Foundations of Data Science area.

3.7.1 FDS/Statistical Learning Theory (CS Core: 1 hr, KA Core: 1 hr)

PAC learning framework, VC dimension, Rademacher complexity, generalization bounds, and the bias-variance trade-off.

Topics:

Core

- PAC learning framework: sample complexity, hypothesis classes, and learnability
- VC dimension: definition, examples, and the fundamental theorem of learning
- Rademacher complexity and uniform convergence bounds
- Bias-variance decomposition and the double descent phenomenon
- Regularization: Tikhonov regularization and structural risk minimization

Learning Outcomes:

Core:

1. State the PAC learning definition and determine whether a hypothesis class is PAC-learnable [Familiarity]
2. Compute the VC dimension of standard hypothesis classes and derive sample complexity bounds [Usage]
3. Analyze generalization error using Rademacher complexity and explain the double descent phenomenon [Assessment]

3.7.2 FDS/Optimization for Machine Learning (CS Core: 1 hr, KA Core: 1 hr)

Convex optimization theory, SGD and its variants, adaptive methods, and non-convex optimization landscapes for training learning models.

Topics:

Core

- Convexity, strong convexity, and Fenchel duality in optimization
- Stochastic gradient descent: convergence rates for convex and non-convex objectives
- Adaptive gradient methods: AdaGrad, RMSProp, and Adam
- Non-convex optimization: saddle points, local minima, and loss landscape geometry of neural networks
- Proximal gradient methods and coordinate descent for structured problems

Learning Outcomes:

Core:

1. Explain the convergence guarantees of SGD for smooth convex functions and identify the role of the learning rate [Familiarity]
2. Apply Adam and proximal gradient methods to train regularized machine learning models [Usage]
3. Analyze the loss landscape of an overparameterized network and explain why SGD finds good minima [Assessment]

3.7.3 FDS/Information Theory (CS Core: 1 hr, KA Core: 1 hr)

Shannon entropy, mutual information, channel capacity, rate-distortion theory, and applications to machine learning and data compression in computing.

Topics:

Core

- Shannon entropy, joint and conditional entropy, and the chain rule
- Mutual information, KL divergence, and the data processing inequality
- Channel capacity: Shannon's noisy channel coding theorem and the binary symmetric channel
- Rate-distortion theory and the trade-off between compression and fidelity
- Information-theoretic foundations of ML: MDL, variational inference (ELBO), and the information bottleneck

Learning Outcomes:

Core:

1. Compute Shannon entropy and mutual information for discrete distributions and interpret their meaning [Familiarity]
2. Apply Shannon's channel capacity theorem to determine the maximum reliable transmission rate [Usage]
3. Analyze a learning algorithm using the information bottleneck principle and relate compression to generalization [Assessment]

3.7.4 FDS/Dimensionality Reduction (CS Core: 1 hr, KA Core: 1 hr)

Linear and nonlinear dimensionality reduction: PCA, random projections, manifold learning, and modern methods (t-SNE, UMAP), widely used in data science and visualization.

Topics:

Core

- PCA: singular value decomposition, explained variance, and optimal low-rank approximation

- Johnson-Lindenstrauss lemma and random projections for dimensionality reduction
- Manifold learning: Isomap, locally linear embedding (LLE), and spectral embedding
- Modern nonlinear methods: t-SNE and UMAP; theory, parameters, and limitations
- Autoencoders and variational autoencoders (VAEs) as nonlinear dimensionality reduction

Learning Outcomes:

Core:

1. Explain the Johnson-Lindenstrauss lemma and justify why random projections preserve distances [Familiarity]
2. Apply PCA and t-SNE to high-dimensional data and interpret the resulting low-dimensional embeddings [Usage]
3. Compare linear and nonlinear dimensionality reduction methods with respect to geometry preservation and computational cost [Assessment]

3.7.5 FDS/Mathematical Foundations of Deep Learning (CS Core: 1 hr, KA Core: 1 hr)

Universal approximation, neural tangent kernel, implicit bias of SGD, overparameterization, and geometric deep learning — the frontier mathematics behind modern AI.

Topics:

Core

- Universal approximation theorems: width and depth trade-offs for ReLU networks
- Neural tangent kernel (NTK) and the infinite-width limit of neural networks
- Implicit regularization and implicit bias of SGD toward minimum-norm solutions
- Benign overfitting and double descent in overparameterized models
- Geometric deep learning: equivariance, group symmetry, and graph neural networks

Learning Outcomes:

Core:

1. State the universal approximation theorem and identify its assumptions and limitations [Familiarity]
2. Explain the neural tangent kernel and its role in understanding training dynamics of wide networks [Usage]
3. Analyze the implicit bias of gradient descent and relate it to generalization in overparameterized regimes [Assessment]

3.8 Quantum Information and Computing (QIC)

This area covers the mathematical theory of quantum information and computation, including the linear algebraic formalism of quantum mechanics, quantum algorithms, entanglement theory, error correction, and tensor network methods. This frontier knowledge area is increasingly essential for Computing students engaged with quantum software and post-quantum security.

Knowledge Area (KA)	CS Core	KA Core
⭐ 3.8.1 Quantum Mechanics Foundations for Computing (Pg.. 160)	1	1
⭐ 3.8.2 Quantum Circuits and Gates (Pg.. 161)	1	1
⭐ 3.8.3 Quantum Algorithms (Pg.. 161)	1	1
⭐ 3.8.4 Entanglement and Bell Inequalities (Pg.. 162)	1	1

Table 3.8: List of KUs in the Quantum Information and Computing area.

3.8.1 QIC/Quantum Mechanics Foundations for Computing (CS Core: 1 hr, KA Core: 1 hr)

Postulates of quantum mechanics formulated for computation: state spaces, observables, measurement, and unitary evolution.

Topics:
Core
- Hilbert space formalism: state vectors, Dirac notation, and inner products
- Observables as Hermitian operators: eigenvalues, eigenstates, and the spectral theorem
- Measurement postulate: Born rule, projective measurements, and collapse of the state
- Unitary evolution: the Schrödinger equation and the time evolution operator
- Density matrices: mixed states, partial trace, and the Bloch sphere representation

Learning Outcomes:
Core:
1. State the postulates of quantum mechanics and represent single-qubit states on the Bloch sphere [Familiarity]
2. Compute measurement probabilities and post-measurement states using the Born rule and projection operators [Usage]
3. Distinguish pure from mixed states using density matrices and the partial trace over a subsystem [Assessment]

3.8.2 QIC/Quantum Circuits and Gates (CS Core: 1 hr, KA Core: 1 hr)

Qubits, quantum gates, quantum circuits, universality, and the circuit model of quantum computation.
Topics:
Core
- Single-qubit gates: Pauli, Hadamard, phase, and rotation gates
- Multi-qubit gates: CNOT, Toffoli, and controlled-U gates
- Quantum circuits: circuit diagrams, depth, and reversibility
- Universality: the Solovay-Kitaev theorem and universal gate sets
- Quantum parallelism, interference, and the role of superposition in computation

Learning Outcomes:
Core:
1. Construct quantum circuits using standard gate sets and represent their action on basis states [Familiarity]
2. Analyze the action of a quantum circuit on an input state using matrix multiplication and Dirac notation [Usage]
3. Justify the universality of a gate set using the Solovay-Kitaev theorem and approximate a target unitary [Assessment]

3.8.3 QIC/Quantum Algorithms (CS Core: 1 hr, KA Core: 1 hr)

Quantum Fourier transform, phase estimation, Shor's factoring algorithm, Grover's search, and HHL for linear systems.
Topics:
Core
- Quantum Fourier transform: circuit construction and complexity analysis
- Quantum phase estimation algorithm and its role as a subroutine
- Shor's factoring algorithm: period finding and reduction to factoring
- Grover's search algorithm: quadratic speedup and amplitude amplification

Non Core
- HHL algorithm for linear systems and quantum speedup for machine learning

Learning Outcomes:
Core:
1. Describe the quantum Fourier transform and explain how phase estimation extracts eigenvalue information [Familiarity]
2. Trace through Shor's and Grover's algorithms step by step and identify the source of quantum speedup [Usage]

NonCore:
3. Analyze the resource requirements of the HHL algorithm and identify the conditions under which it achieves exponential speedup [Assessment]

3.8.4 QIC/Entanglement and Bell Inequalities (CS Core: 1 hr, KA Core: 1 hr)

Quantum entanglement: Bell states, Bell inequalities, CHSH inequality, no-cloning, and entanglement measures.

Topics:

Core

- Bell states, entangled versus separable states, and the Schmidt decomposition
- Bell inequalities and the CHSH inequality: local hidden variable theories vs. quantum mechanics
- No-cloning theorem and no-communication theorem
- Entanglement measures: entanglement entropy, concurrence, and negativity
- Quantum teleportation and superdense coding protocols

Learning Outcomes:

Core:

1. Explain what entanglement is and why Bell inequality violations rule out local hidden variable theories [Familiarity]
2. Construct Bell state circuits and verify entanglement using the CHSH inequality [Usage]
3. Analyze the quantum teleportation protocol and verify that no information travels faster than light [Assessment]

3.9 Electromagnetism (ELM)

This area covers the physics of electric and magnetic fields, circuits, and electromagnetic waves at the level needed for Computing students to understand electronics hardware, digital circuits, signal transmission, and the physical foundations of computing devices.

Knowledge Area (KA)	CS Core	KA Core
⭐ 3.9.1 Electrostatics: Electric Field and Gauss's Law (Pg.. 162)	1	1
⭐ 3.9.2 Electric Current, Circuits, and Kirchhoff's Laws (Pg.. 163)	1	1
⭐ 3.9.3 Maxwell's Equations and Gauge Transformations (Pg.. 163)	1	1

Table 3.9: List of KUs in the Electromagnetism area.

3.9.1 ELM/Electrostatics: Electric Field and Gauss's Law (CS Core: 1 hr, KA Core: 1 hr)

Analysis of stationary charges, the vector nature of the electric field, and the application of Gauss's Law, providing the basis for understanding capacitors and charge storage in computing hardware.

Topics:

Core

- Coulomb's law for point charges and the principle of linear superposition
- The electric field vector $\vec{E}$ for discrete and continuous charge distributions
- Gauss's Law in integral form and its relation to enclosed charge
- Differential form of Gauss's Law: $\nabla \cdot \vec{E} = \rho/\epsilon_0$
- Applications of Gauss's Law to spherical and cylindrical symmetry

Learning Outcomes:

Core:

1. Calculate the electrostatic force and field for charge distributions using superposition and Gauss's Law [Usage]
2. Evaluate the electric flux through surfaces and relate it to enclosed charge [Assessment]
3. Determine volume charge density from a given expression of the electric field using Gauss's law in differential form [Usage]

3.9.2 ELM/Electric Current, Circuits, and Kirchhoff's Laws (CS Core: 1 hr, KA Core: 1 hr)

Study of electric charge flow, resistance, and DC circuit analysis using Kirchhoff's laws, essential for understanding digital logic hardware and embedded systems.

Topics:

Core

- Definition of current, current density, and the continuity equation
- Ohm's law, resistivity, and temperature dependence of resistance
- Kirchhoff's Current Law (KCL) for circuit node analysis
- Kirchhoff's Voltage Law (KVL) for circuit loop analysis
- Series and parallel circuit simplification and equivalent resistance
- RC and RL circuit transients and engineering time constants

Learning Outcomes:

Core:

1. Apply KCL and KVL to solve multi-loop DC circuits relevant to digital hardware design [Usage]
2. Calculate power dissipated and determine time constants for RC and RL circuits [Assessment]
3. Analyze the transient response of RC and RL circuits and relate it to signal timing in digital systems [Usage]

3.9.3 ELM/Maxwell's Equations and Gauge Transformations (CS Core: 1 hr, KA Core: 1 hr)

The unified set of equations governing all classical electromagnetic phenomena and the Poynting vector, providing the physics basis for wireless communication and electromagnetic compatibility.

Topics:

Core

- Maxwell's displacement current and its role in completing Ampere's Law
- The complete set of four Maxwell's equations in integral and differential forms
- Poynting's theorem and energy conservation in electromagnetic fields
- The Poynting vector $\vec{S} = \vec{E} \times \vec{H}$ as energy flux density
- Gauge invariance and the formulation of potentials in electrodynamics

Learning Outcomes:

Core:

1. State all four Maxwell equations and identify their physical content [Familiarity]
2. Calculate the displacement current and the Poynting vector for an electromagnetic configuration [Usage]
3. Analyze energy flow and power transport in electromagnetic systems including transmission lines [Assessment]

3.10 Thermodynamics and Statistical Mechanics (TSM)

This area covers the laws of energy transfer, entropy, equilibrium states, and statistical foundations of macroscopic behavior. For Computing students, it provides the physical basis for energy-efficient system design, thermal management of hardware, and the statistical mechanics underpinning of machine learning algorithms.

Knowledge Area (KA)	CS Core	KA Core
⭐ 3.10.1 Laws of Thermodynamics and State Functions (Pg.. 164)	1	1
⭐ 3.10.2 Kinetic Theory of Gases (Pg.. 164)	1	1
⭐ 3.10.3 Heat Engines, Entropy, and the Second Law (Pg.. 164)	1	1

Table 3.10: List of KUs in the Thermodynamics and Statistical Mechanics area.

3.10.1 TSM/Laws of Thermodynamics and State Functions (CS Core: 1 hr, KA Core: 1 hr)

Fundamental principles governing energy, heat, and work in thermodynamic systems, and the mathematical definition of state variables.

Topics:

Core

- Zeroth Law of Thermodynamics and temperature equilibrium
- First Law of Thermodynamics: $dU = dQ - dW$ for engineering systems
- Enthalpy $H = U + PV$ and its role in flow processes
- Heat capacities C_V and C_P and their measurement and application

Learning Outcomes:

Core:

1. State the three laws of thermodynamics and give a physical interpretation of each [Familiarity]
2. Calculate work done by a gas during isothermal and adiabatic expansions [Usage]
3. Evaluate internal energy changes and apply the First Law to thermodynamic processes in computing systems [Assessment]

3.10.2 TSM/Kinetic Theory of Gases (CS Core: 1 hr, KA Core: 1 hr)

Microscopic model of gases relating molecular motion to macroscopic properties such as pressure, temperature, and transport coefficients.

Topics:

Core

- Derivation of macroscopic pressure from molecular momentum transfer
- Maxwell-Boltzmann speed distribution and its implications
- Equipartition theorem and degrees of freedom for monatomic and polyatomic gases
- Mean free path and its relevance to gas flow and vacuum systems

Learning Outcomes:

Core:

1. Describe the microscopic basis of pressure and temperature in an ideal gas [Familiarity]
2. Compute the rms and average molecular velocities using the Maxwell-Boltzmann distribution [Usage]
3. Apply the equipartition theorem to estimate heat capacities of gases and relate to energy storage in computing environments [Assessment]

3.10.3 TSM/Heat Engines, Entropy, and the Second Law (CS Core: 1 hr, KA Core: 1 hr)

Thermal efficiency limits, entropy generation, and the thermodynamic constraints, relevant to energy efficiency in data centers and computing hardware.

Topics:

Core

- Kelvin-Planck and Clausius statements of the Second Law and their implications
- The Carnot cycle and maximum efficiency $\eta = 1 - T_c/T_h$
- Clausius inequality and the thermodynamic definition of entropy
- Boltzmann's entropy formula $S = k_B \ln \Omega$ and its statistical interpretation
- Landauer's principle: energy cost of irreversible computation and bit erasure

Learning Outcomes:

Core:

1. Explain the Second Law of Thermodynamics and its connection to the direction of spontaneous processes [Familiarity]
2. Calculate maximum thermal efficiency of a heat engine operating between two temperatures [Usage]
3. Apply Landauer's principle to analyze the thermodynamic cost of computing operations and memory erasure [Assessment]

3.11 Computational and Data-Driven Physics (CDP)

This area covers numerical simulation methods, Monte Carlo techniques, machine learning for physical discovery, and physics-informed neural networks. It bridges physics and computing, equipping students to build simulations, analyze scientific data, and construct hybrid physics-ML models.

Knowledge Area (KA)	CS Core	KA Core
⭐ 3.11.1 Numerical Methods for Differential Equations in Physics (Pg.. 165)	1	1
⭐ 3.11.2 Monte Carlo Methods and Simulations (Pg.. 165)	1	1
⭐ 3.11.3 Machine Learning for Physical Discovery (Pg.. 166)	1	1

Table 3.11: List of KUs in the Computational and Data-Driven Physics area.

3.11.1 CDP/Numerical Methods for Differential Equations in Physics (CS Core: 1 hr, KA Core: 1 hr)

Discrete approximations for solving fundamental equations of motion and field theories, essential for building physics simulators in computing.

Topics:

Core

- Runge-Kutta methods (RK2, RK4) for initial value problems
- Finite Difference Time Domain (FDTD) for Maxwell's equations
- Boundary value problems: shooting methods and relaxation techniques
- Symplectic integrators: Verlet and leapfrog algorithms for Hamiltonian systems
- Stability analysis and the Courant-Friedrichs-Lewy (CFL) condition

Learning Outcomes:

Core:

1. Explain the stability requirements of numerical ODE integrators and identify the CFL condition [Familiarity]
2. Implement a velocity Verlet algorithm to simulate a physical system with conserved energy [Usage]
3. Evaluate the scaling of discretization errors for different numerical integration orders [Assessment]

3.11.2 CDP/Monte Carlo Methods and Simulations (CS Core: 1 hr, KA Core: 1 hr)

Stochastic sampling techniques for multi-dimensional integration and statistical physics, relevant to probabilistic inference and simulation in computing.

Topics:

Core

- Pseudo-random and quasi-random number generation algorithms
- The Metropolis-Hastings algorithm for Markov Chain Monte Carlo (MCMC)
- Monte Carlo simulation of the Ising model and phase transitions
- Importance sampling and variance reduction techniques
- Statistical error estimation and autocorrelation time in MCMC chains

Learning Outcomes:

Core:

1. Describe the Metropolis-Hastings acceptance criterion and explain its connection to detailed balance [Familiarity]
2. Develop an importance sampling scheme to integrate a peaked multidimensional function [Usage]
3. Analyze the statistical quality of a Monte Carlo estimate and quantify autocorrelation-induced bias [Assessment]

3.11.3 CDP/Machine Learning for Physical Discovery (CS Core: 1 hr, KA Core: 1 hr)

Applying statistical learning to identify patterns and surrogate models in data, including physics-informed neural networks and neural operators for solving PDEs.

Topics:

Core

- Supervised learning: linear regression, LASSO, and Ridge for physics models
- Unsupervised learning: clustering and PCA for detecting phase transitions
- Gaussian Process Regression for uncertainty-aware surrogate modeling
- Symbolic regression for discovering analytical physical laws from data
- Physics-Informed Neural Networks (PINNs): incorporating PDE constraints into neural network loss functions

Learning Outcomes:

Core:

1. Explain how physics constraints can be encoded in the loss function of a neural network [Familiarity]
2. Train a regression or Gaussian Process model to predict a physical quantity and quantify uncertainty [Usage]
3. Evaluate and compare data-driven surrogate models with traditional numerical solvers on accuracy and computational cost [Assessment]

3.12 Classical Mechanics (CME)

Classical Mechanics provides the foundational principles governing the motion of particles and rigid bodies. This computing-focused subset emphasizes kinematics, energy methods, rigid-body rotation, oscillatory phenomena, and variational mechanics relevant to robotics, game engines, physics simulations, and physics-informed machine learning.

Knowledge Area (KA)	CS Core	KA Core
⭐ 3.12.1 Rotational Motion and Angular Momentum (Pg.. 166)	1	1
⭐ 3.12.2 Systems of Particles and Linear Momentum (Pg.. 167)	1	1
⭐ 3.12.3 Kinematics and Dynamics of Particles (Pg.. 168)	1	1
⭐ 3.12.4 Work, Energy, and Conservation Laws (Pg.. 168)	1	1
⭐ 3.12.5 Rigid Body Dynamics (Pg.. 168)	1	1
⭐ 3.12.6 Oscillations and Waves (Pg.. 169)	1	1
⭐ 3.12.7 Lagrangian and Hamiltonian Mechanics (Pg.. 169)	1	1
⭐ 3.12.8 Elasticity (Pg.. 170)	1	1
⭐ 3.12.9 Fluids: Hydrostatics, Hydrodynamics, and Viscosity (Pg.. 170)	1	1
⭐ 3.12.10 Vibrations and Waves (Pg.. 170)	1	1
⭐ 3.12.11 Transport Phenomena (Pg.. 171)	1	1

Table 3.12: List of KUs in the Classical Mechanics area.

3.12.1 CME/Rotational Motion and Angular Momentum (CS Core: 1 hr, KA Core: 1 hr)

Rotational kinematics and dynamics, torque, moment of inertia, rotational energy, and conservation of angular momentum for particles and systems.

Topics:

Core

- Rotational kinematics including angular displacement, velocity, and acceleration (θ, ω, α)
- Torque definition: $\vec{\tau} = \vec{r} \times \vec{F}$ and its magnitude calculation
- Angular momentum of a particle: $\vec{L} = \vec{r} \times \vec{p}$
- Rotational form of Newton's second law: $\vec{\tau}_{net} = \frac{d\vec{L}}{dt}$

- Moment of inertia $I = \sum m_i r_i^2$ and the parallel axis theorem
- Rotational kinetic energy: $K_{rot} = \frac{1}{2} I\omega^2$
- Conservation of angular momentum for isolated systems
- Gyroscopic motion and the phenomenon of precession
- Rolling without slipping dynamics: $v_{CM} = R\omega$

Elective

- Coriolis effect as a manifestation of rotational dynamics in non-inertial frames

Learning Outcomes:

Core:

1. Relate linear and angular kinematic quantities for circular motion [Usage]
2. Calculate torque about a given axis for a force applied at a specific point [Usage]
3. Determine the moment of inertia for simple geometries such as a rod, hoop, disk, or sphere [Usage]
4. Apply conservation of angular momentum to analyze changes in rotational speed when the moment of inertia changes [Assessment]
5. Compute rotational kinetic energy for a rotating rigid body [Usage]
6. Analyze the motion of objects rolling without slipping down an incline [Assessment]
7. Predict the direction of precession for a spinning top or gyroscope [Assessment]

Elective:

8. Explain the Coriolis effect in terms of rotating reference frames and its effect on projectiles [Familiarity]

3.12.2 CME/Systems of Particles and Linear Momentum (CS Core: 1 hr, KA Core: 1 hr)

Dynamics of multi-particle systems: center of mass motion, linear momentum conservation, internal and external forces, and variable mass systems.

Topics:

Core

- Definition of center of mass: $\vec{R}_{CM} = \frac{1}{M} \sum m_i \vec{r}_i$
- Linear momentum of a particle and system: $\vec{p} = m\vec{v}$ and $\vec{P} = M\vec{V}_{CM}$
- Newton's second law for systems: $\vec{F}_{ext} = \frac{d\vec{P}}{dt}$
- Conservation of linear momentum for isolated systems
- Distinction between internal and external forces in a system
- Two-body collisions including elastic, inelastic, and completely inelastic cases
- Coefficient of restitution and its physical meaning in collisions
- Systems with variable mass such as the rocket equation: $M\frac{dV}{dt} = v_{ex}\frac{dM}{dt}$
- Many-particle systems and their center of mass dynamics

Elective

- Continuous mass distributions and their center of mass calculation using integration

Learning Outcomes:

Core:

1. Locate the center of mass for a discrete set of particles in One, Two, and Three dimensions [Usage]
2. Determine the total linear momentum of a system from the motion of its center of mass [Usage]
3. Apply conservation of linear momentum to solve collision problems in One and Two dimensions [Usage]
4. Classify collisions as elastic, inelastic, or completely inelastic based on kinetic energy conservation [Familiarity]
5. Calculate the coefficient of restitution from pre-collision and post-collision velocities [Usage]
6. Solve the rocket equation to determine velocity as a function of ejected mass [Usage]
7. Analyze how internal forces affect individual particles but not the total momentum of a system [Assessment]

Elective:

8. Integrate to find the center of mass of symmetric continuous bodies such as rods, disks, and spheres [Usage]

3.12.3 CME/Kinematics and Dynamics of Particles (CS Core: 1 hr, KA Core: 1 hr)

Mathematical description of particle motion and the forces governing it, with emphasis on phase space analysis and numerical integration methods used in robotics and simulation.

Topics:

Core

- Vector description of position, velocity, and acceleration; Newton's three laws and inertial reference frames
- Formulation and analytical solution of equations of motion for constant and variable forces
- Phase space diagrams (position vs. velocity) for visualizing mechanical state, stability, and periodicity
- Numerical integration of equations of motion using Euler, Verlet, and Runge-Kutta methods
- Constrained motion and holonomic constraints relevant to robotic linkages and articulated bodies

Learning Outcomes:

Core:

1. Identify all forces acting on a particle in a given mechanical scenario and draw a correct free-body diagram [Familiarity]
2. Implement a numerical integration algorithm such as the Runge-Kutta method to compute the trajectory of a particle under variable forces [Usage]
3. Construct and interpret phase space diagrams to characterize the stability and periodicity of a simulated mechanical system [Assessment]

3.12.4 CME/Work, Energy, and Conservation Laws (CS Core: 1 hr, KA Core: 1 hr)

Energy methods and conservation principles underpinning physics engines, collision resolution, and energy-aware optimization in computing systems.

Topics:

Core

- Work-energy theorem relating net work done to change in kinetic energy
- Potential energy functions and the force-gradient relation for conservative force fields
- Conservation of total mechanical energy and conditions for its applicability
- Energy diagrams for identifying equilibrium points, turning points, and stability
- Elastic and inelastic collision models: momentum and energy exchange in physics engine implementations

Learning Outcomes:

Core:

1. Explain the distinction between conservative and non-conservative forces and the conditions under which mechanical energy is conserved [Familiarity]
2. Apply energy conservation methods to compute speeds and positions in one-dimensional motion problems common in physics engine implementations [Usage]
3. Analyze elastic and inelastic collision events, computing post-collision velocities and quantifying energy dissipation [Assessment]

3.12.5 CME/Rigid Body Dynamics (CS Core: 1 hr, KA Core: 1 hr)

Rotational dynamics of rigid bodies including torque, moment of inertia, and angular momentum, directly applicable to game engine physics and robot kinematics.

Topics:

Core

- Torque definition and moment of inertia for standard geometries: rod, disk, sphere, and box
- Angular momentum vector and its conservation for isolated rotating systems
- Rotational equations of motion and coupling between translational and rotational degrees of freedom

- Euler angles and quaternion representations for three-dimensional orientation tracking
- Rolling without slipping condition and constraint forces in rigid body systems

Learning Outcomes:

Core:

1. Explain how the moment of inertia governs the rotational response of a rigid body to applied torques [Familiarity]
2. Compute the angular velocity and orientation of a rigid body given an applied torque history and initial conditions [Usage]
3. Design a rigid body simulation for a game engine scenario, selecting appropriate numerical integrators and orientation representations [Assessment]

3.12.6 CME/Oscillations and Waves (CS Core: 1 hr, KA Core: 1 hr)

Simple harmonic motion, damping, resonance, and wave propagation, providing the mechanical foundations for signal processing, audio synthesis, and vibration analysis in computing applications.

Topics:

Core

- Simple harmonic oscillator: frequency, period, amplitude, phase, and energy
- Damped oscillators: underdamped, critically damped, and overdamped regimes and their time-domain signatures
- Forced oscillations, resonance frequency, amplitude response, and the quality factor Q
- Coupled oscillators and normal mode decomposition for multi-degree-of-freedom systems
- Mechanical wave propagation: wave equation, dispersion relation, and group versus phase velocity

Learning Outcomes:

Core:

1. Describe the physical behavior of underdamped, critically damped, and overdamped oscillators and relate each regime to applications in control and audio systems [Familiarity]
2. Solve the damped harmonic oscillator equation analytically and numerically for given initial conditions and forcing functions [Usage]
3. Analyze the amplitude-frequency response of a driven oscillator and quantify the effect of damping on resonance bandwidth and peak gain [Assessment]

3.12.7 CME/Lagrangian and Hamiltonian Mechanics (CS Core: 1 hr, KA Core: 1 hr)

Variational formulations of mechanics underpinning symplectic integrators, differentiable physics simulations, and physics-informed machine learning.

Topics:

Core

- Hamilton's principle of stationary action and its role in deriving equations of motion
- Euler-Lagrange equations of motion in generalized coordinates for constrained mechanical systems
- Hamiltonian formulation: canonical coordinates, phase flow, and energy interpretation
- Symplectic integrators (Verlet, leapfrog) and their energy-conserving properties for long-run simulations
- Noether's theorem connecting continuous symmetries to conserved quantities

Learning Outcomes:

Core:

1. Explain the variational principle underlying Lagrangian mechanics and contrast it with the Newtonian force-based approach [Familiarity]
2. Construct the Lagrangian for a mechanical system with holonomic constraints and derive its equations of motion via the Euler-Lagrange equations [Usage]
3. Evaluate the long-run energy conservation of symplectic versus non-symplectic integrators through numerical experiments on a Hamiltonian system [Assessment]

3.12.8 CME/Elasticity (CS Core: 1 hr, KA Core: 1 hr)

Elastic properties of solids under axial, shear, and volumetric load: stress, strain, and scalar elastic moduli.

Topics:

Core

- Definition of stress (force/area) and strain ($\Delta L/L$)
- Young's modulus and Hooke's law for longitudinal strain: $\sigma = E\epsilon$
- Shear modulus (rigidity) and shear strain
- Bulk modulus (compressibility) and volumetric strain
- Elastic limit, plastic region, and fracture point on the stress-strain diagram

Elective

- Elastic energy stored in a deformed material

Learning Outcomes:

Core:

1. Apply the scalar Hooke's law to calculate the deformation of a rod under axial load [Usage]
2. Calculate Young's, shear, or bulk modulus from experimental stress-strain data [Usage]
3. Identify the elastic region, elastic limit, and fracture point on a stress-strain curve [Familiarity]

Elective:

4. Calculate the elastic potential energy stored in a deformed spring or rod [Usage]

3.12.9 CME/Fluids: Hydrostatics, Hydrodynamics, and Viscosity (CS Core: 1 hr, KA Core: 1 hr)

Statics and dynamics of incompressible fluids: pressure, buoyancy, the continuity and Bernoulli equations, and an introduction to viscous flow.

Topics:

Core

- Density, pressure, and its variation with depth in a fluid at rest: $p = p_0 + \rho gh$
- Pascal's principle and its applications (hydraulic press)
- Archimedes' principle, buoyant force, and flotation
- Pressure measurement with manometers and barometers
- Continuity equation for incompressible flow: $A_1 v_1 = A_2 v_2$
- Bernoulli's equation and energy conservation in ideal flow
- Applications: Venturi tube, Torricelli's theorem, Pitot tube

Elective

- Viscosity and Poiseuille's law for laminar flow in pipes
- Reynolds number and the transition to turbulent flow
- Stokes' law for the drag force on a sphere in a viscous fluid

Learning Outcomes:

Core:

1. Calculate the pressure at a point in a fluid at rest given depth and density [Usage]
2. Apply Archimedes' principle to determine whether an object floats, sinks, or remains in equilibrium [Usage]
3. Apply the continuity and Bernoulli equations to solve flow problems in pipes of varying cross-section [Usage]
4. Interpret manometer and barometer readings to determine absolute and gauge pressures [Assessment]

Elective:

5. Apply Poiseuille's law to calculate the flow rate of a viscous fluid in a cylindrical tube [Usage]
6. Estimate the Reynolds number of a flow and predict whether it is laminar or turbulent [Assessment]

3.12.10 CME/Vibrations and Waves (CS Core: 1 hr, KA Core: 1 hr)

Simple harmonic motion, damped and forced oscillations, and the propagation, superposition, and resonance of mechanical waves.

Topics:
Core
- Simple harmonic motion (SHM): equation of motion, position, velocity, and acceleration
- Oscillating systems: spring-mass and simple/physical pendulum
- Energy in simple harmonic motion
- Damped and forced oscillations; introduction to resonance
- Transverse and longitudinal waves; propagation speed
- Wave equation and speed of a wave on a taut string: $v = \sqrt{T/\mu}$
- Superposition principle and standing waves
- Sound waves, intensity, and sound intensity level (decibels)

Elective
- Doppler effect for sound waves
- Beats and interference of waves of nearby frequencies

Learning Outcomes:
Core:
1. Derive the equation of motion and the period of a spring-mass system and a simple pendulum [Usage]
2. Calculate the kinetic, potential, and total energy of a simple harmonic oscillator at any instant [Usage]
3. Analyze the amplitude response of a forced oscillator near the resonance frequency [Assessment]
4. Calculate the propagation speed of a wave on a string from the tension and linear density [Usage]
5. Determine the frequencies and wavelengths of the normal modes of a standing wave on a fixed string [Usage]
6. Calculate the sound intensity level in decibels from the intensity of a sound wave [Usage]

Elective:
7. Apply the Doppler effect formula for a moving source and/or observer [Usage]

3.12.11 CME/Transport Phenomena (CS Core: 1 hr, KA Core: 1 hr)

Introduction to transport phenomena in continuous media: heat conduction and mass diffusion as gradient-driven flux processes.

Topics:
Core
- Heat conduction and Fourier's law: $\vec{q} = -k\nabla T$
- Thermal conductivity of materials and thermal resistance in composite walls
- Molecular diffusion and Fick's first law: $\vec{J} = -D\nabla C$

Elective
- Analogy between heat, mass, and momentum transport (Prandtl and Schmidt numbers)

Learning Outcomes:
Core:
1. Apply Fourier's law to calculate the heat flux through a plane or composite wall [Usage]
2. Apply Fick's first law to calculate the diffusive flux of a species in a concentration gradient [Usage]

Elective:
3. Identify mathematical analogies between heat, mass, and momentum transport processes [Familiarity]

3.13 General Chemistry (GCH)

This knowledge area covers the introductory fundamentals of chemistry: the nature of matter and energy, atomic structure, chemical bonding, stoichiometry, and the states of matter. It forms the conceptual foundation on which the other chemistry areas and their applications in engineering and computer science (materials science, nanotechnology, industrial processes) are built.

Knowledge Area (KA)	CS Core	KA Core
⭐ 3.13.1 Atomic Structure (Pg.. 172)	1	1
⭐ 3.13.2 Chemical Bonding (Pg.. 172)	1	1
⭐ 3.13.3 Matter and Energy (Pg.. 173)	1	1
⭐ 3.13.4 States of Matter (Pg.. 173)	1	1
⭐ 3.13.5 Stoichiometry (Pg.. 173)	1	1
⭐ 3.13.6 Environmental and Water Chemistry (Pg.. 174)	1	1
⭐ 3.13.7 Polymers and Geosynthetics (Pg.. 174)	1	1
⭐ 3.13.8 Chemical Equilibrium (Pg.. 174)	1	1
⭐ 3.13.9 Ionic Equilibrium and Salt Hydrolysis (Pg.. 175)	1	1
⭐ 3.13.10 Salts, Solubility, and Precipitation Equilibria (Pg.. 175)	1	1
⭐ 3.13.11 Electrochemistry and Electrolysis (Pg.. 175)	1	1
⭐ 3.13.12 Cement Chemistry (Pg.. 176)	1	1

Table 3.13: List of KUs in the General Chemistry area.

3.13.1 GCH/Atomic Structure (CS Core: 1 hr, KA Core: 1 hr)

Evolution of the atomic model, subatomic particles, electron configuration, and the organization of the periodic table, along with the periodic trends that derive from it.

Topics:

Core

- Historical evolution of the atomic model: Dalton, Thomson, Rutherford, and Bohr
- Subatomic particles, atomic number, mass number, and isotopes
- Electron configuration, quantum numbers, and the Aufbau principle
- Organization of the periodic table: groups, periods, and blocks
- Periodic trends: atomic radius, ionization energy, electron affinity, and electronegativity

Learning Outcomes:

Core:

1. Describe the historical evolution of the atomic model and each model's contribution to the current understanding of the atom [Familiarity]
2. Determine the electron configuration of an atom or ion from its position in the periodic table [Usage]
3. Predict periodic trends (atomic radius, ionization energy) to compare two or more elements [Assessment]

3.13.2 GCH/Chemical Bonding (CS Core: 1 hr, KA Core: 1 hr)

Types of chemical bonding, Lewis structures, molecular geometry via VSEPR theory, and the intermolecular forces that determine the physical properties of substances.

Topics:

Core

- Ionic bonding: formation, lattice energy, and properties of ionic compounds
- Covalent bonding, Lewis structures, and the concept of resonance
- Molecular geometry via Valence Shell Electron Pair Repulsion (VSEPR) theory
- Electronegativity, bond polarity, and molecular polarity
- Intermolecular forces: dipole-dipole, hydrogen bonding, and London dispersion forces

Learning Outcomes:

Core:

1. Draw correct Lewis structures, including resonance cases, for molecules and polyatomic ions [Usage]
2. Predict the molecular geometry and polarity of a molecule by applying VSEPR theory [Usage]
3. Relate the type of intermolecular force present in a substance to its observed physical properties (boiling point, solubility) [Assessment]

3.13.3 GCH/Matter and Energy (CS Core: 1 hr, KA Core: 1 hr)

Classification of matter, its properties and changes, the system of units and significant figures, and the forms of energy involved in physical and chemical processes.

Topics:

Core

- Classification of matter: elements, compounds, homogeneous and heterogeneous mixtures
- Physical and chemical properties, and physical and chemical changes of matter
- International System of units, significant figures, and dimensional analysis
- Forms of energy, heat and temperature; the law of conservation of mass and energy
- The scientific method applied to chemistry: observation, hypothesis, and experimentation

Learning Outcomes:

Core:

1. Classify a sample of matter as an element, compound, or mixture, and distinguish homogeneous from heterogeneous mixtures [Familiarity]
2. Apply dimensional analysis and significant-figure rules in calculations involving chemical measurements [Usage]
3. Relate observed energy changes to specific physical and chemical processes, applying the law of conservation of energy [Assessment]

3.13.4 GCH/States of Matter (CS Core: 1 hr, KA Core: 1 hr)

The gas laws and kinetic-molecular theory, the properties of liquids and solids, phase changes, and the colligative properties of solutions.

Topics:

Core

- Gas laws: Boyle's, Charles's, Avogadro's, and the ideal gas equation
- Kinetic-molecular theory of gases and its applications
- Phase changes, heating curves, and phase diagrams
- Crystalline and amorphous solids; types of crystalline solids
- Colligative properties of solutions: vapor pressure lowering, boiling point elevation, freezing point depression, and osmotic pressure

Learning Outcomes:

Core:

1. Apply the ideal gas equation and the gas laws to solve problems involving pressure, volume, and temperature [Usage]
2. Interpret a phase diagram to identify the states and transitions of a substance under different conditions [Familiarity]
3. Calculate the colligative properties of a solution (vapor pressure, boiling point, freezing point) from its concentration [Assessment]

3.13.5 GCH/Stoichiometry (CS Core: 1 hr, KA Core: 1 hr)

The mole concept, balancing chemical equations, and stoichiometric calculations, including limiting reagent, percent yield, and solution stoichiometry.

Topics:

Core

- Chemical formulas, molar mass, and the mole concept
- Balancing chemical equations and their quantitative interpretation
- Mole-to-mole, mass-to-mass, and mass-to-mole stoichiometric calculations
- Limiting reagent, excess reagent, and percent yield
- Molarity, solution preparation, and solution stoichiometry

Learning Outcomes:

Core:

1. Balance chemical equations by inspection, verifying conservation of mass [Familiarity]
2. Calculate the amounts of reactants and products in a chemical reaction using mole ratios [Usage]

3. Determine the limiting reagent and percent yield of a reaction from experimental data [Assessment]

3.13.6 GCH/Environmental and Water Chemistry (CS Core: 1 hr, KA Core: 1 hr)

Composition and quality of water as a resource, common environmental pollutants, and the fundamentals of environmental chemistry applied to engineering.

Topics:

Core

- Chemical composition of natural water and quality parameters (pH, hardness, dissolved oxygen, BOD)
- Fundamentals of water treatment: coagulation, flocculation, and disinfection
- Major air and water pollutants and their sources
- Biogeochemical cycles of carbon, nitrogen, and phosphorus

Elective

- Atmospheric chemistry and the greenhouse effect

Learning Outcomes:

Core:

1. Interpret water quality parameters to assess fitness for use [Usage]
2. Describe the basic processes of drinking water and wastewater treatment [Familiarity]
3. Identify the main sources and effects of air and water pollutants [Familiarity]

Elective:

4. Explain the chemical mechanism of the greenhouse effect [Usage]

3.13.7 GCH/Polymers and Geosynthetics (CS Core: 1 hr, KA Core: 1 hr)

Classification and properties of polymers, with emphasis on the geosynthetics used in civil engineering.

Topics:

Core

- Classification of polymers: thermoplastics, thermosets, and elastomers
- Polymerization reactions: addition and condensation
- Mechanical and chemical properties of polymers relevant to engineering
- Geosynthetics: geotextiles, geomembranes, and geogrids, and their applications

Elective

- Degradation and durability of polymers under environmental conditions

Learning Outcomes:

Core:

1. Classify a polymer by its structure and thermal behavior [Usage]
2. Identify whether a polymerization reaction is addition or condensation [Familiarity]
3. Select the appropriate geosynthetic type for a given civil engineering application [Usage]

Elective:

4. Assess the durability of a polymer under environmental exposure [Assessment]

3.13.8 GCH/Chemical Equilibrium (CS Core: 1 hr, KA Core: 1 hr)

Fundamentals of chemical equilibrium: the equilibrium constant, the reaction quotient, and Le Chatelier's principle.

Topics:

Core

- The equilibrium constant K_c and K_p, and its relation to reaction stoichiometry
- The reaction quotient Q and predicting the direction of a reaction
- Le Chatelier's principle: effect of concentration, pressure, and temperature
- ICE tables for calculating equilibrium concentrations

Learning Outcomes:

Core:

1. Write the equilibrium constant expression for a given reaction [Usage]

2. Calculate equilibrium concentrations using ICE tables [Usage]
3. Predict the shift of an equilibrium upon changes in concentration, pressure, or temperature using Le Chatelier's principle [Assessment]

3.13.9 GCH/Ionic Equilibrium and Salt Hydrolysis (CS Core: 1 hr, KA Core: 1 hr)

Acid-base equilibrium, the concept of pH, and the hydrolysis of salts in aqueous solution.

Topics:

Core

- Arrhenius, Brønsted-Lowry, and Lewis acid-base theories
- Calculating pH and pOH for strong acids and bases
- Equilibrium of weak acids and bases: K_a, K_b, and degree of dissociation
- Salt hydrolysis and predicting the acidic, basic, or neutral character of a solution
- Buffer solutions and the Henderson-Hasselbalch equation

Learning Outcomes:

Core:

1. Calculate the pH of solutions of strong and weak acids and bases [Usage]
2. Predict the acid-base character of a salt solution from the hydrolysis of its ions [Assessment]
3. Calculate the pH of a buffer solution using the Henderson-Hasselbalch equation [Usage]

3.13.10 GCH/Salts, Solubility, and Precipitation Equilibria (CS Core: 1 hr, KA Core: 1 hr)

The solubility product K_sp and its application to predicting the precipitation and dissolution of sparingly soluble salts.

Topics:

Core

- The solubility product K_{sp} and its relation to molar solubility
- Predicting precipitation by comparing the ion product with K_{sp}
- Common ion effect on solubility

Learning Outcomes:

Core:

1. Calculate K_{sp} from the molar solubility of a salt, and vice versa [Usage]
2. Predict whether a precipitate will form when mixing two solutions given their concentrations and the corresponding K_{sp} [Assessment]

3.13.11 GCH/Electrochemistry and Electrolysis (CS Core: 1 hr, KA Core: 1 hr)

Oxidation-reduction reactions, galvanic and electrolytic cells, and their applications to the corrosion of metals and reinforced concrete.

Topics:

Core

- Oxidation-reduction reactions and balancing by the ion-electron method
- Galvanic cells, standard electrode potentials, and the Nernst equation
- Electrolytic cells and electrolysis; Faraday's laws
- Electrochemical corrosion of metals and protection methods
- Corrosion of reinforcing steel in reinforced concrete

Learning Outcomes:

Core:

1. Balance redox equations by the ion-electron method in acidic or basic medium [Usage]
2. Calculate the potential of a galvanic cell and predict the spontaneity of a redox reaction [Usage]
3. Apply Faraday's laws to calculate the mass deposited or the time in an electrolysis process [Usage]
4. Analyze the conditions that favor the corrosion of a metal or of reinforcing steel in concrete, and propose protection methods [Assessment]

3.13.12 GCH/Cement Chemistry (CS Core: 1 hr, KA Core: 1 hr)

Chemical composition of Portland cement, its hydration reactions, and the resulting properties of concrete.

Topics:

Core

- Chemical composition of Portland cement: calcium silicates, aluminates, and ferroaluminates
- Cement hydration reactions and strength development
- Role of each compound (C_3S, C_2S, C_3A, C_4AF) in cement properties

Elective

- Supplementary cementitious materials: pozzolans, slag, and fly ash

Learning Outcomes:

Core:

1. Describe the chemical composition of Portland cement and its main compounds [Familiarity]
2. Explain the cement hydration process and its relation to strength development [Usage]

Elective:

3. Evaluate the effect of supplementary cementitious materials on concrete properties [Assessment]

3.14 Computational and Theoretical Chemistry (CTC)

Computational and Theoretical Chemistry provides theoretical foundations and computational tools for predicting and simulating chemical behavior. This computing-focused subset covers molecular modeling and simulation, machine learning for chemical property prediction, and AI-driven drug design, directly relevant to bioinformatics, AI, and data science.

Knowledge Area (KA)	CS Core	KA Core
⭐ 3.14.1 Molecular Modeling and Simulation (Pg.. 176)	1	1
⭐ 3.14.2 Machine Learning in Chemical Discovery (Pg.. 177)	1	1
⭐ 3.14.3 Computational Drug Design (Pg.. 177)	1	1

Table 3.14: List of KUs in the Computational and Theoretical Chemistry area.

3.14.1 CTC/Molecular Modeling and Simulation (CS Core: 1 hr, KA Core: 1 hr)

Principles of molecular mechanics and dynamics simulations for studying molecular structure and motion, with connections to physics-based simulation and sampling methods in computing.

Topics:

Core

- Force field potential energy functions: bond, angle, torsion, and non-bonded interaction terms
- Molecular dynamics: numerical integration algorithms, periodic boundary conditions, and thermostats
- Monte Carlo methods for conformational sampling: Metropolis criterion and importance sampling
- Implicit and explicit solvation models for condensed-phase molecular simulations
- Free energy perturbation and thermodynamic integration for relative binding affinity estimation

Learning Outcomes:

Core:

1. Describe the components of a standard molecular mechanics force field and the physical interpretation of each energy term [Familiarity]
2. Set up and run a molecular dynamics simulation, monitoring energy conservation, temperature equilibration, and structural observables [Usage]
3. Evaluate the stability and convergence of a molecular dynamics trajectory using RMSD, energy profiles, and thermodynamic observables [Assessment]

3.14.2 CTC/Machine Learning in Chemical Discovery (CS Core: 1 hr, KA Core: 1 hr)

Application of statistical learning and deep learning to predict molecular properties, accelerate chemical discovery, and generate novel molecular structures relevant to AI and data science.

Topics:

Core

- Molecular representations: SMILES notation, molecular fingerprints, and molecular graph encodings
- Quantitative structure-activity relationship (QSAR) models using supervised regression and classification
- Graph Neural Networks (GNN) for learning molecular properties directly from chemical graph structures
- Generative models (VAE, diffusion, transformer-based) for de novo molecular design
- Model validation: cross-validation, applicability domain, and chemical space diversity in training sets

Learning Outcomes:

Core:

1. Define and compare different molecular representations and explain their suitability for specific machine learning tasks [Familiarity]
2. Train a QSAR regression model to predict a molecular property from descriptors and evaluate its performance using standard regression metrics [Usage]
3. Evaluate the validity, diversity, and novelty of molecules generated by a deep learning model using standard benchmarking metrics [Assessment]

3.14.3 CTC/Computational Drug Design (CS Core: 1 hr, KA Core: 1 hr)

Structure-based and ligand-based computational approaches for identifying and optimizing candidate drug molecules, relevant to AI-driven drug discovery pipelines.

Topics:

Core

- Molecular docking: conformational search algorithms and scoring functions for binding pose prediction
- Virtual screening of compound libraries against protein targets to prioritize drug candidates
- ADMET property prediction: absorption, distribution, metabolism, excretion, and toxicity profiles
- Pharmacophore modeling and ligand-based virtual screening approaches
- Protein binding site identification and characterization for structure-based drug design

Learning Outcomes:

Core:

1. Identify key intermolecular interactions between a ligand and a protein active site from a computationally docked complex [Familiarity]
2. Perform a molecular docking study, rank predicted binding poses by scoring function values, and select candidates for follow-up analysis [Usage]
3. Design and execute a virtual screening workflow to identify potential enzyme inhibitors from a compound database [Assessment]

3.15 Biochemistry and Molecular Biology (BMB)

Biochemistry and Molecular Biology explores chemical processes within living organisms at the molecular level. This computing-focused subset emphasizes biomolecular structure, enzyme kinetics, molecular genetics, and bioinformatics methods directly relevant to protein structure prediction, AlphaFold, sequence analysis, genomics, and AI applications in biology.

Knowledge Area (KA)	CS Core	KA Core
⭐ 3.15.1 Structure and Function of Biomolecules (Pg.. 178)	1	1
⭐ 3.15.2 Molecular Genetics and Enzymology (Pg.. 178)	1	1
⭐ 3.15.3 Bioinformatics and Computational Biology (Pg.. 179)	1	1

Table 3.15: List of KUs in the Biochemistry and Molecular Biology area.

3.15.1 BMB/Structure and Function of Biomolecules (CS Core: 1 hr, KA Core: 1 hr)

Chemical architecture and biological roles of amino acids, proteins, and nucleic acids, with emphasis on protein folding and three-dimensional structure prediction relevant to AlphaFold and structural bioinformatics.

Topics:

Core

- Amino acids: chemical properties, chirality, side chain classes, and peptide bond formation
- Protein structure hierarchy: primary, secondary, tertiary, and quaternary levels and stabilizing forces
- Protein folding problem, energy landscape perspective, misfolding, and relevance to AlphaFold
- DNA and RNA structure: base pairing rules, double helix geometry, and secondary structure motifs
- Protein Data Bank (PDB) and structural databases as data sources for bioinformatics pipelines

Learning Outcomes:

Core:

1. Describe the chemical properties of amino acids and nucleotides as building blocks of proteins and nucleic acids [Familiarity]
2. Analyze protein structures from PDB files, identifying secondary structure elements, active sites, and key functional residues [Usage]
3. Evaluate protein structure prediction methods including energy-based approaches and deep-learning tools such as AlphaFold [Assessment]

3.15.2 BMB/Molecular Genetics and Enzymology (CS Core: 1 hr, KA Core: 1 hr)

Mechanisms of genetic information flow, enzyme kinetics, and gene regulation providing the molecular foundations for genomics, transcriptomics, and systems biology in computing.

Topics:

Core

- The central dogma of molecular biology: replication, transcription, and translation mechanisms
- Michaelis-Menten enzyme kinetics: Km, Vmax, turnover number, and modes of inhibition
- Gene regulation: transcription factors, operons, epigenetic mechanisms, and regulatory network motifs
- Recombinant DNA techniques: PCR, CRISPR-Cas9, and gene expression systems used in biotechnology
- High-throughput omics technologies: genomics, transcriptomics, proteomics, and metabolomics data generation

Learning Outcomes:

Core:

1. Describe the central dogma of molecular biology and identify known exceptions such as reverse transcription and RNA editing [Familiarity]
2. Analyze enzyme kinetic data using Michaelis-Menten and Lineweaver-Burk plots to determine Km and Vmax [Usage]
3. Interpret omics datasets to identify differentially expressed genes or enriched biological pathways in a given condition [Assessment]

3.15.3 BMB/Bioinformatics and Computational Biology (CS Core: 1 hr, KA Core: 1 hr)

Computational methods for analyzing biological sequences, structures, and genome-scale data, directly relevant to sequence alignment, homology modeling, structural bioinformatics, and AI applications in biology.

Topics:

Core

- Sequence alignment algorithms: Needleman-Wunsch (global), Smith-Waterman (local), and BLAST for database search
- Phylogenetic tree construction, evolutionary distance measures, and multiple sequence alignment
- Comparative (homology) modeling and template-based protein structure prediction
- Structural bioinformatics: molecular docking, structure comparison metrics (RMSD, TM-score), and binding site prediction
- Genome-scale data analysis: variant calling, RNA-seq differential expression, and pathway enrichment
- Machine learning in bioinformatics: protein function prediction, drug-target interaction, and single-cell data analysis

Learning Outcomes:

Core:

1. Explain the algorithmic differences between global and local sequence alignment and identify appropriate use cases for each [Familiarity]
2. Perform sequence database searches using BLAST and interpret results in terms of homology, e-values, and evolutionary conservation [Usage]
3. Evaluate the accuracy of computational protein structure predictions using standard metrics such as GDT-TS and TM-score [Assessment]

3.16 Probability Theory (PRO)

This knowledge area covers the mathematical foundations of probability, including sample spaces, random variables, probability distributions, joint distributions, moments, transformations, and the fundamental limit theorems. It provides the theoretical basis for all of statistical inference and modeling.

Knowledge Area (KA)	CS Core	KA Core
⭐ 3.16.1 Sample Spaces and Events (Pg.. 179)	1	1
⭐ 3.16.2 Probability Axioms and Properties (Pg.. 180)	1	1
⭐ 3.16.3 Conditional Probability and Independence (Pg.. 180)	1	1
⭐ 3.16.4 Discrete Random Variables and Distributions (Pg.. 180)	1	1
⭐ 3.16.5 Continuous Random Variables and Distributions (Pg.. 181)	1	1
⭐ 3.16.6 Joint, Marginal, and Conditional Distributions (Pg.. 181)	1	1
⭐ 3.16.7 Limit Theorems: LLN and CLT (Pg.. 181)	1	1

Table 3.16: List of KUs in the Probability Theory area.

3.16.1 PRO/Sample Spaces and Events (CS Core: 1 hr, KA Core: 1 hr)

Formal definition of random experiments, sample spaces, and events. Covers set-theoretic operations on events and the construction of sigma-algebras as the foundation of probability theory.

Topics:

Core

- Random experiments, outcomes, and sample spaces: finite, countable, and uncountable
- Set-theoretic operations on events: union, intersection, complement, and De Morgan's laws
- Sigma-algebras and measurable spaces as the foundation for rigorous probability

- Borel sets and the Borel sigma-algebra on the real line

Learning Outcomes:

Core:

1. Define sample spaces, events, and sigma-algebras for a given random experiment [Familiarity]
2. Apply set-theoretic operations to compute probabilities of compound events [Usage]
3. Construct appropriate sigma-algebras for measurable spaces arising in statistical problems [Assessment]

3.16.2 PRO/Probability Axioms and Properties (CS Core: 1 hr, KA Core: 1 hr)

Kolmogorov's axioms of probability and derived properties. Covers classical, frequentist, and subjective interpretations of probability and their implications.

Topics:

Core

- Kolmogorov's three axioms of probability and their consequences
- Derived properties: complement rule, addition rule, monotonicity, and continuity of probability
- Classical, frequentist, and subjective (Bayesian) interpretations of probability
- Counting methods: permutations, combinations, and the multiplication principle

Learning Outcomes:

Core:

1. State Kolmogorov's axioms and derive fundamental probability properties [Familiarity]
2. Calculate probabilities of events using the addition rule and complement rule [Usage]
3. Apply counting methods to compute probabilities in finite sample spaces [Usage]

3.16.3 PRO/Conditional Probability and Independence (CS Core: 1 hr, KA Core: 1 hr)

Conditional probability, the multiplication rule, the law of total probability, Bayes' theorem, and the concept of independence between events and collections of events.

Topics:

Core

- Definition and interpretation of conditional probability P(A|B) and its properties
- Bayes' theorem and the law of total probability: derivation and applications
- Independence of events: pairwise and mutual independence
- Conditional independence and its role in probabilistic graphical models

Learning Outcomes:

Core:

1. Explain conditional probability and independence and their relationship [Familiarity]
2. Apply Bayes' theorem and the law of total probability to solve probability problems [Usage]
3. Analyze whether events are independent or conditionally independent in a given probability model [Assessment]

3.16.4 PRO/Discrete Random Variables and Distributions (CS Core: 1 hr, KA Core: 1 hr)

Random variables as measurable functions, probability mass functions, cumulative distribution functions, and the most important discrete probability distributions.

Topics:

Core

- Probability mass functions (PMF) and cumulative distribution functions (CDF) for discrete random variables
- Key discrete distributions: Bernoulli, Binomial, Geometric, Negative Binomial, Poisson, Hypergeometric
- Properties of discrete distributions: mean, variance, skewness, and kurtosis
- Poisson approximation to the Binomial and limiting distribution arguments

Learning Outcomes:
Core:
1. Identify and describe the key discrete probability distributions and their parameters [Familiarity]
2. Calculate probabilities, means, and variances using PMFs of standard discrete distributions [Usage]
3. Select an appropriate discrete distribution to model a given random phenomenon and justify the choice [Assessment]

3.16.5 PRO/Continuous Random Variables and Distributions (CS Core: 1 hr, KA Core: 1 hr)

Probability density functions, CDFs, and the most important continuous probability distributions used in statistical theory and practice.
Topics:
Core
- Probability density functions (PDF) and CDFs for continuous random variables
- Key continuous distributions: Uniform, Normal, Exponential, Gamma, Beta, Chi-squared, t, F
- The Normal distribution: properties, standard normal, and its central role in statistical inference
- The exponential family of distributions: canonical form and sufficient statistics

Learning Outcomes:
Core:
1. Describe the properties and parameters of key continuous probability distributions [Familiarity]
2. Calculate probabilities and quantiles for standard continuous distributions [Usage]
3. Identify distributions belonging to the exponential family and derive their sufficient statistics [Assessment]

3.16.6 PRO/Joint, Marginal, and Conditional Distributions (CS Core: 1 hr, KA Core: 1 hr)

Joint distributions of multiple random variables, marginal and conditional distributions, covariance, correlation, and the bivariate normal distribution.
Topics:
Core
- Joint PMFs and PDFs for two or more random variables
- Marginal and conditional distributions derived from joint distributions
- Covariance, correlation, and their properties; independence implies zero covariance (converse not always true)
- The bivariate normal distribution and its conditional distributions

Learning Outcomes:
Core:
1. Derive marginal and conditional distributions from a given joint distribution [Usage]
2. Calculate covariance and correlation between two random variables [Usage]
3. Analyze the conditional structure of the bivariate normal distribution [Assessment]

3.16.7 PRO/Limit Theorems: LLN and CLT (CS Core: 1 hr, KA Core: 1 hr)

The laws of large numbers and the central limit theorem as the mathematical justification for statistical estimation and inference procedures based on large samples.
Topics:
Core
- Weak law of large numbers: convergence in probability and its implications for estimation
- Central limit theorem: statement, conditions, and role in justifying normal approximations
- Applications of the CLT: normal approximation to the binomial and sampling distributions
- Strong law of large numbers: almost sure convergence
- The delta method: asymptotic distribution of smooth functions of estimators

Learning Outcomes:
Core:
1. State the LLN and CLT and explain the conditions under which they hold [Familiarity]
2. Apply the CLT to derive normal approximations and sampling distributions [Usage]
3. Apply the delta method to derive asymptotic distributions of nonlinear estimators [Assessment]

3.17 Descriptive Statistics and Data Analysis (DSC)

This knowledge area covers methods for summarizing, visualizing, and exploring data. It includes numerical and graphical summaries, exploratory data analysis, data quality assessment, and communication of statistical findings.

Knowledge Area (KA)	CS Core	KA Core
⭐ 3.17.1 Numerical Summaries: Location, Spread, and Shape (Pg.. 182)	1	1

Table 3.17: List of KUs in the Descriptive Statistics and Data Analysis area.

3.17.1 DSC/Numerical Summaries: Location, Spread, and Shape (CS Core: 1 hr, KA Core: 1 hr)

Measures of central tendency, dispersion, and distributional shape. Includes robust alternatives and the interpretation of numerical summaries in context.
Topics:
Core
- Measures of location: mean, median, mode, trimmed mean, and weighted mean
- Measures of spread: variance, standard deviation, IQR, MAD, and range
- Measures of shape: skewness and kurtosis; quantiles and percentiles
- Robust summary statistics: resistance to outliers and heavy-tailed distributions

Learning Outcomes:
Core:
1. Calculate and interpret measures of location, spread, and shape for a univariate dataset [Usage]
2. Select appropriate summary statistics based on the distributional properties of the data [Usage]
3. Compare classical and robust summary statistics and justify the choice for a given dataset [Assessment]

3.18 Statistical Inference (INF)

This knowledge area covers the foundations and methods of statistical inference: point and interval estimation, hypothesis testing, likelihood theory, Bayesian inference, nonparametric methods, multiple testing, and large-sample theory. It is the core of classical and modern statistics.

Knowledge Area (KA)	CS Core	KA Core
⭐ 3.18.1 Point Estimation (Pg.. 182)	1	1
⭐ 3.18.2 Interval Estimation and Confidence Intervals (Pg.. 183)	1	1
⭐ 3.18.3 Hypothesis Testing Foundations (Pg.. 183)	1	1
⭐ 3.18.4 Bayesian Inference (Pg.. 183)	1	1

Table 3.18: List of KUs in the Statistical Inference area.

3.18.1 INF/Point Estimation (CS Core: 1 hr, KA Core: 1 hr)

Methods for estimating unknown parameters from data. Covers desirable properties of estimators, method of moments, maximum likelihood estimation, and the Cramer-Rao lower bound.
Topics:
Core

- Desirable properties: unbiasedness, consistency, efficiency, and sufficiency
- Method of moments estimation: derivation and asymptotic properties
- Maximum likelihood estimation: derivation, properties, and invariance principle
- Cramer-Rao lower bound and the concept of Fisher information

Learning Outcomes:

Core:

1. Describe the key properties of estimators and explain the trade-offs between them [Familiarity]
2. Derive method of moments and MLE estimators for standard distributions [Usage]
3. Evaluate the efficiency of an estimator using the Cramer-Rao lower bound [Assessment]

3.18.2 INF/Interval Estimation and Confidence Intervals (CS Core: 1 hr, KA Core: 1 hr)

Construction and interpretation of confidence intervals. Covers pivotal quantity method, large-sample intervals, and the relationship between confidence intervals and hypothesis tests.

Topics:

Core

- Constructing confidence intervals using pivotal quantities and exact distributions
- Correct interpretation of confidence intervals: coverage probability and common misconceptions
- Large-sample confidence intervals based on asymptotic normality
- Duality between confidence intervals and hypothesis tests

Learning Outcomes:

Core:

1. Construct confidence intervals for means, proportions, and variances using exact and large-sample methods [Usage]
2. Interpret confidence intervals correctly and identify common misinterpretations [Usage]
3. Explain the duality between confidence intervals and hypothesis tests and use it to draw inferences [Assessment]

3.18.3 INF/Hypothesis Testing Foundations (CS Core: 1 hr, KA Core: 1 hr)

The Neyman-Pearson framework for hypothesis testing: null and alternative hypotheses, Type I and II errors, power, p-values, and the most powerful tests.

Topics:

Core

- Neyman-Pearson framework: null/alternative hypotheses, rejection regions, Type I and II errors
- p-values: definition, correct interpretation, and common misuse; power and sample size
- Common parametric tests: z-test, t-test, F-test, chi-squared test
- Neyman-Pearson lemma and uniformly most powerful tests

Learning Outcomes:

Core:

1. Explain the logic of hypothesis testing, Type I and II errors, and the meaning of a p-value [Familiarity]
2. Apply standard parametric tests and interpret the results in context [Usage]
3. Derive the most powerful test for a simple hypothesis using the Neyman-Pearson lemma [Assessment]

3.18.4 INF/Bayesian Inference (CS Core: 1 hr, KA Core: 1 hr)

The Bayesian approach to statistical inference: prior distributions, likelihood, posterior distributions, credible intervals, and Bayesian decision theory.

Topics:

Core

- Bayesian framework: prior, likelihood, posterior, and the role of Bayes' theorem

- Prior distributions: informative, non-informative, and conjugate priors
- Posterior inference: point estimates, credible intervals, and posterior predictive distributions
- Bayesian model comparison: Bayes factors and posterior model probabilities

Learning Outcomes:

Core:

1. Explain the Bayesian approach to inference and contrast it with the frequentist approach [Familiarity]
2. Derive the posterior distribution for standard conjugate models [Usage]
3. Apply Bayes factors to compare statistical models and interpret the evidence [Assessment]

3.19 Regression and Statistical Models (REG)

This knowledge area covers the construction, estimation, interpretation, and validation of statistical models for relationships between variables. It spans simple and multiple linear regression, generalized linear models, nonparametric regression, mixed models, and causal inference.

Knowledge Area (KA)	CS Core	KA Core
⭐ 3.19.1 Simple Linear Regression (Pg.. 184)	1	1

Table 3.19: List of KUs in the Regression and Statistical Models area.

3.19.1 REG/Simple Linear Regression (CS Core: 1 hr, KA Core: 1 hr)

The simple linear regression model: least squares estimation, inference for coefficients, model fit, and the assumptions of the classical linear model.

Topics:

Core

- Ordinary least squares estimation: derivation of coefficients and their properties
- Inference for regression coefficients: t-tests, confidence intervals, and p-values
- Coefficient of determination R-squared and analysis of variance decomposition
- Gauss-Markov theorem: BLUE property of OLS under classical assumptions

Learning Outcomes:

Core:

1. Fit a simple linear regression model and interpret the estimated coefficients [Usage]
2. Conduct hypothesis tests and construct confidence intervals for regression parameters [Usage]
3. Verify the Gauss-Markov assumptions and assess their consequences when violated [Assessment]

Chapter 4

Curricula 2026

4.1 Course Coding

Courses are classified into 4 levels:

- **Introductory** Code 1.. example: CS100
- **Intermediate** Code 2.. example: MA200
- **Advanced** Code 3.. example: IS370
- **Capstone Project** Code 4.. example: CS403

The courses are encoded under the scheme shown in Figure 4.1.

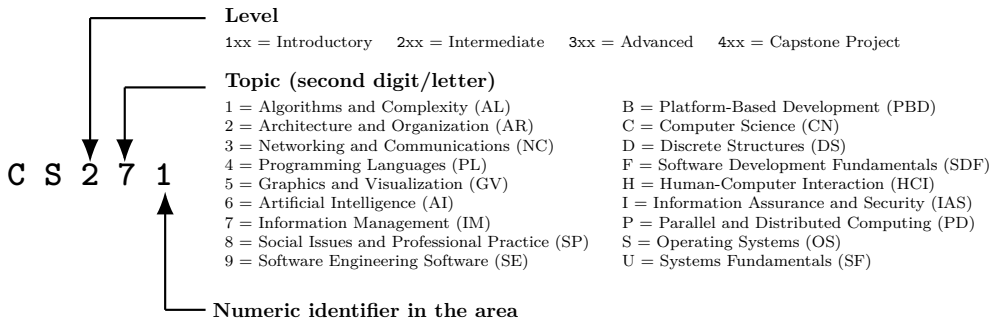

Figure 4.1: Course coding scheme.

The course type is determined by the initial letters. The possible codes for these types are:

1. **CS** Computer Science (CS);
2. **BIC** Computer Science (BIC);
3. **AI** Artificial Intelligence (AI);
4. **MA** Mathematics (MA);
5. **BMA** Mathematics (BMA);
6. **STA** Statistics (ST);
7. **CB** Basic Sciences (Physics, Biology);
8. **FI** Physics
9. **BFI** Physic Sciences
10. **BCH** Chemistry (BCH);
11. **FG** General Education Area;
12. **EX** Extracurricular cctivities
13. **ET** Technological Based Enterprise Formation Area;
14. **BEI** Languages Area

At the same time, and according to current university law, courses are classified into:

- General studies
- Specific
- Specialization

4.2 Curriculum Structure

The list of courses is shown below:

TH = Theory Hours, PR = Practice Hours, CR = Credits, T = Type of Course.

M = Mandatory, E = Elective.

Primer Semestre							
Código	**Curso**	**Área**	**HT**	**HP**	**CR**	**T**	**Requisitos**
BIC101	Introduction to Computing (Pág 239)	General Estudies	2	4	4	O	
BMA101	Linear Algebra (Pág 244)	General Estudies	3	2	4	O	
BMA102	Differential Calculus (Pág 246)	General Estudies	4	2	5	O	
BCH101	Chemistry I (Pág 249)	General Estudies	4	2	5	O	
BEI101	Technical and professional English I (Pág 254)	General Estudies		4	2	O	
					20		

Segundo Semestre							
Código	**Curso**	**Área**	**HT**	**HP**	**CR**	**T**	**Requisitos**
CS100	Introduction to Computer Science (Pág 257)	Specific	2	2	3	O	
CS112	Objects-oriented programming I (Pág 268)	Specific	2	4	4	O	BIC101 (1er Sem)
CS1D1	Discrete Structures (Pág 273)	Specific	2	4	4	O	
BMA103	Integral Calculus (Pág 276)	General Estudies	4	2	5	O	BMA102 (1er Sem)
BFI101	Physics I (Pág 278)	General Estudies	4	4	5	O	
BEI102	Technical and professional English II (Pág 283)	General Estudies		4	2	O	BEI101 (1er Sem)
					23		

	Tercer Semestre							
	Código	**Curso**	**Área**	**HT**	**HP**	**CR**	**T**	**Requisitos**
	CS113	Objects-oriented programming II (Pág 287)	Specific	2	4	4	O	CS112 (2do Sem)
	CS2B1	Platform Based Development (Pág 293)	Specific	2	4	4	O	CS112 (2do Sem)
	AI161	Applied AI (Pág 297)	Specific	2	4	4	O	BIC101 (1er Sem)
	BMA104	Advanced Differential and Integral Calculus (Pág 301)	General Estudies	4	2	5	O	BMA102 (1er Sem)
	STA251	Probability Calculation (Pág 304)	Specific	2	2	3	O	BMA102 (1er Sem)
Elective	BEI201	Technical and professional English III (Pág 309)	General Estudies		4	2	E	BEI102 (2do Sem)
						20		

	Cuarto Semestre							
	Código	**Curso**	**Área**	**HT**	**HP**	**CR**	**T**	**Requisitos**
	CS210	Algorithms and Data Structures (Pág 313)	Specific	2	4	4	O	CS113 (3er Sem)
	CS211	Theory of Computation (Pág 318)	Specific	2	4	4	O	CS1D1 (2do Sem)
	CS221	Computer Systems Architecture (Pág 323)	Specific	2	4	4	O	CS1D1 (2do Sem)
	CS271	Data Management (Pág 327)	Specific	2	4	4	O	CS112 (2do Sem), CS1D1 (2do Sem)
	CS401	Research Methodology (Pág 331)	Specialization	1	2	2	O	CS100 (2do Sem)
	MA202	Numerical Methods (Pág 335)	Specific	2	2	3	O	BMA103 (2do Sem)
Elective	BEI202	Technical and professional English IV (Pág 340)	General Estudies		4	2	E	BEI201 (3er Sem)
						21		

Quinto Semestre							
Código	**Curso**	**Área**	**HT**	**HP**	**CR**	**T**	**Requisitos**
CS212	Analysis and Design of Algorithms (Pág 344)	Specific	2	4	4	O	CS210 (4^{to} Sem), CS211 (4^{to} Sem)
CS261	Artificial Intelligence (Pág 349)	Specific	2	4	4	O	STA251 (3^{er} Sem)
CS272	Databases II (Pág 356)	Specific	1	4	3	O	CS271 (4^{to} Sem)
CS291	Software Engineering I (Pág 360)	Specific	2	4	4	O	CS113 (3^{er} Sem), CS271 (4^{to} Sem)
CS2S1	Operating systems (Pág 365)	Specific	2	4	4	O	CS221 (4^{to} Sem)
FG106	Theater (Pág 370)	Specific	1	2	2	O	
Elective BEI203	Technical and professional English V (Pág 374)	General Estudies		4	2	E	BEI202 (4^{to} Sem)
					21		

Sexto Semestre							
Código	**Curso**	**Área**	**HT**	**HP**	**CR**	**T**	**Requisitos**
CS231	Networking and Communication (Pág 378)	Specific	1	4	3	O	CS2S1 (5^{to} Sem)
CS311	Algorithms for Complex Problems (Pág 383)	Specific	2	2	3	O	CS212 (5^{to} Sem)
CS312	Advanced Data Structures (Pág 386)	Specific	2	4	4	O	CS212 (5^{to} Sem)
CS342	Compilers (Pág 389)	Specific	2	4	4	O	CS211 (4^{to} Sem)
AI263	Introduction to Machine Learning (Pág 392)	Specialization	2	4	4	O	CS261 (5^{to} Sem)
FI201	Computational Physics (Pág 395)	Specific	2	2	3	O	BFI101 (2^{do} Sem)
					21		

Séptimo Semestre							
Código	**Curso**	**Área**	**HT**	**HP**	**CR**	**T**	**Requisitos**
CS251	Computer Graphics (Pág 398)	Specialization	2	4	4	O	MA202 (4^{to} Sem)
CS292	Software Engineering II (Pág 401)	Specific	2	4	4	O	CS291 (5^{to} Sem)
CS2H1	User Experience (UX) (Pág 406)	Specific	2	4	4	O	CS291 (5^{to} Sem)
AI264	Deep Learning (Pág 411)	Specialization	2	4	4	O	AI263 (6^{to} Sem)
FG211	Professional Ethics (Pág 414)	Specific	1	2	2	O	
FG350	Leadership and Performance (Pág 416)	Specific	2		2	O	FG106 (5^{to} Sem)
					20		

	Octavo Semestre							
	Código	**Curso**	**Área**	**HT**	**HP**	**CR**	**T**	**Requisitos**
	CS281	Computing in Society (Pág 419)	Specific	2		2	O	
	CS3I1	Computer Security (Pág 425)	Specific	1	4	3	O	CS231 (6^{to} Sem)
	CS3P1	Parallel and Distributed Computing (Pág 430)	Specific	2	4	4	O	CS212 (5^{to} Sem), CS231 (6^{to} Sem)
	CS402	Capstone Project I (Pág 439)	Specialization	2	4	4	O	CS401 (4^{to} Sem)
	EX301	Extracurricular Activities (Pág 443)	General Estudies	1	2	2	O	
	ET201	Entrerpreneurship I (Pág 445)	Specific	2	2	3	O	FG350 (7^{mo} Sem)
Elective	AI268	Computational Vision (Pág 449)	Specialization	2	2	3	E	AI264 (7^{mo} Sem)
	CS391	Software Engineering III (Pág 455)	Specific	2	2	3	E	CS292 (7^{mo} Sem)
	CS393	Information systems (Pág 459)	Specific	2	2	3	E	CS291 (5^{to} Sem)
	CS394	Software Architecture and DevOps for Cloud Software (Pág 462)	Specific	2	2	3	E	CS292 (7^{mo} Sem)
	CS395	Software Quality (Pág 468)	Specific	2	2	3	E	CS292 (7^{mo} Sem)
						21		

	Noveno Semestre							
	Código	**Curso**	**Área**	**HT**	**HP**	**CR**	**T**	**Requisitos**
	CS370	Big Data (Pág 475)	Specialization	1	4	3	O	CS3P1 (8^{vo} Sem)
	CS400	Pre-professional internships (Pág 478)	Specialization	1	2	2	O	110Cr
	CS403	Capstone Project II (Pág 480)	Specialization	2	4	3	O	CS402 (8^{vo} Sem)
	AI365	Advanced Generative AI Models (Pág 484)	Specific	2	4	4	O	AI264 (7^{mo} Sem)
	CB309	Bioinformatics (Pág 486)	Specialization	2	4	4	O	
	ET301	Entrerpreneurship II (Pág 489)	Specific	2	2	3	O	ET201 (8^{vo} Sem)
Elective	AI369	Robotics (Pág 492)	Specialization	2	2	3	E	AI268 (8^{vo} Sem)
	CS351	Topics in Computer Graphics (Pág 496)	Specialization	2	2	3	E	CS251 (7^{mo} Sem)
	CS353	Quantum Computing (Pág 500)	Specialization	2	2	3	E	CS3I1 (8^{vo} Sem)
	CS392	Tópicos en Ingeniería de Software (Pág 504)	Specialization	2	2	3	E	CS391 (8^{vo} Sem)
	CS3P3	Internet of Things (Pág 511)	Specialization	1	4	3	E	CS3P1 (8^{vo} Sem)
						22		

Décimo Semestre							
Código	**Curso**	**Área**	**HT**	**HP**	**CR**	**T**	**Requisitos**
CS3P2	Cloud Computing (Pág 513)	Specific	1	4	3	O	CS370 (9no Sem)
CS404	Research Workshop (Pág 517)	Specialization	2	4	4	O	CS403 (9no Sem)
AI367	Topics in Artificial Intelligence (Pág 521)	Specialization	2	4	4	O	AI264 (7mo Sem)
AI368	Evolutionary Computing (Pág 527)	Specific	2	4	4	O	AI264 (7mo Sem)
					11		

Total de créditos de la carrera: 200 .

4.3 Credits for Diverse Activities

Extracurricular Activities must total a minimum of 80 hours across 3 different activities (Art. 27, RR 544, 2017).

Item	Credits	Extracurricular Activities (Art. 31, RR 1817, 2019)
01	01	• Artistic activities: music, theater, literature, visual arts, folklore, etc. • Physical activities and sports. • Participation in courses, conferences, workshops. • Participation in Scientific Activities: scientific and technological contests. • Social outreach (fairs and vocational outreach in schools). • Personal development courses: interpersonal communication, leadership, coaching, entrepreneurship, personal marketing, etc. • Student assistantships in course-reinforcement workshops, for students in academic risk, per item d) of Art. 26° of the UNI Tutoring Regulation for Undergraduate Students in Academic Risk, approved by R.R. N° 0477 of 04.21.2017. • Student assistantships in laboratories and workshops.

4.4 Graduation Requirements

Type	School of Computer Science	N6
Mandatory	Credits in mandatory courses	180
	BIE01 Foreign Language or Native Language at Intermediate Level*	2
	BAE01 Extracurricular Activities	1
	YA100 Academic Assistantship I	1
	XP200 Pre-Professional Internship II	2
	Total mandatory credits	**186**
Electives	Minimum credits in elective courses (specialization)	0
	Minimum sum of credits in elective courses (specialization) and complementary elective courses	**22**
Total	**Minimum credits to graduate**	**208**

*Students who request exemption from the BIE01 course in the academic years 2020, 2021, and 2022, including 2023-2, are exempted from taking the course under the exception provided by Law N° 31359.

4.5 Topics distributed by course

4.5.1 Computer Science topics/course

		First Sem					Second Sem						.
	Knowledge Units of Computer Science **Courses →**	BIC101	BMA101	BMA102	BCH101	BEI101	CS100	CS112	CS1D1	BMA103	BFI101	BEI102	Total
⭐	AI/Machine Learning Fundamentals (1 hour Core-Tier1, 2 hours Core-Tier2)						1						1
⭐	AI/Neural Networks and Ethics (1 hour Core-Tier1, 1 hour Core-Tier2)						1						1
⭐	AL/Foundational Data Structures (5 hours Core-Tier1, 3 hours Core-Tier2)						3						3
⭐	AL/Foundational Algorithms (4 hours Core-Tier1, 2 hours Core-Tier2)						3						3
⭐	AL/Advanced Algorithms (3 hours Core-Tier1, 2 hours Core-Tier2)						2						2
⭐	AL/Algorithmic Strategies (6 hours Core-Tier1)	6											6
⭐	AL/Complexity Analysis Framework (2 hours Core-Tier1, 1 hour Core-Tier2)						2						2
⭐	AL/Asymptotic Notation and Complexity Classes (1 hour Core-Tier1, 1 hour Core-Tier2)						1						1
⭐	AL/Complexity Analysis: Recursion, Amortization, and Tight Bounds (2 hours Core-Tier1, 1 hour Core-Tier2)						1						1
⭐	AR/Machine-Level Data Representation (1 hour Core-Tier1)						4						4
⭐	DM/Core Database System Concepts (2 hours Core-Tier1, 1 hour Core-Tier2)						4						4
⭐	FPL/Object-Oriented Programming Fundamentals (2 hours Core-Tier1, 1 hour Core-Tier2)	24					4	14					42
⭐	FPL/Object-Oriented Programming: Encapsulation, Subtyping, and Reflection (2 hours Core-Tier1, 1 hour Core-Tier2)							8					8
⭐	FPL/Functional Programming (4 hours Core-Tier1, 3 hours Core-Tier2)							10					10
⭐	FPL/Shell Scripting (2 hours Core-Tier1)							4					4
⭐	FPL/Type Systems Fundamentals (2 hours Core-Tier1, 2 hours Core-Tier2)	6						8					14
⭐	FPL/Systems Execution and Memory Model (2 hours Core-Tier1)							12					12
⭐	FPL/Language Translation and Execution (2 hours Core-Tier1, 3 hours Core-Tier2)							4					4
⭐	FPL/Program Abstraction and Representation (3 hours Core-Tier2)						2						2
	FPL/Advanced Programming Constructs							6					6
	FPL/Language Pragmatics							2					2
⭐	GIT/Fundamental Concepts of Graphics and Interactive Techniques (4 hours Core-Tier1, 3 hours Core-Tier2)						2						2
⭐	GIT/Applied Rendering and Techniques (15 hours Core-Tier2)						2						2
⭐	MSF/Discrete Mathematics (15 hours Core-Tier1, 6 hours Core-Tier2)								48				48

Table 4.1: Computer Science topics by course from 1st to 2nd Semester (Part 1 of 2)

Knowledge Units of Computer Science	Courses →	First Sem					Second Sem					.
		BIC101	BMA101	BMA102	BCH101	BEI101	CS100	CS112	CS1D1	BMA103	BFI101	BEI102	Total
★	NC/Fundamentals of Networks and Communications (3 hours Core-Tier1)						4						4
★	NC/Networked Applications (4 hours Core-Tier1)							4					4
★	OS/Role and Purpose of Operating Systems (2 hours Core-Tier1)							2					2
★	OS/Principles of Operating System (2 hours Core-Tier1)								2				2
★	OS/Memory Management (1 hour Core-Tier2)												
★	SE/Version Control and CI/CD (1 hour Core-Tier1, 2 hours Core-Tier2)						2						2
★	SE/Requirements Engineering (2 hours Core-Tier2)												
★	SEC/Cryptography Fundamentals (1 hour Core-Tier1, 2 hours Core-Tier2)						2						2
	Total	36	0	0	0	0	54	68	48	0	0	0	2

Table 4.2: Computer Science topics by course from 1st to 2nd Semester (Part 2 of 2)

		Third Sem						Fourth Sem							.
	Knowledge Units of Computer Science **Courses →**	CS113	CS2B1	AI161	BMA104	STA251	BEI201	CS210	CS211	CS221	CS271	CS401	MA202	BEI202	Total
★	AL/Foundational Data Structures (5 hours Core-Tier1, 3 hours Core-Tier2)							32							32
★	AL/Complexity Analysis Framework (2 hours Core-Tier1, 1 hour Core-Tier2)								3						3
★	AL/Asymptotic Notation and Complexity Classes (1 hour Core-Tier1, 1 hour Core-Tier2)								4						4
★	AL/Complexity Analysis: Recursion, Amortization, and Tight Bounds (2 hours Core-Tier1, 1 hour Core-Tier2)								7						7
★	AL/Computational Complexity Theory (2 hours Core-Tier1, 1 hour Core-Tier2)								6						6
★	AL/Formal Languages and Automata (4 hours Core-Tier1, 10 hours Core-Tier2)								62						62
★	AR/Assembly Level Machine Organization (1 hour Core-Tier1, 2 hours Core-Tier2)									20					20
★	AR/Memory Hierarchy (6 hours Core-Tier1)									18					18
★	AR/Interfacing and Communication (1 hour Core-Tier1)									8					8
★	AR/Functional Organization (2 hours Core-Tier2)									8					8
★	AR/Heterogeneous Architectures (2 hours Core-Tier2)									10					10
★	AR/Secure Processor Architectures (2 hours Core-Tier2)									8					8
★	DM/Core Database System Concepts (2 hours Core-Tier1, 1 hour Core-Tier2)										8				8
★	DM/Data Modeling (2 hours Core-Tier1, 3 hours Core-Tier2)										8				8
★	DM/Relational Databases (1 hour Core-Tier1, 3 hours Core-Tier2)										16				16
★	DM/Query Construction (2 hours Core-Tier1, 4 hours Core-Tier2)										16				16
★	DM/DBMS Internals (4 hours Core-Tier2)										8				8
★	DM/Data Security and Privacy (1 hour Core-Tier1, 2 hours Core-Tier2)										8				8
★	FPL/Object-Oriented Programming: Encapsulation, Subtyping, and Reflection (2 hours Core-Tier1, 1 hour Core-Tier2)	8													8
★	FPL/Parallel and Distributed Computing (2 hours Core-Tier1, 2 hours Core-Tier2)	12													12
★	FPL/Type Systems: Polymorphism and Complementary Typing (1 hour Core-Tier1, 1 hour Core-Tier2)	12													12
★	FPL/Systems Execution and Memory Model (2 hours Core-Tier1)	8													8
★	FPL/Memory Management (1 hour Core-Tier1)	8													8
	FPL/Advanced Programming Constructs	20													20
★	SE/Version Control and CI/CD (1 hour Core-Tier1, 2 hours Core-Tier2)	4													4

	Knowledge Units of Computer Science	Courses →	Third Sem							Fourth Sem						.
			CS113	CS2B1	AI161	BMA104	STA251	BEI201	CS210	CS211	CS221	CS271	CS401	MA202	BEI202	Total
⭐	SPD/Common Aspects: Platform Languages and Web Programming Patterns (2 hours Core-Tier1, 1 hour Core-Tier2)			3												3
⭐	SPD/Frontend, Frameworks and Web Standards (3 hours Core-Tier2)			12												12
⭐	SPD/Mobile Platforms: Development Fundamentals (2 hours Core-Tier2)			6												6
⭐	SPD/Mobile Platforms: Architecture and Constraints (2 hours Core-Tier2)			3												3
⭐	SPD/Mobile Platforms: Capabilities and Asynchronous Computing (1 hour Core-Tier2)			3												3
⭐	SPD/Interactive Data and Analytics Platforms (2 hours Core-Tier2)			6												6
	SPD/SPD-SEP/Interactive			10	0	0	0	0	32	82	72	64	0	0	0	10
	Total		72	48												

Table 4.4: Computer Science topics by course from 3rd to 4th Semester (Part 2 of 2)

		Fifth Sem							Sixth Sem						.
	Knowledge Units of Computer Science **Courses →**	CS212	CS261	CS272	CS291	CS2S1	FG106	BEI203	CS231	CS311	CS312	CS342	AI263	FI201	Total
★	AI/Introduction to AI: Concepts and Intelligent Behavior (2 hours Core-Tier1, 1 hour Core-Tier2)		4												4
★	AI/Problem Characteristics and Agent Nature (1 hour Core-Tier1, 1 hour Core-Tier2)		4												4
★	AI/State and Uninformed Search (1 hour Core-Tier1, 2 hours Core-Tier2)		5												5
★	AI/Heuristic Search and CSP (1 hour Core-Tier1, 2 hours Core-Tier2)		6												6
★	AI/Adversarial Search and Stochastic Optimization (1 hour Core-Tier1, 3 hours Core-Tier2)		5												5
★	AI/Knowledge Representation and Probabilistic Reasoning Foundations (1 hour Core-Tier1, 1 hour Core-Tier2)		6												6
★	AI/Independence, Markov Models and Decision Making (1 hour Core-Tier1, 1 hour Core-Tier2)		6												6
★	AI/Machine Learning Fundamentals (1 hour Core-Tier1, 2 hours Core-Tier2)		7												7
★	AI/Machine Learning Data, Representations and Evaluation (1 hour Core-Tier1, 2 hours Core-Tier2)		7												7
★	AI/Neural Networks and Ethics (1 hour Core-Tier1, 1 hour Core-Tier2)		6												6
★	AI/Machine Learning Formulation and Evaluation (1 hour Core-Tier1, 2 hours Core-Tier2)		3												3
★	AI/Deep Learning, RL and Ethics (1 hour Core-Tier1, 2 hours Core-Tier2)		3												3
★	AI/AI Applications and Societal Impact (2 hours Core-Tier1, 2 hours Core-Tier2)		4												4
★	AI/Additional AI Applications and Societal Impact (2 hours Core-Tier1, 2 hours Core-Tier2)		4												4
★	AL/Foundational Data Structures (5 hours Core-Tier1, 3 hours Core-Tier2)	6													6
★	AL/Foundational Algorithms (4 hours Core-Tier1, 2 hours Core-Tier2)	6								36					42
★	AL/Advanced Algorithms (3 hours Core-Tier1, 2 hours Core-Tier2)	4													4
★	AL/Algorithmic Strategies (6 hours Core-Tier1)	24								12					36
★	AL/Complexity Analysis Framework (2 hours Core-Tier1, 1 hour Core-Tier2)	6													6
★	AL/Asymptotic Notation and Complexity Classes (1 hour Core-Tier1, 1 hour Core-Tier2)	6													6
★	AL/Complexity Analysis: Recursion, Amortization, and Tight Bounds (2 hours Core-Tier1, 1 hour Core-Tier2)	12													12
★	AL/Computational Complexity Theory (2 hours Core-Tier1, 1 hour	6													6

		Fifth Sem							Sixth Sem						.
	Knowledge Units of Computer Science **Courses →**	CS212	CS261	CS272	CS291	CS2S1	FG106	BEI203	CS231	CS311	CS312	CS342	AI263	FI201	Total
	DM/Semi-structured and Unstructured Databases			12											12
⭐	FPL/Object-Oriented Programming: Encapsulation, Subtyping, and Reflection (2 hours Core-Tier1, 1 hour Core-Tier2)										12				12
⭐	FPL/Shell Scripting (2 hours Core-Tier1)					8									8
⭐	FPL/Language Translation and Execution (2 hours Core-Tier1, 3 hours Core-Tier2)											24			24
	FPL/Program Analysis and Analyzers											20			20
	FPL/Run-time Behavior and Systems											20			20
⭐	NC/Fundamentals of Networks and Communications (3 hours Core-Tier1)								6						6
⭐	NC/Networked Applications (4 hours Core-Tier1)								6						6
⭐	NC/Reliability Support (6 hours Core-Tier2)								6						6
⭐	NC/Routing and Forwarding (4 hours Core-Tier2)								6						6
⭐	NC/Single Hop Communication (3 hours Core-Tier2)								6						6
⭐	NC/Network Security (3 hours Core-Tier2)								6						6
⭐	NC/Mobility Support (4 hours Core-Tier2)								6						6
⭐	NC/Emerging Topics (4 hours Core-Tier2)								6						6
⭐	OS/Role and Purpose of Operating Systems (2 hours Core-Tier1)					8									8
⭐	OS/Protection and Safety (2 hours Core-Tier1, 1 hour Core-Tier2)					12									12
⭐	OS/Scheduling (2 hours Core-Tier2)					16									16
⭐	OS/Memory Management (1 hour Core-Tier2)					16									16
⭐	OS/File Systems API and Implementation (2 hours Core-Tier2)					12									12
⭐	SE/Team Collaboration and Dynamics (2 hours Core-Tier1, 2 hours Core-Tier2)				14										14
⭐	SE/Version Control and CI/CD (1 hour Core-Tier1, 2 hours Core-Tier2)				12										12
⭐	SE/Requirements Engineering (2 hours Core-Tier2)				14										14
⭐	SE/Principles and Software Architecture (1 hour Core-Tier1, 2 hours Core-Tier2)				12										12
⭐	SE/Verification and validation Concepts (1 hour Core-Tier1, 1 hour Core-Tier2)				12										12
⭐	SE/Testing Tools and Analysis (1 hour Core-Tier1, 1 hour Core-Tier2)				4										4
⭐	SE/Performance Testing and Benchmarking (1 hour Core-Tier1, 1 hour Core-Tier2)				4										4

Table 4.6: Computer Science topics by course from 5th to 6th Semester (Part 2 of 3)

Table 4.7: Computer Science topics by course from 5th to 6th Semester (Part 3 of 3)

		Seventh Sem						.
	Knowledge Units of Computer Science **Courses →**	CS251	CS292	CS2H1	AI264	FG211	FG350	Total
★	AI/Neural Network Training and Optimization (1 hour Core-Tier1, 1 hour Core-Tier2)				10			10
★	AI/Regularization Techniques for Deep Networks (1 hour Core-Tier1, 1 hour Core-Tier2)				5			5
★	AI/Convolutional and Residual Network Architectures (1 hour Core-Tier1, 1 hour Core-Tier2)				9			9
★	AI/Transformer Architectures and Generative Models (1 hour Core-Tier1, 1 hour Core-Tier2)				16			16
	AI/Agents and Cognitive Systems			8				8
	AI/Natural Language Processing			8				8
★	GIT/Fundamental Concepts of Graphics and Interactive Techniques (4 hours Core-Tier1, 3 hours Core-Tier2)	16						16
★	GIT/Visualization (6 hours Core-Tier2)	16						16
★	GIT/Applied Rendering and Techniques (15 hours Core-Tier2)	16						16
★	GIT/Geometric Modeling (6 hours Core-Tier2)	16						16
★	GIT/Shading and Advanced Rendering (6 hours Core-Tier2)	12						12
★	HCI/Understanding the User: Individual goals and interactions with others (2 hours Core-Tier1, 5 hours Core-Tier2)			8				8
★	HCI/Accountability and Responsibility in Design (2 hours Core-Tier1, 2 hours Core-Tier2)			8				8
★	HCI/Accessibility and Inclusive Design (2 hours Core-Tier1, 2 hours Core-Tier2)			8				8
★	HCI/Evaluating the Design (1 hour Core-Tier1, 2 hours Core-Tier2)			8				8
★	HCI/System Design (1 hour Core-Tier1, 5 hours Core-Tier2)			8				8
	HCI/Society, Ethics, and the Profession			8				8
★	SE/Team Collaboration and Dynamics (2 hours Core-Tier1, 2 hours Core-Tier2)		14					14
★	SE/Principles and Software Architecture (1 hour Core-Tier1, 2 hours Core-Tier2)		16					16
★	SE/Coding Practices (1 hour Core-Tier1, 2 hours Core-Tier2)		14					14
★	SE/Verification and validation Concepts (1 hour Core-Tier1, 1 hour Core-Tier2)		10					10
★	SE/Testing Tools and Analysis (1 hour Core-Tier1, 1 hour Core-Tier2)		4					4
★	SE/Performance Testing and Benchmarking (1 hour Core-Tier1, 1 hour Core-Tier2)		4					4
★	SE/Reliability Concepts (2 hours Core-Tier2)		10					10
	Total	76	72	64	40	0	0	

Table 4.8: Computer Science topics by course 7th Semester

		Eighth Sem											.
	Knowledge Units of Computer Science **Courses →**	CS281	CS3I1	CS3P1	CS402	EX301	ET201	AI268	CS391	CS393	CS394	CS395	Total
⭐	AI/Introduction to AI: Concepts and Intelligent Behavior (2 hours Core-Tier1, 1 hour Core-Tier2)							5					5
⭐	AI/Problem Characteristics and Agent Nature (1 hour Core-Tier1, 1 hour Core-Tier2)							5					5
⭐	AI/Machine Learning Fundamentals (1 hour Core-Tier1, 2 hours Core-Tier2)							5					5
⭐	AI/Machine Learning Data, Representations and Evaluation (1 hour Core-Tier1, 2 hours Core-Tier2)							5					5
⭐	AI/Neural Networks and Ethics (1 hour Core-Tier1, 1 hour Core-Tier2)							4					4
⭐	AI/Machine Learning Formulation and Evaluation (1 hour Core-Tier1, 2 hours Core-Tier2)							3					3
⭐	AI/Deep Learning, RL and Ethics (1 hour Core-Tier1, 2 hours Core-Tier2)							3					3
	AI/Perception and Computer Vision							24					24
⭐	NC/Networked Applications (4 hours Core-Tier1)										6		6
⭐	OS/Virtualization (1 hour Core-Tier2)										6		6
⭐	PDC/Programs: Declarative Parallelism (1 hour Core-Tier1, 1 hour Core-Tier2)			4									4
⭐	PDC/Programs: Starting Activities (1 hour Core-Tier1, 1 hour Core-Tier2)			3									3
⭐	PDC/Programs: Execution Properties (1 hour Core-Tier1, 1 hour Core-Tier2)			3									3
⭐	PDC/Programs: Distribution (1 hour Core-Tier1, 1 hour Core-Tier2)			2									2
⭐	PDC/Programs: Implementation Mappings (1 hour Core-Tier1, 1 hour Core-Tier2)			2									2
	PDC/GPU Programming			4									4
⭐	PDC/Communication (2 hours Core-Tier1, 4 hours Core-Tier2)			8									8
⭐	PDC/Communication: Properties and Extensions (1 hour Core-Tier1, 1 hour Core-Tier2)			3									3
⭐	PDC/Memory and Consistency (1 hour Core-Tier1, 1 hour Core-Tier2)			3									3
⭐	PDC/Coordination (1 hour Core-Tier1, 3 hours Core-Tier2)			7							6		13
⭐	PDC/Coordination: Synchronization and Atomicity (1 hour Core-Tier1, 2 hours Core-Tier2)			3									3
⭐	PDC/Coordination: Advanced Properties (1 hour Core-Tier1, 2 hours Core-Tier2)			4									4
⭐	PDC/Evaluation (1 hour Core-Tier1, 3 hours Core-Tier2)			7									7
⭐	PDC/Evaluation: Formal Methods (1 hour Core-Tier1, 1 hour Core-Tier2)			3									3
⭐	PDC/Evaluation: Testing and Measurement (1 hour Core-Tier1, 1 hour Core-Tier2)			2									2
⭐	PDC/Algorithms (1 hour Core-Tier1, 3 hours Core-Tier2)			4									4

Table 4.9: Computer Science topics by course 8th Semester (Part 1 of 3)

		Eighth Sem											
	Knowledge Units of Computer Science **Courses →**	CS281	CS3I1	CS3P1	CS402	EX301	ET201	AI268	CS391	CS393	CS394	CS395	Total
⭐	PDC/Algorithms: Application Domains Survey (1 hour Core-Tier1, 3 hours Core-Tier2)			4									4
⭐	PDC/Algorithms: Algorithmic Domains (1 hour Core-Tier1, 3 hours Core-Tier2)			4									4
⭐	SE/Version Control and CI/CD (1 hour Core-Tier1, 2 hours Core-Tier2)								12		12	12	36
⭐	SE/Principles and Software Architecture (1 hour Core-Tier1, 2 hours Core-Tier2)								12		8		20
⭐	SE/API Design and Modeling (1 hour Core-Tier1, 1 hour Core-Tier2)										6		6
⭐	SE/Software Design Quality and Evaluation (1 hour Core-Tier1, 2 hours Core-Tier2)										8	4	12
⭐	SE/Large-Scale Construction and Process (1 hour Core-Tier1, 2 hours Core-Tier2)											4	4
⭐	SE/Verification and validation Concepts (1 hour Core-Tier1, 1 hour Core-Tier2)								12				12
⭐	SE/Test Planning and Types (1 hour Core-Tier1, 1 hour Core-Tier2)											8	8
⭐	SE/Advanced Testing Practices (1 hour Core-Tier1, 1 hour Core-Tier2)											4	4
⭐	SE/Testing Tools and Analysis (1 hour Core-Tier1, 1 hour Core-Tier2)								3			8	11
⭐	SE/Performance Testing and Benchmarking (1 hour Core-Tier1, 1 hour Core-Tier2)								3			4	7
⭐	SE/API Compatibility and Versioning (2 hours Core-Tier2)								12			4	16
⭐	SE/Refactoring Techniques (1 hour Core-Tier2)											4	4
⭐	SE/Reliability Engineering (1 hour Core-Tier2)										6	4	10
⭐	SEC/Security Mindset and Principles (1 hour Core-Tier1, 3 hours Core-Tier2)		6										6
⭐	SEC/Security Ethics, Law and Society (1 hour Core-Tier1, 2 hours Core-Tier2)		6										6
⭐	SEC/Injection and Input Validation (1 hour Core-Tier1, 2 hours Core-Tier2)		8										8
⭐	SEC/Malware Analysis and Advanced Security (1 hour Core-Tier1, 3 hours Core-Tier2)		6										6
⭐	SEC/Cryptography Fundamentals (1 hour Core-Tier1, 2 hours Core-Tier2)		8										8
⭐	SEC/Threat Analysis and Security Engineering (1 hour Core-Tier1, 4 hours Core-Tier2)		8										8
⭐	SEC/Digital Forensics Techniques (4 hours Core-Tier2)		6										6
⭐	SEC/Security Governance, Management and Policy (2 hours Core-Tier2)		6										6
⭐	SEP/Social Impact and AI (2 hours Core-Tier1, 1 hour Core-Tier2)	2											2
⭐	SEP/Ethical Foundations and Frameworks (2 hours Core-Tier1, 1 hour Core-Tier2)	4											4
⭐	SEP/Professional Values, Nature and Continuing Development (1 hour Core-Tier1, 1 hour Core-Tier2)	4											4

Table 4.10: Computer Science topics by course 8th Semester (Part 2 of 3)

		Eighth Sem											.
	Knowledge Units of Computer Science **Courses →**	CS281	CS311	CS3P1	CS402	EX301	ET201	AI268	CS391	CS393	CS394	CS395	Total
⭐	SEP/Intellectual Property Foundations (1 hour Core-Tier1, 1 hour Core-Tier2)	4											4
⭐	SEP/Digital IP Rights and Copyright (1 hour Core-Tier1, 1 hour Core-Tier2)	6											6
⭐	SEP/Data, Surveillance and Privacy Technology (1 hour Core-Tier1, 1 hour Core-Tier2)	4											4
⭐	SEP/Technical and Professional Communication (1 hour Core-Tier1, 1 hour Core-Tier2)	2											2
⭐	SEP/Environmental Footprint of Computing Systems (1 hour Core-Tier1, 1 hour Core-Tier2)	2											2
⭐	SEP/Foundations and Early History of Computing (1 hour Core-Tier1, 1 hour Core-Tier2)	2											2
⭐	SEP/Economic Models and Strategies (1 hour Core-Tier2)	2											2
⭐	SEP/Computer Crimes and Attacks (1 hour Core-Tier1, 1 hour Core-Tier2)	2											2
⭐	SEP/Identity and Diversity in Computing (1 hour Core-Tier1, 1 hour Core-Tier2)	2											2
⭐	SF/System Reliability (2 hours Core-Tier1, 1 hour Core-Tier2)										6		6
	Total	36	54	70	0	0	0	54	54	0	64	56	

Table 4.11: Computer Science topics by course 8th Semester (Part 3 of 3)

		Ninth Sem											Tenth Sem				.
	Knowledge Units of Computer Science **Courses →**	CS370	CS400	CS403	AI365	CB309	ET301	AI369	CS351	CS353	CS392	CS3P3	CS3P2	CS404	AI367	AI368	Total
★	AI/Machine Learning Fundamentals (1 hour Core-Tier1, 2 hours Core-Tier2)														12		12
★	AI/Machine Learning Data, Representations and Evaluation (1 hour Core-Tier1, 2 hours Core-Tier2)														10		10
★	AI/Neural Networks and Ethics (1 hour Core-Tier1, 1 hour Core-Tier2)														10		10
★	AI/Machine Learning Formulation and Evaluation (1 hour Core-Tier1, 2 hours Core-Tier2)														3		3
★	AI/Deep Learning, RL and Ethics (1 hour Core-Tier1, 2 hours Core-Tier2)														3		3
★	AI/AI Applications and Societal Impact (2 hours Core-Tier1, 2 hours Core-Tier2)														6		6
★	AI/Additional AI Applications and Societal Impact (2 hours Core-Tier1, 2 hours Core-Tier2)														6		6
	AI/Natural Language Processing														20		20
	AI/Robotics							36									36
	AI/Perception and Computer Vision							12									12
★	AL/Algorithmic Strategies (6 hours Core-Tier1)									12							12
★	AL/Formal Languages and Automata (4 hours Core-Tier1, 10 hours Core-Tier2)									10							10
★	AR/Quantum Architectures (2 hours Core-Tier2)									10							10
★	DM/Core Database System Concepts (2 hours Core-Tier1, 1 hour Core-Tier2)	12															12
★	DM/NoSQL Systems (2 hours Core-Tier2)	12															12
★	DM/Data Analytics (3 hours Core-Tier2)	12															12
	DM/Distributed Databases/Cloud Computing	12											10				22
★	FPL/Parallel and Distributed Computing (2 hours Core-Tier1, 2 hours Core-Tier2)												8				8
★	GIT/Geometric Modeling (6 hours Core-Tier2)								10								10
★	GIT/Shading and Advanced Rendering (6 hours Core-Tier2)								12								12
★	GIT/Computer Animation (6 hours Core-Tier2)								8								8
★	GIT/Simulation (6 hours Core-Tier2)								8								8
★	GIT/Immersion (6 hours Core-Tier2)								5								5
★	GIT/Interaction (4 hours Core-Tier2)								5								5
★	GIT/Tangible/Physical Computing (6 hours Core-Tier2)								8								8
	GIT/Society, Ethics, and the Profession								8								8

Table 4.12: Computer Science topics by course from 9th to 10th Semester (Part 1 of 2)

		Ninth Sem											Tenth Sem				.
	Knowledge Units of Computer Science **Courses →**	CS370	CS400	CS403	AI365	CB309	ET301	AI369	CS351	CS353	CS392	CS3P3	CS3P2	CS404	AI367	AI368	Total
★	NC/Networked Applications (4 hours Core-Tier1)											24	6				30
★	NC/Mobility Support (4 hours Core-Tier2)											12					12
★	PDC/Algorithms: Application Domains Survey (1 hour Core-Tier1, 3 hours Core-Tier2)												4				4
★	SE/Team Collaboration and Dynamics (2 hours Core-Tier1, 2 hours Core-Tier2)										4						4
★	SE/Version Control and CI/CD (1 hour Core-Tier1, 2 hours Core-Tier2)										4						4
★	SE/Requirements Engineering (2 hours Core-Tier2)										6						6
★	SE/Principles and Software Architecture (1 hour Core-Tier1, 2 hours Core-Tier2)										6						6
★	SE/Coding Practices (1 hour Core-Tier1, 2 hours Core-Tier2)										6						6
★	SE/Verification and validation Concepts (1 hour Core-Tier1, 1 hour Core-Tier2)										6						6
★	SE/Testing Tools and Analysis (1 hour Core-Tier1, 1 hour Core-Tier2)										3						3
★	SE/Performance Testing and Benchmarking (1 hour Core-Tier1, 1 hour Core-Tier2)										3						3
★	SE/API Compatibility and Versioning (2 hours Core-Tier2)										6						6
★	SE/Reliability Concepts (2 hours Core-Tier2)										5						5
	SE/Formal Methods										5						5
★	SEC/Security Mindset and Principles (1 hour Core-Tier1, 3 hours Core-Tier2)												8				8
★	SEC/Security Ethics, Law and Society (1 hour Core-Tier1, 2 hours Core-Tier2)											12					12
★	SPD/Cloud Architecture and Data Management (3 hours Core-Tier2)												12				12
	Total	48	0	0	0	0	0	48	64	32	54	48	48	0	70	0	

Table 4.13: Computer Science topics by course from 9th to 10th Semester (Part 2 of 2)

4.5.2 Basic Sciences for Computing topics/course

		First Sem					Second Sem						
	Knowledge Units of Basic Sciences for Computing — Courses →	BIC101	BMA101	BMA102	BCH101	BEI101	CS100	CS112	CS1D1	BMA103	BFI101	BEI102	Total
★	CAN/Parametric Equations and Polar Coordinates (1 hour Core-Tier1, 1 hour Core-Tier2)			6									6
★	CAN/Limits and Continuity (1 hour Core-Tier1, 1 hour Core-Tier2)			12									12
★	CAN/Differential Calculus (1 hour Core-Tier1, 1 hour Core-Tier2)			36									36
★	CAN/Antiderivatives and Integration Techniques (1 hour Core-Tier1, 1 hour Core-Tier2)									20			20
★	CAN/Improper Integrals (1 hour Core-Tier1, 1 hour Core-Tier2)									8			8
★	CAN/Sequences, Series, and Power Series (1 hour Core-Tier1, 1 hour Core-Tier2)									20			20
★	ANT/Matrices and Systems of Linear Equations (1 hour Core-Tier1, 1 hour Core-Tier2)		19										19
★	ANT/Vector Spaces and Linear Transformations (1 hour Core-Tier1, 1 hour Core-Tier2)		14										14
★	ANT/Eigenvalues, Inner Product Spaces, and Quadratic Forms (1 hour Core-Tier1, 1 hour Core-Tier2)		15										15
★	CME/Rotational Motion and Angular Momentum (1 hour Core-Tier1, 1 hour Core-Tier2)										12		12
★	CME/Systems of Particles and Linear Momentum (1 hour Core-Tier1, 1 hour Core-Tier2)										8		8
★	CME/Kinematics and Dynamics of Particles (1 hour Core-Tier1, 1 hour Core-Tier2)										22		22
★	CME/Work, Energy, and Conservation Laws (1 hour Core-Tier1, 1 hour Core-Tier2)										12		12
★	CME/Elasticity (1 hour Core-Tier1, 1 hour Core-Tier2)										2		2
★	CME/Fluids: Hydrostatics, Hydrodynamics, and Viscosity (1 hour Core-Tier1, 1 hour Core-Tier2)										10		10
★	CME/Vibrations and Waves (1 hour Core-Tier1, 1 hour Core-Tier2)										14		14
★	CME/Transport Phenomena (1 hour Core-Tier1, 1 hour Core-Tier2)										4		4
★	GCH/Atomic Structure (1 hour Core-Tier1, 1 hour Core-Tier2)				10								10
★	GCH/Chemical Bonding (1 hour Core-Tier1, 1 hour Core-Tier2)				10								10
★	GCH/Matter and Energy (1 hour Core-Tier1, 1 hour Core-Tier2)				8								8
★	GCH/States of Matter (1 hour Core-Tier1, 1 hour Core-Tier2)				10								10
★	GCH/Stoichiometry (1 hour Core-Tier1, 1 hour Core-Tier2)				10								10
★	GCH/Environmental and Water Chemistry (1 hour Core-Tier1, 1 hour Core-Tier2)				4								4
★	GCH/Polymers and Geosynthetics (1 hour Core-Tier1, 1 hour Core-Tier2)				4								4

		First Sem					Second Sem						
	Knowledge Units of Basic Sciences for Computing **Courses →**	BIC101	BMA101	BMA102	BCH101	BEI101	CS100	CS112	CSIDI	BMA103	BFI101	BEI102	Total
★	GCH/Chemical Equilibrium (1 hour Core-Tier1, 1 hour Core-Tier2)				4								4
★ ★	GCH/Ionic Equilibrium and Salt Hydrolysis (1 hour Core-Tier1, 1 hour Core-Tier2)				12								12
★	GCH/Salts, Solubility, and Precipitation Equilibria (1 hour Core-Tier1, 1 hour Core-Tier2)				6								6
★ ★	GCH/Electrochemistry and Electrolysis (1 hour Core-Tier1, 1 hour Core-Tier2)				8								8
★ ★	GCH/Cement Chemistry (1 hour Core-Tier1, 1 hour Core-Tier2)				4								4
	Total	0	48	54	90	0	0	0	0	48	84	0	

Table 4.15: Basic Sciences for Computing topics by course from 1st to 2nd Semester (Part 2 of 2)

		Third Sem						Fourth Sem							
	Knowledge Units of Basic Sciences for Computing — Courses →	CS113	CS2B1	AI161	BMA104	STA251	BEI201	CS210	CS211	CS221	CS271	CS401	MA202	BEI202	Total
★	CAN/Line Integrals and Vector Fields (1 hour Core-Tier1, 1 hour Core-Tier2)				10										10
★	CAN/Multiple Integrals (1 hour Core-Tier1, 1 hour Core-Tier2)				18										18
★	CAN/Multivariable Transformations (1 hour Core-Tier1, 1 hour Core-Tier2)				12										12
★	CAN/Surface Integrals and Fundamental Theorems (1 hour Core-Tier1, 1 hour Core-Tier2)				12										12
★	CAN/Vector Functions of a Real Variable (1 hour Core-Tier1, 1 hour Core-Tier2)				14										14
★	CAN/Multivariable Differential Calculus (1 hour Core-Tier1, 1 hour Core-Tier2)				18										18
★	DMC/Graph Theory (1 hour Core-Tier1, 1 hour Core-Tier2)												4		4
★	PST/Time Series Analysis (1 hour Core-Tier1, 1 hour Core-Tier2)												2		2
★	NSA/Approximation Theory and Interpolation (1 hour Core-Tier1, 1 hour Core-Tier2)												10		10
★	NSA/Numerical Integration and Quadrature (1 hour Core-Tier1, 1 hour Core-Tier2)												10		10
★	NSA/Error Analysis and Floating-Point Arithmetic (1 hour Core-Tier1, 1 hour Core-Tier2)												6		6
★	NSA/Root-Finding for Nonlinear Equations (1 hour Core-Tier1, 1 hour Core-Tier2)												10		10
★	NSA/Numerical Linear Algebra (1 hour Core-Tier1, 1 hour Core-Tier2)												12		12
★	NSA/Numerical Methods for Differential Equations (1 hour Core-Tier1, 1 hour Core-Tier2)												12		12
★	NSA/Optimization Algorithms (1 hour Core-Tier1, 1 hour Core-Tier2)												4		4
★	NSA/Finite Element Method (1 hour Core-Tier1, 1 hour Core-Tier2)												8		8
★	NSA/Parallel and High-Performance Computing (1 hour Core-Tier1, 1 hour Core-Tier2)												4		4
★	PRO/Sample Spaces and Events (1 hour Core-Tier1, 1 hour Core-Tier2)					4									4
★	PRO/Probability Axioms and Properties (1 hour Core-Tier1, 1 hour Core-Tier2)					4									4
★	PRO/Conditional Probability and Independence (1 hour Core-Tier1, 1 hour Core-Tier2)					4									4
★	PRO/Discrete Random Variables and Distributions (1 hour Core-Tier1, 1 hour Core-Tier2)					4									4
★	PRO/Continuous Random Variables and Distributions (1 hour Core-Tier1, 1 hour Core-Tier2)					4									4

Third Sem								Fourth Sem								.
Knowledge Units of Basic Sciences for Computing	**Courses →**	CS113	CS2B1	AI161	BMA104	STA251	BEI201	CS210	CS211	CS221	CS271	CS401	MA202	BEI202	Total
INF/Interval Estimation and Confidence Intervals (1 hour Core-Tier1, 1 hour Core-Tier2)						4									4
INF/Hypothesis Testing Foundations (1 hour Core-Tier1, 1 hour Core-Tier2)															4
INF/Bayesian Inference (1 hour Core-Tier1, 1 hour Core-Tier2)						4									4
REG/Simple Linear Regression (1 hour Core-Tier1, 1 hour Core-Tier2)															4
Total		0	0	0	84	52	0	0	0	0	0	0	82	0	

Table 4.17: Basic Sciences for Computing topics by course from 3rd to 4th Semester (Part 2 of 2)

Fifth Sem								Sixth Sem							.
Knowledge Units of Basic Sciences for Computing	**Courses →**	CS212	CS261	CS272	CS291	CS251	FG106	BEI203	CS231	CS311	CS312	CS342	AI263	FI201	Total
Total		0	0	0	0	0	0	0	0	0	0	0	0	0	

Table 4.18: Basic Sciences for Computing topics by course from 5th to 6th Semester

Seventh Sem						.
Knowledge Units of Basic Sciences for Computing	**Courses →**	CS251	CS292	CS2H1	AI264	FG211	FG350	Total
Total		0	0	0	0	0	0	

Table 4.19: Basic Sciences for Computing topics by course 7th Semester

Eighth Sem												
Knowledge Units of Basic Sciences for Computing	Courses →	CS281	CS311	CS3P1	CS402	EX301	ET201	AI268	CS391	CS393	CS394	CS395
Total		0	0	0	0	0	0	0	0	0	0	0

Table 4.20: Basic Sciences for Computing topics by course 8th Semester

Ninth Sem												
Knowledge Units of Basic Sciences for Computing	Courses →	CS370	CS400	CS403	AI365	CB309	ET301	AI369	CS351	CS353	CS392	CS3P3
Total		0	0	0	0	0	0	0	0	0	0	0

Table 4.21: Basic Sciences for Computing topics by course from 9th to 10th Semester

Tenth Sem						
Knowledge Units of Basic Sciences for Computing	Courses →	CS3P2	CS404	AI367	AI368	Total
Total		0	0	0	0	0

4.6 Expected outcomes distributed by course

The following tables provide an overview of the outcomes expected to be achieved in each course of the present curriculum. The complete list of expected outcomes can be found in Section: 1.5.1.

Skill/Course	First Sem							Second Sem						
	BIC101	BMA101	BMA102	BCH101	BEI101	CS100	CS112	CS1D1	BMA103	BFI101	BEI102
AG-C01) Social impact of computing solutions.											
AG-C02) Professional ethics in computing.						1					
AG-C03) Individual and teamwork.					1						1
AG-C04) Effective communication in computing.					2						2
AG-C05) Project management in computing.											
AG-C06) Continuous learning in computing.											
AG-C07) Computing knowledge.		1	1	1				2	1	1	
AG-C08) Analysis of computing problems.	1					1					
AG-C09) Design and development of solutions.							2				
AG-C10) Inquiry in computing.											
AG-C11) Use of computing tools.											
AG-C12) Management of secure computing technologies.	2					1	2	2			

Table 4.22: Outcomes by course 1st to 2nd Semester

		Third Sem						Fourth Sem						
	Skill/Course	CS113	CS2B1	AI161	BMA104	STA251	BEI201	CS210	CS211	CS221	CS271	CS401	MA202	BEI202
AG-C01)	Social impact of computing solutions.			1										
AG-C02)	Professional ethics in computing.											3		
AG-C03)	Individual and teamwork.						1			2				1
AG-C04)	Effective communication in computing.		2				2					3		2
AG-C05)	Project management in computing.											3		
AG-C06)	Continuous learning in computing.													
AG-C07)	Computing knowledge.				3	1								
AG-C08)	Analysis of computing problems.							2	2			3	2	
AG-C09)	Design and development of solutions.	2	2					2		2	2	3	2	
AG-C10)	Inquiry in computing.													
AG-C11)	Use of computing tools.			2									2	
AG-C12)	Management of secure computing technologies.	3							2		2			

Table 4.23: Outcomes by course 3rd to 4th Semester

Skill/Course	Fifth Sem									Sixth Sem						
	CS212	CS261	CS272	CS291	CS251	FG106	BE1203	CS231	CS311	CS312	CS342	AI263	FI201			
AG-C01) Social impact of computing solutions.																
AG-C02) Professional ethics in computing.						2										
AG-C03) Individual and teamwork.				2	2		1	2								
AG-C04) Effective communication in computing.						2	2									
AG-C05) Project management in computing.																
AG-C06) Continuous learning in computing.																
AG-C07) Computing knowledge.																
AG-C08) Analysis of computing problems.	2	2	2						2	2	2	2	2			
AG-C09) Design and development of solutions.		2		2	2			2				2				
AG-C10) Inquiry in computing.																
AG-C11) Use of computing tools.													2			
AG-C12) Management of secure computing technologies.	2		2						2	2	2					

Table 4.24: Outcomes by course 5th to 6th Semester

Seventh Sem	CS251	CS292	CS2H1	AI264	FG211	FG350
Skill/Course						
AG-C01) Social impact of computing solutions.						
AG-C02) Professional ethics in computing.					2	2
AG-C03) Individual and teamwork.		2	2			
AG-C04) Effective communication in computing.					2	2
AG-C05) Project management in computing.						
AG-C06) Continuous learning in computing.						
AG-C07) Computing knowledge.	2					
AG-C08) Analysis of computing problems.	2					
AG-C09) Design and development of solutions.	3	2	2	2		
AG-C10) Inquiry in computing.						
AG-C11) Use of computing tools.				3		
AG-C12) Management of secure computing technologies.						

Table 4.25: Outcomes by course 7th Semester

Eighth Sem	CS281	CS311	CS3P1	CS402	EX301	ET201	AI268	CS391	CS393	CS394	CS395
Skill/Course											
AG-C01) Social impact of computing solutions.	2				1						
AG-C02) Professional ethics in computing.	2			3							
AG-C03) Individual and teamwork.								2	2		
AG-C04) Effective communication in computing.				3							
AG-C05) Project management in computing.				3							
AG-C06) Continuous learning in computing.					1						
AG-C07) Computing knowledge.							2				
AG-C08) Analysis of computing problems.				3			3	2	2	3	3
AG-C09) Design and development of solutions.		2	2	3							
AG-C10) Inquiry in computing.										3	3
AG-C11) Use of computing tools.		2	2				2			2	2
AG-C12) Management of secure computing technologies.											

Table 4.26: Outcomes by course 8th Semester

		Ninth Sem												Tenth Sem			
	Skill/Course	CS370	CS400	CS403	AI365	CB309	ET301	AI369	CS351	CS353	CS392	CS3P3	CS3P2	CS404	AI367	AI368
AG-C01)	Social impact of computing solutions.		2													
AG-C02)	Professional ethics in computing.		3	3										3		
AG-C03)	Individual and teamwork.		3								2					
AG-C04)	Effective communication in computing.		3	3										3		
AG-C05)	Project management in computing.		2	3										3		
AG-C06)	Continuous learning in computing.		3												2	
AG-C07)	Computing knowledge.															
AG-C08)	Analysis of computing problems.			3										3		2
AG-C09)	Design and development of solutions.	2	3	3	3			2	2		2	2	2	3		
AG-C10)	Inquiry in computing.					2				2						
AG-C11)	Use of computing tools.	2			3	2		3	2	2		2	2			
AG-C12)	Management of secure computing technologies.															2

Table 4.27: Outcomes by course 9$^{\text{th}}$ to 10$^{\text{th}}$ Semester

4.7 Courses by Outcome

4.7.1 Outcome: AG-C01) Social impact of computing solutions.

- AI161 Applied AI (3rd Sem-Pg. 297)
- CS281 Computing in Society (8th Sem-Pg. 419)
- EX301 Extracurricular Activities (8th Sem-Pg. 443)
- CS400 Pre-professional internships (9th Sem-Pg. 478)

4.7.2 Outcome: AG-C02) Professional ethics in computing.

- CS100 Introduction to Computer Science (2nd Sem-Pg. 257)
- CS401 Research Methodology (4th Sem-Pg. 331)
- FG106 Theater (5th Sem-Pg. 370)
- FG211 Professional Ethics (7th Sem-Pg. 414)
- FG350 Leadership and Performance (7th Sem-Pg. 416)
- CS281 Computing in Society (8th Sem-Pg. 419)
- CS402 Capstone Project I (8th Sem-Pg. 439)
- CS400 Pre-professional internships (9th Sem-Pg. 478)
- CS403 Capstone Project II (9th Sem-Pg. 480)
- CS404 Research Workshop (10th Sem-Pg. 517)

4.7.3 Outcome: AG-C03) Individual and teamwork.

- BEI101 Technical and professional English I (1st Sem-Pg. 254)
- BEI102 Technical and professional English II (2nd Sem-Pg. 283)
- BEI201 Technical and professional English III (3rd Sem-Pg. 309)
- CS221 Computer Systems Architecture (4th Sem-Pg. 323)
- BEI202 Technical and professional English IV (4th Sem-Pg. 340)
- CS291 Software Engineering I (5th Sem-Pg. 360)
- CS2S1 Operating systems (5th Sem-Pg. 365)
- BEI203 Technical and professional English V (5th Sem-Pg. 374)
- CS231 Networking and Communication (6th Sem-Pg. 378)
- CS292 Software Engineering II (7th Sem-Pg. 401)
- CS2H1 User Experience (UX) (7th Sem-Pg. 406)
- CS391 Software Engineering III (8th Sem-Pg. 455)
- CS393 Information systems (8th Sem-Pg. 459)
- CS400 Pre-professional internships (9th Sem-Pg. 478)
- CS392 Tópicos en Ingeniería de Software (9th Sem-Pg. 504)

4.7.4 Outcome: AG-C04) Effective communication in computing.

- BEI101 Technical and professional English I (1st Sem-Pg. 254)
- BEI102 Technical and professional English II (2nd Sem-Pg. 283)
- CS2B1 Platform Based Development (3rd Sem-Pg. 293)
- BEI201 Technical and professional English III (3rd Sem-Pg. 309)
- CS401 Research Methodology (4th Sem-Pg. 331)
- BEI202 Technical and professional English IV (4th Sem-Pg. 340)
- FG106 Theater (5th Sem-Pg. 370)
- BEI203 Technical and professional English V (5th Sem-Pg. 374)
- FG211 Professional Ethics (7th Sem-Pg. 414)

- FG350 Leadership and Performance (7th Sem-Pg. 416)
- CS402 Capstone Project I (8th Sem-Pg. 439)
- CS400 Pre-professional internships (9th Sem-Pg. 478)
- CS403 Capstone Project II (9th Sem-Pg. 480)
- CS404 Research Workshop (10th Sem-Pg. 517)

4.7.5 Outcome: AG-C05) Project management in computing.

- CS401 Research Methodology (4th Sem-Pg. 331)
- CS402 Capstone Project I (8th Sem-Pg. 439)
- CS400 Pre-professional internships (9th Sem-Pg. 478)
- CS403 Capstone Project II (9th Sem-Pg. 480)
- CS404 Research Workshop (10th Sem-Pg. 517)

4.7.6 Outcome: AG-C06) Continuous learning in computing.

- EX301 Extracurricular Activities (8th Sem-Pg. 443)
- CS400 Pre-professional internships (9th Sem-Pg. 478)
- AI367 Topics in Artificial Intelligence (10th Sem-Pg. 521)

4.7.7 Outcome: AG-C07) Computing knowledge.

- BMA101 Linear Algebra (1st Sem-Pg. 244)
- BMA102 Differential Calculus (1st Sem-Pg. 246)
- BCH101 Chemistry I (1st Sem-Pg. 249)
- CS1D1 Discrete Structures (2nd Sem-Pg. 273)
- BMA103 Integral Calculus (2nd Sem-Pg. 276)
- BFI101 Physics I (2nd Sem-Pg. 278)
- BMA104 Advanced Differential and Integral Calculus (3rd Sem-Pg. 301)
- STA251 Probability Calculation (3rd Sem-Pg. 304)
- CS251 Computer Graphics (7th Sem-Pg. 398)

4.7.8 Outcome: AG-C08) Analysis of computing problems.

- BIC101 Introduction to Computing (1st Sem-Pg. 239)
- CS100 Introduction to Computer Science (2nd Sem-Pg. 257)
- CS210 Algorithms and Data Structures (4th Sem-Pg. 313)
- CS211 Theory of Computation (4th Sem-Pg. 318)
- CS401 Research Methodology (4th Sem-Pg. 331)
- MA202 Numerical Methods (4th Sem-Pg. 335)
- CS212 Analysis and Design of Algorithms (5th Sem-Pg. 344)
- CS261 Artificial Intelligence (5th Sem-Pg. 349)
- CS272 Databases II (5th Sem-Pg. 356)
- CS311 Algorithms for Complex Problems (6th Sem-Pg. 383)
- CS312 Advanced Data Structures (6th Sem-Pg. 386)
- CS342 Compilers (6th Sem-Pg. 389)
- AI263 Introduction to Machine Learning (6th Sem-Pg. 392)
- FI201 Computational Physics (6th Sem-Pg. 395)
- CS251 Computer Graphics (7th Sem-Pg. 398)
- CS402 Capstone Project I (8th Sem-Pg. 439)
- AI268 Computational Vision (8th Sem-Pg. 449)

- CS403 Capstone Project II (9th Sem-Pg. 480)
- CS404 Research Workshop (10th Sem-Pg. 517)
- AI368 Evolutionary Computing (10th Sem-Pg. 527)

4.7.9 Outcome: AG-C09) Design and development of solutions.

- CS112 Objects-oriented programming I (2nd Sem-Pg. 268)
- CS113 Objects-oriented programming II (3rd Sem-Pg. 287)
- CS2B1 Platform Based Development (3rd Sem-Pg. 293)
- CS210 Algorithms and Data Structures (4th Sem-Pg. 313)
- CS221 Computer Systems Architecture (4th Sem-Pg. 323)
- CS271 Data Management (4th Sem-Pg. 327)
- CS401 Research Methodology (4th Sem-Pg. 331)
- MA202 Numerical Methods (4th Sem-Pg. 335)
- CS261 Artificial Intelligence (5th Sem-Pg. 349)
- CS291 Software Engineering I (5th Sem-Pg. 360)
- CS2S1 Operating systems (5th Sem-Pg. 365)
- CS231 Networking and Communication (6th Sem-Pg. 378)
- AI263 Introduction to Machine Learning (6th Sem-Pg. 392)
- CS251 Computer Graphics (7th Sem-Pg. 398)
- CS292 Software Engineering II (7th Sem-Pg. 401)
- CS2H1 User Experience (UX) (7th Sem-Pg. 406)
- AI264 Deep Learning (7th Sem-Pg. 411)
- CS3I1 Computer Security (8th Sem-Pg. 425)
- CS3P1 Parallel and Distributed Computing (8th Sem-Pg. 430)
- CS402 Capstone Project I (8th Sem-Pg. 439)
- AI268 Computational Vision (8th Sem-Pg. 449)
- CS391 Software Engineering III (8th Sem-Pg. 455)
- CS393 Information systems (8th Sem-Pg. 459)
- CS394 Software Architecture and DevOps for Cloud Software (8th Sem-Pg. 462)
- CS395 Software Quality (8th Sem-Pg. 468)
- CS370 Big Data (9th Sem-Pg. 475)
- CS400 Pre-professional internships (9th Sem-Pg. 478)
- CS403 Capstone Project II (9th Sem-Pg. 480)
- AI365 Advanced Generative AI Models (9th Sem-Pg. 484)
- AI369 Robotics (9th Sem-Pg. 492)
- CS351 Topics in Computer Graphics (9th Sem-Pg. 496)
- CS392 Tópicos en Ingeniería de Software (9th Sem-Pg. 504)
- CS3P3 Internet of Things (9th Sem-Pg. 511)
- CS3P2 Cloud Computing (10th Sem-Pg. 513)
- CS404 Research Workshop (10th Sem-Pg. 517)

4.7.10 Outcome: AG-C10) Inquiry in computing.

- CB309 Bioinformatics (9th Sem-Pg. 486)
- CS353 Quantum Computing (9th Sem-Pg. 500)

4.7.11 Outcome: AG-C11) Use of computing tools.

- AI161 Applied AI (3rd Sem-Pg. 297)

- MA202 Numerical Methods (4th Sem-Pg. 335)
- FI201 Computational Physics (6th Sem-Pg. 395)
- AI264 Deep Learning (7th Sem-Pg. 411)
- CS3I1 Computer Security (8th Sem-Pg. 425)
- CS3P1 Parallel and Distributed Computing (8th Sem-Pg. 430)
- CS394 Software Architecture and DevOps for Cloud Software (8th Sem-Pg. 462)
- CS395 Software Quality (8th Sem-Pg. 468)
- CS370 Big Data (9th Sem-Pg. 475)
- AI365 Advanced Generative AI Models (9th Sem-Pg. 484)
- CB309 Bioinformatics (9th Sem-Pg. 486)
- AI369 Robotics (9th Sem-Pg. 492)
- CS351 Topics in Computer Graphics (9th Sem-Pg. 496)
- CS353 Quantum Computing (9th Sem-Pg. 500)
- CS3P3 Internet of Things (9th Sem-Pg. 511)
- CS3P2 Cloud Computing (10th Sem-Pg. 513)

4.7.12 Outcome: AG-C12) Management of secure computing technologies.

- BIC101 Introduction to Computing (1st Sem-Pg. 239)
- CS100 Introduction to Computer Science (2nd Sem-Pg. 257)
- CS112 Objects-oriented programming I (2nd Sem-Pg. 268)
- CS1D1 Discrete Structures (2nd Sem-Pg. 273)
- CS113 Objects-oriented programming II (3rd Sem-Pg. 287)
- CS211 Theory of Computation (4th Sem-Pg. 318)
- CS271 Data Management (4th Sem-Pg. 327)
- CS212 Analysis and Design of Algorithms (5th Sem-Pg. 344)
- CS272 Databases II (5th Sem-Pg. 356)
- CS311 Algorithms for Complex Problems (6th Sem-Pg. 383)
- CS312 Advanced Data Structures (6th Sem-Pg. 386)
- CS342 Compilers (6th Sem-Pg. 389)
- AI268 Computational Vision (8th Sem-Pg. 449)
- CS394 Software Architecture and DevOps for Cloud Software (8th Sem-Pg. 462)
- CS395 Software Quality (8th Sem-Pg. 468)
- AI368 Evolutionary Computing (10th Sem-Pg. 527)

4.8 Course distribution in the Career

This proposal can be analyzed by the number of credits dedicated to each area, by course levels (Introductory, Intermediate, Advanced, and Projects).

	General Estudies	Specific	Specialization	
1st Sem	20			20
2nd Sem	12	11		23
3rd Sem	7	15		22
4th Sem	2	19	2	23
5th Sem	2	21		23
6th Sem		17	4	21
7th Sem		12	8	20
8th Sem	2	15	4	21
9th Sem		7	15	22
10th Sem		7	8	15
Total	45	124	41	210
	21.4%	59%	19.5%	

Table 4.28: Distribution of credits by area by semester

Figure 4.2: Credits by area by semester

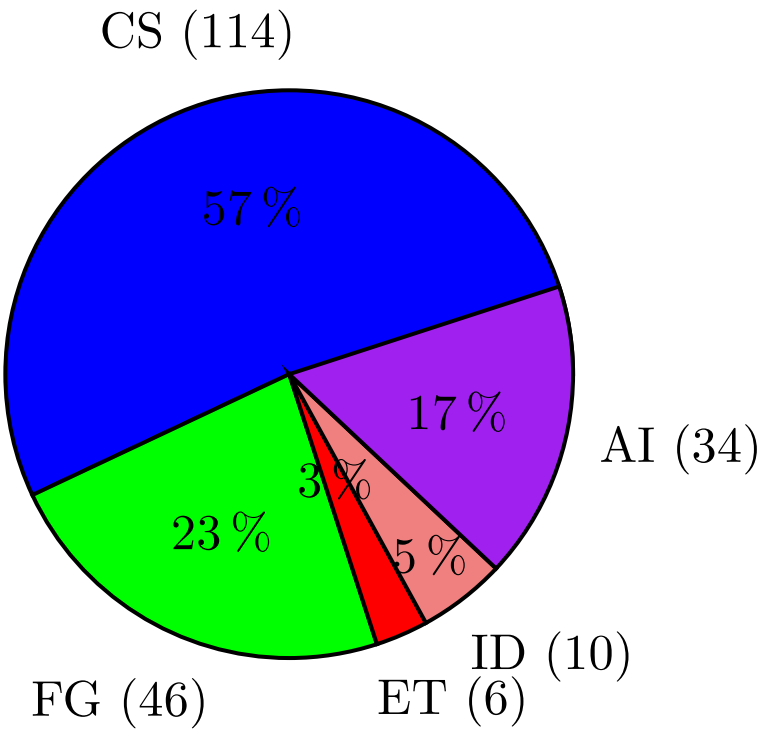

Figure 4.3: Distribution of courses by area considering credits.

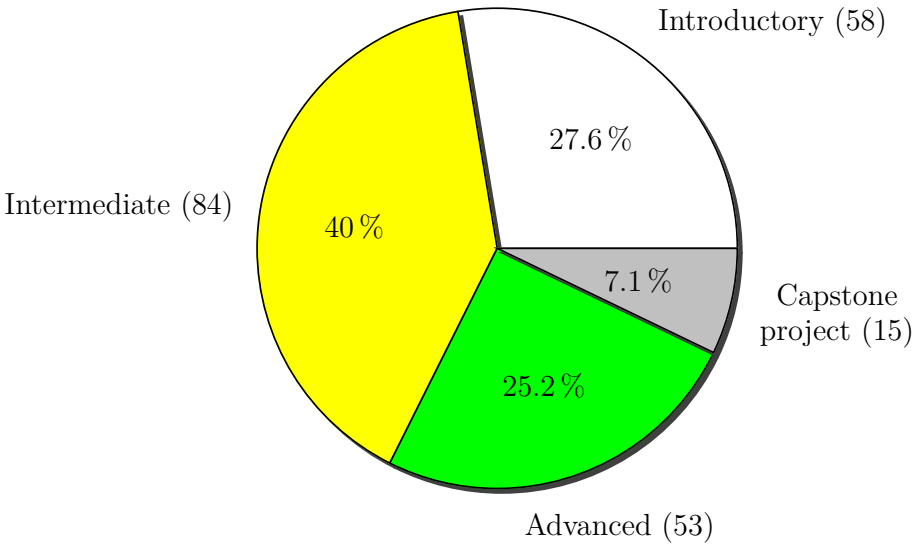

Figure 4.4: Distribution of credits by course levels.

4.9 Departments

4.9.1 DAISI

The list of courses for this department is:

- CB309 Bioinformatics (9th Sem)
- ET301 Entrerpreneurship II (9th Sem)
- AI367 Topics in Artificial Intelligence (10th Sem)
- AI368 Evolutionary Computing (10th Sem)

- CS3P2 Cloud Computing (10th Sem)

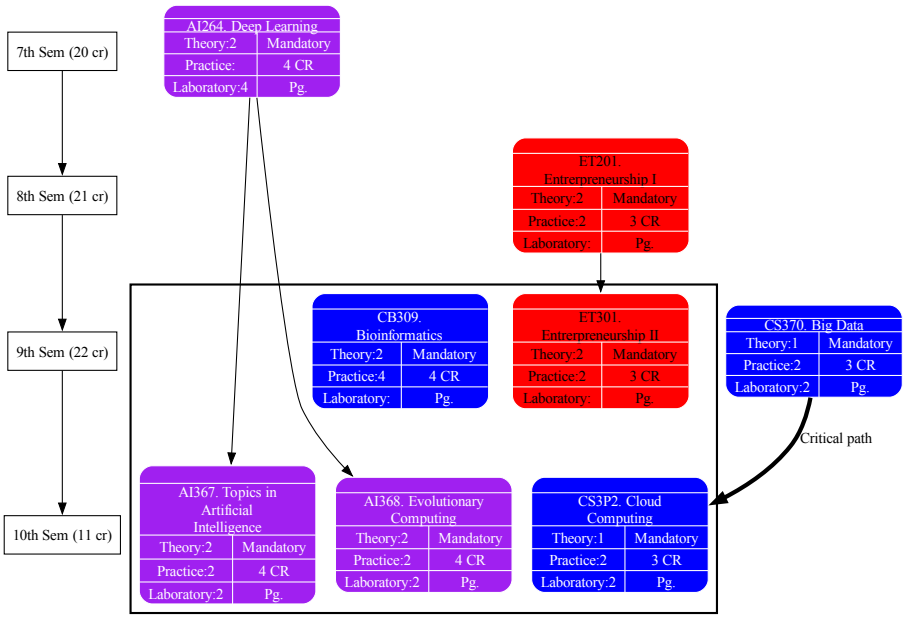

Figure 4.5: DAISI

4.10 Compatibility of the Career with International Standards

In this section, we present the distribution of courses by concentration areas in contrast to the international proposals of *Computing Curricula* from IEEE-CS/ACM.

It is necessary to note that in some cases, courses may appear in more than one axis as they have content from more than one area. For example, the operating systems course contains application units that can be classified in Information Technology but at the same time, it contains fundamentals of how an Operating System is structured, which belongs to the Computer Science axis. In these cases, the credits have been divided between the corresponding axes.

1. Hardware and Architecture (9.50 Créditos)
   - CS221 Computer Systems Architecture (4^{th} Sem-Pg. 323)
   - AI369 Robotics (9^{th} Sem-Pg. 492)
2. Computer Science (134.17 Créditos)
   - BIC101 Introduction to Computing (1^{st} Sem-Pg. 239)
   - CS100 Introduction to Computer Science (2^{nd} Sem-Pg. 257)
   - CS112 Objects-oriented programming I (2^{nd} Sem-Pg. 268)
   - CS113 Objects-oriented programming II (3^{rd} Sem-Pg. 287)
   - CS210 Algorithms and Data Structures (4^{th} Sem-Pg. 313)
   - CS211 Theory of Computation (4^{th} Sem-Pg. 318)
   - CS401 Research Methodology (4^{th} Sem-Pg. 331)
   - CS212 Analysis and Design of Algorithms (5^{th} Sem-Pg. 344)
   - CS2S1 Operating systems (5^{th} Sem-Pg. 365)
   - CS231 Networking and Communication (6^{th} Sem-Pg. 378)
   - CS312 Advanced Data Structures (6^{th} Sem-Pg. 386)
   - CS342 Compilers (6^{th} Sem-Pg. 389)
   - CS251 Computer Graphics (7^{th} Sem-Pg. 398)

- CS2H1 User Experience (UX) (7th Sem-Pg. 406)
- CS281 Computing in Society (8th Sem-Pg. 419)
- CS3P1 Parallel and Distributed Computing (8th Sem-Pg. 430)
- CS402 Capstone Project I (8th Sem-Pg. 439)
- CS403 Capstone Project II (9th Sem-Pg. 480)
- CB309 Bioinformatics (9th Sem-Pg. 486)
- CS351 Topics in Computer Graphics (9th Sem-Pg. 496)
- CS353 Quantum Computing (9th Sem-Pg. 500)
- CS3P3 Internet of Things (9th Sem-Pg. 511)
- CS3P2 Cloud Computing (10th Sem-Pg. 513)
- CS404 Research Workshop (10th Sem-Pg. 517)

3. Artificial Intelligence (60.50 Créditos)
   - AI161 Applied AI (3rd Sem-Pg. 297)
   - CS261 Artificial Intelligence (5th Sem-Pg. 349)
   - AI263 Introduction to Machine Learning (6th Sem-Pg. 392)
   - AI264 Deep Learning (7th Sem-Pg. 411)
   - AI268 Computational Vision (8th Sem-Pg. 449)
   - AI365 Advanced Generative AI Models (9th Sem-Pg. 484)
   - AI369 Robotics (9th Sem-Pg. 492)
   - AI367 Topics in Artificial Intelligence (10th Sem-Pg. 521)
   - AI368 Evolutionary Computing (10th Sem-Pg. 527)
4. Software Engineering (41.67 Créditos)
   - CS112 Objects-oriented programming I (2nd Sem-Pg. 268)
   - CS113 Objects-oriented programming II (3rd Sem-Pg. 287)
   - CS291 Software Engineering I (5th Sem-Pg. 360)
   - CS292 Software Engineering II (7th Sem-Pg. 401)
   - CS2H1 User Experience (UX) (7th Sem-Pg. 406)
   - CS391 Software Engineering III (8th Sem-Pg. 455)
   - CS393 Information systems (8th Sem-Pg. 459)
   - CS394 Software Architecture and DevOps for Cloud Software (8th Sem-Pg. 462)
   - CS395 Software Quality (8th Sem-Pg. 468)
   - CS392 Tópicos en Ingeniería de Software (9th Sem-Pg. 504)
5. Information Systems
   - Ninguno
6. Information Technology (34.17 Créditos)
   - CS2B1 Platform Based Development (3rd Sem-Pg. 293)
   - CS271 Data Management (4th Sem-Pg. 327)
   - CS272 Databases II (5th Sem-Pg. 356)
   - CS2S1 Operating systems (5th Sem-Pg. 365)
   - CS231 Networking and Communication (6th Sem-Pg. 378)
   - CS2H1 User Experience (UX) (7th Sem-Pg. 406)
   - CS281 Computing in Society (8th Sem-Pg. 419)
   - CS370 Big Data (9th Sem-Pg. 475)
   - CS3P3 Internet of Things (9th Sem-Pg. 511)
   - CS3P2 Cloud Computing (10th Sem-Pg. 513)
7. Data Science (10.00 Créditos)
   - CS271 Data Management (4th Sem-Pg. 327)
   - CS272 Databases II (5th Sem-Pg. 356)
   - CS370 Big Data (9th Sem-Pg. 475)
8. Cybersecurity (6.00 Créditos)

- CS3I1 Computer Security (8th Sem-Pg. 425)

9. Mathematics for Computing (14.00 Créditos)
   - CS1D1 Discrete Structures (2nd Sem-Pg. 273)
   - CS311 Algorithms for Complex Problems (6th Sem-Pg. 383)
10. Organizational Issues
    - Ninguno
11. Humanities
    - Ninguno
12. Basic Sciences (76.00 Créditos)
    - BMA101 Linear Algebra (1st Sem-Pg. 244)
    - BMA102 Differential Calculus (1st Sem-Pg. 246)
    - BCH101 Chemistry I (1st Sem-Pg. 249)
    - BMA103 Integral Calculus (2nd Sem-Pg. 276)
    - BFI101 Physics I (2nd Sem-Pg. 278)
    - BMA104 Advanced Differential and Integral Calculus (3rd Sem-Pg. 301)
    - STA251 Probability Calculation (3rd Sem-Pg. 304)
    - MA202 Numerical Methods (4th Sem-Pg. 335)
    - FI201 Computational Physics (6th Sem-Pg. 395)
13. Business (12.00 Créditos)
    - ET201 Entrerpreneurship I (8th Sem-Pg. 445)
    - ET301 Entrerpreneurship II (9th Sem-Pg. 489)
14. General Knowledge (34.00 Créditos)
    - BEI101 Technical and professional English I (1st Sem-Pg. 254)
    - BEI102 Technical and professional English II (2nd Sem-Pg. 283)
    - BEI201 Technical and professional English III (3rd Sem-Pg. 309)
    - BEI202 Technical and professional English IV (4th Sem-Pg. 340)
    - FG106 Theater (5th Sem-Pg. 370)
    - BEI203 Technical and professional English V (5th Sem-Pg. 374)
    - FG211 Professional Ethics (7th Sem-Pg. 414)
    - FG350 Leadership and Performance (7th Sem-Pg. 416)
    - EX301 Extracurricular Activities (8th Sem-Pg. 443)
    - CS400 Pre-professional internships (9th Sem-Pg. 478)

Considering this distribution, figures 4.6 to 4.18 allow us to have a graphical view of this curriculum against the proposals of programs presented by IEEE-CS/ACM in the *Computing Curricula*.

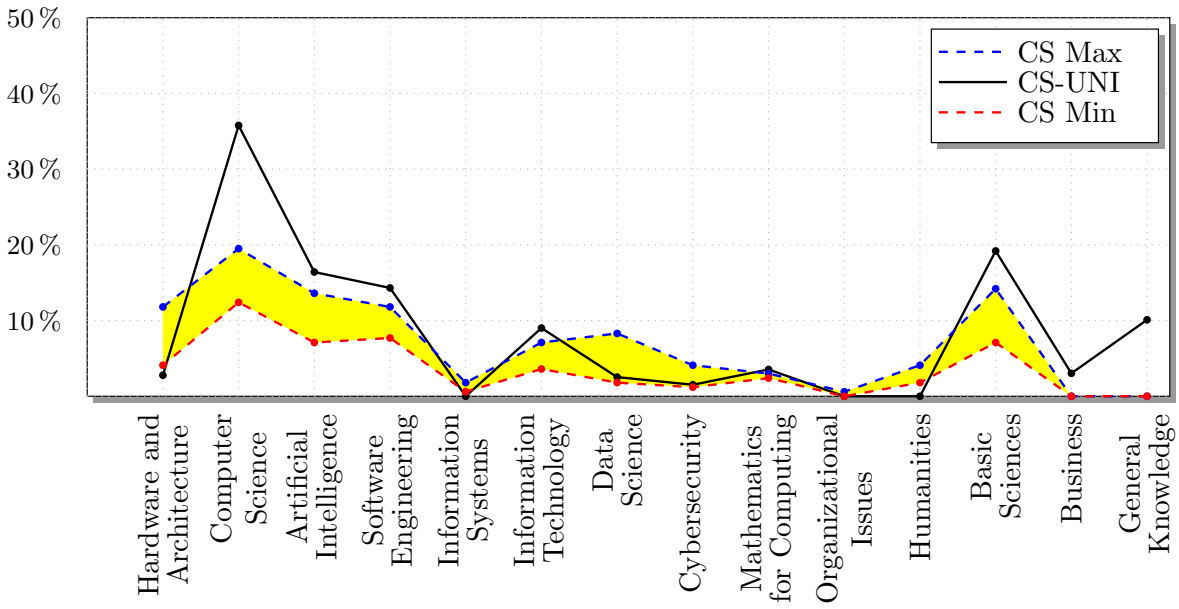

Figure 4.6: Comparing CS-UNI and CS proposed by ACM/IEEE-CS (Curve chart).

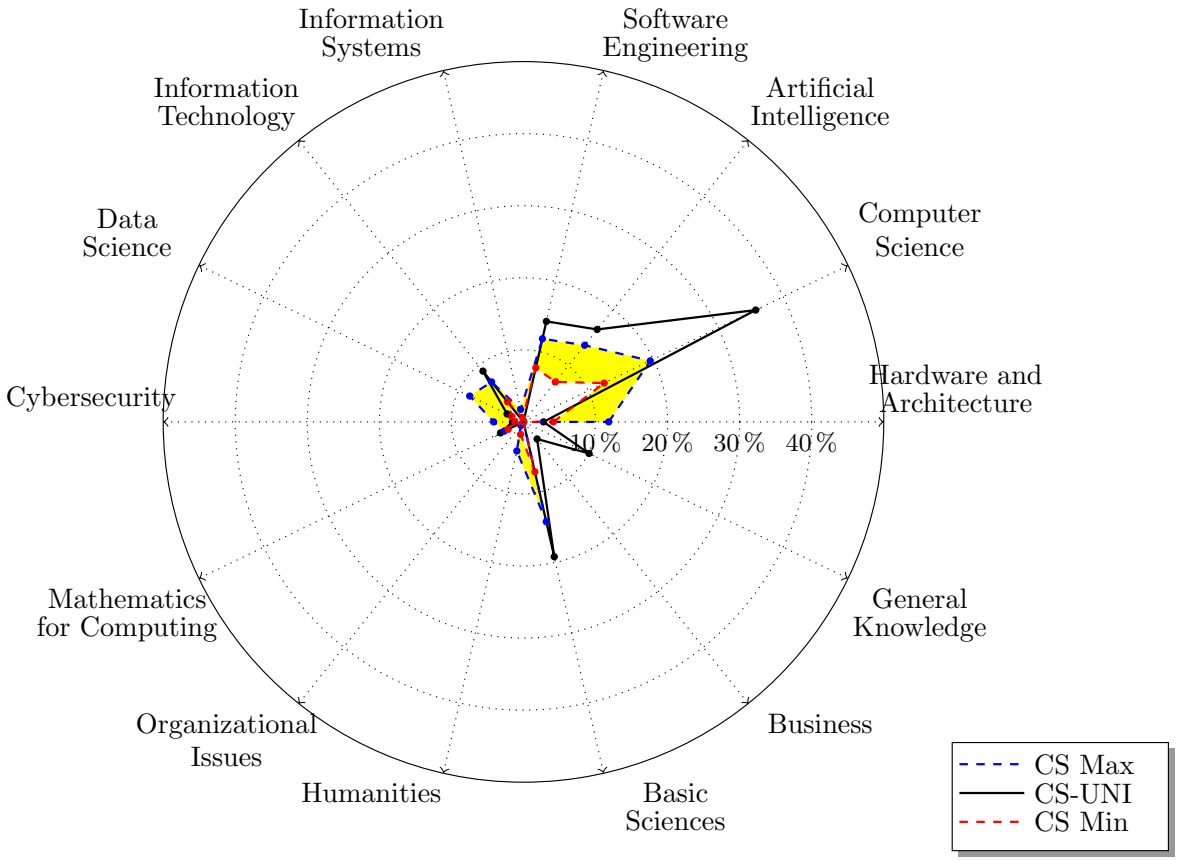

Figure 4.7: Comparing CS-UNI and CS proposed by ACM/IEEE-CS (Spider chart).

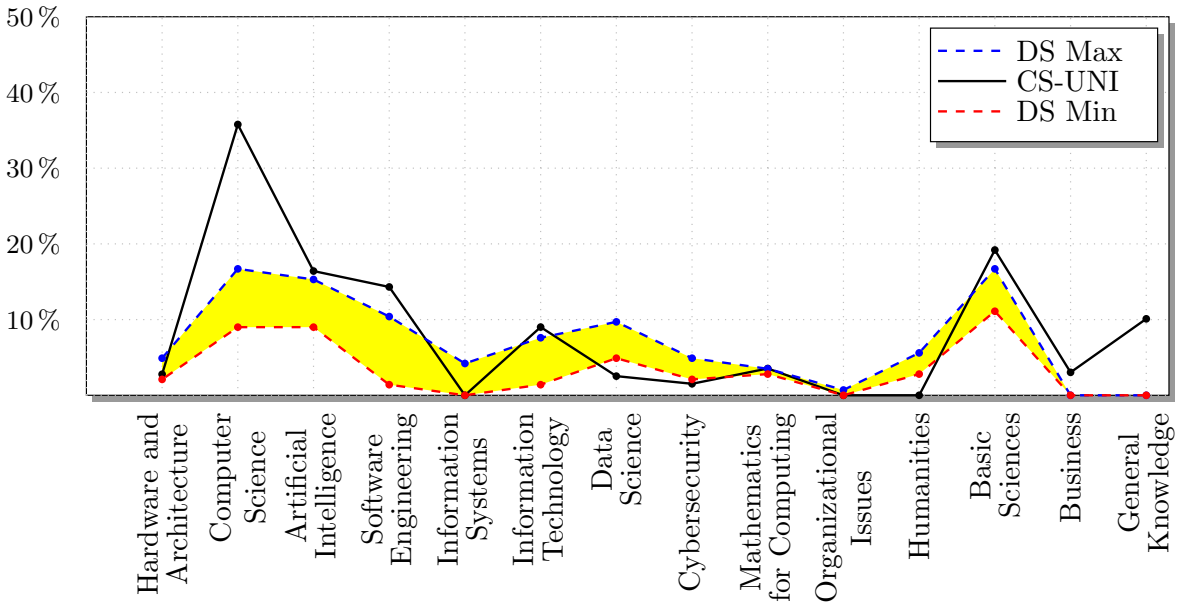

Figure 4.8: Comparing CS-UNI and DS proposed by ACM/IEEE-CS (Curve chart).

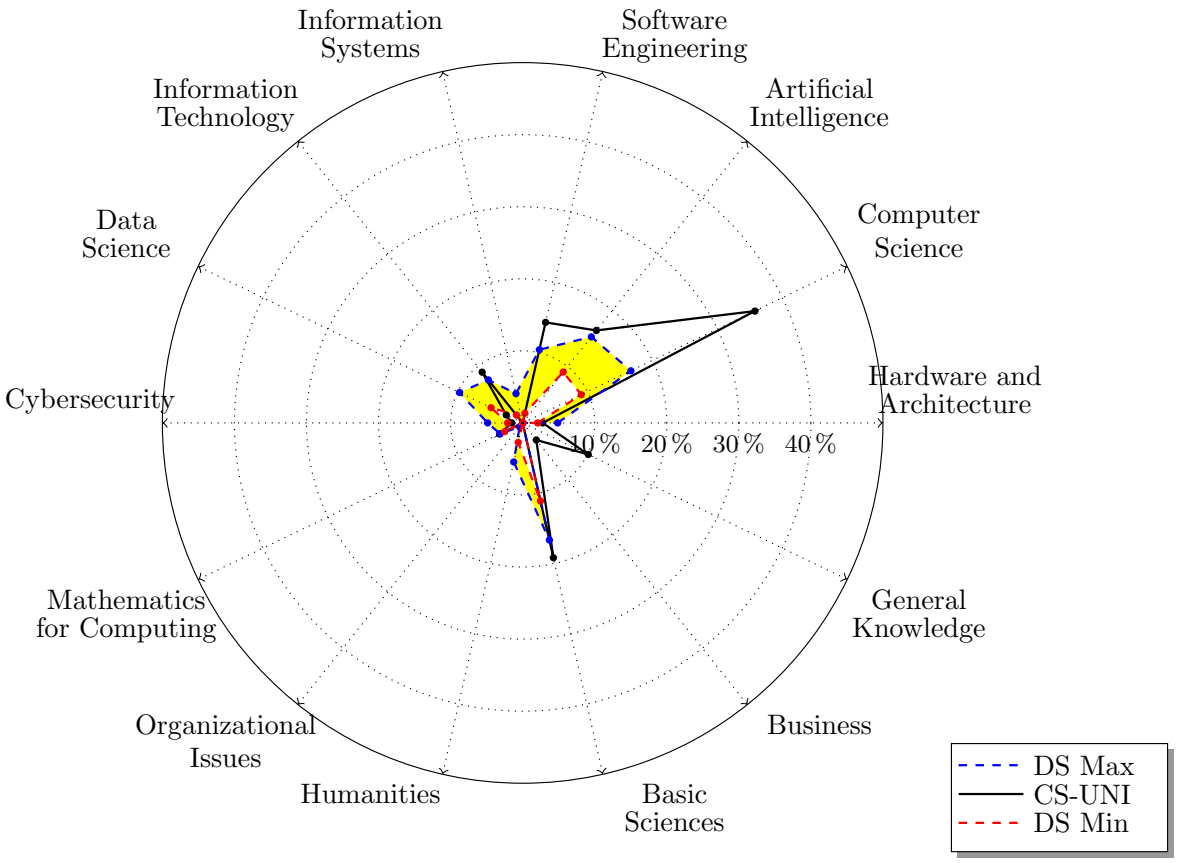

Figure 4.9: Comparing CS-UNI and DS proposed by ACM/IEEE-CS (Spider chart).

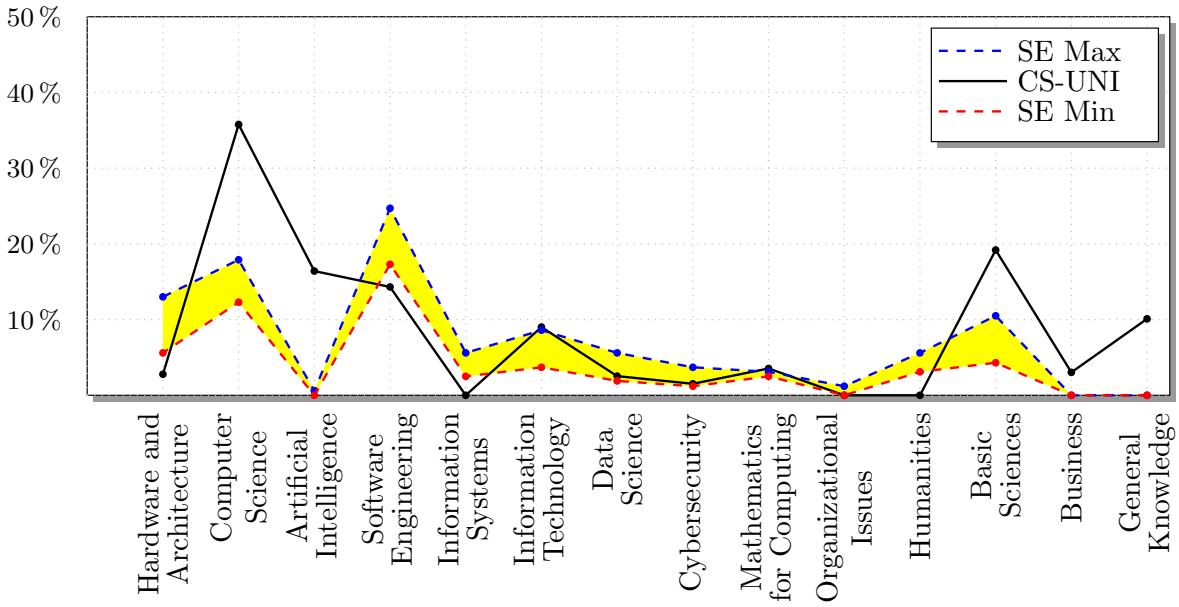

Figure 4.10: Comparing CS-UNI and SE proposed by ACM/IEEE-CS (Curve chart).

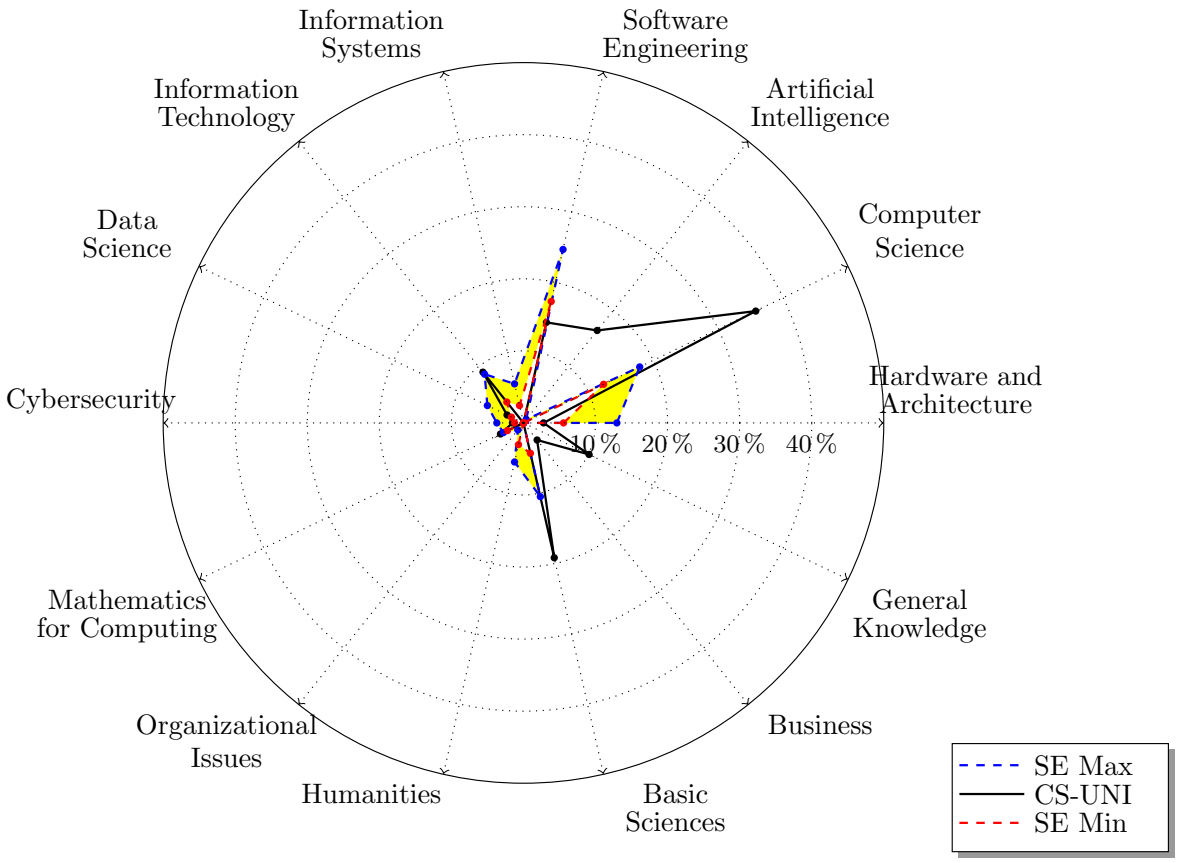

Figure 4.11: Comparing CS-UNI and SE proposed by ACM/IEEE-CS (Spider chart).

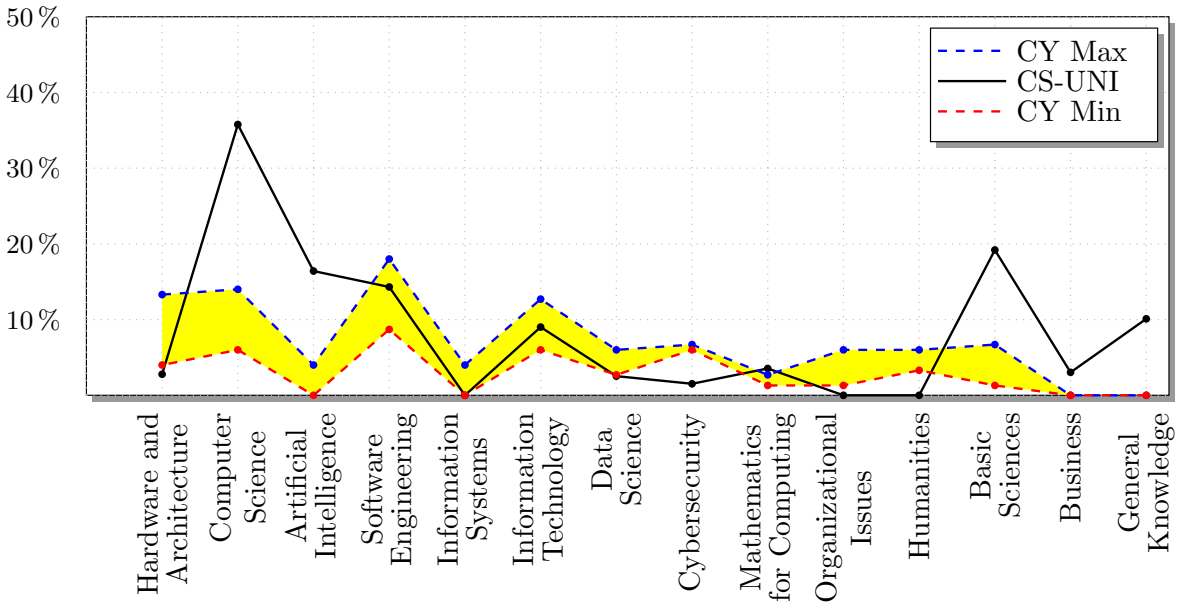

Figure 4.12: Comparing CS-UNI and CY proposed by ACM/IEEE-CS (Curve chart).

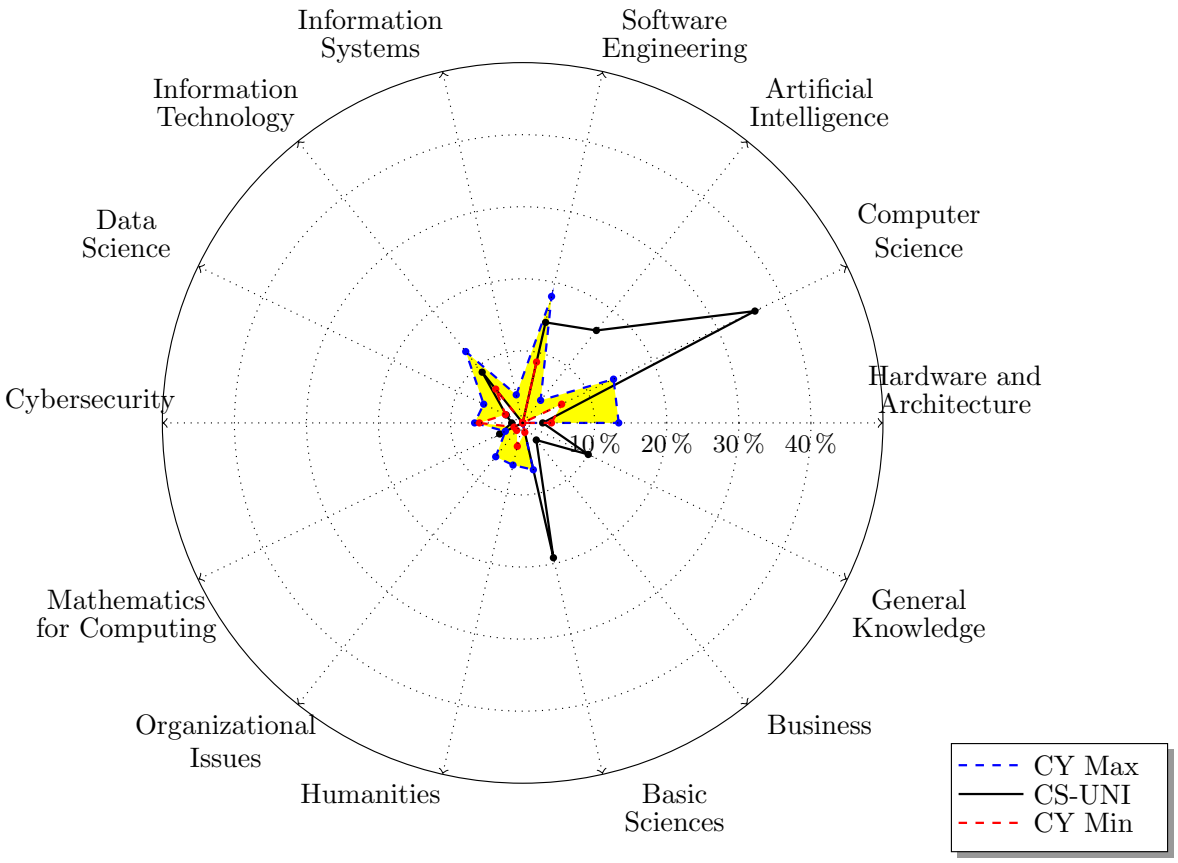

Figure 4.13: Comparing CS-UNI and CY proposed by ACM/IEEE-CS (Spider chart).

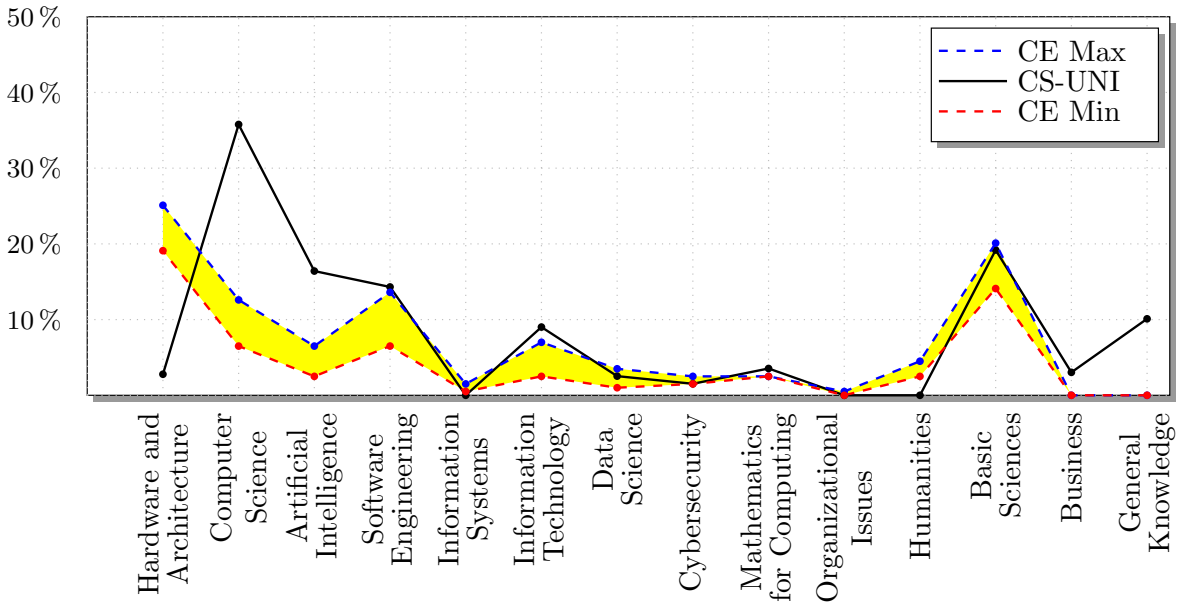

Figure 4.14: Comparing CS-UNI and CE proposed by ACM/IEEE-CS (Curve chart).

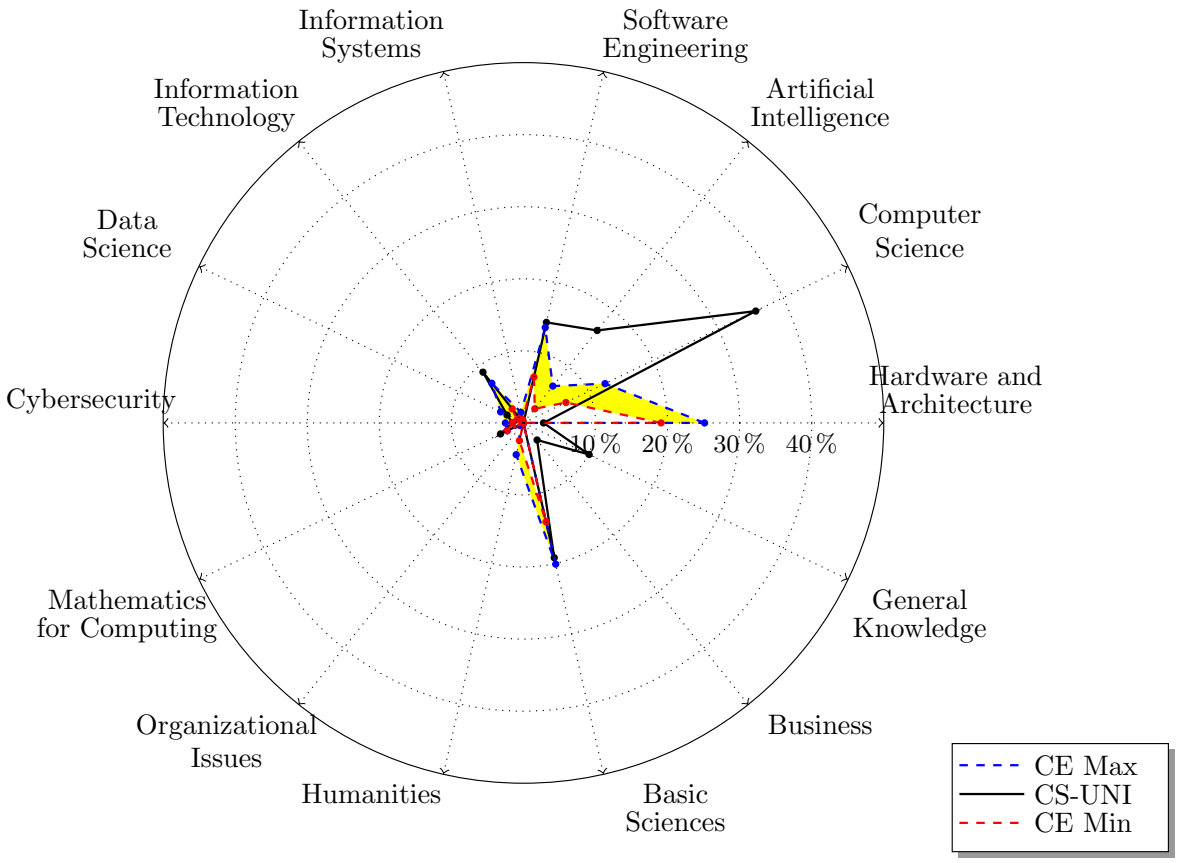

Figure 4.15: Comparing CS-UNI and CE proposed by ACM/IEEE-CS (Spider chart).

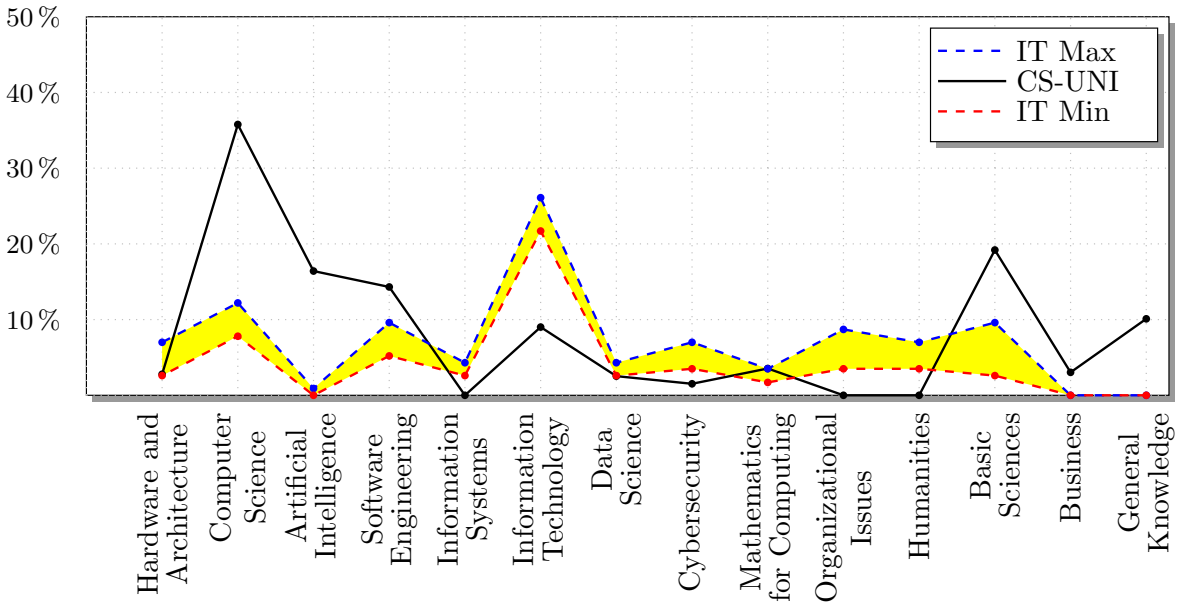

Figure 4.16: Comparing CS-UNI and IT proposed by ACM/IEEE-CS (Curve chart).

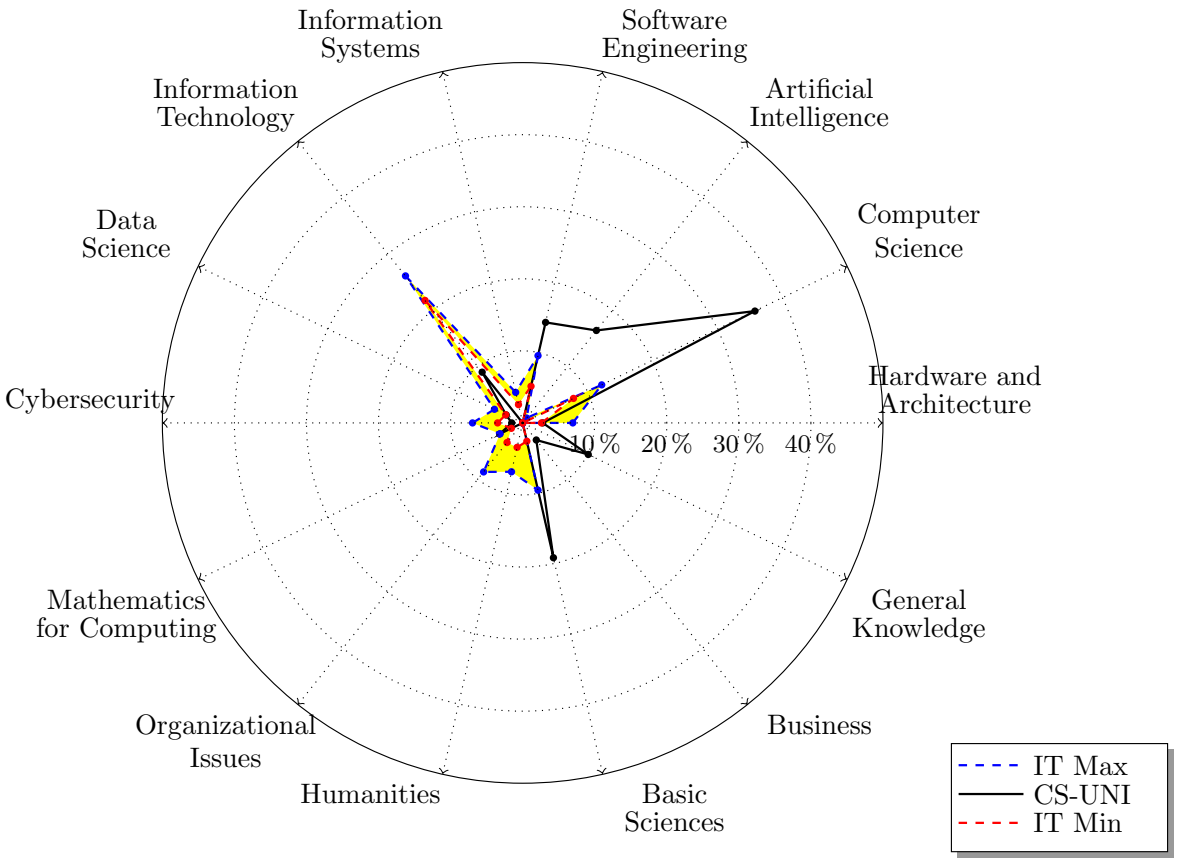

Figure 4.17: Comparing CS-UNI and IT proposed by ACM/IEEE-CS (Spider chart).

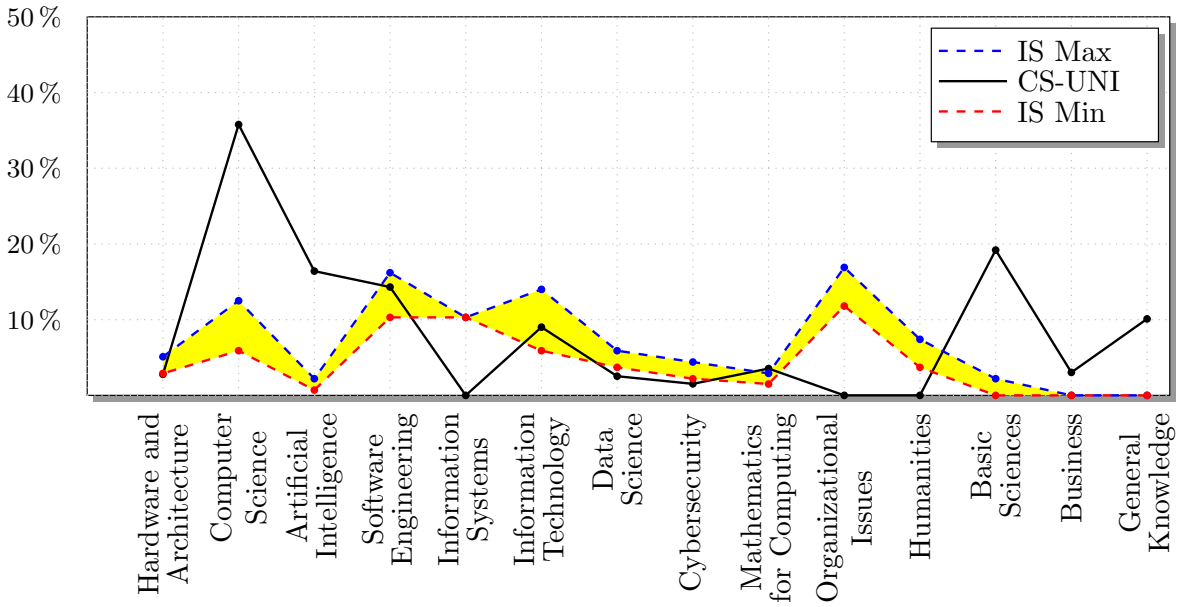

Figure 4.18: Comparing CS-UNI and IS proposed by ACM/IEEE-CS (Curve chart).

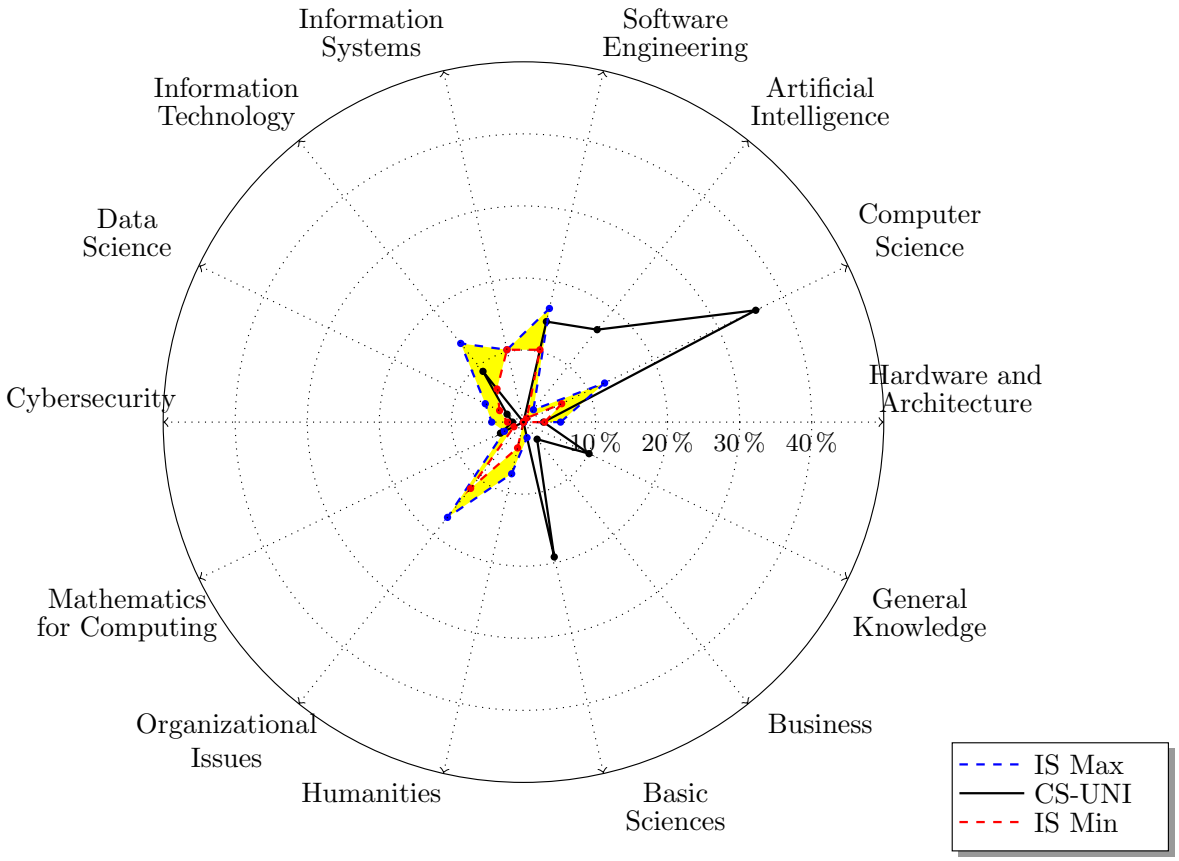

Figure 4.19: Comparing CS-UNI and IS proposed by ACM/IEEE-CS (Spider chart).

Referencias Bibliográficas

[ACM and IEEE-CS, 2017] ACM and IEEE-CS (2017). Information technology curricula 2017. Technical report, ACM and IEEE-CS.

[ACM Data Science Task Force, 2021] ACM Data Science Task Force (2021). Computing competencies for undergraduate data science curricula. Technical report, Association for Computing Machinery (ACM).

[ACM/IEEE-CS, 2020] ACM/IEEE-CS (2020). Computing curricula 2020. Technical report, ACM Press and IEEE Computer Society Press.

[ACM/IEEE-CS Joint Task Force on Cybersecurity Education, 2017] ACM/IEEE-CS Joint Task Force on Cybersecurity Education (2017). Cybersecurity curricula 2017. Technical report, ACM Press and IEEE Computer Society Press and Association for Information Systems Special Interest Group on Information Security and Privacy (AIS SIGSEC) and International Federation for Information Processing Technical Committee on Information Security Education (IFIP WG 11.8).

[ACM/IEEE-CS/AAAI Joint Task Force, 2023] ACM/IEEE-CS/AAAI Joint Task Force (2023). Cs2023: Acm/ieee-cs/aaai computer science curricula. Technical report, ACM Press and IEEE Computer Society Press and AAAI Press.

[Díaz-Herrera and Hilburn, 2004] Díaz-Herrera, J. L. and Hilburn, T. B. (2004). Software engineering: Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering. Technical report, ACM, IEEE.

[LeBlanc et al., 2015] LeBlanc, R., Sobel, A., Ben-Menachem, M., Lethbridge, T. C., Díaz-Herrera, J. L., Hilburn, T. B., McGettrick, A., Atlee, J. M., Hawthorne, E. K., Leaney, J., Budgen, D., Matsumoto, Y., Thompson, J. B., Ardis, M., Hislop, G., Offutt, J., Sebern, M., and Visser, W. (2015). Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering. Technical report, IEEE Computer Society, ACM.

[Shakelford et al., 2005] Shakelford, R., Cross, J. H., Davies, G., Impagliazzo, J., Kamali, R., LeBlanc, R., Lunt, B., McGettrick, A., Sloan, R., and Topi, H. (2005). Computing curricula 2005. Technical report, ACM/IEEE.

[Soldan et al., 2016] Soldan, D., Aylor, J., Clements, A., Engel, G., Hoelzeman, R., Hughes, E. A., Hughes, J. L., Impagliazzo, J., Jaeger, R. C., Klenke, R., Lyon, D. A., McGettrick, A., Nelson, V. P., Neebel, D. J., Page, I., Peterson, G. D., Ranganathan, N., Sloan, R., Srimani, P. K., Theys, M. D., Wolf, W., and Varanasi, M. (2016). Computer engineering curricula 2016. Technical report, ACM and IEEE-CS.

[The Joint ACM/AIS IS2020 Task Force, 2020] The Joint ACM/AIS IS2020 Task Force (2020). A competency model for undergraduate programs in information systems. Technical report, ACM Press and AIS Press.

Chapter 5

Detailed content by course

First Semester

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

BIC101 Introduction to Computing (Mandatory)

1. General information

- **Semester**: 1st Sem. **Credits**: 4
- **Hour of this course**: **Theory**: 2 hours; **Laboratory**: 4 hours;
- **Prerrequisites**: None

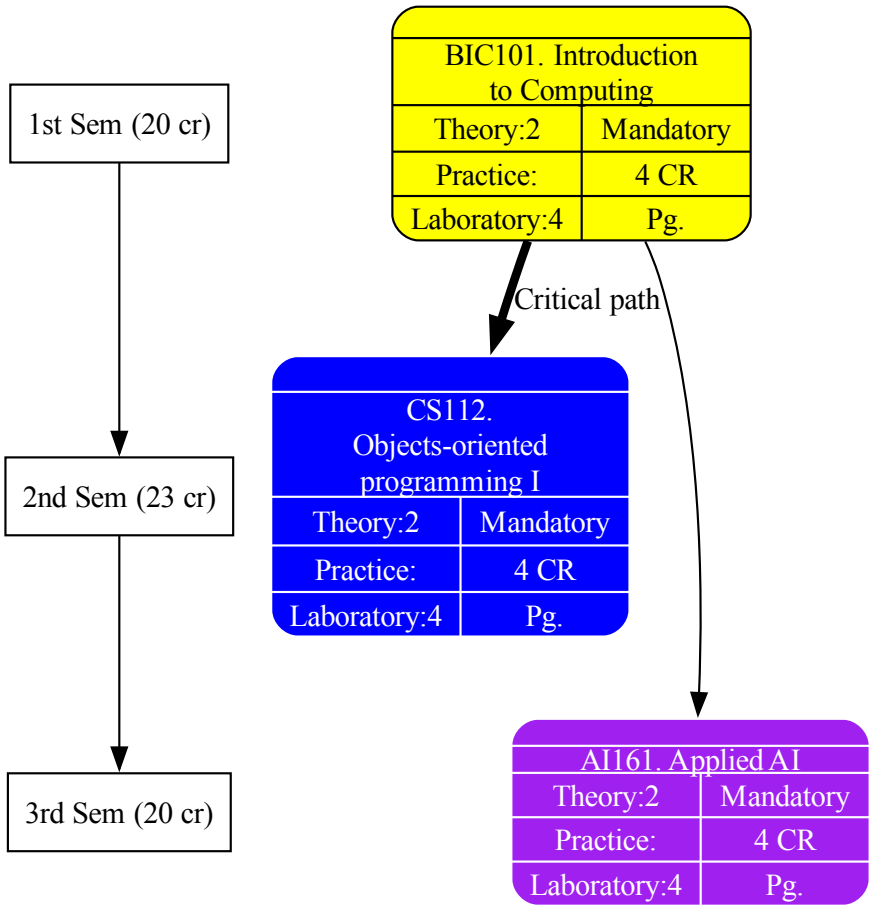

Figure 5.1: Connection Map. BIC101 Introduction to Computing

2. Course Rationale

This is the first course in the sequence of introductory courses to Computer Science. This course introduces participants to the fundamental concepts of programming using Python. Topics include data types, control structures, functions, lists, recursion, and the mechanics of execution, testing, and debugging. Programming is one of the pillars of Computer Science; any professional in the area will need to program to materialize their models and proposals.

3. Goals of the Course

- Introduce the fundamental concepts of programming.
- Develop the ability of abstraction using a programming language.
- Develop problem-solving skills through algorithmic thinking.
- Understand basic data structures and their applications.
- Master the Python programming language for problem solving.

4. General Competences (*Outcomes*)

AG-C08) Problem Analysis: Identifies, formulates, and analyzes complex computing problems. (Familiarity)

AG-C12) Applies computer science theory and software development fundamentals to produce computer-based solutions. (Usage)

5. Content

5.1 Introduction to Programming and History, 4 hr(s)

Competences: AG-C08,AG-C12

Topics: **I:** Brief history of computing and programming languages. **II:** Overview of Python programming language. **III:** Programming environments and tools. **IV:** Basic program structure and execution. **V:** Writing, running, and debugging simple programs.

Learning Outcomes **I:** Identify important trends in the history of computing. [Familiarity] **II:** Discuss the historical context of programming language paradigms. [Familiarity] **III:** Compare daily life before and after personal computers and the Internet. [Assessment] **IV:** Write, compile, and run a simple Python program. [Usage]

Bibliography: [Zelle, 2010, Guttag, 2013, Brookshear and Brylow, 2019]

5.2 Type Systems Fundamentals, 6 hr(s)

Competences: AG-C08,AG-C12

Topics: **I:** A type as a set of values together with a set of operations: (a) Primitive types (e.g., numbers, Booleans) (b) Compound types built from other types (e.g., records/structs, unions, arrays, lists, functions, references using set operations) **II:** Type safety and errors caused by using values inconsistently given their intended types **III:** Goals and limitations of static and dynamic typing: detecting and eliminating errors as early as possible. **IV:** Primitive data types in Python (int, float, str, bool). **V:** Type conversion and casting.

Learning Outcomes **I:** Describe, for both a primitive and a compound type, the values that have that type [Familiarity] **II:** Describe examples of program errors detected by a type system [Familiarity] **III:** Use types and type-error messages to write and debug programs [Usage] **IV:** Explain Python's dynamic typing system. [Familiarity]

Bibliography: [Zelle, 2010, Guttag, 2013]

5.3 Object-Oriented Programming Fundamentals, 14 hr(s)

Competences: AG-C08,AG-C12

Topics: **I:** Imperative programming as a subset of object-oriented programming. **II:** Definition of classes: fields, methods, and constructors. **III:** Dynamic vs static properties. **IV:** Variables, expressions, and assignments. **V:** Input and output operations. **VI:** Conditional statements (if, elif, else). **VII:** Iteration structures (for loops, while loops). **VIII:** Nested loops and loop control statements. **IX:** Exception handling basics.

Learning Outcomes **I:** Compose a class through design, implementation, and testing to meet behavioral requirements [Usage] **II:** Predict and validate control flow in a program using dynamic dispatch [Assessment] **III:** Use object-oriented encapsulation mechanisms such as interfaces and private members [Usage] **IV:** Implement programs using conditional and iterative structures. [Usage] **V:** Debug programs with syntax and logical errors. [Usage] **VI:** Design and implement simple algorithms using loops. [Usage]

Bibliography: [Zelle, 2010, Guttag, 2013, Brookshear and Brylow, 2019]

5.4 Functions and Parameter Passing, 10 hr(s)

Competences: AG-C08,AG-C12

Topics: **I:** Function definition and invocation. **II:** Parameter passing mechanisms (by value, by reference). **III:** Return values and multiple returns. **IV:** Variable scope and lifetime (local, global, nonlocal). **V:** Recursion fundamentals. **VI:** Lambda expressions and higher-order functions. **VII:** Functions as first-class objects. **VIII:** Documentation and testing of functions.

Learning Outcomes **I:** Define and implement functions with proper parameter passing. [Usage] **II:** Explain the difference between pass-by-value and pass-by-reference. [Familiarity] **III:** Implement recursive solutions to simple problems. [Usage] **IV:** Use lambda expressions for functional programming constructs. [Usage] **V:** Design functions with appropriate scope and documentation. [Assessment]

Bibliography: [Zelle, 2010, Guttag, 2013, Hunt and Thomas, 1999]

5.5 Basic Data Structures, 10 hr(s)

Competences: AG-C08,AG-C12

Topics: **I:** Lists and list operations (indexing, slicing, mutability). **II:** Tuples and immutability. **III:** Dictionaries and key-value mapping. **IV:** Sets and set operations. **V:** Strings as sequences. **VI:** List comprehensions and generator expressions. **VII:** Basic searching and sorting algorithms. **VIII:** Introduction to algorithmic complexity for basic operations.

Learning Outcomes **I:** Implement programs using lists, dictionaries, and sets. [Usage] **II:** Choose appropriate data structures for given problems. [Assessment] **III:** Explain the trade-offs between different data structures. [Familiarity] **IV:** Implement basic search and sort operations on lists. [Usage]

Bibliography: [Zelle, 2010, Guttag, 2013]

5.6 Object-Oriented Programming Fundamentals, 10 hr(s)

Competences: AG-C08,AG-C12

Topics: **I:** Subclasses, inheritance (including multiple inheritance), and method overriding. **II:** Dynamic dispatch: definition of method-call. **III:** Object-oriented idioms for encapsulation: (a) Privacy, data hiding, and visibility of class members. (b) Interfaces revealing only method signatures. (c) Abstract base classes, traits and mixins. **IV:** Classes and objects in Python. **V:** Attributes and methods. **VI:** Inheritance and polymorphism basics. **VII:** Encapsulation and information hiding.

Learning Outcomes **I:** Build a simple class hierarchy utilizing subclassing that allows code to be reused for distinct subclasses [Usage] **II:** Explain the relationship between object-oriented inheritance (code-sharing and overriding) and subtyping (the idea of a subtype being usable in a context that expects the supertype) [Familiarity] **III:** Compare and contrast the benefits and costs/impact of using inheritance (subclasses) and composition (specifically, how to base composition on higher order functions) [Assessment] **IV:** Design and implement simple classes in Python. [Usage] **V:** Explain the benefits of OOP for code organization. [Familiarity]

Bibliography: [Zelle, 2010, Guttag, 2013]

5.7 Algorithmic Strategies, 6 hr(s)

Competences: AG-C08,AG-C12

Topics: **I:** Paradigms: (a) Brute-Force (e.g., linear search, selection sort, traveling salesperson, knapsack) (b) Decrease-and-Conquer: i. By a Constant (e.g., insertion sort, topological sort) ii. By a Constant Factor (e.g., binary search) iii. By a Variable Size (e.g., Euclid's) (c) Divide-and-Conquer (e.g., binary search, quicksort, mergesort, Strassen's) (d) Greedy (e.g., Dijkstra's, Kruskal's, Knapsack) (e) Transform-and-Conquer: i. Instance simplification (e.g., find duplicates via list presort) ii. Representation change (e.g., heapsort) iii. Problem reduction (e.g., least-common-multiple, linear programming) iv. Dynamic programming (e.g., Floyd's, Marshall, Bellman-Ford) (f) Space vs time tradeoffs (e.g., hashing) **II:** Iteration vs recursion (e.g., factorial, tree search) **III:** Problem decomposition strategies. **IV:** Algorithm design techniques. **V:** Debugging and testing strategies. **VI:** Tracing program execution.

Learning Outcomes **I:** For each of the paradigms in this unit: (a) Explain its definitional characteristics (b) Explain an example that demonstrates the paradigm including how this example satisfies the paradigm's characteristics. [Familiarity] **II:** Give examples of iterative and recursive algorithms that solve the same problem, explain the benefits and disadvantages of each approach [Usage] **III:** Evaluate whether a greedy approach leads to an optimal solution [Assessment] **IV:** Apply systematic debugging techniques to locate and fix errors. [Usage] **V:** Analyze algorithm complexity for simple problems. [Familiarity]

Bibliography: [Guttag, 2013, Cormen et al., 2009]

5.8 Modern Development Methods and Tool Usage, 4 hr(s)

Competences: AG-C08,AG-C12

Topics: **I:** Using modern programming environments (IDEs, notebooks). **II:** Code search and reuse best practices. **III:** Using standard libraries and APIs. **IV:** Debugging and testing techniques. **V:** Version control basics. **VI:** Code documentation and style guidelines. **VII:** Virtual environments and package management.

Learning Outcomes **I:** Use modern development tools effectively. [Usage] **II:** Apply code reuse principles through libraries and APIs. [Usage] **III:** Debug and test programs systematically. [Usage] **IV:** Follow coding standards and documentation practices. [Familiarity] **V:** Manage Python packages and environments. [Familiarity]

Bibliography: [Zelle, 2010, Hunt and Thomas, 1999]

6. Bibliography

[Brookshear and Brylow, 2019] Brookshear, J. G. and Brylow, D. (2019). *Computer Science: An Overview*. Pearson, global edition edition.

[Cormen et al., 2009] Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2009). *Introduction to Algorithms*. MIT Press, 3rd edition.

[Guttag, 2013] Guttag, J. V. (2013). *Introduction to Computation and Programming Using Python*. MIT Press, 1st edition.

[Hunt and Thomas, 1999] Hunt, A. and Thomas, D. (1999). *The Pragmatic Programmer: From Journeyman to Master*. Addison-Wesley, 1st edition.

[Zelle, 2010] Zelle, J. M. (2010). *Python Programming: An Introduction to Computer Science*. Franklin, Beedle & Associates Inc, 2nd edition.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

BMA101 Linear Algebra (Mandatory)

1. General information

- **Semester**: 1st Sem. **Credits**: 4
- **Hour of this course**: **Theory**: 3 hours; **Practice**: 2 hours;
- **Prerrequisites**: None

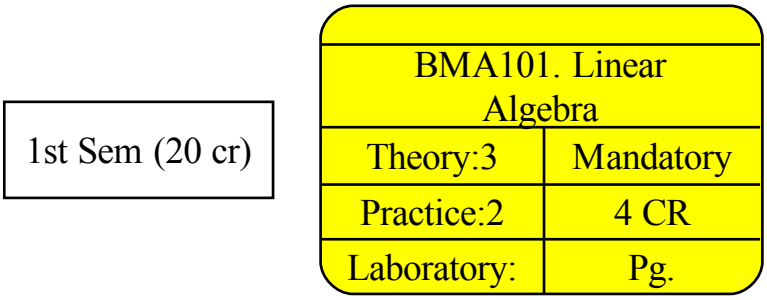

Figure 5.2: Connection Map. BMA101 Linear Algebra

2. Course Rationale

Linear algebra is a fundamental mathematical tool shared by computer science and engineering disciplines, providing the foundations for structural analysis, circuit networks, control systems, data processing, and algorithm design. This course provides a solid foundation in the concepts and techniques of elementary and advanced linear algebra, with emphasis on its application in scientific and engineering contexts.

3. Goals of the Course

- Understand the fundamental concepts of linear algebra, including vector spaces, linear transformations, and matrix theory.
- Apply linear algebra techniques to solve systems of equations and eigenvalue/eigenvector problems arising from scientific and engineering contexts.
- Develop skills in abstract reasoning and logical thinking to address mathematical problems.

4. General Competences (*Outcomes*)

AG-C07) Computing Knowledge: Applies knowledge of mathematics, science, and computing. (Familiarity)

5. Content

5.1 Matrices and Systems of Linear Equations, 19 hr(s)

Competences: AG-C07

Topics: **I:** Matrix algebra: addition, scalar multiplication, matrix multiplication, transpose, types of matrices, and the inverse of a matrix **II:** Solving systems of linear equations via Gaussian

elimination and Gauss-Jordan; row echelon form, rank, and consistency of systems **III:** Determinants: definition, properties, Cramer's rule, and geometric interpretation **IV:** Matrix decompositions: LU, QR, and singular value decomposition (SVD)

Learning Outcomes **I:** Identify types of matrices and perform basic matrix algebra operations, including computing the inverse of a matrix [Familiarity] **II:** Solve systems of linear equations using Gaussian elimination or Gauss-Jordan, determining the rank and consistency of the system [Usage] **III:** Apply Cramer's rule and the LU, QR, and SVD decompositions to solve linear systems and factor matrices [Assessment]

Bibliography: [Strang, 2016, Lay et al., 2016]

5.2 Vector Spaces and Linear Transformations, 14 hr(s)

Competences: AG-C07

Topics: **I:** Vector spaces and subspaces; bases, dimension, and linear independence **II:** Linear maps, kernel and image, rank-nullity theorem, and matrix representations

Learning Outcomes **I:** Identify whether a set with given operations forms a vector space and determine a basis [Familiarity] **II:** Determine the kernel, image, and matrix representation of a linear map, verifying the rank-nullity theorem [Usage] **III:** Apply linear transformations to model and solve problems involving change of basis and composition of maps [Assessment]

Bibliography: [Strang, 2016, Lay et al., 2016]

5.3 Eigenvalues, Inner Product Spaces, and Quadratic Forms, 15 hr(s)

Competences: AG-C07

Topics: **I:** Eigenvalues and eigenvectors; diagonalization and the spectral theorem **II:** Inner product spaces, orthogonality, Gram-Schmidt process, and least squares **III:** Quadratic forms and definiteness: classification as positive, negative, or indefinite via eigenvalues

Learning Outcomes **I:** Identify the eigenvalues and eigenvectors of a matrix and determine whether it is diagonalizable [Familiarity] **II:** Compute the diagonalization of matrices via the spectral theorem and apply the Gram-Schmidt process to obtain orthogonal bases [Usage] **III:** Apply least squares methods and the classification of quadratic forms via eigenvalues to solve approximation and optimization problems [Assessment]

Bibliography: [Strang, 2016, Lay et al., 2016]

6. Bibliography

[Lay et al., 2016] Lay, D. C., Lay, S. R., and McDonald, J. J. (2016). *Linear Algebra and Its Applications*. Pearson.

[Strang, 2016] Strang, G. (2016). *Introduction to Linear Algebra*. Wellesley-Cambridge Press.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

BMA102 Differential Calculus (Mandatory)

1. General information

- **Semester**: 1st Sem. **Credits**: 5
- **Hour of this course**: **Theory**: 4 hours; **Practice**: 2 hours;
- **Prerrequisites**: None

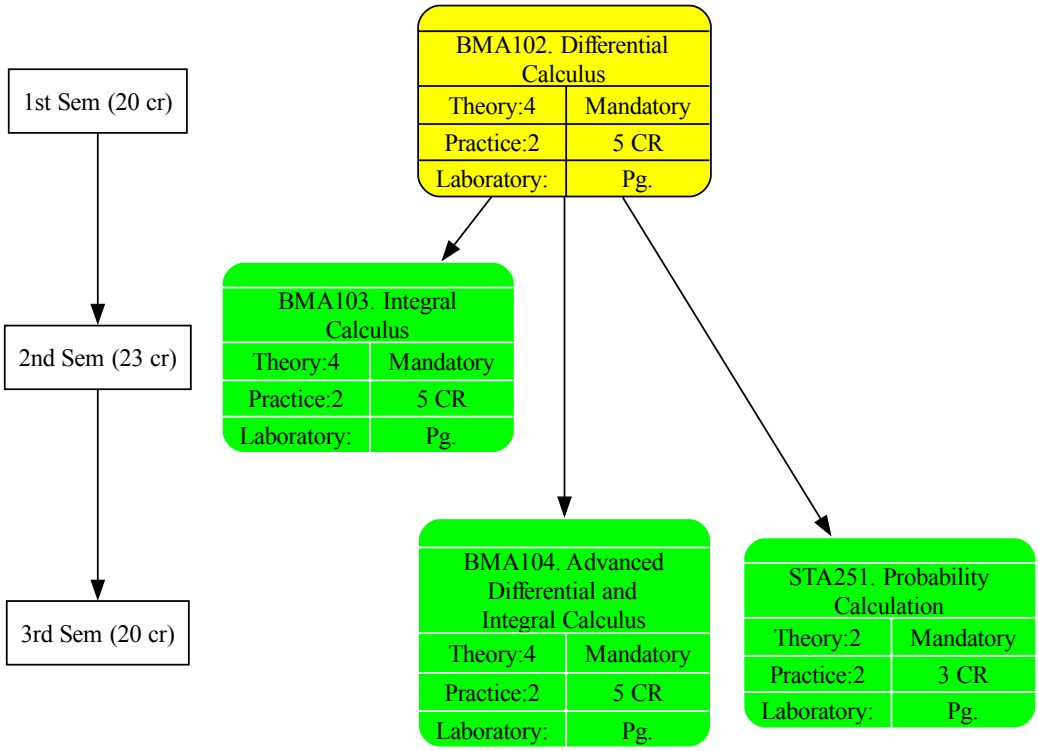

Figure 5.3: Connection Map. BMA102 Differential Calculus

2. Course Rationale

Differential calculus is a fundamental mathematical tool shared by computer science and engineering disciplines, providing the foundations for the rigorous study of change, the analysis of rates of change, the optimization of systems and functions, and the modeling of continuous phenomena. This course provides a solid foundation in the concepts and techniques of differential calculus of one variable, with emphasis on its application in scientific and engineering contexts.

3. Goals of the Course

- Rigorously understand the concepts of limit, continuity, and derivative of functions of one variable.
- Apply differentiation rules to compute derivatives of algebraic, composite, and implicit functions arising from scientific and engineering contexts.
- Use differential calculus to solve optimization problems, related rates problems, and function behavior analysis.

4. General Competences (*Outcomes*)

AG-C07) Computing Knowledge: Applies knowledge of mathematics, science, and computing. (Familiarity)

5. Content

5.1 Limits and Continuity, 12 hr(s)

Competences: AG-C07

Topics: **I:** Epsilon-delta definition of a limit and formal proof techniques **II:** Limit laws, one-sided limits, and limits at infinity **III:** Continuity: types, properties, and the Intermediate Value Theorem **IV:** L'Hôpital's rule and indeterminate forms **V:** Limits of sequences: convergence, Squeeze Theorem, and Bolzano-Weierstrass

Learning Outcomes **I:** State the epsilon-delta definition of a limit and explain its geometric meaning [Familiarity] **II:** Evaluate limits using limit laws, L'Hôpital's rule, and asymptotic analysis [Usage] **III:** Prove continuity or discontinuity of a function and apply the Intermediate Value Theorem [Assessment]

Bibliography: [Stewart, 2015, Larson and Edwards, 2014]

5.2 Differential Calculus, 36 hr(s)

Competences: AG-C07

Topics: **I:** Derivative as a limit; geometric and physical interpretations; differentiability **II:** Product, quotient, and chain rules; implicit differentiation; related rates **III:** Rolle's theorem, Mean Value Theorem, and monotonicity criteria **IV:** Higher-order derivatives, Taylor polynomials, and remainder estimation **V:** Extrema: first and second derivative tests, curve sketching, and applied optimization

Learning Outcomes **I:** Interpret the derivative as a rate of change and as the slope of the tangent line [Familiarity] **II:** Apply differentiation rules and implicit differentiation to compute derivatives of composite and implicit functions [Usage] **III:** Solve optimization problems and sketch curves using the Mean Value Theorem and derivative tests [Assessment]

Bibliography: [Stewart, 2015, Larson and Edwards, 2014]

5.3 Parametric Equations and Polar Coordinates, 6 hr(s)

Competences: AG-C07

Topics: **I:** Parametric curves and their calculus, applied to trajectories in kinematics **II:** Polar coordinates and their conversion to Cartesian coordinates

Learning Outcomes **I:** Calculate the velocity and acceleration of a trajectory described parametrically [Usage] **II:** Convert between polar and Cartesian coordinates when modeling engineering geometries [Usage]

Bibliography: [Stewart, 2016, Thomas et al., 2014]

6. Bibliography

[Larson and Edwards, 2014] Larson, R. and Edwards, B. H. (2014). *Calculus.* Cengage Learning.

[Stewart, 2015] Stewart, J. (2015). *Calculus: Early Transcendentals.* Cengage Learning.

[Stewart, 2016] Stewart, J. (2016). *Cálculo de una variable: Trascendentes tempranas.* Cengage Learning, 8th edition.

[Thomas et al., 2014] Thomas, G. B., Weir, M. D., and Hass, J. (2014). *Cálculo de una variable*. Pearson, 13th edition.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

BCH101 Chemistry I (Mandatory)

1. General information

- **Semester**: 1st Sem. **Credits**: 5
- **Hour of this course**: **Theory**: 4 hours; **Laboratory**: 2 hours;
- **Prerrequisites**: None

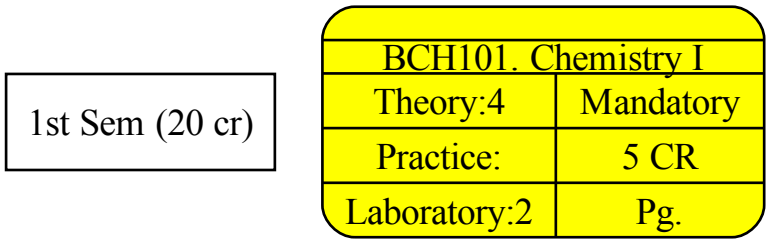

Figure 5.4: Connection Map. BCH101 Chemistry I

2. Course Rationale

Chemistry provides the foundation for understanding the composition, structure, and properties of matter – knowledge required across computing, and engineering programs alike as part of a rigorous grounding in the natural sciences. This course introduces the basic principles of general chemistry and extends them to chemical equilibrium and electrochemistry, together with the applied chemistry of water, polymers, and cement, topics that recur in materials science, environmental analysis, and engineering practice.

3. Goals of the Course

- Understand the structure of matter at the atomic and molecular level.
- Apply the principles of stoichiometry to perform chemical calculations.
- Understand the different types of chemical bonds and their influence on the properties of substances.
- Apply chemical equilibrium, including acid-base and solubility equilibria, to predict the behavior of aqueous systems.
- Apply electrochemical principles to analyze corrosion of metals and reinforced concrete.
- Relate the composition and properties of water, polymers, and cement to their applications in materials science and civil engineering.

4. General Competences (*Outcomes*)

AG-C07) Computing Knowledge: Applies knowledge of mathematics, science, and computing. (Familiarity)

5. Content

5.1 Matter and Energy, 8 hr(s)

Competences: AG-C07

Topics: **I:** Classification of matter: elements, compounds, homogeneous and heterogeneous mixtures **II:** Physical and chemical properties, and physical and chemical changes of matter **III:** International System of units, significant figures, and dimensional analysis **IV:** Forms of energy, heat and temperature; the law of conservation of mass and energy **V:** The scientific method applied to chemistry: observation, hypothesis, and experimentation

Learning Outcomes **I:** Classify a sample of matter as an element, compound, or mixture, and distinguish homogeneous from heterogeneous mixtures [Familiarity] **II:** Apply dimensional analysis and significant-figure rules in calculations involving chemical measurements [Usage] **III:** Relate observed energy changes to specific physical and chemical processes, applying the law of conservation of energy [Assessment]

Bibliography: [Brown et al., 2017, Chang and Goldsby, 2016]

5.2 Atomic Structure, 10 hr(s)

Competences: AG-C07

Topics: **I:** Historical evolution of the atomic model: Dalton, Thomson, Rutherford, and Bohr **II:** Subatomic particles, atomic number, mass number, and isotopes **III:** Electron configuration, quantum numbers, and the Aufbau principle **IV:** Organization of the periodic table: groups, periods, and blocks **V:** Periodic trends: atomic radius, ionization energy, electron affinity, and electronegativity

Learning Outcomes **I:** Describe the historical evolution of the atomic model and each model's contribution to the current understanding of the atom [Familiarity] **II:** Determine the electron configuration of an atom or ion from its position in the periodic table [Usage] **III:** Predict periodic trends (atomic radius, ionization energy) to compare two or more elements [Assessment]

Bibliography: [Brown et al., 2017, Chang and Goldsby, 2016]

5.3 Chemical Bonding, 10 hr(s)

Competences: AG-C07

Topics: **I:** Ionic bonding: formation, lattice energy, and properties of ionic compounds **II:** Covalent bonding, Lewis structures, and the concept of resonance **III:** Molecular geometry via Valence Shell Electron Pair Repulsion (VSEPR) theory **IV:** Electronegativity, bond polarity, and molecular polarity **V:** Intermolecular forces: dipole-dipole, hydrogen bonding, and London dispersion forces

Learning Outcomes **I:** Draw correct Lewis structures, including resonance cases, for molecules and polyatomic ions [Usage] **II:** Predict the molecular geometry and polarity of a molecule by applying VSEPR theory [Usage] **III:** Relate the type of intermolecular force present in a substance to its observed physical properties (boiling point, solubility) [Assessment]

Bibliography: [Brown et al., 2017, Chang and Goldsby, 2016]

5.4 Stoichiometry, 10 hr(s)

Competences: AG-C07

Topics: **I:** Chemical formulas, molar mass, and the mole concept **II:** Balancing chemical equations and their quantitative interpretation **III:** Mole-to-mole, mass-to-mass, and mass-to-mole stoichiometric calculations **IV:** Limiting reagent, excess reagent, and percent yield **V:** Molarity, solution preparation, and solution stoichiometry

Learning Outcomes **I:** Balance chemical equations by inspection, verifying conservation of mass [Familiarity] **II:** Calculate the amounts of reactants and products in a chemical reaction using mole ratios [Usage] **III:** Determine the limiting reagent and percent yield of a reaction from experimental data [Assessment]

Bibliography: [Brown et al., 2017, Chang and Goldsby, 2016]

5.5 States of Matter, 10 hr(s)

Competences: AG-C07

Topics: **I:** Gas laws: Boyle's, Charles's, Avogadro's, and the ideal gas equation **II:** Kinetic-molecular theory of gases and its applications **III:** Phase changes, heating curves, and phase diagrams **IV:** Crystalline and amorphous solids; types of crystalline solids **V:** Colligative properties of solutions: vapor pressure lowering, boiling point elevation, freezing point depression, and osmotic pressure

Learning Outcomes **I:** Apply the ideal gas equation and the gas laws to solve problems involving pressure, volume, and temperature [Usage] **II:** Interpret a phase diagram to identify the states and transitions of a substance under different conditions [Familiarity] **III:** Calculate the colligative properties of a solution (vapor pressure, boiling point, freezing point) from its concentration [Assessment]

Bibliography: [Brown et al., 2017, Chang and Goldsby, 2016]

5.6 Environmental and Water Chemistry, 4 hr(s)

Competences: AG-C07

Topics: **I:** Chemical composition of natural water and quality parameters (pH, hardness, dissolved oxygen, BOD) **II:** Fundamentals of water treatment: coagulation, flocculation, and disinfection **III:** Major air and water pollutants and their sources **IV:** Biogeochemical cycles of carbon, nitrogen, and phosphorus **V:** Atmospheric chemistry and the greenhouse effect

Learning Outcomes **I:** Interpret water quality parameters to assess fitness for use [Usage] **II:** Describe the basic processes of drinking water and wastewater treatment [Familiarity] **III:** Identify the main sources and effects of air and water pollutants [Familiarity] **IV:** Explain the chemical mechanism of the greenhouse effect [Usage]

Bibliography: [Brown et al., 2017, Chang and Goldsby, 2016]

5.7 Polymers and Geosynthetics, 4 hr(s)

Competences: AG-C07

Topics: **I:** Classification of polymers: thermoplastics, thermosets, and elastomers **II:** Polymerization reactions: addition and condensation **III:** Mechanical and chemical properties of polymers relevant to engineering **IV:** Geosynthetics: geotextiles, geomembranes, and geogrids, and their applications **V:** Degradation and durability of polymers under environmental conditions

Learning Outcomes **I:** Classify a polymer by its structure and thermal behavior [Usage] **II:** Identify whether a polymerization reaction is addition or condensation [Familiarity] **III:** Select the appropriate geosynthetic type for a given civil engineering application [Usage] **IV:** Assess the durability of a polymer under environmental exposure [Assessment]

Bibliography: [Chang and Goldsby, 2016, Mamlouk and Zaniewski, 2017]

5.8 Chemical Equilibrium, 4 hr(s)

Competences: AG-C07

Topics: **I:** The equilibrium constant K_c and K_p, and its relation to reaction stoichiometry **II:** The reaction quotient Q and predicting the direction of a reaction **III:** Le Chatelier's principle: effect of concentration, pressure, and temperature **IV:** ICE tables for calculating equilibrium concentrations

Learning Outcomes **I:** Write the equilibrium constant expression for a given reaction [Usage] **II:** Calculate equilibrium concentrations using ICE tables [Usage] **III:** Predict the shift of an equilibrium upon changes in concentration, pressure, or temperature using Le Chatelier's principle [Assessment]

Bibliography: [Brown et al., 2017, Chang and Goldsby, 2016]

5.9 Ionic Equilibrium and Salt Hydrolysis, 12 hr(s)

Competences: AG-C07

Topics: **I:** Arrhenius, Brønsted-Lowry, and Lewis acid-base theories **II:** Calculating pH and pOH for strong acids and bases **III:** Equilibrium of weak acids and bases: K_a, K_b, and degree of dissociation **IV:** Salt hydrolysis and predicting the acidic, basic, or neutral character of a solution **V:** Buffer solutions and the Henderson-Hasselbalch equation

Learning Outcomes **I:** Calculate the pH of solutions of strong and weak acids and bases [Usage] **II:** Predict the acid-base character of a salt solution from the hydrolysis of its ions [Assessment] **III:** Calculate the pH of a buffer solution using the Henderson-Hasselbalch equation [Usage]

Bibliography: [Brown et al., 2017, Chang and Goldsby, 2016]

5.10 Salts, Solubility, and Precipitation Equilibria, 6 hr(s)

Competences: AG-C07

Topics: **I:** The solubility product K_{sp} and its relation to molar solubility **II:** Predicting precipitation by comparing the ion product with K_{sp} **III:** Common ion effect on solubility

Learning Outcomes **I:** Calculate K_{sp} from the molar solubility of a salt, and vice versa [Usage] **II:** Predict whether a precipitate will form when mixing two solutions given their concentrations and the corresponding K_{sp} [Assessment]

Bibliography: [Brown et al., 2017, Chang and Goldsby, 2016]

5.11 Electrochemistry and Electrolysis, 8 hr(s)

Competences: AG-C07

Topics: **I:** Oxidation-reduction reactions and balancing by the ion-electron method **II:** Galvanic cells, standard electrode potentials, and the Nernst equation **III:** Electrolytic cells and electrolysis; Faraday's laws **IV:** Electrochemical corrosion of metals and protection methods **V:** Corrosion of reinforcing steel in reinforced concrete

Learning Outcomes **I:** Balance redox equations by the ion-electron method in acidic or basic medium [Usage] **II:** Calculate the potential of a galvanic cell and predict the spontaneity of a redox reaction [Usage] **III:** Apply Faraday's laws to calculate the mass deposited or the time in an electrolysis process [Usage] **IV:** Analyze the conditions that favor the corrosion of a metal or of reinforcing steel in concrete, and propose protection methods [Assessment]

Bibliography: [Brown et al., 2017, Chang and Goldsby, 2016]

5.12 Cement Chemistry, 4 hr(s)

Competences: AG-C07

Topics: **I:** Chemical composition of Portland cement: calcium silicates, aluminates, and ferroaluminates **II:** Cement hydration reactions and strength development **III:** Role of each compound (C_3S, C_2S, C_3A, C_4AF) in cement properties **IV:** Supplementary cementitious materials: pozzolans, slag, and fly ash

Learning Outcomes **I:** Describe the chemical composition of Portland cement and its main compounds [Familiarity] **II:** Explain the cement hydration process and its relation to strength development [Usage] **III:** Evaluate the effect of supplementary cementitious materials on concrete properties [Assessment]

Bibliography: [Mamlouk and Zaniewski, 2017, Chang and Goldsby, 2016]

6. Bibliography

[Brown et al., 2017] Brown, T. L., Jr., H. E. L., Bursten, B. E., Murphy, C. J., and Woodward, P. M. (2017). *Chemistry: The Central Science*. Pearson.

[Chang and Goldsby, 2016] Chang, R. and Goldsby, K. A. (2016). *Chemistry*. McGraw-Hill Education.

[Mamlouk and Zaniewski, 2017] Mamlouk, M. S. and Zaniewski, J. P. (2017). *Materials for Civil and Construction Engineers*. Pearson, 4th edition.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

BEI101 Technical and professional English I (Mandatory)

1. General information

- **Semester**: 1st Sem. **Credits**: 2
- **Hour of this course**: **Practice**: 4 hours;
- **Prerrequisites**: None

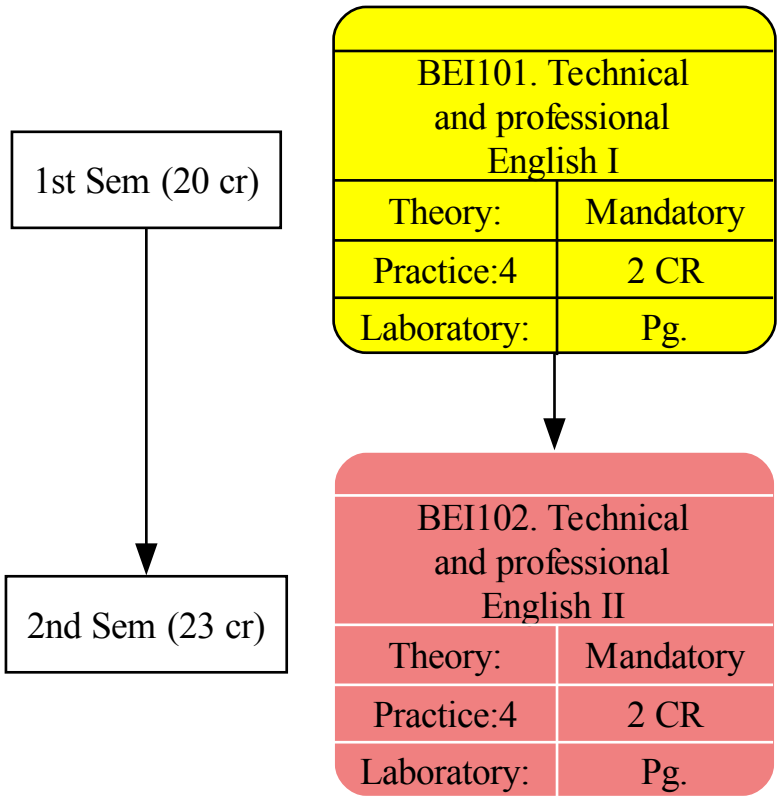

Figure 5.5: Connection Map. BEI101 Technical and professional English I

2. Course Rationale

English is the primary language of science, engineering, and technology worldwide. For computing and engineering students, mastering English means gaining direct access to technical documentation, research papers, and professional collaboration without translation barriers. This course builds foundational English skills using contexts drawn from computing systems, hardware, and the everyday work of engineers and scientists.

3. Goals of the Course

- Acquire foundational English grammar structures applied to computing and engineering contexts.
- Identify and use technical vocabulary related to hardware, software, and scientific concepts.

4. General Competences (*Outcomes*)

AG-C03) Individual and Team Work: Performs effectively as an individual and as a member or leader in diverse teams. (Familiarity)

AG-C04) Communication: Communicates effectively in complex computing activities. (Usage)

5. Content

5.1 Welcome to Computing, 12 hr(s)

Competences: AG-C03,AG-C04

Topics: I: Verb To Be: Affirmative, Negative, and Questions. **II:** Personal Pronouns and Possessive Adjectives. **III:** Number Expressions (1-100): Ports, addresses, and binary basics. **IV:** Social Expressions in academic and lab environments.

Learning Outcomes I: By the end of the unit, students use the verb 'to be' to introduce themselves as computing students and describe basic properties of computing systems (e.g., "The processor is fast. It is a 64-bit CPU.").

Bibliography: [Soars and John, 2002d, Cambridge, 2006, MacGrew, 1999]

5.2 Parts and Systems, 12 hr(s)

Competences: AG-C03,AG-C04

Topics: I: Possessive 's and Plural Nouns applied to components (the CPU's speed, the system's memory). **II:** Hardware components and their relationships. **III:** Adjective + Noun structures for technical descriptions. **IV:** Lab interactions: Asking about equipment and tools.

Learning Outcomes I: Students describe computer hardware components and their relationships using possessives and noun phrases (e.g., "the motherboard's slots", "the server's processors").

Bibliography: [Soars and John, 2002d, Cambridge, 2006, MacGrew, 1999]

5.3 How It Works, 12 hr(s)

Competences: AG-C03,AG-C04

Topics: I: Present Simple I (Third person: He/She/It) to describe processes. **II:** Vocabulary: How programs, algorithms, and devices operate. **III:** Telling the time and interpreting system clocks and schedules. **IV:** Verbs of operation: runs, compiles, executes, stores, transmits.

Learning Outcomes I: Students explain how a program or device works using the present simple in the third person (e.g., "The compiler translates source code. It generates an executable file.").

Bibliography: [Soars and John, 2002d, Cambridge, 2006, MacGrew, 1999]

5.4 What Engineers Do, 12 hr(s)

Competences: AG-C03,AG-C04

Topics: I: Present Simple II (I/You/We/They) for describing professional routines. **II:** Questions and negatives with 'Do/Does' in technical contexts. **III:** Engineering roles: developer, analyst, researcher, systems engineer. **IV:** Adverbs of frequency in describing work cycles and testing schedules.

Learning Outcomes I: Students describe the daily tasks and professional roles of computing engineers and scientists using the present simple with accuracy.

Bibliography: [Soars and John, 2002d, Cambridge, 2006, MacGrew, 1999]

5.5 Inside a Computer, 12 hr(s)

Competences: AG-C03,AG-C04

Topics: I: Existential: There is / There are applied to architecture descriptions. **II:** Prepositions of place: components on a motherboard, files in a directory. **III:** Vocabulary: memory, storage, buses, slots, ports, layers. **IV:** Describing network topology and physical infrastructure.

Learning Outcomes I: Students describe the spatial organization of hardware components and file systems, enabling them to interpret and produce technical architecture descriptions.

Bibliography: [Soars and John, 2002d, Cambridge, 2006, MacGrew, 1999]

5.6 Can Computers Think?, 12 hr(s)

Competences: AG-C03,AG-C04

Topics: I: Modal verb Can/Can't to describe computing capabilities and limitations. **II:** Adverbs of manner applied to system performance. **III:** Past of verb 'To Be' (Was/Were): early computing systems and pioneers. **IV:** Basic expressions for technical support and troubleshooting.

Learning Outcomes I: Students describe what computing systems can and cannot do, and identify basic facts about the early history of computing using simple past structures.

Bibliography: [Soars and John, 2002d, Cambridge, 2006, MacGrew, 1999]

5.7 Computing History, 12 hr(s)

Competences: AG-C03,AG-C04

Topics: I: Past Simple: Regular and Irregular verbs to narrate tech history. **II:** Time expressions for the past: invented, developed, published, launched. **III:** Key figures: Turing, Babbage, Lovelace, von Neumann. **IV:** Vocabulary: Generations of computers, milestones, inventions.

Learning Outcomes I: Students narrate key events in the history of computing and science using the simple past correctly, identifying the contributions of foundational figures.

Bibliography: [Soars and John, 2002d, Cambridge, 2006, MacGrew, 1999]

5.8 My Professional Goals in Computing, 12 hr(s)

Competences: AG-C03,AG-C04

Topics: I: Introduction to Technical English for Computing and Engineering. **II:** Reading and understanding simple technical specifications. **III:** Working in multicultural engineering teams: communication norms. **IV:** Oral presentation: "My role in the computing field".

Learning Outcomes I: Students present their professional goals in computing using basic technical vocabulary and connect them to the roles and responsibilities of software and hardware engineers.

Bibliography: [Soars and John, 2002d, Cambridge, 2006, MacGrew, 1999]

6. Bibliography

[Cambridge, 2006] Cambridge (2006). *Diccionario Inglés-Espanol Cambridge*. Editorial Oxford.

[MacGrew, 1999] MacGrew, J. (1999). *Focus on Grammar Basic*. Editorial Oxford.

[Soars and John, 2002d] Soars, L. and John (2002d). *American Headway N 2 Student Book*. Editorial Oxford.

Second Semester

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

CS100 Introduction to Computer Science (Mandatory)

1. General information

- **Semester**: 2nd Sem. **Credits**: 3
- **Hour of this course**: **Theory**: 2 hours; **Practice**: 2 hours;
- **Prerrequisites**: None

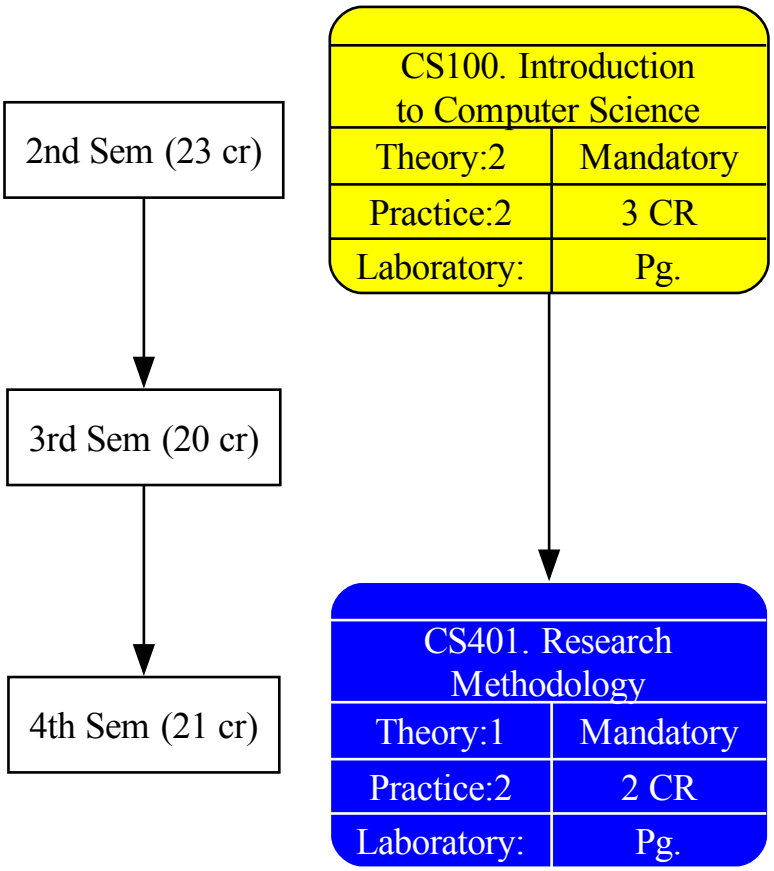

Figure 5.6: Connection Map. CS100 Introduction to Computer Science

2. Course Rationale

This course serves as the foundation for understanding the fundamental concepts of computational thinking applicable across various professions.

The course provides, starting from ground zero, a panoramic view of: introductory computational thinking, data storage, computer architecture, operating systems, networks and the Internet, algorithms, sorting methods, software engineering, databases, data structures, software engineering, computer graphics, artificial intelligence among others.

Designed as an introductory course to Computer Science, the concepts are presented in a playful manner and using an Active Learning methodology. Throughout the course, active audience participation is encouraged, akin to a theatrical performance.

The related knowledge areas covered are directly aligned with the Computing Curricula ACM/IEEE-CS.

The course **does not require** any prior knowledge in computer handling topics and can be taken by student from any field.

3. Goals of the Course

- Introduce the fundamental concepts of Computational Thinking and Computer Science to students from any professional background.
- Develop their ability to abstract.
- Understand how Computational Thinking is applied in each of their professions.
- Apply advanced concepts in a simplified manner in any career.

4. General Competences (*Outcomes*)

AG-C02) Ethics: Applies ethical principles and commits to professional ethics and the standards of professional computing practice. (Familiarity)

AG-C08) Problem Analysis: Identifies, formulates, and analyzes complex computing problems. (Familiarity)

AG-C12) Applies computer science theory and software development fundamentals to produce computer-based solutions. (Familiarity)

5. Content

5.1 Computational Thinking (Part I), 4 hr(s)

Competences: AG-C02,AG-C08,AG-C12

Topics: **I:** Explanation of the evaluation system. **II:** General course instructions. **III:** Definition of Computing. **IV:** Computing as a Human-Computer binomial. **V:** Distortions in the definition of computing. **VI:** Computing as the automation of abstraction.

VII: Computing and Engineering: similarities and differences. **VIII:** Algorithmic problem-solving. **IX:** Dynamics: Understanding the execution of an algorithm at human speed.

Learning Outcomes **I:** Apply the fundamental concepts of computing in real-life situations. [Usage]
II: Identify distortions of Computing in real-life situations. [Usage]

III: Clearly identify at least 3 contexts of using the word“Engineer” in English. [Assessment]
IV: Identify the limitations of humans in solving computational problems. [Usage]

Bibliography: [Brookshear and Brylow, 2019]

5.2 Computational Thinking. Part II, 4 hr(s)

Competences: AG-C02,AG-C08,AG-C12

Topics: **I:** Decomposition (Breaking a problem in smaller pieces) **II:** Abstraction (Focus on important things) **III:** Pattern recongnition (Identify similar sequences based on previous problems) **IV:** Designing algorithms

Learning Outcomes **I:** Apply Computational Thinking to real-world problems. [Usage]

Bibliography: [Brookshear and Brylow, 2019]

5.3 General concepts, 4 hr(s)

Competences: AG-C02,AG-C08,AG-C12

Topics: **I:** Binary vs. decimal numbering. **II:** Character representation: the ASCII table. **III:** Binary search. **IV:** Computational complexity of an algorithm.

Learning Outcomes **I:** Apply various numbering systems to real-world problems. [Usage] **II:** Understand the internal representation of characters in the ASCII and UTF-8 tables. [Familiarity] **III:** Apply Divide and Conquer algorithmic strategy. [Usage] **IV:** Determine basic analysis of algorithmic complexity. [Usage]

Bibliography: [Brookshear and Brylow, 2019]

5.4 Machine-Level Data Representation, 4 hr(s)

Competences: AG-C02,AG-C08,AG-C12

Topics: **I:** Overview and history of computer architecture **II:** Bits, bytes, and words **III:** Unsigned, signed and two's complement representations **IV:** Numeric data representation and number bases: (a) Fixed-point (b) Floating-point **V:** Representation of non-numeric data **VI:** Representation of records, arrays and UTF data types

Learning Outcomes **I:** Discuss why everything in computers are data, including instructions [Debate] **II:** Explain how fixed-length number representations can affect accuracy and precision [Explain] **III:** Describe how negative integers are stored in sign-magnitude and two's-complement representations [Describe] **IV:** Discuss how different formats can represent numerical data [Debate] **V:** Explain the bit-level representation of non-numeric data, such as characters, strings, records, and arrays [Explain] **VI:** Translate numerical data from one format to another [Translate] **VII:** Describe how a single adder (without overflow detection) can handle both signed (two's complement) and unsigned (binary) input without "knowing" which format a given input is using [Describe]

Bibliography: [Brookshear and Brylow, 2019]

5.5 Program Abstraction and Representation, 2 hr(s)

Competences: AG-C02,AG-C08,AG-C12

Topics: **I:** Programs that take (other) programs as input such as interpreters, compilers, type-checkers, documentation generators **II:** Components of a language: (a) Definitions of alphabets, delimiters, sentences, syntax, and semantics (b) Syntax vs semantics **III:** Program as a set of non-ambiguous meaningful sentences **IV:** Basic programming abstractions: constants, variables, declarations (including nested declarations), command, expression, assignment, selection, definite and indefinite iteration, iterators, function, procedure, modules, exception handling **V:** Types of variables: static, local, nonlocal, global; need and issues with nonlocal and global variables. **VI:** Scope rules: static vs dynamic; visibility of variables; side-effects. **VII:** Environment vs store and their properties **VIII:** Data and control abstraction **IX:** Mechanisms for information exchange between program units such as procedures, functions, and modules: nonlocal variables, global variables, parameter-passing, import-export between modules **X:** Low level instruction representation such as virtual machine instructions, assembly language, and binary representation

Learning Outcomes **I:** Illustrate the scope of variables and visibility using simple programs [Apply] **II:** Illustrate different types of parameter-passing using simple pseudo programming language [Apply] **III:** Explain side-effect using global and nonlocal variables and how to fix such programs [Explain] **IV:** Explain how programs that process other programs treat the other programs as their input data [Explain] **V:** Describe a grammar and an abstract syntax tree for a small language [Describe]

Bibliography: [Brookshear and Brylow, 2019]

5.6 Cryptography Fundamentals, 2 hr(s)

Competences: AG-C02,AG-C08,AG-C12

Topics: **I:** Basic concepts on Cryptography **II:** Public-keys, Private-keys

Learning Outcomes **I:** Explain the role of cryptography in supporting security and privacy [Explain] **II:** Discuss the risks of inventing one's own cryptographic methods [Debate] **III:** Discuss the importance of prime numbers in cryptography and explain their use in cryptographic algorithms [Debate] **IV:** Implement and cryptanalyze classical ciphers [Implement]

Bibliography: [Brookshear and Brylow, 2019]

5.7 Memory Management, 4 hr(s)

Competences: AG-C02,AG-C08,AG-C12

Topics: **I:** Review of physical memory, address translation and memory management hardware See also: Memory Hierarchy, Discrete Mathematics **II:** Impact of memory hierarchy including cache concept, cache lookup, and per-CPU caching on operating system mechanisms and policy See also: Memory Hierarchy, Latency, Cache and Memory Hierarchy, Virtualization and Isolation **III:** Logical and physical addressing, address space virtualization. See also: Memory Hierarchy, Discrete Mathematics **IV:** Concepts of paging, page replacement, thrashing and allocation of pages and frames **V:** Allocation/deallocation/storage techniques (algorithms and data structure) performance and flexibility (a) Arenas, slab allocators, free lists, size classes, heterogeneously sized pages (huge pages) **VI:** Memory caching and cache coherence and the effect of flushing the cache to avoid speculative execution vulnerabilities. See also: Functional Organization, Memory Hierarchy, Latency, Cache and Memory Hierarchy, Virtualization and Isolation **VII:** Virtual memory: leveraging virtual memory hardware for OS services and efficiency

Learning Outcomes **I:** Explain memory hierarchy and cost-performance tradeoffs [Explain] **II:** Summarize the principles of virtual memory as applied to caching and paging [Summarize] **III:** Evaluate the tradeoffs in terms of memory size (main memory, cache memory, auxiliary memory) and processor speed [Evaluate] **IV:** Describe the reason for and use of cache memory (performance and proximity, how caches complicate isolation and virtual machine abstraction) [Describe]

Bibliography: [Brookshear and Brylow, 2019]

5.8 Role and Purpose of Operating Systems, 2 hr(s)

Competences: AG-C02,AG-C08,AG-C12

Topics: **I:** Operating systems mediate between general purpose hardware and application-specific software. **II:** Universal operating system functions (e.g., process, user and device interfaces, persistence of data) **III:** Influences of security, networking, multimedia, parallel and distributed computing

Learning Outcomes **I:** Understand the objectives and functions of modern operating systems [Explain] **II:** Evaluate the design issues in different usage scenarios (e.g., real time OS, mobile, server) [Evaluate] **III:** Understand how evolution and stability are desirable and mutually antagonistic in operating systems function [Explain]

Bibliography: [Brookshear and Brylow, 2019]

5.9 Principles of Operating System, 2 hr(s)

Competences: AG-C02,AG-C08,AG-C12

Topics: **I:** Operating system software design and approaches (e.g., monolithic, layered, modular, microkernel, unikernel) **II:** Abstractions, processes, and resources **III:** The evolution of the link between hardware architecture and the operating system functions **IV:** Leveraging interrupts from hardware level: service routines and implementations. See also: Assembly Level Machine Organization (a) Timer interrupts for implementing time slices (b) I/O interrupts for putting blocking threads to sleep without polling

Learning Outcomes **I:** Understand how the application of software design approaches to operating systems design/implementation (e.g., layered, modular, etc.) affects the robustness and maintainability of an operating system [Explain] **II:** Categorize system calls by purpose [Categorize] **III:** Apply OS techniques for isolation, protection, and throughput across OS functions (e.g., starvation similarities in process scheduling, disk request scheduling, semaphores, etc.) and beyond [Apply]

Bibliography: [Brookshear and Brylow, 2019]

5.10 Fundamentals of Networks and Communications, 4 hr(s)

Competences: AG-C02,AG-C08,AG-C12

Topics: **I:** Importance of networking in contemporary computing, and associated challenges. See also: Digital Equity and Accessibility, Social Impact and AI, Civil Liberties and Expression, Data, Surveillance and Privacy Technology, Privacy Philosophical Foundations and Legislation **II:** Organization of the internet (e.g., users, Internet Service Providers, autonomous systems, content providers, content delivery networks) **III:** Layers and their roles (application, transport, network, datalink, and physical) **IV:** Network elements (e.g., routers, switches, hubs, access points, and hosts) **V:** Basic queueing concepts (e.g., relationship with latency, congestion, service levels, etc.)

Learning Outcomes **I:** Articulate the organization of the internet [Articulate] **II:** List and define the appropriate network terminology [List] **III:** Describe the layered structure of a typical networked architecture [Describe]

Bibliography: [Brookshear and Brylow, 2019]

5.11 Networked Applications, 4 hr(s)

Competences: AG-C02,AG-C08,AG-C12

Topics: **I:** Naming and address schemes (e.g., DNS, and Uniform Resource Identifiers) **II:** Distributed application paradigms (e.g., client/server, peer-to-peer, cloud, edge, and fog). See also: Communication, Coordination **III:** Diversity of networked application demands (e.g., latency, bandwidth, and loss tolerance). See also: Communication, Sustainable Design and Pervasive Computing, Environmental Footprint of Computing Systems, Systemic Effects and Social Context, Digital Equity and Accessibility, Social Impact and AI **IV:** Coverage of application-layer protocols (e.g., HTTP) **V:** Interactions with TCP, UDP, and Socket APIs. See also: Programs: Distribution, Programs: Execution Properties, GPU Programming, Programs: Implementation Mappings, Programs: Declarative Parallelism, Programs: Starting Activities

Learning Outcomes **I:** Define the principles of naming, addressing, resource location [Define] **II:** Analyze the needs of specific networked application demands [Analyze]

Bibliography: [Brookshear and Brylow, 2019]

5.12 Complexity Analysis Framework, 2 hr(s)

Competences: AG-C02,AG-C08,AG-C12

Topics: **I:** Complexity Analysis Framework: (a) Best, average, and worst-case performance of an algorithm (b) Empirical and relative (Order of Growth) measurements (c) Input size and primitive operations (d) Time and space efficiency **II:** Empirical measurements of performance **III:** Time and space tradeoffs in algorithms

Learning Outcomes **I:** Prepare a presentation that explains to first year students the basic concepts of algorithmic complexity including best, average, and worst-case algorithm behavior, Big-O, Omega, and Theta notations, complexity classes, time and space tradeoffs, empirical measurement, and impact on practical problems [Explain] **II:** For each algorithm in the Foundational Data Structures, Foundational Algorithms, Advanced Algorithms unit, explain its runtime complexity class and why it belongs to this class [Explain] **III:** Informally evaluate the foundational complexity class of simple algorithms [Evaluate] **IV:** Develop empirical studies to determine and validate hypotheses about the runtime complexity of various algorithms by running algorithms on input of various sizes and comparing actual performance to the theoretical analysis [Create] **V:** Explain examples that illustrate time-space tradeoffs of algorithms [Explain] **VI:** Explain how tree balance affects the efficiency of binary search tree operations [Explain]

Bibliography: [Brookshear and Brylow, 2019]

5.13 Asymptotic Notation and Complexity Classes, 1 hr(s)

Competences: AG-C02,AG-C08,AG-C12

Topics: **I:** Asymptotic complexity analysis (average and worst-case bounds): (a) Big-O, Big-Omega, and Big-Theta formal notations (b) Foundational Complexity Classes and Representative Examples/Problems: i. $O(1)$ Constant (e.g., array access) ii. $O(\log_2 n)$ Logarithmic (e.g., binary search) iii. $O(n)$ Linear (e.g., linear search) iv. $O(n \log_2 n)$ Log Linear (e.g., mergesort) v. $O(n^2)$ Quadratic (e.g., selection sort) vi. $O(n^c)$ Polynomial (e.g., $O(n^3)$ Gaussian elimination) vii. $O(2^n)$ Exponential (e.g., Knapsack, Satisfiability (SAT), Traveling Sales-Person (TSP), all subsets) viii. $O(n!)$ Factorial (e.g., Hamiltonian circuit, all permutations)

Learning Outcomes **I:** Using examples, explain each of the foundational complexity classes in this unit [Explain] **II:** For each foundational complexity class in this unit, explain an algorithm that demonstrates the associated runtime complexity [Explain] **III:** Explain to a non-technical audience the significance of tractable versus intractable algorithms using an intuitive explanation of Big-O complexity [Explain]

Bibliography: [Brookshear and Brylow, 2019]

5.14 Complexity Analysis: Recursion, Amortization, and Tight Bounds, 1 hr(s)

Competences: AG-C02,AG-C08,AG-C12

Topics: **I:** Little-o, Little-Omega, and Little Theta notations

Learning Outcomes **I:** Given a problem to program for which there may be several algorithmic approaches, evaluate them and determine which are feasible, and select one that is optimal in implementation and run-time behavior [Evaluate]

Bibliography: [Brookshear and Brylow, 2019]

5.15 Foundational Data Structures, 3 hr(s)

Competences: AG-C02,AG-C08,AG-C12

Topics: **I:** Abstract Data Type (ADT) and operations on an ADT: (a) Dictionary operations (insert, delete, find) **II:** Arrays: (a) Numeric vs non-numeric, character strings (b) Single (vector) vs multidimensional (matrix) **III:** Records/Structs/Tuples and Objects **IV:** Linked lists (for historical reasons): (a) Single vs Double and Linear vs Circular **V:** Stacks **VI:** Queues and deques: (a) Heap-based priority queue **VII:** Hash tables/maps: (a) Collision resolution and

complexity (e.g., probing, chaining, rehash) **VIII:** Trees: (a) Binary, n-ary, and search trees (b) Balanced (e.g., AVL, Red-Black, Heap)

Learning Outcomes I: For each ADT/Data-Structure in this unit: (a) Explain its definition, properties, representation(s), and associated ADT operations. (b) Explain step-by-step how the ADT operations associated with the data structure transform it. [Explain] **II:** Explain how collision avoidance and collision resolution is handled in hash tables [Explain] **III:** Explain the heap property and the use of heaps as an implementation of a priority queue [Explain]

Bibliography: [Brookshear and Brylow, 2019]

5.16 Foundational Algorithms, 3 hr(s)

Competences: AG-C02,AG-C08,AG-C12

Topics: I: Graphs (e.g., [un]directed, [a]cyclic, [un]connected, and [un]weighted): (a) Graph representation: adjacency list vs matrix **II:** Search algorithms: (a) $O(n)$ complexity (e.g., linear/sequential array/list search) (b) $O(\log_2 n)$ complexity (e.g., binary search) (c) $O(\log_b n)$ complexity (e.g., uninformed depth/breadth-first tree search) **III:** Sorting algorithms (e.g., stable, unstable): (a) $O(n^2)$ complexity (e.g., insertion, selection) (b) $O(n \log n)$ complexity (e.g., quicksort, merge, timsort) **IV:** Graph algorithms: (a) Shortest path (e.g., Dijkstra's, Floyd's) (b) Minimal spanning tree (e.g., Prim's, Kruskal's) **V:** Sorting algorithms: (a) $O(n \log n)$ complexity heapsort (b) Pseudo $O(n)$ complexity (e.g., bucket, counting, radix) **VI:** Graph algorithms: (a) Transitive closure (e.g., Warshall's) (b) Topological sort

Learning Outcomes I: For each algorithm in this unit explain step-by-step how the algorithm operates [Explain] **II:** For each algorithmic approach (e.g., sorting) in this unit apply a prototypical example of the approach (e.g., merge sort) [Apply] **III:** Given requirements for a problem, develop multiple solutions using various data structures and algorithms. Subsequently, evaluate the suitability, strengths, and weaknesses selecting an approach that best satisfies the requirements [Create] **IV:** Explain factors beyond computational efficiency that influence the choice of algorithms, such as programming time, maintainability, and the use of application-specific patterns in the input data [Explain]

Bibliography: [Brookshear and Brylow, 2019]

5.17 Advanced Algorithms, 2 hr(s)

Competences: AG-C02,AG-C08,AG-C12

Topics: I: Cryptography algorithms (e.g., SHA-256) **II:** Parallel algorithms

Learning Outcomes I: An appreciation of quantum computation and its application to certain problems [Explain]

Bibliography: [Brookshear and Brylow, 2019]

5.18 Core Database System Concepts, 4 hr(s)

Competences: AG-C02,AG-C08,AG-C12

Topics: I: Purpose and advantages of database systems **II:** Components of database systems **III:** Design of core DBMS functions (e.g., query mechanisms, transaction management, buffer management, access methods) **IV:** Transaction management **V:** Normalization **VI:** Approaches for managing large volumes of data (e.g., NoSQL database systems, use of MapReduce) **VII:** Distributed databases/cloud-based systems **VIII:** Use of a declarative query language

Learning Outcomes I: Identify at least four advantages that using a database system provides [Analyze] **II:** Enumerate the components of a (relational) database system [Enumerate] **III:** Defend the value of data independence [Defend] **IV:** Compose a simple select-project-join query in SQL [Compose] **V:** Describe the advantages for eliminating duplicate repeated data [Describe] **VI:** Outline how MapReduce uses parallelism to process data efficiently [Outline]

Bibliography: [Brookshear and Brylow, 2019]

5.19 Object-Oriented Programming Fundamentals, 4 hr(s)

Competences: AG-C02,AG-C08,AG-C12

Topics: I: Imperative programming as a subset of object-oriented programming. **II:** Object-oriented design: (a) Decomposition into objects carrying state and having behavior. (b) Class-hierarchy design for modeling. **III:** Definition of classes: fields, methods, and constructors. **IV:** Subclasses, inheritance (including multiple inheritance), and method overriding. **V:** Dynamic dispatch: definition of method-call. **VI:** Object-oriented idioms for encapsulation: (a) Privacy, data hiding, and visibility of class members. (b) Interfaces revealing only method signatures. (c) Abstract base classes, traits and mixins.

Learning Outcomes I: Enumerate the differences between imperative and object-oriented programming paradigms [Enumerate] **II:** Compose a class through design, implementation, and testing to meet behavioral requirements [Compose] **III:** Use object-oriented encapsulation mechanisms such as interfaces and private members [Use]

Bibliography: [Brookshear and Brylow, 2019]

5.20 Version Control and CI/CD, 2 hr(s)

Competences: AG-C02,AG-C08,AG-C12

Topics: I: Software configuration management and version control: Software Development Practices (a) Configuration in version control, reproducible builds/configuration. (b) Version control branching strategies. Development branches vs release branches. Trunk-based development. (c) Merging/rebasing strategies, when relevant. **II:** Release management. **III:** Software process automation: (a) Build systems - the value of fast, hermetic, reproducible builds, compare/contrast approaches to building a project. (b) Continuous Integration (CI) - the use of automation and automated tests to do preliminary validation that the current head/trunk revision builds and passes (basic) tests. (c) Continuous Deployment (CD) - the use of automation to automatically release every change that passes the automated tests to the production environment, ensuring frequent and reliable deliveries. (d) Dependency management - updating external/upstream dependencies, package management, SemVer. **IV:** Tool integration concepts and mechanisms. Software Development Practices **V:** Use of modern IDE facilities - debugging, refactoring, searching/indexing, ML-powered code assistants, etc. Software Development Practices

Learning Outcomes I: Describe the difference between centralized and distributed software configuration management [Describe] **II:** Describe how version control can be used to help manage software release management [Describe] **III:** Identify configuration items and use a source code control tool in a small team-based project [Analyze] **IV:** Understand the use of CI/CD systems as a ground-truth for the state of the team's shared code (build and test success) [Explain] **V:** Demonstrate the capability to use software tools in support of the development of a software product of medium size [Demonstrate]

Bibliography: [Brookshear and Brylow, 2019]

5.21 Requirements Engineering, 2 hr(s)

Competences: AG-C02,AG-C08,AG-C12

Topics: **I:** Describe functional requirements using, for example, use cases or user stories. (a) Using at least one method of documenting and structuring functional requirements. (b) Understanding how the method supports design and implementation. (c) Strengths and weaknesses of using a specific approach. **II:** Requirements elicitation. (a) Sources of requirements, for example, users, administrators, or support personnel. (b) Methods of requirement gathering, for example, surveys, interviews, or behavioral analysis. **III:** Non-functional requirements, for example, security, usability, or performance, also called as Quality Attributes. Sustainable Design and Pervasive Computing, Environmental Footprint of Computing Systems, Systemic Effects and Social Context **IV:** Risk identification and management, including ethical considerations surrounding the proposed product. Ethics Codes and Legal Accountability, Professional Responsibility and Public Policy, Professional Values, Nature and Continuing Development, Ethical Dissent and Cultural Context, Workplace Ethics **V:** Communicating and/or formalizing requirement specifications. **VI:** Prototyping a tool for both eliciting and validating/confirming requirements. **VII:** Product evolution: when requirements change, how to understand what effect that has and what changes need to be made. **VIII:** Effort estimation: (a) Learning techniques for better estimating the effort required to complete a task; (b) Practicing estimation and comparing it to how long tasks take; (c) Effort estimation is quite difficult, so students are likely to be way off in many cases, but seeing the process play out with their own work is valuable.

Learning Outcomes **I:** Compare different methods of eliciting requirements along multiple axes [Compare] **II:** Identify differences between two methods of describing functional requirements (e.g., customer interviews, user studies) and the situations where each would be preferred [Analyze] **III:** Identify which behaviors are required, allowed, or barred from a given set of requirements and a list of candidate behaviors [Analyze] **IV:** Collect a set of requirements for a simple software system [Analyze] **V:** Identify areas of a software system that need to be changed, given a description of the system and a set of new requirements to be implemented [Analyze] **VI:** Identify the functional and non-functional requirements in a set of requirements [Analyze] **VII:** Estimate the time to complete a set of tasks, then compare estimates to the actual time taken [Estimate] **VIII:** Determine an implementation sequence for a set of tasks, adhering to dependencies between them, with a goal to retire risk as early as possible [Determine] **IX:** Write a requirement specification for a simple software system [Write]

Bibliography: [Brookshear and Brylow, 2019]

5.22 Machine Learning Fundamentals, 1 hr(s)

Competences: AG-C02,AG-C08,AG-C12

Topics: **I:** Definition and examples of a broad variety of machine learning tasks: (a) Supervised learning: i. Classification ii. Regression (b) Reinforcement learning (c) Unsupervised learning: i. Clustering **II:** Fundamental ideas: (a) No free lunch theorem: no one learner can solve all problems; representational design decisions have consequences. (b) Sources of error and undecidability in machine learning **III:** A simple statistical-based supervised learning such as linear regression or decision trees: (a) Focus on how they work without going into mathematical or optimization details; enough to understand and use existing implementations correctly

Learning Outcomes **I:** Describe the differences among the three main styles of learning (supervised, reinforcement, and unsupervised) and determine which is appropriate to a particular problem domain [Explain] **II:** Differentiate the terms of AI, machine learning, and deep learning [Evaluate] **III:** Explain how machine learning works as an optimization/search process [Explain]

Bibliography: [Brookshear and Brylow, 2019]

5.23 Neural Networks and Ethics, 1 hr(s)

Competences: AG-C02,AG-C08,AG-C12

Topics: **I:** Basic neural networks: (a) Fundamentals of understanding how neural networks work and their training process, without details of the calculations (b) Basic introduction to generative neural networks (e.g., large language models) **II:** Ethics for Machine Learning: (a) Focus on real data, real scenarios, and case studies (b) Dataset/algorithmic/evaluation bias and unintended consequences **III:** Deep learning: (a) Deep feed-forward networks (intuition only, no mathematics) (b) Convolutional neural networks (intuition only, no mathematics) (c) Visualization of learned feature representations from deep nets (d) Other architectures (generative NN, recurrent NN, transformers, etc.) **IV:** Ethics for Machine Learning: (a) Continued focus on real data, real scenarios, and case studies (b) Privacy (c) Fairness (d) Intellectual property (e) Explainability

Learning Outcomes **I:** Describe the neural network training process and resulting learned representations [Explain] **II:** Visualize the training progress of a neural network through learning curves in a well-established toolkit (e.g., TensorBoard) and visualize the learned features of the network [Apply] **III:** Given a real-world application of machine learning, describe ethical issues regarding the choices of data, preprocessing steps, algorithm selection, and visualization/presentation of results [Apply]

Bibliography: [Brookshear and Brylow, 2019]

5.24 Fundamental Concepts of Graphics and Interactive Techniques, 2 hr(s)

Competences: AG-C02,AG-C08,AG-C12

Topics: **I:** Overview of computer graphics pipeline and interactive systems **II:** Basic rendering concepts: rasterization, ray tracing, shading models **III:** Color theory and color spaces **IV:** Coordinate systems and transformations **V:** Basic interaction techniques: input devices, event handling **VI:** Advanced rendering concepts: global illumination, physically based rendering **VII:** Graphics hardware and acceleration techniques **VIII:** Human perception and visual cognition

Learning Outcomes **I:** Explain the basic stages of the computer graphics pipeline [Explain] **II:** Describe basic rendering techniques such as rasterization and ray tracing [Describe] **III:** Apply color theory to design visually effective graphics [Apply] **IV:** Implement coordinate transformations for 2D and 3D graphics [Implement] **V:** Design basic interactive applications using common input devices and event handling [Design] **VI:** Compare advanced rendering techniques such as global illumination and physically based rendering [Compare] **VII:** Analyze the role of graphics hardware in accelerating rendering tasks [Analyze] **VIII:** Explain how human perception influences graphics and interaction design [Explain]

Bibliography: [Brookshear and Brylow, 2019]

5.25 Applied Rendering and Techniques, 2 hr(s)

Competences: AG-C02,AG-C08,AG-C12

Topics: **I:** Ray tracing: acceleration structures, sampling, shading **II:** Texture mapping and filtering

Learning Outcomes **I:** Implement rasterization algorithms for primitive rendering [Implement] **II:** Implement basic ray tracing with acceleration structures [Implement]

Bibliography: [Brookshear and Brylow, 2019]

5.26 Closure Class: How Does a Search Engine Like Google Work?, 2 hr(s)

Competences: AG-C02,AG-C08,AG-C12

Topics: **I:** Problem analysis **II:** The index does not grow linearly with the size of the indexed information. **III:** Response time does not depend on the size of the "database." **IV:** Response time does not depend on the number of occurrences found. **V:** Combining various data structures to reach a solution. **VI:** Analyzing the scalability of the solution.

Learning Outcomes **I:** Understand the principles under which a search engine is created [Usage] **II:** Correctly apply data structures to solve the problem [Usage] **III:** Apply concepts related to algorithmic complexity in a search engine [Usage].

Bibliography: [Brookshear and Brylow, 2019]

6. Bibliography

[Brookshear and Brylow, 2019] Brookshear, J. G. and Brylow, D. (2019). *Computer Science: An Overview*. Pearson, global edition edition.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

CS112 Objects-oriented programming I (Mandatory)

1. General information

- **Semester**: 2nd Sem. **Credits**: 4
- **Hour of this course**: **Theory**: 2 hours; **Laboratory**: 4 hours;
- **Prerrequisites**:
  - BIC101 Introduction to Computing (1st Sem-Pg. 239)

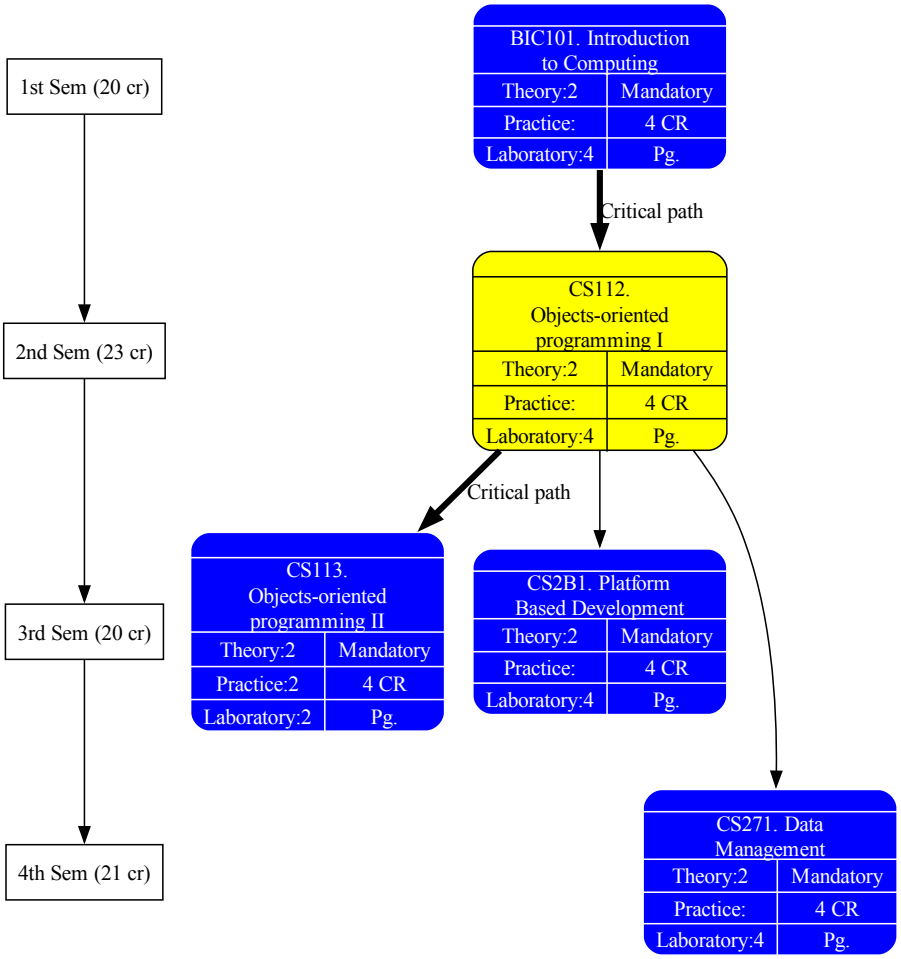

Figure 5.7: Connection Map. CS112 Objects-oriented programming I

2. Course Rationale

This is the second course in the introductory Computer Science programming sequence. It incorporates a programming language transition as a pedagogical strategy and focuses on core Object-Oriented Programming (OOP) concepts in C++17/C++20, expanding on the foundations from the first course while introducing low-level programming concepts. The course provides a solid foundation for advanced topics covered in the subsequent course of the curriculum.

3. Goals of the Course

- Introduce the student to the foundations of the object orientation paradigm in C++
- Develop skills in C++17/C++20 programming including templates and STL
- Understand memory management and low-level programming concepts
- Develop problem-solving skills using OOP principles
- Master intermediate C++ programming techniques

4. General Competences (*Outcomes*)

AG-C09) Design and Development of Solutions: Designs, implements, and evaluates solutions for complex computing problems. (Usage)

AG-C12) Applies computer science theory and software development fundamentals to produce computer-based solutions. (Usage)

5. Content

5.1 Language Pragmatics, 2 hr(s)

Competences: AG-C09,AG-C12

Topics: **I:** Problem domains and programming paradigm **II:** Criteria for good programming language design: (a) Principles of language design such as orthogonality (b) Defining control and iteration constructs (c) Modularization of large software **III:** Brief review of programming paradigms **IV:** Comparison between functional and imperative programming **V:** History of programming languages (emphasis on C and C++)

Learning Outcomes **I:** Discuss the historical context of programming language paradigms [Familiarity]. **II:** Evaluate language design trade-offs for specific applications [Assessment].

Bibliography: [Stroustrup, 2013, Josuttis, 2019]

5.2 Language Translation and Execution, 4 hr(s)

Competences: AG-C09,AG-C12

Topics: **I:** Execution models for JIT (Just-In-Time), compiler, interpreter **II:** Use of intermediate code, e.g., bytecode **III:** Execution as native code or within a virtual machine **IV:** Virtual machines and intermediate languages **V:** Compilation vs. interpretation models

Learning Outcomes **I:** Explain and understand the differences between compiled, JIT, and interpreted language implementations, including the benefits and limitations of each [Familiarity] **II:** Differentiate syntax and parsing from semantics and evaluation [Usage] **III:** Illustrate ambiguity in parsing using nested if-then-else/arithmetic expression and show resolution using precedence order [Assessment]

Bibliography: [Stroustrup, 2013, Deitel and Deitel, 2017]

5.3 Type Systems Fundamentals, 8 hr(s)

Competences: AG-C09,AG-C12

Topics: **I:** A type as a set of values together with a set of operations: (a) Primitive types (e.g., numbers, Booleans) (b) Compound types built from other types (e.g., records/structs, unions, arrays, lists, functions, references using set operations) **II:** Generic types (parametric polymorphism): (a) Definition and advantages of polymorphism: parametric, subtyping, overloading, and coercion (b) Comparison of monomorphic and polymorphic types (c) Comparison with ad-hoc polymorphism (overloading) and subtype polymorphism (d) Generic parameters and typing (e) Use of generic libraries such as collections (f) Comparison with ad hoc polymorphism (overloading)

and subtype polymorphism (g) Prescriptive vs descriptive polymorphism (h) Implementation models of polymorphic types (i) Subtyping **III:** Declaration models (linking, visibility, scope, lifetime) **IV:** Overview of type checking in C++ **V:** Type conversion and casting in C++

Learning Outcomes **I:** Define and use program pieces (such as functions, classes, methods) that use generic types, including for collections [Usage] **II:** Discuss the differences among generics, subtyping, and overloading [Familiarity] **III:** Explain multiple benefits and limitations of static typing in writing, maintaining, and debugging software [Assessment]

Bibliography: [Stroustrup, 2013, Josuttis, 2019]

5.4 Object-Oriented Programming Fundamentals, 14 hr(s)

Competences: AG-C09,AG-C12

Topics: **I:** Definition of classes: fields, methods, and constructors. **II:** Subclasses, inheritance (including multiple inheritance), and method overriding. **III:** Dynamic dispatch: definition of method-call. **IV:** Subtyping: (a) Subtype polymorphism; implicit upcasts in typed languages. (b) Notion of behavioral replacement: subtypes acting like supertype. (c) Relationship between subtyping and inheritance. **V:** Classes and objects in C++ **VI:** Constructors and destructors **VII:** Access specifiers and encapsulation **VIII:** Inheritance hierarchies **IX:** Polymorphism and virtual functions

Learning Outcomes **I:** Compose a class through design, implementation, and testing to meet behavioral requirements [Usage] **II:** Build a simple class hierarchy utilizing subclassing that allows code to be reused for distinct subclasses [Usage] **III:** Predict and validate control flow in a program using dynamic dispatch [Assessment] **IV:** Design C++ classes with proper encapsulation [Usage]. **V:** Implement inheritance hierarchies for code reuse [Usage].

Bibliography: [Stroustrup, 2013, Josuttis, 2019]

5.5 Systems Execution and Memory Model, 12 hr(s)

Competences: AG-C09,AG-C12

Topics: **I:** Direct, indirect, and indexed access to memory location **II:** Run-time representation of data abstractions such as variables, arrays, vectors, records, pointer-based data elements such as linked-lists and trees, and objects **III:** Low level allocation and accessing of high-level data structures such as basic data types, n-dimensional array, vector, record, and objects **IV:** Return from procedure as automatic deallocation mechanism for local data elements in the stack **V:** Manual memory management: allocating, de-allocating, and reusing heap memory **VI:** Automated memory management: garbage collection as an automated technique using the notion of reachability **VII:** Memory layout (stack, heap, static) **VIII:** Pointer arithmetic and operations **IX:** Dynamic memory allocation (new/delete) **X:** Memory leaks and dangling pointers **XI:** Smart pointers (unique_ptr, shared_ptr)

Learning Outcomes **I:** Explain why memory leaks and dangling pointer problems occur, and what can be done by a programmer to avoid/fix them [Usage] **II:** Explain how programming language implementations typically organize memory into global data, text, heap, and stack sections and how features such as recursion and memory management map to this memory model [Assessment] **III:** Explain how a core language construct, such as data abstractions and control abstractions, is executed [Assessment] **IV:** Implement RAII for resource management [Usage]. **V:** Debug memory-related errors in C++ programs [Usage].

Bibliography: [Stroustrup, 2013, Deitel and Deitel, 2017]

5.6 Functional Programming, 10 hr(s)

Competences: AG-C09,AG-C12

Topics: **I:** Lambda expressions and evaluation: (a) Variable binding and scope rules. (b) Parameter-passing. (c) Nested lambda expressions and reduction order. **II:** Using higher-order functions (taking, returning, and storing functions). **III:** Function templates and class templates **IV:** Template specialization **V:** Lambda expressions in C++11/14 **VI:** Standard Template Library concepts

Learning Outcomes **I:** Develop useful functions that take and return other functions [Usage] **II:** Correctly interpret variables and lexical scope in a program using function closures [Assessment] **III:** Use functional encapsulation mechanisms such as closures and modular interfaces [Usage] **IV:** Implement generic functions using templates [Usage]. **V:** Use lambda expressions for functional programming [Usage].

Bibliography: [Stroustrup, 2013, Josuttis, 2019]

5.7 Object-Oriented Programming: Encapsulation, Subtyping, and Reflection, 8 hr(s)

Competences: AG-C09,AG-C12

Topics: **I:** Collection classes, iterators, and other common library components. **II:** Sequence containers (vector, list, deque) **III:** Associative containers (set, map, unordered_set) **IV:** Container adapters (stack, queue, priority_queue) **V:** Iterators and algorithms **VI:** Functors and predicates

Learning Outcomes **I:** Use collection classes and iterators effectively to solve a problem [Usage] **II:** Define and use iterators and other operations on aggregates, including operations that take functions as arguments, in multiple programming languages, selecting the most natural idioms for each language [Usage] **III:** Compare and contrast mechanisms to define and protect data elements within procedural, functional, and object-oriented approaches [Assessment] **IV:** Select appropriate STL containers for specific needs [Usage].

Bibliography: [Stroustrup, 2013, Josuttis, 2019]

5.8 Advanced Programming Constructs, 6 hr(s)

Competences: AG-C09,AG-C12

Topics: **I:** Object-oriented abstractions: multiple inheritance, mixins, traits, multimethods **II:** Metaprogramming: macros, generative programming, model-based development **III:** Operator overloading **IV:** Move semantics and rvalue references **V:** Exception handling in C++ **VI:** Namespaces and modular programming

Learning Outcomes **I:** Use various advanced programming constructs and idioms correctly [Usage] **II:** Discuss how various advanced programming constructs aim to improve program structure, software quality, and programmer productivity [Assessment] **III:** Discuss how various advanced programming constructs interact with the definition and implementation of other language features [Familiarity] **IV:** Implement move semantics for performance optimization [Usage].

Bibliography: [Stroustrup, 2013, Josuttis, 2019]

5.9 Shell Scripting, 4 hr(s)

Competences: AG-C09,AG-C12

Topics: **I:** System commands: (a) Interface with operating systems **II:** Piping **III:** File abstraction and operators **IV:** Programs and processes **V:** Automating compilation and testing of C++ programs through scripts (Makefiles, build/test scripts)

Learning Outcomes I: Create and execute automated scripts to manage various system tasks [Usage]

Bibliography: [Robbins and Beebe, 2005]

6. Bibliography

[Deitel and Deitel, 2017] Deitel, H. and Deitel, P. (2017). *C++ How to Program*. Pearson, Boston, MA, 10th edition.

[Josuttis, 2019] Josuttis, N. M. (2019). *C++17: The Complete Guide*. Pearson Education, Indianapolis, IN, 1st edition.

[Robbins and Beebe, 2005] Robbins, A. and Beebe, N. H. (2005). *Classic Shell Scripting*. O'Reilly Media, Sebastopol, CA, 1st edition.

[Stroustrup, 2013] Stroustrup, B. (2013). *The C++ Programming Language*. Addison-Wesley Professional, Upper Saddle River, NJ, 4th edition.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

CS1D1 Discrete Structures (Mandatory)

1. General information

- **Semester**: 2nd Sem. **Credits**: 4
- **Hour of this course**: **Theory**: 2 hours; **Practice**: 4 hours;
- **Prerrequisites**: None

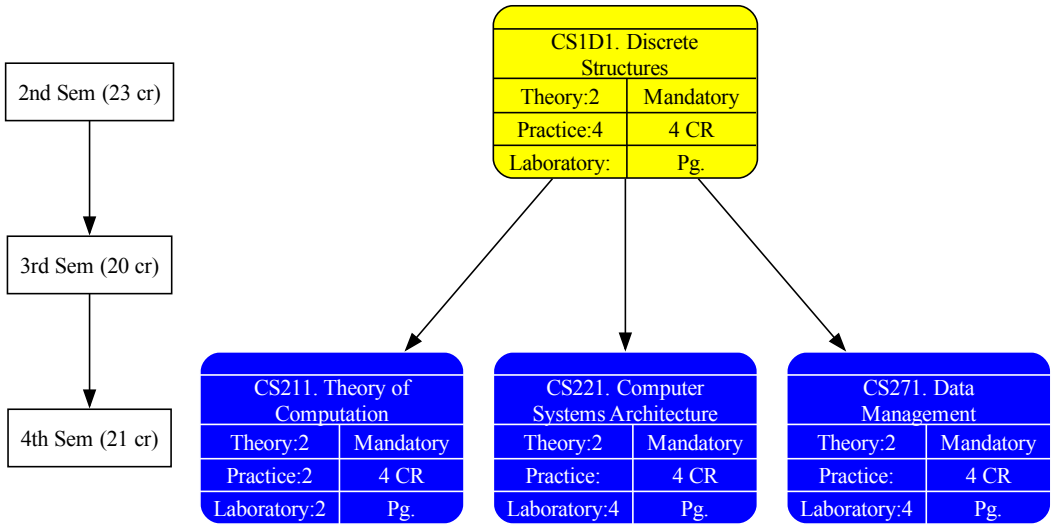

Figure 5.8: Connection Map. CS1D1 Discrete Structures

2. Course Rationale

Discrete structures provide the theoretical foundation for computer science. This course introduces the essential mathematical tools used to model computational problems, including logic, set theory, and functions. Understanding these concepts is critical for developing rigorous reasoning skills and for the subsequent study of algorithms, databases, and formal languages.

3. Goals of the Course

- Understand the principles of mathematical logic and proof techniques.
- Perform operations with sets, functions, and relations.
- Apply basic counting techniques and recurrence relations to problems.
- Develop the ability to think abstractly and solve problems formally.

4. General Competences (*Outcomes*)

AG-C07) Computing Knowledge: Applies knowledge of mathematics, science, and computing. (Usage)

AG-C12) Applies computer science theory and software development fundamentals to produce computer-based solutions. (Usage)

5. Content

5.1 Logic and Foundations, 12 hr(s)

Competences: AG-C07,AG-C12

Topics: **I:** Logic: propositional and predicate logic, quantifiers, proofs **II:** Sets: operations, cardinality, power sets, Cartesian products **III:** Relations: properties, equivalence relations, partial orders **IV:** Functions: injective, surjective, bijective, composition, inverse **V:** Advanced logic: proof systems, completeness, decidability **VI:** Advanced set theory: infinite sets, Cantor's diagonal argument

Learning Outcomes **I:** Explain propositional and predicate logic [Explain] **II:** Construct simple logical proofs [Apply] **III:** Define basic set operations and relations [Explain] **IV:** Define properties of relations [Explain] **V:** Define types of functions [Explain]

Bibliography: [Rosen, 2019, Lehman et al., 2018]

5.2 Proof Techniques and Advanced Structures, 12 hr(s)

Competences: AG-C07,AG-C12

Topics: **I:** Logic: propositional and predicate logic, quantifiers, proofs **II:** Advanced logic: proof systems, completeness, decidability **III:** Advanced set theory: infinite sets, Cantor's diagonal argument **IV:** Advanced relations: closures, lattices **V:** Advanced functions: recursive functions, computability

Learning Outcomes **I:** Construct simple logical proofs [Apply] **II:** Explain concepts of infinite sets and cardinality [Explain] **III:** Apply closure operations to relations [Apply] **IV:** Determine if a relation is an equivalence relation [Apply]

Bibliography: [Rosen, 2019, Velleman, 2019]

5.3 Relations, Functions and Graphs, 12 hr(s)

Competences: AG-C07,AG-C12

Topics: **I:** Relations: properties, equivalence relations, partial orders **II:** Functions: injective, surjective, bijective, composition, inverse **III:** Graphs: definitions, paths, cycles, trees, connectivity **IV:** Advanced graph theory: planarity, coloring, networks

Learning Outcomes **I:** Determine if a relation is an equivalence relation [Apply] **II:** Compose functions and find inverses [Apply] **III:** Define basic graph concepts [Explain] **IV:** Solve basic graph problems [Apply] **V:** Analyze advanced graph properties [Evaluate]

Bibliography: [Rosen, 2019, Scheinerman, 2012]

5.4 Counting and Recurrence, 12 hr(s)

Competences: AG-C07,AG-C12

Topics: **I:** Counting: permutations, combinations, binomial theorem **II:** Advanced counting: generating functions, inclusion-exclusion **III:** Recurrence relations: solving simple recurrences **IV:** Advanced recurrence: generating functions, asymptotic behavior

Learning Outcomes **I:** Explain permutations and combinations [Explain] **II:** Apply counting techniques to solve problems [Apply] **III:** Define recurrence relations [Explain] **IV:** Solve simple recurrence relations [Apply] **V:** Develop generating functions for counting problems [Create]

Bibliography: [Rosen, 2019, Grimaldi, 2003]

6. Bibliography

[Grimaldi, 2003] Grimaldi, R. P. (2003). *Discrete and Combinatorial Mathematics: An Applied Introduction*. Pearson, 5th edition.

[Lehman et al., 2018] Lehman, E., Leighton, F. T., and Meyer, A. R. (2018). *Mathematics for Computer Science*. MIT OpenCourseWare.

[Rosen, 2019] Rosen, K. H. (2019). *Discrete Mathematics and Its Applications*. McGraw-Hill Education, 8th edition.

[Scheinerman, 2012] Scheinerman, E. A. (2012). *Mathematics: A Discrete Introduction*. Cengage Learning, 3rd edition.

[Velleman, 2019] Velleman, D. J. (2019). *How to Prove It: A Structured Approach*. Cambridge University Press, 3rd edition.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

BMA103 Integral Calculus (Mandatory)

1. General information

- **Semester**: 2nd Sem. **Credits**: 5
- **Hour of this course**: **Theory**: 4 hours; **Practice**: 2 hours;
- **Prerrequisites**:
  - BMA102 Differential Calculus (1st Sem-Pg. 246)

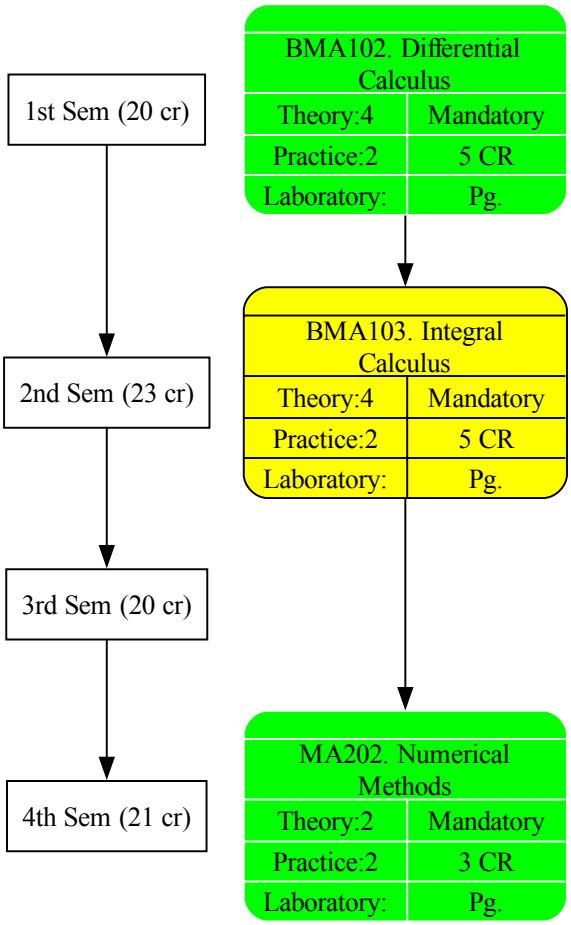

Figure 5.9: Connection Map. BMA103 Integral Calculus

2. Course Rationale

Integral calculus is a fundamental mathematical tool shared by computer science and engineering disciplines, essential for modeling and solving problems involving accumulation, change, and areas under curves. This course provides the foundations of integral calculus, including integration techniques and their applications in scientific and engineering contexts.

3. Goals of the Course

- Understand the concept of definite and indefinite integrals.

- Apply various integration techniques to solve problems.
- Use integral calculus to model and solve problems in scientific and engineering contexts.

4. General Competences (*Outcomes*)

AG-C07) Computing Knowledge: Applies knowledge of mathematics, science, and computing. (Familiarity)

5. Content

5.1 Antiderivatives and Integration Techniques, 20 hr(s)

Competences: AG-C07

Topics: **I:** The Fundamental Theorem of Calculus and the antiderivative **II:** Integration techniques: substitution, parts, partial fractions, and trigonometric substitution

Learning Outcomes **I:** State the Fundamental Theorem of Calculus and explain the relationship between differentiation and integration [Familiarity] **II:** Compute definite and indefinite integrals using standard techniques including substitution and integration by parts [Usage]

Bibliography: [Stewart, 2015, Larson and Edwards, 2014]

5.2 Improper Integrals, 8 hr(s)

Competences: AG-C07

Topics: **I:** Improper integrals: convergence criteria and comparison tests

Learning Outcomes **I:** Determine convergence of improper integrals using comparison and other convergence tests [Assessment]

Bibliography: [Stewart, 2015, Larson and Edwards, 2014]

5.3 Sequences, Series, and Power Series, 20 hr(s)

Competences: AG-C07

Topics: **I:** Sequences and series: convergence tests (ratio, root, comparison, integral) **II:** Power series, radius of convergence, Taylor and Maclaurin series

Learning Outcomes **I:** Determine convergence of power series and construct Taylor series for standard functions [Assessment]

Bibliography: [Stewart, 2015, Larson and Edwards, 2014]

6. Bibliography

[Larson and Edwards, 2014] Larson, R. and Edwards, B. H. (2014). *Calculus*. Cengage Learning.

[Stewart, 2015] Stewart, J. (2015). *Calculus: Early Transcendentals*. Cengage Learning.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

BFI101 Physics I (Mandatory)

1. General information

- **Semester**: 2nd Sem. **Credits**: 5
- **Hour of this course**: **Theory**: 4 hours; **Laboratory**: 4 hours;
- **Prerrequisites**: None

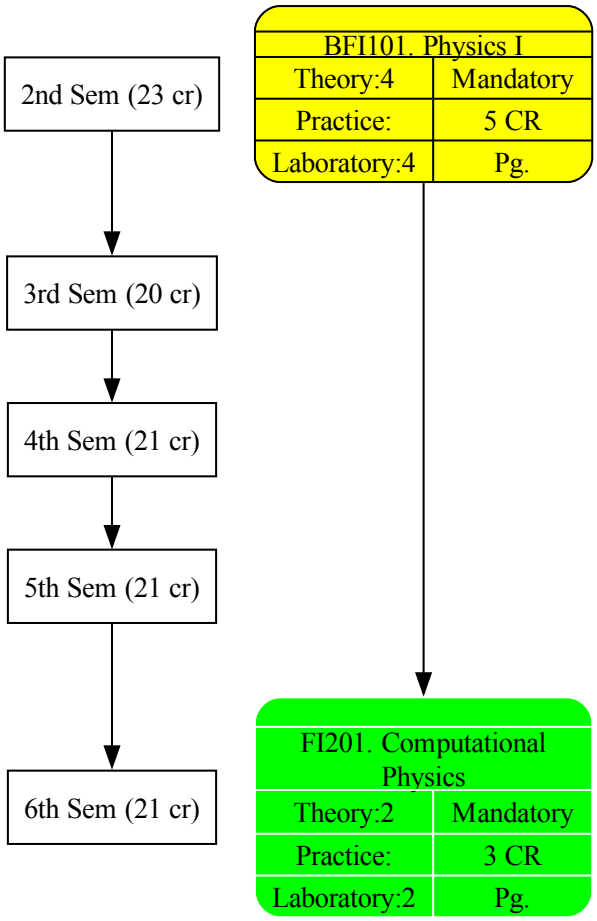

Figure 5.10: Connection Map. BFI101 Physics I

2. Course Rationale

Physics I introduces the foundations of classical mechanics as a mathematical and experimental discipline. Through the study of kinematics, Newtonian dynamics, conservation principles, rotational motion, elasticity, fluid mechanics, vibrations and waves, and an introduction to transport phenomena, students acquire the analytical tools needed to model and solve problems of the physical world. This course provides the indispensable foundation for the study of oscillatory, wave, and electromagnetic phenomena addressed in subsequent courses.

3. Goals of the Course

- Describe the motion of particles in different coordinate systems and formulate the equations of motion from Newton's laws.
- Apply the principles of work, potential energy, and conservation of mechanical energy to systems with conservative and non-conservative forces.
- Solve particle-system dynamics problems using conservation of linear momentum and collision analysis.
- Analyze rotational motion and angular momentum dynamics in systems of particles and extended bodies.
- Apply elastic moduli and Hooke's law to relate stress and strain in solids.
- Apply the principles of hydrostatics and hydrodynamics, including the continuity and Bernoulli equations, to fluid flow problems.
- Analyze simple harmonic motion and mechanical wave propagation, including superposition and resonance phenomena.
- Apply Fourier's and Fick's laws to describe basic heat conduction and mass diffusion processes.

4. General Competences (*Outcomes*)

AG-C07) Computing Knowledge: Applies knowledge of mathematics, science, and computing. (Familiarity)

5. Content

5.1 Kinematics and Dynamics of Particles, 22 hr(s)

Competences: AG-C07

Topics: **I:** Vector description of position, velocity, and acceleration; Newton's three laws and inertial reference frames **II:** Formulation and analytical solution of equations of motion for constant and variable forces **III:** Phase space diagrams (position vs. velocity) for visualizing mechanical state, stability, and periodicity **IV:** Numerical integration of equations of motion using Euler, Verlet, and Runge-Kutta methods **V:** Constrained motion and holonomic constraints relevant to robotic linkages and articulated bodies

Learning Outcomes **I:** Identify all forces acting on a particle in a given mechanical scenario and draw a correct free-body diagram [Familiarity] **II:** Implement a numerical integration algorithm such as the Runge-Kutta method to compute the trajectory of a particle under variable forces [Usage] **III:** Construct and interpret phase space diagrams to characterize the stability and periodicity of a simulated mechanical system [Assessment]

Bibliography: [Serway and Jewett, 2018, Young and Freedman, 2015]

5.2 Work, Energy, and Conservation Laws, 12 hr(s)

Competences: AG-C07

Topics: **I:** Work-energy theorem relating net work done to change in kinetic energy **II:** Potential energy functions and the force-gradient relation for conservative force fields **III:** Conservation of total mechanical energy and conditions for its applicability **IV:** Energy diagrams for identifying equilibrium points, turning points, and stability **V:** Elastic and inelastic collision models: momentum and energy exchange in physics engine implementations

Learning Outcomes **I:** Explain the distinction between conservative and non-conservative forces and the conditions under which mechanical energy is conserved [Familiarity] **II:** Apply energy conservation methods to compute speeds and positions in one-dimensional motion problems common in physics engine implementations [Usage] **III:** Analyze elastic and inelastic collision events, computing post-collision velocities and quantifying energy dissipation [Assessment]

Bibliography: [Serway and Jewett, 2018, Tipler and Mosca, 2007]

5.3 Systems of Particles and Linear Momentum, 8 hr(s)

Competences: AG-C07

Topics: **I:** Definition of center of mass: $\vec{R}_{CM} = \frac{1}{M}\sum m_i \vec{r}_i$ **II:** Linear momentum of a particle and system: $\vec{p} = m\vec{v}$ and $\vec{P} = M\vec{V}_{CM}$ **III:** Newton's second law for systems: $\vec{F}_{ext} = \frac{d\vec{P}}{dt}$ **IV:** Conservation of linear momentum for isolated systems **V:** Distinction between internal and external forces in a system **VI:** Two-body collisions including elastic, inelastic, and completely inelastic cases **VII:** Coefficient of restitution and its physical meaning in collisions **VIII:** Systems with variable mass such as the rocket equation: $M\frac{dV}{dt} = v_{ex}\frac{dM}{dt}$ **IX:** Many-particle systems and their center of mass dynamics **X:** Continuous mass distributions and their center of mass calculation using integration

Learning Outcomes **I:** Locate the center of mass for a discrete set of particles in One, Two, and Three dimensions [Usage] **II:** Determine the total linear momentum of a system from the motion of its center of mass [Usage] **III:** Apply conservation of linear momentum to solve collision problems in One and Two dimensions [Usage] **IV:** Classify collisions as elastic, inelastic, or completely inelastic based on kinetic energy conservation [Familiarity] **V:** Calculate the coefficient of restitution from pre-collision and post-collision velocities [Usage] **VI:** Solve the rocket equation to determine velocity as a function of ejected mass [Usage] **VII:** Analyze how internal forces affect individual particles but not the total momentum of a system [Assessment] **VIII:** Integrate to find the center of mass of symmetric continuous bodies such as rods, disks, and spheres [Usage]

Bibliography: [Young and Freedman, 2015, Tipler and Mosca, 2007]

5.4 Rotational Motion and Angular Momentum, 12 hr(s)

Competences: AG-C07

Topics: **I:** Rotational kinematics including angular displacement, velocity, and acceleration (θ, ω, α) **II:** Torque definition: $\vec{\tau} = \vec{r} \times \vec{F}$ and its magnitude calculation **III:** Angular momentum of a particle: $\vec{L} = \vec{r} \times \vec{p}$ **IV:** Rotational form of Newton's second law: $\vec{\tau}_{net} = \frac{d\vec{L}}{dt}$ **V:** Moment of inertia $I = \sum m_i r_i^2$ and the parallel axis theorem **VI:** Rotational kinetic energy: $K_{rot} = \frac{1}{2}I\omega^2$ **VII:** Conservation of angular momentum for isolated systems **VIII:** Gyroscopic motion and the phenomenon of precession **IX:** Rolling without slipping dynamics: $v_{CM} = R\omega$ **X:** Coriolis effect as a manifestation of rotational dynamics in non-inertial frames

Learning Outcomes **I:** Relate linear and angular kinematic quantities for circular motion [Usage] **II:** Calculate torque about a given axis for a force applied at a specific point [Usage] **III:** Determine the moment of inertia for simple geometries such as a rod, hoop, disk, or sphere [Usage] **IV:** Apply conservation of angular momentum to analyze changes in rotational speed when the moment of inertia changes [Assessment] **V:** Compute rotational kinetic energy for a rotating rigid body [Usage] **VI:** Analyze the motion of objects rolling without slipping down an incline [Assessment] **VII:** Predict the direction of precession for a spinning top or gyroscope [Assessment] **VIII:** Explain the Coriolis effect in terms of rotating reference frames and its effect on projectiles [Familiarity]

Bibliography: [Serway and Jewett, 2018, Kleppner and Kolenkow, 2013]

5.5 Elasticity, 2 hr(s)

Competences: AG-C07

Topics: **I:** Definition of stress (force/area) and strain $(\Delta L/L)$ **II:** Young's modulus and Hooke's law for longitudinal strain: $\sigma = E\epsilon$ **III:** Shear modulus (rigidity) and shear strain **IV:** Bulk modulus (compressibility) and volumetric strain **V:** Elastic limit, plastic region, and fracture point on the stress-strain diagram **VI:** Elastic energy stored in a deformed material

Learning Outcomes **I:** Apply the scalar Hooke's law to calculate the deformation of a rod under axial load [Usage] **II:** Calculate Young's, shear, or bulk modulus from experimental stress-strain data [Usage] **III:** Identify the elastic region, elastic limit, and fracture point on a stress-strain curve [Familiarity] **IV:** Calculate the elastic potential energy stored in a deformed spring or rod [Usage]

Bibliography: [Serway and Jewett, 2018, Young and Freedman, 2015]

5.6 Fluids: Hydrostatics, Hydrodynamics, and Viscosity, 10 hr(s)

Competences: AG-C07

Topics: **I:** Density, pressure, and its variation with depth in a fluid at rest: $p = p_0 + \rho g h$ **II:** Pascal's principle and its applications (hydraulic press) **III:** Archimedes' principle, buoyant force, and flotation **IV:** Pressure measurement with manometers and barometers **V:** Continuity equation for incompressible flow: $A_1 v_1 = A_2 v_2$ **VI:** Bernoulli's equation and energy conservation in ideal flow **VII:** Applications: Venturi tube, Torricelli's theorem, Pitot tube **VIII:** Viscosity and Poiseuille's law for laminar flow in pipes **IX:** Reynolds number and the transition to turbulent flow **X:** Stokes' law for the drag force on a sphere in a viscous fluid

Learning Outcomes **I:** Calculate the pressure at a point in a fluid at rest given depth and density [Usage] **II:** Apply Archimedes' principle to determine whether an object floats, sinks, or remains in equilibrium [Usage] **III:** Apply the continuity and Bernoulli equations to solve flow problems in pipes of varying cross-section [Usage] **IV:** Interpret manometer and barometer readings to determine absolute and gauge pressures [Assessment] **V:** Apply Poiseuille's law to calculate the flow rate of a viscous fluid in a cylindrical tube [Usage] **VI:** Estimate the Reynolds number of a flow and predict whether it is laminar or turbulent [Assessment]

Bibliography: [Serway and Jewett, 2018, Young and Freedman, 2015]

5.7 Vibrations and Waves, 14 hr(s)

Competences: AG-C07

Topics: **I:** Simple harmonic motion (SHM): equation of motion, position, velocity, and acceleration **II:** Oscillating systems: spring-mass and simple/physical pendulum **III:** Energy in simple harmonic motion **IV:** Damped and forced oscillations; introduction to resonance **V:** Transverse and longitudinal waves; propagation speed **VI:** Wave equation and speed of a wave on a taut string: $v = \sqrt{T/\mu}$ **VII:** Superposition principle and standing waves **VIII:** Sound waves, intensity, and sound intensity level (decibels) **IX:** Doppler effect for sound waves **X:** Beats and interference of waves of nearby frequencies

Learning Outcomes **I:** Derive the equation of motion and the period of a spring-mass system and a simple pendulum [Usage] **II:** Calculate the kinetic, potential, and total energy of a simple harmonic oscillator at any instant [Usage] **III:** Analyze the amplitude response of a forced oscillator near the resonance frequency [Assessment] **IV:** Calculate the propagation speed of a wave on a string from the tension and linear density [Usage] **V:** Determine the frequencies and wavelengths of the normal modes of a standing wave on a fixed string [Usage] **VI:** Calculate the sound intensity level in decibels from the intensity of a sound wave [Usage] **VII:** Apply the Doppler effect formula for a moving source and/or observer [Usage]

Bibliography: [Serway and Jewett, 2018, Young and Freedman, 2015]

5.8 Transport Phenomena, 4 hr(s)

Competences: AG-C07

Topics: **I:** Heat conduction and Fourier's law: $\vec{q} = -k\nabla T$ **II:** Thermal conductivity of materials and thermal resistance in composite walls **III:** Molecular diffusion and Fick's first law: $\vec{J} = -D\nabla C$ **IV:** Analogy between heat, mass, and momentum transport (Prandtl and Schmidt numbers)

Learning Outcomes **I:** Apply Fourier's law to calculate the heat flux through a plane or composite wall [Usage] **II:** Apply Fick's first law to calculate the diffusive flux of a species in a concentration gradient [Usage] **III:** Identify mathematical analogies between heat, mass, and momentum transport processes [Familiarity]

Bibliography: [Serway and Jewett, 2018, Tipler and Mosca, 2007]

6. Bibliography

[Kleppner and Kolenkow, 2013] Kleppner, D. and Kolenkow, R. J. (2013). *An Introduction to Mechanics*. Cambridge University Press, 2nd edition.

[Serway and Jewett, 2018] Serway, R. A. and Jewett, J. W. (2018). *Physics for Scientists and Engineers with Modern Physics*. Cengage Learning, 10th edition.

[Tipler and Mosca, 2007] Tipler, P. A. and Mosca, G. (2007). *Physics for Scientists and Engineers*. W. H. Freeman and Company, 6th edition.

[Young and Freedman, 2015] Young, H. D. and Freedman, R. A. (2015). *Sears and Zemansky's University Physics with Modern Physics*. Pearson Education, 14th edition.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

BEI102 Technical and professional English II (Mandatory)

1. General information

- **Semester**: 2nd Sem. **Credits**: 2
- **Hour of this course**: **Practice**: 4 hours;
- **Prerrequisites**:
  - BEI101 Technical and professional English I (1st Sem-Pg. 254)

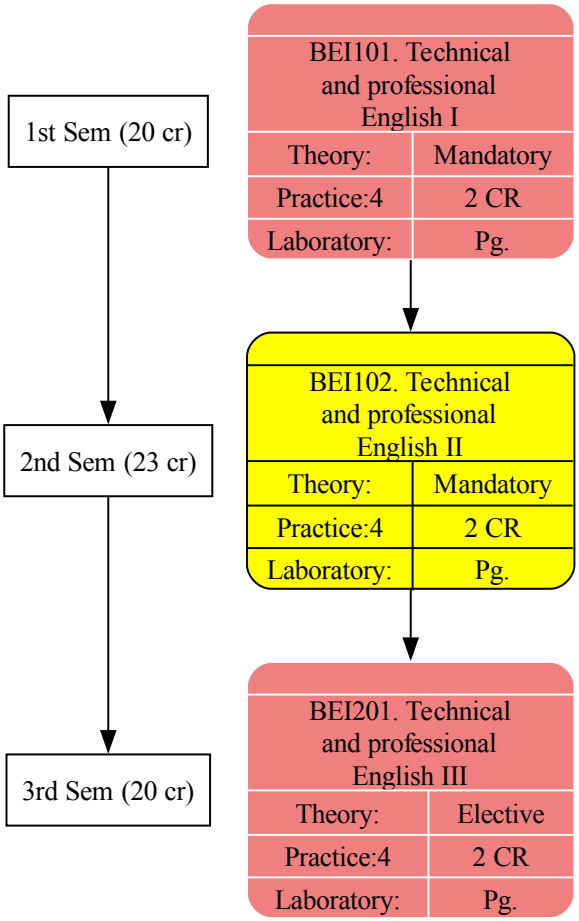

Figure 5.11: Connection Map. BEI102 Technical and professional English II

2. Course Rationale

Building on foundational English skills, this course develops the student's ability to communicate in technical and scientific contexts with greater fluency. Grammar structures are practiced through reading and producing descriptions of technological systems, scientific phenomena, and engineering processes. Students gain vocabulary relevant to computing, data, and the physical sciences.

3. Goals of the Course

- Develop fluency in English applied to technical and scientific communication.

- Expand vocabulary in computing, data, and engineering domains.

4. General Competences (*Outcomes*)

AG-C03) Individual and Team Work: Performs effectively as an individual and as a member or leader in diverse teams. (Familiarity)

AG-C04) Communication: Communicates effectively in complex computing activities. (Usage)

5. Content

5.1 Breakthroughs and Discoveries, 12 hr(s)

Competences: AG-C03,AG-C04

Topics: I: Simple Past Review: Affirmative and Negative in scientific contexts. **II:** Time expressions with 'ago': When technologies were invented or released. **III:** Conjunctions: and, so, but, because — structuring technical explanations. **IV:** Vocabulary: Scientific milestones, inventions, and discoveries.

Learning Outcomes I: By the end of the unit, students narrate past technological breakthroughs and connect cause-and-effect relationships using simple past and conjunctions (e.g., "Scientists discovered X, so engineers developed Y").

Bibliography: [Soars and John, 2002, Cambridge, 2006, MacGrew, 1999]

5.2 Data and Variables, 12 hr(s)

Competences: AG-C03,AG-C04

Topics: I: Countable and Uncountable nouns in computing: files, data, code, memory. **II:** Use of some, any, a lot of in quantifying technical resources. **III:** Expressions of quantity: a little memory, a few errors, too many processes. **IV:** Vocabulary: Data types, variables, constants, and storage units.

Learning Outcomes I: Students correctly use countable and uncountable noun structures to describe data, resources, and system states in technical contexts.

Bibliography: [Soars and John, 2002, Cambridge, 2006, MacGrew, 1999]

5.3 Future of Technology, 12 hr(s)

Competences: AG-C03,AG-C04

Topics: I: Future plans: Verb + to + infinitive in project planning. **II:** Use of 'Will' vs 'Going to' for predictions and scheduled releases. **III:** Auxiliary verbs in forecasting: AI trends, cloud computing, quantum computing. **IV:** Vocabulary: Emerging technologies and career pathways in computing.

Learning Outcomes I: Students express technological intentions and project plans, as well as make evidence-based predictions about computing and engineering trends.

Bibliography: [Soars and John, 2002, Cambridge, 2006, MacGrew, 1999]

5.4 Comparing Technologies, 12 hr(s)

Competences: AG-C03,AG-C04

Topics: **I:** Comparative and superlative adjectives applied to hardware and software. **II:** Comparing programming languages, operating systems, and algorithms. **III:** Synonyms and antonyms in technical vocabulary. **IV:** Reading: Short technology comparison articles and benchmark reports.

Learning Outcomes **I:** Students make detailed technical comparisons between systems, languages, and tools, using comparative and superlative structures accurately.

Bibliography: [Soars and John, 2002, Cambridge, 2006, MacGrew, 1999]

5.5 Running Processes, 12 hr(s)

Competences: AG-C03,AG-C04

Topics: **I:** Present Continuous to describe active and ongoing technical processes. **II:** Current use: "The server is processing requests." **III:** Future use with Present Continuous: scheduled deployments and experiments. **IV:** Writing: Status updates and progress reports in technical projects.

Learning Outcomes **I:** Students describe active computing processes, ongoing experiments, and scheduled technical activities using the present continuous form.

Bibliography: [Soars and John, 2002, Cambridge, 2006, MacGrew, 1999]

5.6 Software Milestones, 12 hr(s)

Competences: AG-C03,AG-C04

Topics: **I:** Introduction to Present Perfect to describe achievements with current relevance. **II:** Keywords: ever, never, yet, just — applied to software releases and research. **III:** Vocabulary: Version history, patches, upgrades, published papers. **IV:** Reading: Technology news and software changelogs.

Learning Outcomes **I:** Students communicate technical achievements and software milestones with present relevance, without specifying an exact time, using the present perfect.

Bibliography: [Soars and John, 2002, Cambridge, 2006, MacGrew, 1999]

5.7 Technical Challenges, 12 hr(s)

Competences: AG-C03,AG-C04

Topics: **I:** Adjectives ending in -ed and -ing to describe technical experiences and processes. **II:** Modal verbs of obligation: must, have to — applied to engineering requirements. **III:** Vocabulary: Engineering challenges, constraints, trade-offs, and standards. **IV:** Writing: Short technical problem statements.

Learning Outcomes **I:** Students describe the emotional and technical aspects of engineering challenges and express obligations and constraints using specialized vocabulary.

Bibliography: [Soars and John, 2002, Cambridge, 2006, MacGrew, 1999]

5.8 English for Technology II, 12 hr(s)

Competences: AG-C03,AG-C04

Topics: I: Technical terminology: Input/output devices, storage, and peripherals. **II:** Reading and following software installation and configuration instructions. **III:** Communication in technical support: Describing symptoms and solutions. **IV:** Presentation: Describing a computing system to a non-specialist audience.

Learning Outcomes I: By the end of the course, students interpret technical instructions, describe hardware components, and communicate basic computing concepts to diverse audiences in English.

Bibliography: [Soars and John, 2002, Cambridge, 2006, MacGrew, 1999]

6. Bibliography

[Cambridge, 2006] Cambridge (2006). *Diccionario Inglés-Espanol Cambridge*. Editorial Oxford.

[MacGrew, 1999] MacGrew, J. (1999). *Focus on Grammar Basic*. Editorial Oxford.

[Soars and John, 2002] Soars, L. and John (2002). *American Headway N 1 Student Book*. Editorial Oxford.

Third Semester

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

CS113 Objects-oriented programming II (Mandatory)

1. General information

- **Semester**: 3rd Sem. **Credits**: 4
- **Hour of this course**: **Theory**: 2 hours; **Practice**: 2 hours; **Laboratory**: 2 hours;
- **Prerrequisites**:
  - CS112 Objects-oriented programming I (2nd Sem-Pg. 268)

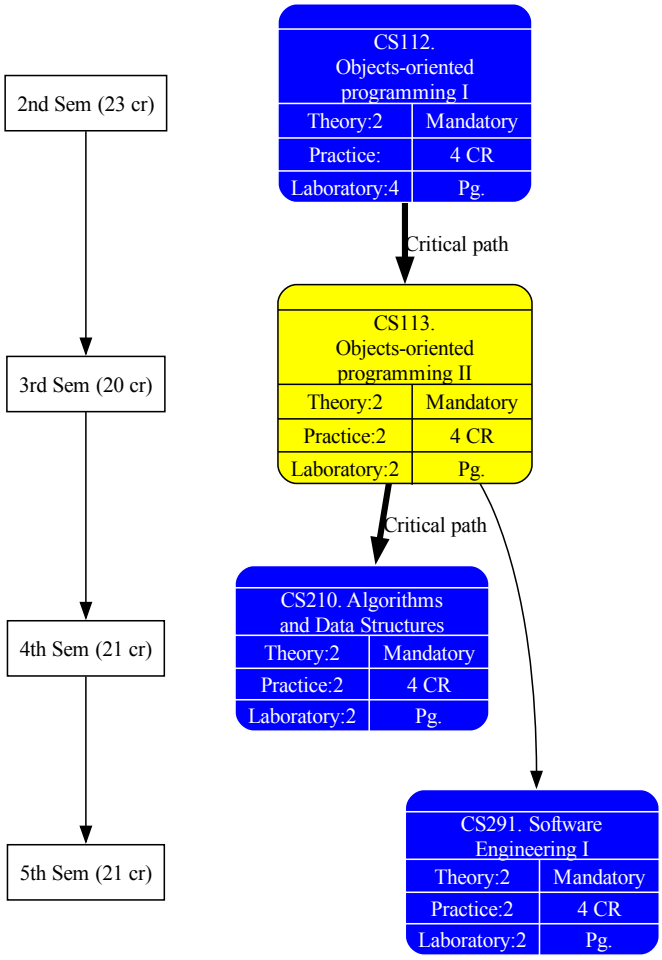

Figure 5.12: Connection Map. CS113 Objects-oriented programming II

2. Course Rationale

This is the third course in the sequence of introductory courses in computer science. This course explores advanced C++17/C++20 features focusing on high-performance systems development. Key topics include template metaprogramming, move semantics, perfect forwarding, RAII optimization,

multiple/virtual inheritance patterns, concurrency with std::thread and async/await, lock-free programming, modern C++ paradigms such as CRTP and expression templates, and interfacing C++ with other languages. The course prepares students for game engines, high-performance computing, and embedded development where C++ dominates.

3. Goals of the Course

- Master advanced C++17/C++20 features including templates and metaprogramming
- Develop skills in concurrent and systems programming
- Understand performance optimization techniques
- Prepare for high-performance computing applications
- Master template metaprogramming and compile-time computation

4. General Competences (*Outcomes*)

AG-C09) Design and Development of Solutions: Designs, implements, and evaluates solutions for complex computing problems. (Usage)

AG-C12) Applies computer science theory and software development fundamentals to produce computer-based solutions. (Assessment)

5. Content

5.1 Object-Oriented Programming: Encapsulation, Subtyping, and Reflection, 8 hr(s)

Competences: AG-C09,AG-C12

Topics: **I:** Collection classes, iterators, and other common library components. **II:** Associative containers (std::set, std::map, unordered variants) **III:** Container adapters (stack, queue, priority_queue) **IV:** Advanced STL algorithms **V:** Functors and predicates **VI:** Iterator categories and traits

Learning Outcomes **I:** Use collection classes and iterators effectively to solve a problem [Usage] **II:** Define and use iterators and other operations on aggregates, including operations that take functions as arguments, in multiple programming languages, selecting the most natural idioms for each language [Assessment] **III:** Compare and contrast the benefits and costs/impact of using inheritance (subclasses) and composition (specifically, how to base composition on higher order functions) [Usage] **IV:** Implement custom allocators for STL containers [Usage].

Bibliography: [Stroustrup, 2013, Josuttis, 2019, Vandevoorde et al., 2017]

5.2 Type Systems: Polymorphism and Complementary Typing, 12 hr(s)

Competences: AG-C09,AG-C12

Topics: **I:** Generic types (parametric polymorphism): (a) Definition and advantages of polymorphism: parametric, subtyping, overloading, and coercion (b) Comparison of monomorphic and polymorphic types (c) Comparison with ad-hoc polymorphism (overloading) and subtype polymorphism (d) Generic parameters and typing (e) Use of generic libraries such as collections (f) Comparison with ad hoc polymorphism (overloading) and subtype polymorphism (g) Prescriptive vs descriptive polymorphism (h) Implementation models of polymorphic types (i) Subtyping **II:** SFINAE (Substitution Failure Is Not An Error): historical foundation of constrained generic programming in C++, the basis of pre-C++20 libraries. **III:** Concepts (C++20): declarative syntax to constrain template parameters, the modern successor to SFINAE with better error diagnostics. **IV:** if constexpr (C++17): compile-time branching, replacing much of the tag dispatch previously handled with SFINAE. **V:** Perfect forwarding and universal references **VI:** Variadic templates **VII:** Compile-time computation with constexpr and consteval

Learning Outcomes **I:** Define and use program pieces (such as functions, classes, methods) that use generic types, including for collections [Assessment] **II:** Discuss the differences among generics, subtyping, and overloading [Assessment] **III:** Apply Concepts to constrain template parameters and improve error diagnostics [Usage]. **IV:** Distinguish when to use Concepts, if constexpr, or legacy SFINAE when maintaining existing C++ code [Assessment]. **V:** Implement perfect forwarding for efficient argument passing [Usage].

Bibliography: [Stroustrup, 2013, Vandevoorde et al., 2017, Stroustrup, 2022]

5.3 Memory Management, 8 hr(s)

Competences: AG-C09,AG-C12

Topics: **I:** Low level allocation and accessing of high-level data structures such as basic data types, n-dimensional array, vector, record, and objects **II:** Return from procedure as automatic deallocation mechanism for local data elements in the stack **III:** Manual memory management: allocating, de-allocating, and reusing heap memory **IV:** Automated memory management: garbage collection as an automated technique using the notion of reachability **V:** Lvalues and rvalues in C++11/14/17/20 **VI:** Move constructors and move assignment operators **VII:** Rvalue references and reference collapsing **VIII:** Resource Acquisition Is Initialization (RAII) pattern **IX:** Smart pointer customization **X:** Memory leak detection and prevention

Learning Outcomes **I:** Explain why memory leaks and dangling pointer problems occur, and what can be done by a programmer to avoid/fix them [Usage] **II:** Explain how programming language implementations typically organize memory into global data, text, heap, and stack sections and how features such as recursion and memory management map to this memory model [Assessment] **III:** Explain how a core language construct, such as data abstractions and control abstractions, is executed [Assessment] **IV:** Implement move semantics for resource optimization [Usage]. **V:** Design RAII classes for exception-safe resource management [Usage].

Bibliography: [Meyers, 2014, Stroustrup, 2013]

5.4 Parallel and Distributed Computing, 12 hr(s)

Competences: AG-C09,AG-C12

Topics: **I:** Safety and liveness: (a) Race conditions (b) Dependencies/preconditions (c) Fault models (d) Termination **II:** Programming models: (a) Actor models (b) Procedural and reactive models (c) Synchronous/asynchronous programming models (d) Data parallelism **III:** Communication and coordination: (a) Mutexes (b) Message-passing (c) Shared memory (d) Cobegin-coend (e) Monitors (f) Channels (g) Threads (h) Guards **IV:** std::thread and thread management **V:** std::jthread (C++20): automatic joining (RAII) and cooperative cancellation via stop_token **VI:** Mutexes, locks, and condition variables **VII:** C++20 synchronization primitives: std::latch, std::barrier, std::counting_semaphore **VIII:** Atomic operations and lock-free programming **IX:** std::async and futures **X:** std::execution and the senders-receivers model (C++26, experimental): a preview of where standard C++ concurrency is heading, presented at a conceptual level only, not evaluated in lab given its limited availability (partial in GCC) **XI:** Thread-safe design patterns **XII:** Data races and deadlock prevention

Learning Outcomes **I:** Implement correct concurrent programs using multiple programming models, such as shared memory, actors, futures, synchronization constructs, and data-parallelism primitives [Usage] **II:** Use synchronization constructions such as monitor/synchronized methods in a simple program [Usage] **III:** Use a message-passing model to analyze a communication protocol [Assessment] **IV:** Implement lock-free data structures using atomics [Usage]. **V:** Differentiate std::thread from std::jthread and justify when to use each [Usage]. **VI:** Analyze thread-safety of concurrent algorithms [Assessment].

Bibliography: [Williams, 2019, Stroustrup, 2013]

5.5 Advanced Programming Constructs, 8 hr(s)

Competences: AG-C09,AG-C12

Topics: **I:** Object-oriented abstractions: multiple inheritance, mixins, traits, multimethods **II:** Metaprogramming: macros, generative programming, model-based development **III:** Multiple inheritance and virtual inheritance **IV:** CRTP (Curiously Recurring Template Pattern) **V:** deducing this (C++23): simplifies CRTP by removing the need for static_cast<Derived&>(*this) for static polymorphism **VI:** Expression templates **VII:** Type traits and compile-time introspection **VIII:** Policy-based design **IX:** Interface design for libraries

Learning Outcomes **I:** Use various advanced programming constructs and idioms correctly [Usage] **II:** Discuss how various advanced programming constructs aim to improve program structure, software quality, and programmer productivity [Assessment] **III:** Discuss how various advanced programming constructs interact with the definition and implementation of other language features [Assessment] **IV:** Implement CRTP for static polymorphism [Usage]. **V:** Design extensible library interfaces using policy-based design [Usage].

Bibliography: [Stroustrup, 2013, Gregor, 2017]

5.6 Systems Execution and Memory Model, 8 hr(s)

Competences: AG-C09,AG-C12

Topics: **I:** Run-time representation of data abstractions such as variables, arrays, vectors, records, pointer-based data elements such as linked-lists and trees, and objects **II:** Data structures for translation, execution, translation, and code mobility such as stack, heap, aliasing (sharing using pointers), indexed sequence and string **III:** Implementation of the dynamic dispatch introduced in OOP I: virtual function table (vtable), vtable pointer (vptr), and its memory/performance cost versus static dispatch **IV:** Interfacing C++ with other languages (FFI) **V:** System calls and OS interaction **VI:** Memory-mapped I/O **VII:** Performance profiling and benchmarking **VIII:** Optimization techniques for HPC

Learning Outcomes **I:** Explain how a core language construct, such as data abstractions and control abstractions, is executed [Usage] **II:** Explain how programming language implementations typically organize memory into global data, text, heap, and stack sections and how features such as recursion and memory management map to this memory model [Assessment] **III:** Explain why memory leaks and dangling pointer problems occur, and what can be done by a programmer to avoid/fix them [Usage] **IV:** Profile and optimize C++ programs for performance [Usage]. **V:** Explain how the compiler implements runtime polymorphism via vtables [Assessment]. **VI:** Interface C++ code with C libraries [Usage].

Bibliography: [Stroustrup, 2013, Williams, 2019]

5.7 Advanced Programming Constructs, 6 hr(s)

Competences: AG-C09,AG-C12

Topics: **I:** Object-oriented abstractions: multiple inheritance, mixins, traits, multimethods **II:** Creational, structural, and behavioral patterns **III:** Thread-safe singleton patterns **IV:** Modern C++ design idioms **V:** Dependency injection and inversion of control

Learning Outcomes **I:** Use various advanced programming constructs and idioms correctly [Usage] **II:** Discuss how various advanced programming constructs aim to improve program structure, software quality, and programmer productivity [Assessment] **III:** Discuss how various advanced programming constructs interact with the definition and implementation of other language features [Familiarity] **IV:** Implement thread-safe design patterns [Usage]. **V:** Evaluate architectural trade-offs in C++ systems [Assessment].

Bibliography: [Gamma et al., 1994, Stroustrup, 2013]

5.8 Advanced Programming Constructs, 6 hr(s)

Competences: AG-C09,AG-C12

Topics: **I:** Control abstractions: exception handling, continuations, monads. **II:** Compiler optimizations and flags **III:** Benchmarking methodologies **IV:** Cache-friendly data structures **V:** Branch prediction and alignment **VI:** Reducing compilation time **VII:** Continuous integration for performance testing

Learning Outcomes **I:** Use various advanced programming constructs and idioms correctly [Usage] **II:** Discuss how various advanced programming constructs aim to improve program structure, software quality, and programmer productivity [Assessment] **III:** Discuss how various advanced programming constructs interact with the definition and implementation of other language features [Familiarity] **IV:** Apply optimization techniques to improve program performance [Usage]. **V:** Interpret benchmark results to guide optimization decisions [Usage].

Bibliography: [Meyers, 2014, Stroustrup, 2013]

5.9 Version Control and CI/CD, 4 hr(s)

Competences: AG-C09,AG-C12

Topics: **I:** Software process automation: (a) Build systems - the value of fast, hermetic, reproducible builds, compare/contrast approaches to building a project. (b) Continuous Integration (CI) - the use of automation and automated tests to do preliminary validation that the current head/trunk revision builds and passes (basic) tests. (c) Continuous Deployment (CD) - the use of automation to automatically release every change that passes the automated tests to the production environment, ensuring frequent and reliable deliveries. (d) Dependency management - updating external/upstream dependencies, package management, SemVer. **II:** Build systems for C++: CMake, the de facto industry standard **III:** Modern alternatives: xmake, Meson, and Bazel – comparing declarative approaches versus Lua/Python-based ones **IV:** Organizing multi-file projects: translation units, headers, static and dynamic libraries **V:** Dependency management in C++ (vcpkg, Conan)

Learning Outcomes **I:** Understand the use of CI/CD systems as a ground-truth for the state of the team's shared code (build and test success) [Familiarity] **II:** Configure a multi-file C++ project using a declarative build system [Usage]. **III:** Compare CMake with at least one modern alternative (xmake, Meson, or Bazel) in terms of build speed and reproducibility [Assessment].

Bibliography: [Stroustrup, 2013, Stroustrup, 2022]

6. Bibliography

[Gamma et al., 1994] Gamma, E., Helm, R., Johnson, R., and Vlissides, J. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional, Reading, MA, 1st edition.

[Gregor, 2017] Gregor, D. (2017). *Advanced C++: Master the Technique of Template Metaprogramming*. Addison-Wesley Professional, Upper Saddle River, NJ, 1st edition.

[Josuttis, 2019] Josuttis, N. M. (2019). *C++17: The Complete Guide*. Pearson Education, Indianapolis, IN, 1st edition.

[Meyers, 2014] Meyers, S. (2014). *Effective Modern C++: 42 Specific Ways to Improve Your Use of C++11 and C++14*. O'Reilly Media, Sebastopol, CA, 1st edition.

[Stroustrup, 2013] Stroustrup, B. (2013). *The C++ Programming Language*. Addison-Wesley Professional, Upper Saddle River, NJ, 4th edition.

[Stroustrup, 2022] Stroustrup, B. (2022). *A Tour of C++*. Addison-Wesley Professional, Boston, MA, 3rd edition.

[Vandevoorde et al., 2017] Vandevoorde, D., Josuttis, N. M., and Gregor, D. (2017). *C++ Templates: The Complete Guide*. Addison-Wesley Professional, Upper Saddle River, NJ, 2nd edition.

[Williams, 2019] Williams, A. (2019). *C++ Concurrency in Action*. Manning Publications, Shelter Island, NY, 2nd edition.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

CS2B1 Platform Based Development (Mandatory)

1. General information

- **Semester**: 3rd Sem. **Credits**: 4
- **Hour of this course**: **Theory**: 2 hours; **Laboratory**: 4 hours;
- **Prerrequisites**:
  - CS112 Objects-oriented programming I (2nd Sem-Pg. 268)

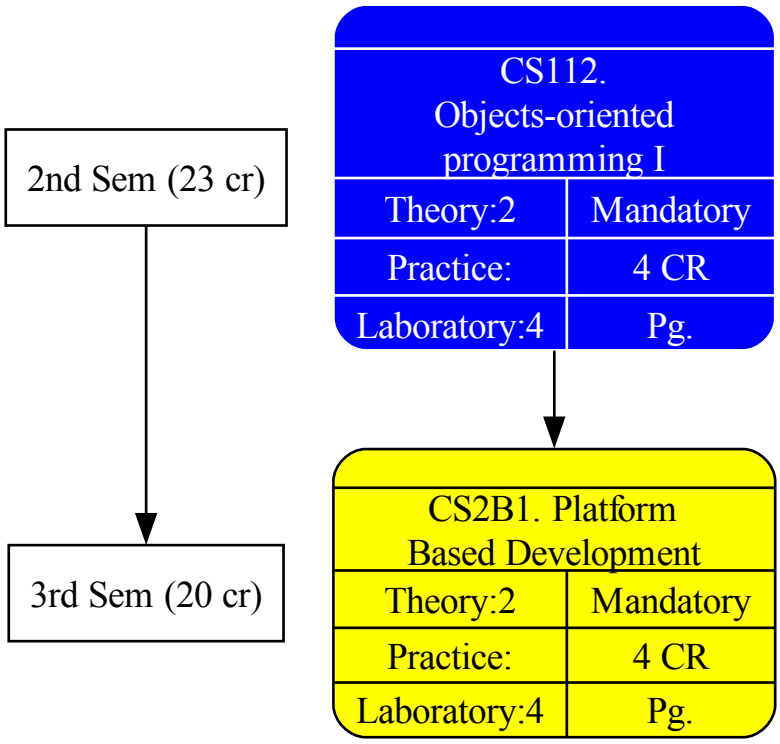

Figure 5.13: Connection Map. CS2B1 Platform Based Development

2. Course Rationale

Platform-based development focuses on the design and implementation of applications within specific computing environments. This course utilizes the Software Process and Design (SPD) framework to explore the unique constraints and capabilities of Web, Mobile, and Interactive platforms. Students will learn to apply shared architectural concerns and platform-specific APIs to build robust, scalable, and user-centric software solutions.

3. Goals of the Course

- Understand the shared aspects and unique constraints of modern platforms.
- Develop professional applications for Web and Mobile environments.
- Apply interactive computing principles to enhance user experience.
- Design software architectures that leverage platform-specific services.

4. General Competences (*Outcomes*)

AG-C04) Communication: Communicates effectively in complex computing activities. (Usage)

AG-C09) Design and Development of Solutions: Designs, implements, and evaluates solutions for complex computing problems. (Usage)

5. Content

5.1 Common Aspects: Platforms, APIs and Constraints, 5 hr(s)

Competences: AG-C04,AG-C09

Topics: I: Overview of development platforms (i.e., web, mobile, game, robotics, embedded, and interactive). (a) Input/sensors/control devices/haptic devices (b) Resource constraints i. Computational ii. Data storage iii. Memory iv. Communication (c) Requirements - security, uptime availability, fault tolerance Reliability Concepts, Reliability Engineering, Security Design and Controls Engineering, Threat Analysis and Security Engineering, Trusted Computing and Privacy Engineering (d) Output/actuators/haptic devices **II:** Programming via platform-specific Application Programming Interface (API) vs traditional application construction **III:** Programming under platform constraints and requirements (e.g., available development tools, development, security considerations) AI and Hardware Security, Access Control and Applications, Security Mindset and Principles

Learning Outcomes I: List the constraints of mobile programming [List]

Bibliography: [Fielding, 2000, Gasca, 2021]

5.2 Common Aspects: Platform Languages and Web Programming Patterns, 3 hr(s)

Competences: AG-C04,AG-C09

Topics: I: Overview of platform Languages (e.g., Python, Swift, Lua, Kotlin) **II:** Techniques for learning and mastering a platform-specific programming language

Learning Outcomes I: List the characteristics of scripting languages [List] **II:** Describe the three-tier model of web programming [Describe] **III:** Describe how the state is maintained in web programming [Describe]

Bibliography: [Fielding, 2000, Gasca, 2021]

5.3 Frontend, Frameworks and Web Standards, 12 hr(s)

Competences: AG-C04,AG-C09

Topics: I: Web programming languages (e.g., HTML5, JavaScript, Typescript, CSS) **II:** Web platforms, frameworks, or meta-frameworks (a) Cloud services (b) API, Web Components **III:** Software as a Service (SaaS) **IV:** Web standards such as document object model, accessibility Accessibility and Inclusive Design **V:** Security and Privacy Considerations Computer Crimes and Attacks, Security Laws, Policies and Responsibility

Learning Outcomes I: Compare and contrast web programming with general-purpose programming [Compare] **II:** Describe the differences between Software-as-a-Service (SaaS) and traditional software products [Describe] **III:** Discuss how web standards impact software development [Debate] **IV:** Evaluate an existing web application against current web standards [Evaluate]

Bibliography: [Flanagan, 2020, Duckett, 2014, Fielding, 2000]

5.4 Mobile Platforms: Development Fundamentals, 6 hr(s)

Competences: AG-C04,AG-C09

Topics: **I:** Development with (a) Mobile programming languages (b) Mobile programming environments **II:** Mobile platform constraints (a) User interface design Understanding the User: Individual goals and interactions with others (b) Security **III:** Access (a) Accessing data through APIs Query Construction (b) Designing API endpoints for mobile apps: pitfalls and design considerations (c) Network and web interfaces Data Modeling, Fundamentals of Networks and Communications

Learning Outcomes **I:** Compare mobile programming with general-purpose programming [Compare] **II:** Evaluate the pros and cons of native and cross-platform mobile application development [Evaluate]

Bibliography: [Deitel et al., 2017, Hellman, 2013]

5.5 Mobile Platforms: Architecture and Constraints, 3 hr(s)

Competences: AG-C04,AG-C09

Topics: **I:** Development (a) Native versus cross-platform development (b) Software design/architecture patterns for mobile applications API Design and Modeling, Principles and Software Architecture, Software Design Quality and Evaluation **II:** Mobile platform constraints (a) Responsive user interface design Accessibility and Inclusive Design (b) Heterogeneity and mobility of devices (c) Differences in user experiences (e.g., between mobile and web-based applications) (d) Power and performance tradeoff **III:** Specification and testing Software Development Practices, Performance Testing and Benchmarking, Testing Tools and Analysis, Verification and validation Concepts, Test Planning and Types, Advanced Testing Practices

Learning Outcomes **I:** Develop a location-aware mobile application with data API integration [Create] **II:** Build a sensor-driven mobile application capable of logging data on a remote server [Create] **III:** Create a communication app incorporating telephony and instant messaging [Create]

Bibliography: [Deitel et al., 2017, Hellman, 2013]

5.6 Mobile Platforms: Capabilities and Asynchronous Computing, 3 hr(s)

Competences: AG-C04,AG-C09

Topics: **I:** Mobile computing affordances (a) Location-aware applications (b) Sensor-driven computing (e.g., gyroscope, accelerometer, health data from a watch) (c) Telephony and instant messaging (d) Augmented reality Immersion **II:** Asynchronous computing Algorithms (a) Difference from traditional synchronous programming (b) Handling success via callbacks (c) Handling errors asynchronously (d) Testing asynchronous code and typical problems in testing

Learning Outcomes **I:** Develop a location-aware mobile application with data API integration [Create] **II:** Build a sensor-driven mobile application capable of logging data on a remote server [Create] **III:** Create a communication app incorporating telephony and instant messaging [Create]

Bibliography: [Deitel et al., 2017, Hellman, 2013]

5.7 SPD-SEP/Interactive, 10 hr(s)

Competences: AG-C04,AG-C09

Topics: **I:** Ethical guidelines when using AI models to assist in journalism and content creation **II:** Accountability for AI-generated outputs **III:** Behavior among prompt programmers and AI developers **IV:** Trust with the public when using AI models

Learning Outcomes I: Indicate a framework for accountability in AI model deployment, including clear documentation and attribution [Evaluate] **II:** Discuss ethical codes of conduct and professional standards relevant to prompt programming and AI development [Debate] **III:** Create communication plans and materials to educate the public about AI capabilities, limitations, and ethical safeguards [Create]

Bibliography: [Duckett, 2014, Deitel et al., 2017]

5.8 Interactive Data and Analytics Platforms, 6 hr(s)

Competences: AG-C04,AG-C09

Topics: I: Computational Notebook Platforms (e.g., Jupyter notebooks; Google Colab; R; SPSS; Observable) **II:** Cloud SQL and Data Analysis Platforms (e.g., BigQuery) Query Construction (a) Apache Spark (b) Data Visualizations Visualization **III:** Interactive Presentations Backed by Data **IV:** Design Tools Requiring Low-Latency Feedback Loops (a) Rendering tools (b) Graphic design tools

Learning Outcomes I: Analyze large datasets interactively [Analyze] **II:** Create a backing track for a musical performance, such as live coding [Create] **III:** Create compelling computational notebooks that construct a narrative for a given journalistic goal/story [Create] **IV:** Implement interactive code that uses a dataset and generates exploratory graphics [Implement] **V:** Contextualize the attributes of different data analysis styles, such as interactive vs engineered pipeline [Contextualize] **VI:** Write a program using a notebook computing platform (e.g., searching, sorting, or graph manipulation) [Write]

Bibliography: [Flanagan, 2020]

6. Bibliography

[Deitel et al., 2017] Deitel, P., Deitel, H., and Deitel, A. (2017). *Android How to Program*. Pearson, 3rd edition.

[Duckett, 2014] Duckett, J. (2014). *JavaScript and JQuery: Interactive Front-End Web Development*. Wiley.

[Fielding, 2000] Fielding, R. T. (2000). *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, University of California, Irvine.

[Flanagan, 2020] Flanagan, D. (2020). *JavaScript: The Definitive Guide*. O'Reilly Media, 7th edition.

[Gasca, 2021] Gasca, J. (2021). *Web Security for Developers*. No Starch Press.

[Hellman, 2013] Hellman, E. (2013). *Android Programming: Pushing the Limits*. Wiley.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

AI161 Applied AI (Mandatory)

1. General information

- **Semester**: 3rd Sem. **Credits**: 4
- **Hour of this course**: **Theory**: 2 hours; **Laboratory**: 4 hours;
- **Prerrequisites**:
  - BIC101 Introduction to Computing (1st Sem-Pg. 239)

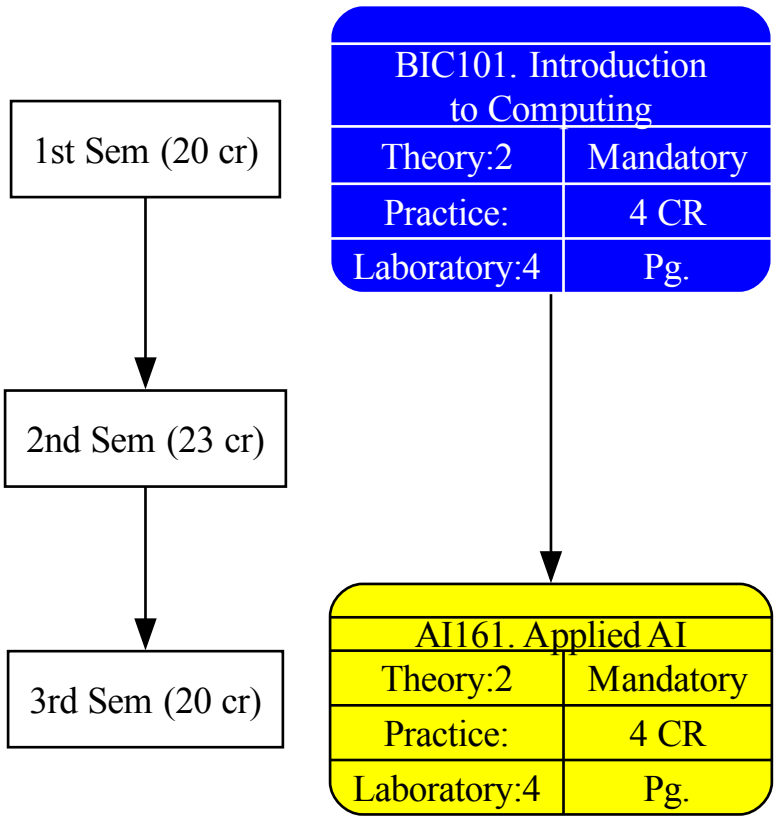

Figure 5.14: Connection Map. AI161 Applied AI

2. Course Rationale

This course provides a practical introduction to Artificial Intelligence (AI) for students from all scientific and engineering disciplines. Focused on developing AI literacy and practical skills, it covers fundamental concepts, modern AI tools (including Western and Chinese platforms), and responsible usage. Students will learn to effectively interact with diverse AI systems, write quality prompts, and apply AI solutions to problems across various domains while understanding ethical implications and cultural contexts of AI deployment.

3. Goals of the Course

- Develop comprehensive AI literacy by understanding fundamental concepts, capabilities, and limitations of modern AI systems across different platforms and cultures.
- Master effective prompt engineering techniques and interaction patterns with various AI tools (Western: ChatGPT, Claude, Gemini; Chinese: DeepSeek, Kimi, ERNIE).
- Apply diverse AI tools to solve practical problems in scientific, engineering, and academic contexts while maintaining critical evaluation of outputs across platforms.
- Understand ethical considerations, biases, cultural contexts, and responsible usage of AI technologies in global professional and academic settings.
- Develop workflows that integrate multiple AI tools to enhance productivity and problem-solving capabilities while understanding regional strengths and specializations.

4. General Competences (*Outcomes*)

AG-C01) The Professional and the World: Analyzes and evaluates the impact of solutions to complex computing problems on the sustainable development of society. (Familiarity)

AG-C11) Use of Tools: Applies modern computing tools in problem solving. (Usage)

5. Content

5.1 AI Fundamentals and Global Landscape, 16 hr(s)

Competences: AG-C01,AG-C11

Topics: **I:** What is AI? Definitions, history, and current global landscape. **II:** Types of AI systems: chatbots, image generators, research assistants across regions. **III:** Western AI ecosystems: OpenAI, Anthropic, Google, Microsoft. **IV:** Chinese AI ecosystems: DeepSeek, Kimi, ERNIE, Zhipu AI, Baidu. **V:** AI capabilities and limitations: comparative analysis across platforms. **VI:** Digital literacy in the AI era: critical thinking about diverse AI outputs.

Learning Outcomes **I:** Apply fundamental AI knowledge to identify system types and their capabilities [Usage]. **II:** Analyze global intelligent systems and their distinctive characteristics [Assessment]. **III:** Critically evaluate AI outputs across different cultural contexts [Assessment].

Bibliography: [Ng, 2019, Mollick and Mollick, 2024, UNESCO, 2023, Institute, 2024]

5.2 Effective AI Interaction and Cross-Platform Prompt Engineering, 20 hr(s)

Competences: AG-C01,AG-C11

Topics: **I:** Principles of effective prompt writing: clarity, context, constraints across platforms. **II:** Prompt patterns: persona, template, chain-of-thought, few-shot for different AI systems. **III:** Western tools: specific techniques for ChatGPT, Claude, Gemini, Copilot. **IV:** Chinese tools: specific features and best practices for DeepSeek, Kimi, ERNIE. **V:** Cross-platform strategies: leveraging different AI strengths for complex tasks. **VI:** Iterative refinement: how to improve prompts based on outputs from diverse systems. **VII:** Practical sessions: comparative prompt writing workshops across platforms.

Learning Outcomes **I:** Design effective prompt solutions for diverse AI systems [Assessment]. **II:** Develop cross-platform workflows that integrate different AI tools [Usage]. **III:** Implement iterative refinement strategies to optimize AI outputs [Usage].

Bibliography: [White et al., 2023, Prompt Engineering Institute, 2024, Microsoft, 2024, DeepSeek, 2024]

5.3 AI Applications Across Disciplines and Platforms, 16 hr(s)

Competences: AG-C01,AG-C11

Topics: **I:** AI for research: literature review, data analysis using multiple AI assistants. **II:** AI for writing: academic papers, reports with cross-platform verification. **III:** AI for problem-solving: scientific calculations, engineering design with specialized tools. **IV:** AI for creativity: brainstorming, concept development across cultural contexts. **V:** Platform-specific strengths: when to use Western vs. Chinese AI tools. **VI:** Discipline-specific workshops: tailored applications using diverse AI ecosystems. **VII:** Case studies: real-world applications in scientific research across regions.

Learning Outcomes **I:** Apply AI fundamentals to solve discipline-specific problems [Usage]. **II:** Design solutions that integrate multiple intelligent systems for complex tasks [Assessment]. **III:** Develop practical applications using Western and Chinese AI tools [Usage]. **IV:** Implement workflows that leverage specific strengths of each platform [Assessment].

Bibliography: [Mollick, 2023, Cloud, 2024b, Cloud, 2024a, OpenAI, 2024]

5.4 Global AI Ethics and Responsible Usage, 12 hr(s)

Competences: AG-C01,AG-C11

Topics: **I:** Understanding AI biases: Western and Eastern cultural perspectives. **II:** Ethical frameworks: comparing EU AI Act, Chinese regulations, and global standards. **III:** Academic integrity: proper citation and AI usage in coursework across platforms. **IV:** Privacy and data security: regional differences in AI data handling. **V:** Cultural sensitivity: navigating AI outputs in global contexts. **VI:** Environmental impact: sustainability considerations of different AI systems. **VII:** Developing personal guidelines for ethical AI usage in international settings. **VIII:** Case studies: ethical dilemmas in AI deployment across regions.

Learning Outcomes **I:** Demonstrate professional responsibility in using AI technologies [Assessment]. **II:** Apply professional ethical principles in global AI contexts [Usage]. **III:** Develop ethical guidelines to mitigate biases in AI systems [Assessment]. **IV:** Evaluate ethical implications of AI deployment across different cultures [Usage].

Bibliography: [UNESCO, 2023, European Union, 2024, Bender et al., 2021, Chinese Academy of Cyberspace Studies, 2024, Tencent AI Lab, 2023]

6. Bibliography

[Bender et al., 2021] Bender, E. M., Gebru, T., McMillan-Major, A., and Shmitchell, S. (2021). On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency (FAccT '21)*, pages 610–623. Association for Computing Machinery. Foundational paper on AI ethics with ongoing relevance.

[Chinese Academy of Cyberspace Studies, 2024] Chinese Academy of Cyberspace Studies (2024). Ai governance and ethics in china. Technical report, Cyberspace Administration of China. Official Chinese perspective on AI governance and ethical standards.

[Cloud, 2024a] Cloud, A. (2024a). AI Ethics and best practices in the chinese context. Chinese perspective on AI ethics and implementation guidelines.

[Cloud, 2024b] Cloud, G. (2024b). Introduction to responsible ai. Google's updated framework for responsible AI development and use.

[DeepSeek, 2024] DeepSeek (2024). Deepseek model documentation and best practices. Official documentation for DeepSeek AI models and usage guidelines.

[European Union, 2024] European Union (2024). AI Act: Regulatory framework. Technical report, European Commission. Comprehensive AI regulation framework effective in 2024.

[Institute, 2024] Institute, T. R. (2024). Ai development in china: Current status and future trends. Technical report, Tencent Research Institute. Comprehensive analysis of Chinese AI ecosystem development.

[Microsoft, 2024] Microsoft (2024). Prompt crafting for ai systems. Official Microsoft prompt engineering guidance updated for 2024.

[Mollick, 2023] Mollick, E. (2023). Chatgpt and how ai disrupts industries. *Harvard Business Review*, 101(5):42–51. Analysis of AI's practical impact across global sectors.

[Mollick and Mollick, 2024] Mollick, E. and Mollick, L. (2024). *Co-Intelligence: Living and Working with AI*. Portfolio. Practical guide for human-AI collaboration.

[Ng, 2019] Ng, A. (2019). AI for everyone. Technical report, DeepLearning.AI. Continuously updated online course, available on Coursera.

[OpenAI, 2024] OpenAI (2024). Best practices for prompt engineering. Updated official prompt engineering guidelines from OpenAI.

[Prompt Engineering Institute, 2024] Prompt Engineering Institute (2024). Prompt engineering guide. `https://www.promptingguide.ai/`. Recurso integral de ingeniería de prompts actualizado continuamente. Accedido: 15 de enero de 2024.

[Tencent AI Lab, 2023] Tencent AI Lab (2023). Responsible ai practices in the chinese technology industry. Technical report, Tencent AI Lab. Industry perspective on AI ethics from a leading Chinese technology company.

[UNESCO, 2023] UNESCO (2023). AI and education: Guidance for policy-makers. Technical report, UNESCO. Latest guidance on AI literacy and ethical implementation in education.

[White et al., 2023] White, J., Fu, Q., Hays, S., Sandborn, M., Olea, C., Gilbert, H., Elnashar, A., Spencer-Smith, J., and Schmidt, D. C. (2023). A prompt pattern catalog to enhance prompt engineering with chatgpt. Peer-reviewed research on prompt engineering patterns.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

BMA104 Advanced Differential and Integral Calculus (Mandatory)

1. General information

- **Semester**: 3rd Sem. **Credits**: 5
- **Hour of this course**: **Theory**: 4 hours; **Practice**: 2 hours;
- **Prerrequisites**:
  - BMA102 Differential Calculus (1st Sem-Pg. 246)

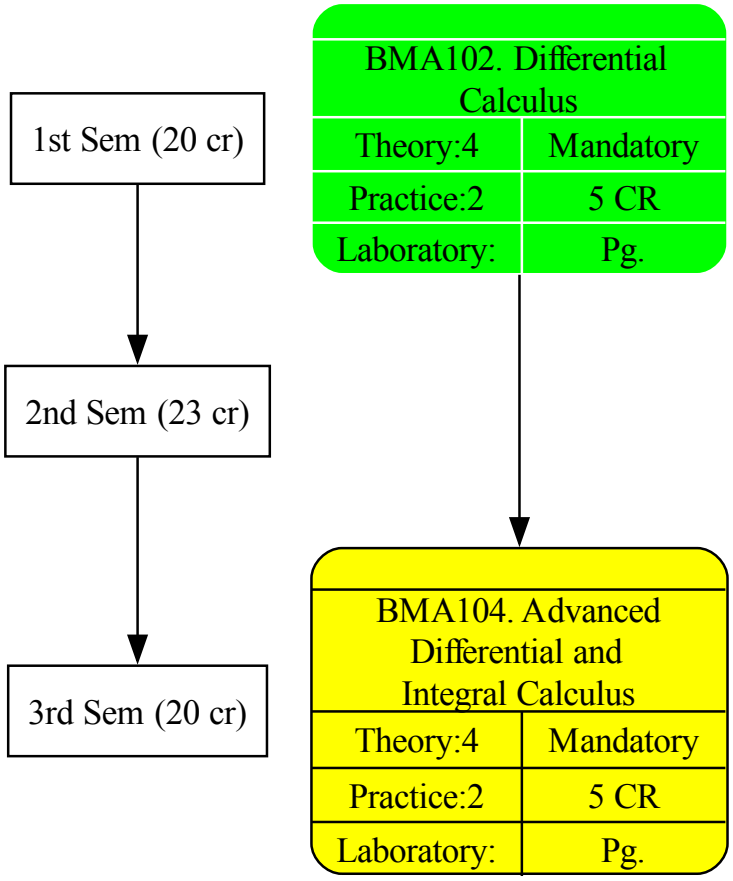

Figure 5.15: Connection Map. BMA104 Advanced Differential and Integral Calculus

2. Course Rationale

Advanced differential and integral calculus provides the rigorous foundations for understanding multidimensional phenomena in engineering and computer science. This course covers vector functions, multivariable calculus, differentiable transformations, multiple integrals, and integral calculus on manifolds, including the fundamental theorems of Green, Stokes, and Gauss. These tools are essential for modeling complex systems, optimizing multivariable functions, and solving problems in computer graphics, computer vision, and physical simulations.

3. Goals of the Course

- Study vector functions of a real variable and the differential geometry of space curves.
- Apply differential calculus to functions of several variables: partial derivatives, gradient, tangent plane, optimization, and Taylor's theorem.
- Analyze differentiable transformations between vector spaces using the Jacobian matrix and the inverse and implicit function theorems.
- Evaluate multiple integrals in different coordinate systems and compute areas, volumes, and centers of mass.
- Compute line integrals and apply Green's Theorem and the concept of conservative fields.
- Compute surface integrals and apply Stokes' Theorem and the Divergence Theorem.

4. General Competences (*Outcomes*)

AG-C07) Computing Knowledge: Applies knowledge of mathematics, science, and computing. (Assessment)

5. Content

5.1 Vector Functions of a Real Variable, 14 hr(s)

Competences: AG-C07

Topics: **I:** Definition, domain, and range of vector functions of a real variable **II:** Limits, continuity, and derivatives of vector functions **III:** Arc length and natural parametrization of a curve **IV:** Tangent, normal, and binormal vectors; moving Frenet frame **V:** Curvature, torsion, and Frenet-Serret formulas

Learning Outcomes **I:** Identify limits, continuity, and derivatives of vector functions of a real variable [Familiarity] **II:** Compute tangent, normal, and binormal vectors and reparametrize a curve by arc length [Usage] **III:** Calculate curvature and torsion using the Frenet-Serret formulas [Assessment]

Bibliography: [Apostol, 1997, Marsden and Tromba, 2011]

5.2 Multivariable Differential Calculus, 18 hr(s)

Competences: AG-C07

Topics: **I:** Limits and continuity of functions of several variables in $\mathbb{R}^n$ **II:** Partial derivatives, directional derivatives, and the gradient vector **III:** Tangent plane, total differential, and linear approximation **IV:** Extrema of multivariable functions and Lagrange multipliers **V:** Taylor's theorem for functions of several variables

Learning Outcomes **I:** Evaluate limits and determine continuity of functions of several variables [Familiarity] **II:** Compute gradients, directional derivatives, and tangent planes for multivariable functions [Usage] **III:** Apply Lagrange multipliers to solve constrained optimization problems in computing and data science [Usage] **IV:** Expand functions of several variables using Taylor's theorem and estimate error bounds [Assessment]

Bibliography: [Apostol, 1997, Marsden and Tromba, 2011]

5.3 Multivariable Transformations, 12 hr(s)

Competences: AG-C07

Topics: **I:** Transformations from $\mathbb{R}^n$ to $\mathbb{R}^m$ and their differentiability **II:** Jacobian matrix and Jacobian determinant **III:** Change of variables in polar, cylindrical, and spherical coordinates **IV:** Inverse function theorem **V:** Implicit function theorem in multivariable contexts

Learning Outcomes **I:** Describe transformations between coordinate systems using the Jacobian matrix [Familiarity] **II:** Compute the Jacobian and apply change of variables to evaluate multiple integrals [Usage] **III:** Apply the inverse and implicit function theorems to analyze local invertibility [Assessment]

Bibliography: [Apostol, 1997, Marsden and Tromba, 2011]

5.4 Multiple Integrals, 18 hr(s)

Competences: AG-C07

Topics: **I:** Double integrals over rectangles and general regions; iterated integrals (Fubini's theorem) **II:** Double integrals in polar coordinates and change-of-variables formula **III:** Triple integrals in Cartesian coordinates and applications to volume and mass **IV:** Triple integrals in cylindrical and spherical coordinates **V:** Applications: area, volume, centers of mass, and moments of inertia

Learning Outcomes **I:** Evaluate double and triple integrals over general regions using iterated integrals [Usage] **II:** Apply coordinate changes in polar, cylindrical, and spherical systems to simplify multiple integrals [Usage] **III:** Compute areas, volumes, and centers of mass using multiple integrals [Assessment]

Bibliography: [Apostol, 1997, Marsden and Tromba, 2011]

5.5 Line Integrals and Vector Fields, 10 hr(s)

Competences: AG-C07

Topics: **I:** Vector fields: definition, visualization, divergence, and curl **II:** Line integrals of scalar functions: arc length and mass **III:** Line integrals of vector fields: work and circulation **IV:** Conservative fields, potential functions, and path independence **V:** Green's Theorem and its applications in the plane

Learning Outcomes **I:** Compute scalar and vector line integrals over oriented curves [Usage] **II:** Determine whether a vector field is conservative and construct its potential function [Usage] **III:** Apply Green's Theorem to evaluate line integrals via double integrals [Assessment]

Bibliography: [Apostol, 1997, Marsden and Tromba, 2011]

5.6 Surface Integrals and Fundamental Theorems, 12 hr(s)

Competences: AG-C07

Topics: **I:** Parametric surfaces, tangent plane, and surface area **II:** Surface integrals of scalar functions **III:** Flux integrals: surface integrals of vector fields **IV:** Stokes' Theorem: relation between line integrals and surface integrals **V:** Divergence Theorem (Gauss): relation between flux and volume integral

Learning Outcomes **I:** Parametrize surfaces and evaluate scalar and flux surface integrals [Usage] **II:** Apply Stokes' and Divergence Theorems to reduce complex integrals [Assessment] **III:** Relate local differential properties (curl, divergence) to global behavior via integral theorems [Assessment]

Bibliography: [Apostol, 1997, Marsden and Tromba, 2011]

6. Bibliography

[Apostol, 1997] Apostol, T. M. (1997). *Calculus, Vol. II: Multi-Variable Calculus and Linear Algebra with Applications*. John Wiley & Sons, 2nd edition.

[Marsden and Tromba, 2011] Marsden, J. E. and Tromba, A. (2011). *Vector Calculus*. W. H. Freeman and Company, 6th edition.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

STA251 Probability Calculation (Mandatory)

1. General information

- **Semester**: 3rd Sem. **Credits**: 3
- **Hour of this course**: **Theory**: 2 hours; **Practice**: 2 hours;
- **Prerrequisites**:
  - BMA102 Differential Calculus (1st Sem-Pg. 246)

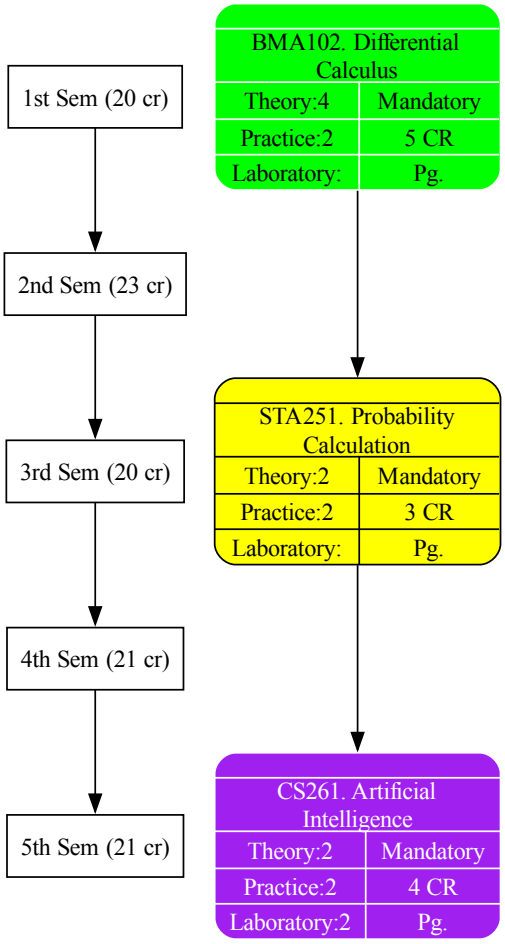

Figure 5.16: Connection Map. STA251 Probability Calculation

2. Course Rationale

Probability and statistics provide the mathematical foundation for reasoning under uncertainty, analyzing data, and building predictive models – skills that are essential across computing, artificial intelligence, computational mathematics, and engineering alike. This course develops a rigorous understanding of probability theory and its use in describing random phenomena, together with the fundamentals of descriptive statistics, statistical inference (estimation, hypothesis testing, and Bayesian reasoning), and regression modeling, so that students can summarize and analyze data, quantify uncertainty, and draw sound quantitative conclusions in their own field of application.

3. Goals of the Course

- Understand the fundamentals of probability theory and descriptive statistics.
- Apply probability distributions and statistical inference methods to analyze data and quantify uncertainty.
- Develop skills to model random phenomena and build predictive statistical models such as linear regression.

4. General Competences (*Outcomes*)

AG-C07) Computing Knowledge: Applies knowledge of mathematics, science, and computing. (Familiarity)

5. Content

5.1 Numerical Summaries: Location, Spread, and Shape, 4 hr(s)

Competences: AG-C07

Topics: **I:** Measures of location: mean, median, mode, trimmed mean, and weighted mean **II:** Measures of spread: variance, standard deviation, IQR, MAD, and range **III:** Measures of shape: skewness and kurtosis; quantiles and percentiles **IV:** Robust summary statistics: resistance to outliers and heavy-tailed distributions

Learning Outcomes **I:** Calculate and interpret measures of location, spread, and shape for a univariate dataset [Usage] **II:** Select appropriate summary statistics based on the distributional properties of the data [Usage] **III:** Compare classical and robust summary statistics and justify the choice for a given dataset [Assessment]

Bibliography: [Devore, 2016]

5.2 Sample Spaces and Events, 4 hr(s)

Competences: AG-C07

Topics: **I:** Random experiments, outcomes, and sample spaces: finite, countable, and uncountable **II:** Set-theoretic operations on events: union, intersection, complement, and De Morgan's laws **III:** Sigma-algebras and measurable spaces as the foundation for rigorous probability **IV:** Borel sets and the Borel sigma-algebra on the real line

Learning Outcomes **I:** Define sample spaces, events, and sigma-algebras for a given random experiment [Familiarity] **II:** Apply set-theoretic operations to compute probabilities of compound events [Usage] **III:** Construct appropriate sigma-algebras for measurable spaces arising in statistical problems [Assessment]

Bibliography: [Ross, 2014, Devore, 2016]

5.3 Probability Axioms and Properties, 4 hr(s)

Competences: AG-C07

Topics: **I:** Kolmogorov's three axioms of probability and their consequences **II:** Derived properties: complement rule, addition rule, monotonicity, and continuity of probability **III:** Classical, frequentist, and subjective (Bayesian) interpretations of probability **IV:** Counting methods: permutations, combinations, and the multiplication principle

Learning Outcomes **I:** State Kolmogorov's axioms and derive fundamental probability properties [Familiarity] **II:** Calculate probabilities of events using the addition rule and complement rule [Usage] **III:** Apply counting methods to compute probabilities in finite sample spaces [Usage]

Bibliography: [Ross, 2014, Devore, 2016]

5.4 Conditional Probability and Independence, 4 hr(s)

Competences: AG-C07

Topics: **I:** Definition and interpretation of conditional probability P(A|B) and its properties **II:** Bayes' theorem and the law of total probability: derivation and applications **III:** Independence of events: pairwise and mutual independence **IV:** Conditional independence and its role in probabilistic graphical models

Learning Outcomes **I:** Explain conditional probability and independence and their relationship [Familiarity] **II:** Apply Bayes' theorem and the law of total probability to solve probability problems [Usage] **III:** Analyze whether events are independent or conditionally independent in a given probability model [Assessment]

Bibliography: [Ross, 2014, Devore, 2016]

5.5 Discrete Random Variables and Distributions, 4 hr(s)

Competences: AG-C07

Topics: **I:** Probability mass functions (PMF) and cumulative distribution functions (CDF) for discrete random variables **II:** Key discrete distributions: Bernoulli, Binomial, Geometric, Negative Binomial, Poisson, Hypergeometric **III:** Properties of discrete distributions: mean, variance, skewness, and kurtosis **IV:** Poisson approximation to the Binomial and limiting distribution arguments

Learning Outcomes **I:** Identify and describe the key discrete probability distributions and their parameters [Familiarity] **II:** Calculate probabilities, means, and variances using PMFs of standard discrete distributions [Usage] **III:** Select an appropriate discrete distribution to model a given random phenomenon and justify the choice [Assessment]

Bibliography: [Ross, 2014, Devore, 2016]

5.6 Continuous Random Variables and Distributions, 4 hr(s)

Competences: AG-C07

Topics: **I:** Probability density functions (PDF) and CDFs for continuous random variables **II:** Key continuous distributions: Uniform, Normal, Exponential, Gamma, Beta, Chi-squared, t, F **III:** The Normal distribution: properties, standard normal, and its central role in statistical inference **IV:** The exponential family of distributions: canonical form and sufficient statistics

Learning Outcomes **I:** Describe the properties and parameters of key continuous probability distributions [Familiarity] **II:** Calculate probabilities and quantiles for standard continuous distributions [Usage] **III:** Identify distributions belonging to the exponential family and derive their sufficient statistics [Assessment]

Bibliography: [Ross, 2014, Devore, 2016]

5.7 Joint, Marginal, and Conditional Distributions, 4 hr(s)

Competences: AG-C07

Topics: **I:** Joint PMFs and PDFs for two or more random variables **II:** Marginal and conditional distributions derived from joint distributions **III:** Covariance, correlation, and their properties; independence implies zero covariance (converse not always true) **IV:** The bivariate normal distribution and its conditional distributions

Learning Outcomes **I:** Derive marginal and conditional distributions from a given joint distribution [Usage] **II:** Calculate covariance and correlation between two random variables [Usage] **III:** Analyze the conditional structure of the bivariate normal distribution [Assessment]

Bibliography: [Ross, 2014, Devore, 2016]

5.8 Limit Theorems: LLN and CLT, 4 hr(s)

Competences: AG-C07

Topics: **I:** Weak law of large numbers: convergence in probability and its implications for estimation **II:** Central limit theorem: statement, conditions, and role in justifying normal approximations **III:** Applications of the CLT: normal approximation to the binomial and sampling distributions **IV:** Strong law of large numbers: almost sure convergence **V:** The delta method: asymptotic distribution of smooth functions of estimators

Learning Outcomes **I:** State the LLN and CLT and explain the conditions under which they hold [Familiarity] **II:** Apply the CLT to derive normal approximations and sampling distributions [Usage] **III:** Apply the delta method to derive asymptotic distributions of nonlinear estimators [Assessment]

Bibliography: [Ross, 2014, Devore, 2016]

5.9 Point Estimation, 4 hr(s)

Competences: AG-C07

Topics: **I:** Desirable properties: unbiasedness, consistency, efficiency, and sufficiency **II:** Method of moments estimation: derivation and asymptotic properties **III:** Maximum likelihood estimation: derivation, properties, and invariance principle **IV:** Cramer-Rao lower bound and the concept of Fisher information

Learning Outcomes **I:** Describe the key properties of estimators and explain the trade-offs between them [Familiarity] **II:** Derive method of moments and MLE estimators for standard distributions [Usage] **III:** Evaluate the efficiency of an estimator using the Cramer-Rao lower bound [Assessment]

Bibliography: [Devore, 2016]

5.10 Interval Estimation and Confidence Intervals, 4 hr(s)

Competences: AG-C07

Topics: **I:** Constructing confidence intervals using pivotal quantities and exact distributions **II:** Correct interpretation of confidence intervals: coverage probability and common misconceptions **III:** Large-sample confidence intervals based on asymptotic normality **IV:** Duality between confidence intervals and hypothesis tests

Learning Outcomes **I:** Construct confidence intervals for means, proportions, and variances using exact and large-sample methods [Usage] **II:** Interpret confidence intervals correctly and identify common misinterpretations [Usage] **III:** Explain the duality between confidence intervals and hypothesis tests and use it to draw inferences [Assessment]

Bibliography: [Devore, 2016]

5.11 Hypothesis Testing Foundations, 4 hr(s)

Competences: AG-C07

Topics: **I:** Neyman-Pearson framework: null/alternative hypotheses, rejection regions, Type I and II errors **II:** p-values: definition, correct interpretation, and common misuse; power and sample size **III:** Common parametric tests: z-test, t-test, F-test, chi-squared test **IV:** Neyman-Pearson lemma and uniformly most powerful tests

Learning Outcomes **I:** Explain the logic of hypothesis testing, Type I and II errors, and the meaning of a p-value [Familiarity] **II:** Apply standard parametric tests and interpret the results in context [Usage] **III:** Derive the most powerful test for a simple hypothesis using the Neyman-Pearson lemma [Assessment]

Bibliography: [Devore, 2016]

5.12 Bayesian Inference, 4 hr(s)

Competences: AG-C07

Topics: **I:** Bayesian framework: prior, likelihood, posterior, and the role of Bayes' theorem **II:** Prior distributions: informative, non-informative, and conjugate priors **III:** Posterior inference: point estimates, credible intervals, and posterior predictive distributions **IV:** Bayesian model comparison: Bayes factors and posterior model probabilities

Learning Outcomes **I:** Explain the Bayesian approach to inference and contrast it with the frequentist approach [Familiarity] **II:** Derive the posterior distribution for standard conjugate models [Usage] **III:** Apply Bayes factors to compare statistical models and interpret the evidence [Assessment]

Bibliography: [Ross, 2014, Devore, 2016]

5.13 Simple Linear Regression, 4 hr(s)

Competences: AG-C07

Topics: **I:** Ordinary least squares estimation: derivation of coefficients and their properties **II:** Inference for regression coefficients: t-tests, confidence intervals, and p-values **III:** Coefficient of determination R-squared and analysis of variance decomposition **IV:** Gauss-Markov theorem: BLUE property of OLS under classical assumptions

Learning Outcomes **I:** Fit a simple linear regression model and interpret the estimated coefficients [Usage] **II:** Conduct hypothesis tests and construct confidence intervals for regression parameters [Usage] **III:** Verify the Gauss-Markov assumptions and assess their consequences when violated [Assessment]

Bibliography: [Devore, 2016]

6. Bibliography

[Devore, 2016] Devore, J. L. (2016). *Probability and Statistics for Engineering and the Sciences*. Cengage Learning.

[Ross, 2014] Ross, S. M. (2014). *A First Course in Probability*. Pearson.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

BEI201 Technical and professional English III (Elective)

1. General information

- **Semester**: 3rd Sem. **Credits**: 2
- **Hour of this course**: **Practice**: 4 hours;
- **Prerrequisites**:
  - BEI102 Technical and professional English II (2nd Sem-Pg. 283)

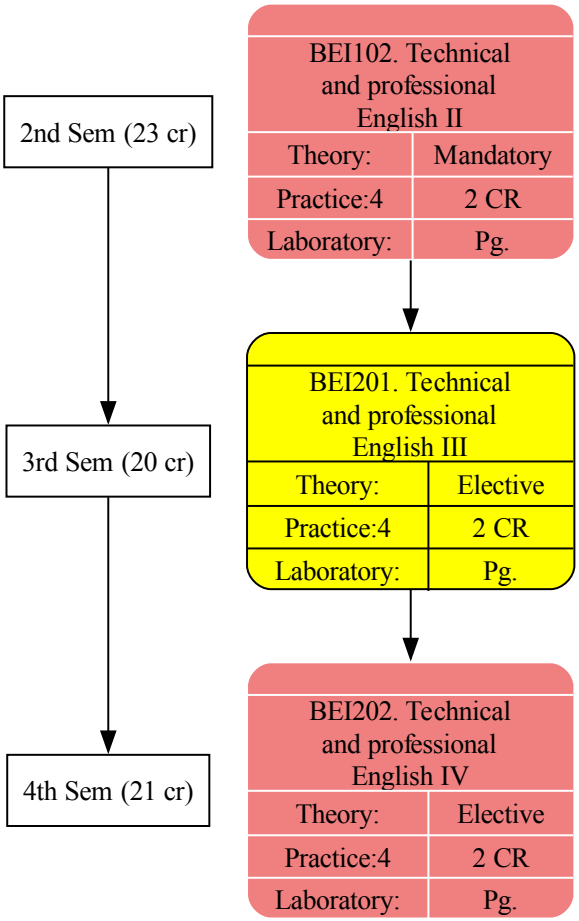

Figure 5.17: Connection Map. BEI201 Technical and professional English III

2. Course Rationale

At this level, students engage with English as the primary medium for accessing and producing technical content in computing, engineering, and the sciences. Grammar structures are applied to reading scientific articles, writing technical reports, describing engineering processes, and participating in professional discussions. The course bridges everyday English fluency with the demands of technical communication.

3. Goals of the Course

- Train students to read, understand, and produce technical communication in English.
- Provide linguistic tools for connecting ideas in descriptive and analytical writing about computing and engineering.

4. General Competences (*Outcomes*)

AG-C03) Individual and Team Work: Performs effectively as an individual and as a member or leader in diverse teams. (Familiarity)

AG-C04) Communication: Communicates effectively in complex computing activities. (Usage)

5. Content

5.1 Technical Communication, 12 hr(s)

Competences: AG-C03,AG-C04

Topics: **I:** Review of Tenses: Present, Past, and Future in technical documentation. **II:** Interrogative sentences with complex Wh- questions for system analysis. **III:** Words with more than one meaning in technical contexts (Polysemy). **IV:** Expressions for describing professional activities and research interests.

Learning Outcomes **I:** By the end of the unit, students use various tenses to exchange detailed technical information and formulate precise questions about computing systems and engineering processes.

Bibliography: [**?** line:45 , **?** line:45 , **?** line:45]

5.2 Comparing Architectures, 12 hr(s)

Competences: AG-C03,AG-C04

Topics: **I:** Use of "What's it like?": Describing system characteristics and interfaces. **II:** Advanced comparative and superlative adjectives in benchmark analysis. **III:** Collocations in computing: high-performance, low-level, fault-tolerant. **IV:** Reading: Architecture comparison reports (RISC vs. CISC, SQL vs. NoSQL).

Learning Outcomes **I:** Students analyze and describe differences between computing architectures, frameworks, and systems using varied comparative structures with technical precision.

Bibliography: [**?** line:58 , **?** line:58 , **?** line:58]

5.3 Then and Now in Computing, 12 hr(s)

Competences: AG-C03,AG-C04

Topics: **I:** Present Perfect vs. Simple Past: Contrasting historic events with current state. **II:** Time expressions: for, since, ever, never applied to technology evolution. **III:** Adverbs of time and manner in scientific narrative. **IV:** Vocabulary: Generations of technology, legacy systems, current platforms.

Learning Outcomes **I:** Students differentiate and correctly apply the present perfect and the simple past to narrate the evolution of computing technologies and contrast historical achievements with the current state of the field.

Bibliography: [**?** line:71 , **?** line:71 , **?** line:71]

5.4 Engineering Standards and Protocols, 12 hr(s)

Competences: AG-C03,AG-C04

Topics: **I:** Modals of obligation and permission: Have to, Must, Can in professional norms. **II:** Giving technical recommendations: Should, Ought to — code reviews and design. **III:** Vocabulary: IEEE/ACM standards, coding conventions, security policies. **IV:** Formal writing: Technical memos and engineering recommendations.

Learning Outcomes **I:** Students express engineering rules, standards, and professional recommendations using appropriate modal verbs in formal technical writing and peer discussion.

Bibliography: [? line:84, ? line:84, ? line:84]

5.5 How Systems Are Built, 12 hr(s)

Competences: AG-C03,AG-C04

Topics: **I:** Passive Voice: Present and Simple Past to describe engineering processes. **II:** Vocabulary: Manufacturing, deployment, integration, and testing. **III:** Expressions for giving technical opinions and discussing design trade-offs. **IV:** Writing: Process description for a software or hardware system.

Learning Outcomes **I:** Students use the passive voice to describe how systems are designed, built, tested, and deployed, participating in technical discussions about engineering processes.

Bibliography: [? line:97, ? line:97, ? line:97]

5.6 Debugging and Analysis, 12 hr(s)

Competences: AG-C03,AG-C04

Topics: **I:** Past Continuous vs. Simple Past in incident and failure reports. **II:** Narrative and sequence connectors: while, when, suddenly, as a result. **III:** Vocabulary: Bugs, exceptions, crashes, logs, root cause analysis. **IV:** Writing: Incident report for a software or network failure.

Learning Outcomes **I:** Students narrate debugging sessions, system failures, and experimental incidents using narrative tenses and technical sequencing connectors fluently.

Bibliography: [? line:110, ? line:110, ? line:110]

5.7 Research and Innovation, 12 hr(s)

Competences: AG-C03,AG-C04

Topics: **I:** Future Forms: Will, Going to, Present Continuous for research projection. **II:** Expressions of probability and certainty in scientific contexts. **III:** Vocabulary: Innovation cycles, research proposals, technology forecasting. **IV:** Oral presentations: ”The future of computing in my field”.

Learning Outcomes **I:** Students project research goals and discuss future directions in computing and engineering using various future structures with appropriate degrees of certainty.

Bibliography: [? line:123, ? line:123, ? line:123]

5.8 Technical English III, 12 hr(s)

Competences: AG-C03,AG-C04

Topics: **I:** Programming and Web Development terminology: functions, APIs, frameworks. **II:** User Interface (UI/UX) descriptions and usability concepts. **III:** Basic project management vocabulary: sprints, milestones, deliverables. **IV:** Writing: Simple technical specification or README document.

Learning Outcomes **I:** By the end of the course, students use specialized software development vocabulary to describe logical and functional components and produce basic technical documentation in English.

Bibliography: [**?** line:136 , **?** line:136 , **?** line:136]

6. Bibliography

line:141 line:141 line:141

Fourth Semester

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

CS210 Algorithms and Data Structures (Mandatory)

1. General information

- **Semester**: 4th Sem. **Credits**: 4
- **Hour of this course**: **Theory**: 2 hours; **Practice**: 2 hours; **Laboratory**: 2 hours;
- **Prerrequisites**:
  - CS113 Objects-oriented programming II (3rd Sem-Pg. 287)

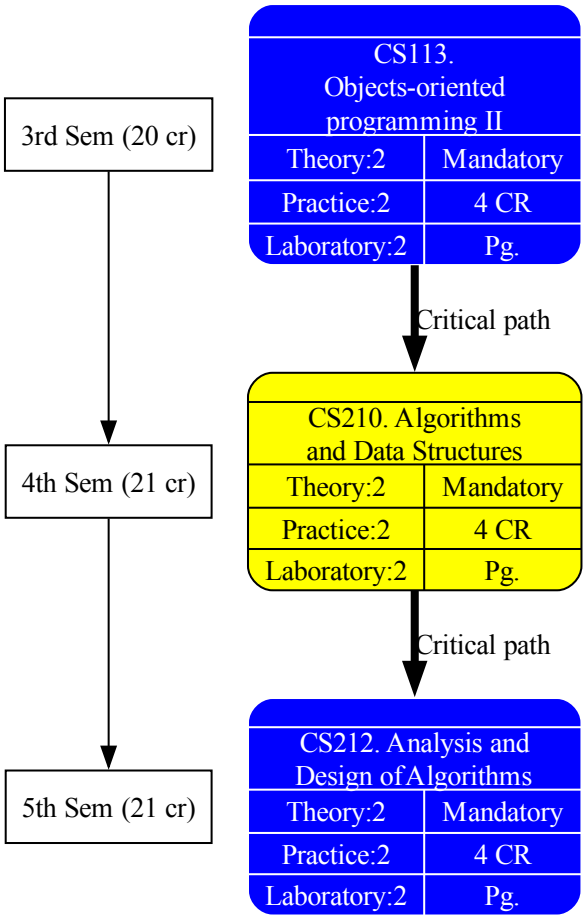

Figure 5.18: Connection Map. CS210 Algorithms and Data Structures

2. Course Rationale

This course provides a comprehensive study of fundamental data structures and algorithms with emphasis on implementation using advanced C++ features. Building on prior C++ knowledge, students will implement generic data structures using templates and traits, ensuring type flexibility and performance. The course covers linear structures (vectors, various linked lists), tree structures (binary,

AVL, B-trees), heaps, hash tables, tries, and concurrent data structures. Special focus is placed on modern C++ techniques including variadic templates for generic operations and concurrent programming for thread-safe data structures.

3. Goals of the Course

- Implement generic data structures using templates and traits.
- Master advanced tree structures and balanced trees.
- Understand and implement concurrent data structures.
- Apply variadic templates for generic operations on containers.
- Analyze algorithmic complexity of all implemented structures.
- Design efficient data structures for specific problem constraints.

4. General Competences (*Outcomes*)

AG-C08) Problem Analysis: Identifies, formulates, and analyzes complex computing problems. (Usage)

AG-C09) Design and Development of Solutions: Designs, implements, and evaluates solutions for complex computing problems. (Usage)

5. Content

5.1 Advanced Foundation for Data Structures, 8 hr(s)

Competences: AG-C08,AG-C09

Topics: **I:** Template metaprogramming review and advanced traits. **II:** Type traits and SFINAE for conditional compilation. **III:** Variadic templates and parameter packs. **IV:** Perfect forwarding and universal references. **V:** Lambda expressions with generic captures. **VI:** Policy-based design and CRTP (Curiously Recurring Template Pattern). **VII:** Concept-based constraints (C++20 concepts if applicable).

Learning Outcomes **I:** Design type traits for customizing data structure behavior [Usage]. **II:** Implement variadic templates for generic container operations [Usage]. **III:** Apply policy-based design to create configurable data structures [Assessment]. **IV:** Use perfect forwarding to implement efficient constructors [Usage].

Bibliography: [Stroustrup, 2013, Vandevoorde et al., 2017]

5.2 Foundational Data Structures, 8 hr(s)

Competences: AG-C08,AG-C09

Topics: **I:** Abstract Data Type (ADT) and operations on an ADT: (a) Dictionary operations (insert, delete, find) **II:** Arrays: (a) Numeric vs non-numeric, character strings (b) Single (vector) vs multidimensional (matrix) **III:** Records/Structs/Tuples and Objects **IV:** Linked lists (for historical reasons): (a) Single vs Double and Linear vs Circular **V:** Stacks **VI:** Queues and deques: (a) Heap-based priority queue **VII:** Hash tables/maps: (a) Collision resolution and complexity (e.g., probing, chaining, rehash) **VIII:** Trees: (a) Binary, n-ary, and search trees (b) Balanced (e.g., AVL, Red-Black, Heap) **IX:** Sets

Learning Outcomes **I:** For each ADT/Data-Structure in this unit: (a) Explain its definition, properties, representation(s), and associated ADT operations. (b) Explain step-by-step how the ADT operations associated with the data structure transform it. [Usage] **II:** For each algorithm in this unit explain step-by-step how the algorithm operates [Usage] **III:** Implement vector with customizable allocation policy using templates [Usage]. **IV:** Design generic linked list nodes with trait-based memory management [Usage]. **V:** Compare time/space complexity of different linear structures [Assessment]. **VI:** Apply variadic templates to initialize containers with multiple elements [Usage].

Bibliography: [Cormen et al., 2009, Stroustrup, 2013]

5.3 Foundational Data Structures, 8 hr(s)

Competences: AG-C08,AG-C09

Topics: **I:** Trees: (a) Binary, n-ary, and search trees (b) Balanced (e.g., AVL, Red-Black, Heap) **II:** Binary tree template implementation with iterator support. **III:** Binary search tree with trait-based comparison operators. **IV:** Tree traversal algorithms (iterative and recursive). **V:** Expression trees and their applications. **VI:** Template-based node structures with configurable payload.

Learning Outcomes **I:** Given requirements for a problem, develop multiple solutions using various data structures and algorithms. Subsequently, evaluate the suitability, strengths, and weaknesses selecting an approach that best satisfies the requirements [Assessment] **II:** Explain factors beyond computational efficiency that influence the choice of algorithms, such as programming time, maintainability, and the use of application-specific patterns in the input data [Familiarity] **III:** Implement generic binary tree with configurable traversal order [Usage]. **IV:** Design expression tree evaluator using visitor pattern [Usage]. **V:** Analyze tree operation complexities for different balancing strategies [Assessment].

Bibliography: [Cormen et al., 2009, Knuth, 1997]

5.4 Balanced and Multi-way Trees, 8 hr(s)

Competences: AG-C08,AG-C09

Topics: **I:** AVL tree implementation with template-based balancing trait. **II:** Red-black tree properties and operations. **III:** B-Tree template implementation for disk-based storage. **IV:** 2-3 Tree and 2-3-4 Tree structures. **V:** Digital trie (prefix tree) implementation. **VI:** Suffix trees and their applications in string processing. **VII:** Node splitting/merging strategies for B-trees.

Learning Outcomes **I:** Implement self-balancing trees using policy-based design [Usage]. **II:** Design B-tree for database index applications [Usage]. **III:** Evaluate trade-offs between different balanced tree structures [Assessment]. **IV:** Implement trie for efficient string prefix matching [Usage].

Bibliography: [Cormen et al., 2009, Knuth, 1997, Sedgewick and Wayne, 2011]

5.5 Foundational Data Structures, 8 hr(s)

Competences: AG-C08,AG-C09

Topics: **I:** Trees: (a) Binary, n-ary, and search trees (b) Balanced (e.g., AVL, Red-Black, Heap) **II:** Binary heap implementation using array representation. **III:** Min-heap and max-heap variants. **IV:** Heapify algorithms and complexity analysis. **V:** Priority queue ADT implementation. **VI:** Binomial heap and Fibonacci heap overview. **VII:** Heap applications: Dijkstra's algorithm, heap sort.

Learning Outcomes **I:** Explain the heap property and the use of heaps as an implementation of a priority queue [Familiarity] **II:** Implement generic heap with configurable comparison trait [Usage]. **III:** Design priority queue supporting multiple priority update strategies [Usage]. **IV:** Analyze heap operations complexity for different heap types [Assessment]. **V:** Apply heap in graph algorithm implementations [Usage].

Bibliography: [Cormen et al., 2009, Sedgewick and Wayne, 2011]

5.6 Foundational Data Structures, 8 hr(s)

Competences: AG-C08,AG-C09

Topics: **I:** Hash tables/maps: (a) Collision resolution and complexity (e.g., probing, chaining, rehash) **II:** Hash function design and properties. **III:** Separate chaining implementation with linked lists. **IV:** Open addressing: linear probing, quadratic probing, double hashing. **V:** Perfect hashing and minimal perfect hashing. **VI:** Cuckoo hashing and hopscotch hashing. **VII:** Hash table resizing strategies and load factor management.

Learning Outcomes **I:** Explain how collision avoidance and collision resolution is handled in hash tables [Familiarity] **II:** Design generic hash table with configurable collision resolution [Usage]. **III:** Implement custom hash functions for user-defined types [Usage]. **IV:** Evaluate performance of different hashing schemes [Assessment]. **V:** Apply hash tables in compiler symbol table implementation [Usage].

Bibliography: [Cormen et al., 2009, Knuth, 1997]

5.7 Concurrent Data Structures, 8 hr(s)

Competences: AG-C08,AG-C09

Topics: **I:** Thread-safe data structure design principles. **II:** Lock-based concurrent linked lists. **III:** Lock-free and wait-free algorithms. **IV:** Concurrent hash tables with fine-grained locking. **V:** Atomic operations and memory ordering. **VI:** Concurrent queues (bounded and unbounded). **VII:** Concurrent stacks and elimination techniques.

Learning Outcomes **I:** Design thread-safe data structures using mutex and lock guards [Usage]. **II:** Implement lock-free linked list using atomic operations [Usage]. **III:** Analyze performance of concurrent vs sequential data structures [Assessment]. **IV:** Apply concurrent queues in producer-consumer patterns [Usage].

Bibliography: [Herlihy and Shavit, 2012, Williams, 2019]

5.8 Specialized Structures and Applications, 8 hr(s)

Competences: AG-C08,AG-C09

Topics: **I:** Persistent data structures. **II:** Disjoint-set union (union-find) with path compression. **III:** Bloom filters and their probabilistic properties. **IV:** Skip lists for probabilistic balanced search. **V:** Segment trees and Fenwick trees for range queries. **VI:** Spatial data structures: k-d trees, quadtrees. **VII:** Cache-oblivious data structures.

Learning Outcomes **I:** Implement persistent data structures with structural sharing [Usage]. **II:** Design union-find with both union by rank and path compression [Usage]. **III:** Apply specialized structures to solve domain-specific problems [Assessment]. **IV:** Evaluate space-time tradeoffs in probabilistic data structures [Assessment].

Bibliography: [Okasaki, 1999, Sedgewick and Wayne, 2011]

6. Bibliography

[Cormen et al., 2009] Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2009). *Introduction to Algorithms*. MIT Press, 3rd edition.

[Herlihy and Shavit, 2012] Herlihy, M. and Shavit, N. (2012). *The Art of Multiprocessor Programming*. Morgan Kaufmann, revised 1st edition.

[Knuth, 1997] Knuth, D. E. (1997). *The Art of Computer Programming, Volume 1: Fundamental Algorithms*. Addison-Wesley, 3rd edition.

[Okasaki, 1999] Okasaki, C. (1999). *Purely Functional Data Structures*. Cambridge University Press.

[Sedgewick and Wayne, 2011] Sedgewick, R. and Wayne, K. (2011). *Algorithms*. Addison-Wesley, 4th edition.

[Stroustrup, 2013] Stroustrup, B. (2013). *The C++ Programming Language*. Addison-Wesley, 4th edition.

[Vandevoorde et al., 2017] Vandevoorde, D., Josuttis, N. M., and Gregor, D. (2017). *C++ Templates: The Complete Guide*. Addison-Wesley, 2nd edition.

[Williams, 2019] Williams, A. (2019). *C++ Concurrency in Action*. Manning Publications, 2nd edition.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

CS211 Theory of Computation (Mandatory)

1. General information

- **Semester**: 4th Sem. **Credits**: 4
- **Hour of this course**: **Theory**: 2 hours; **Practice**: 2 hours; **Laboratory**: 2 hours;
- **Prerrequisites**:
  - CS1D1 Discrete Structures (2nd Sem-Pg. 273)

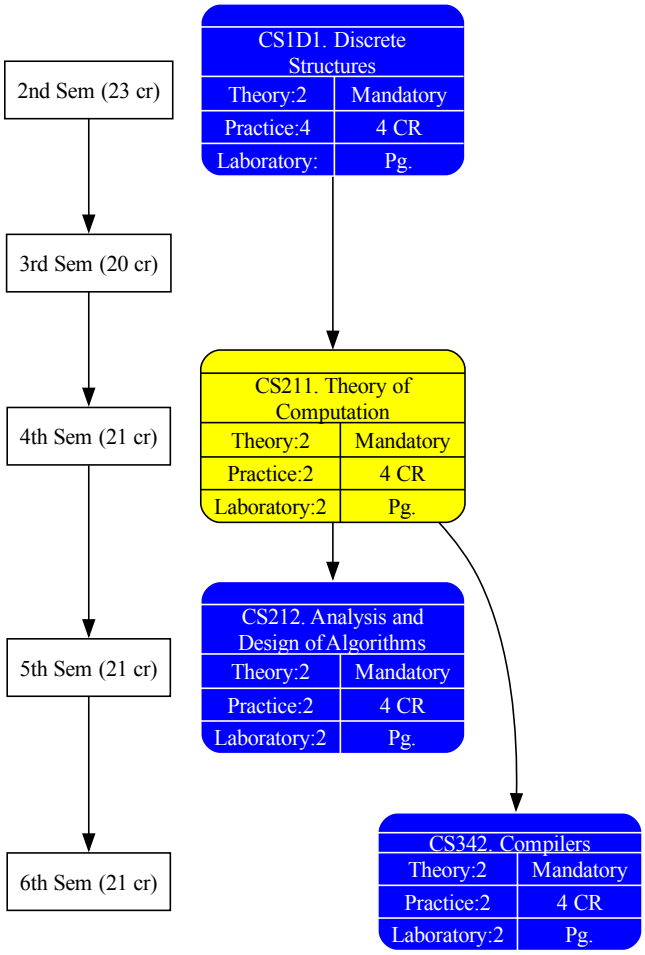

Figure 5.19: Connection Map. CS211 Theory of Computation

2. Course Rationale

Theory of Computation provides the mathematical foundations to understand what can be computed and how efficiently. It introduces formal languages, automata, and the limits of algorithmic solvability. This knowledge is fundamental for understanding compiler construction, formal verification, and the inherent complexity of computational problems.

3. Goals of the Course

- Master the concepts of finite automata and regular expressions.

- Analyze context-free grammars and pushdown automata.
- Understand the Universal Turing Machine as a model of computation.
- Comprehend the boundaries of computability and the Halting Problem.
- Differentiate between complexity classes like P and NP.

4. General Competences (*Outcomes*)

AG-C08) Problem Analysis: Identifies, formulates, and analyzes complex computing problems. (Usage)

AG-C12) Applies computer science theory and software development fundamentals to produce computer-based solutions. (Usage)

5. Content

5.1 Finite Automata and Regular Languages, 18 hr(s)

Competences: AG-C08,AG-C12

Topics: **I:** Formal automata: (a) Finite State (b) Pushdown (c) Linear Bounded (d) Turing Machine **II:** Formal languages, grammars and Chomsky Hierarchy: (a) Regular (Type-3): i. Regular Expressions (b) Context-Free (Type-2) (c) Context-Sensitive (Type-1) (d) Recursively Enumerable (Type-0)) **III:** Deterministic and nondeterministic automata **IV:** Pumping Lemma proofs: (a) Proof of Finite State/Regular-Language limitation (b) Pushdown Automata/Context-Free-Language limitation

Learning Outcomes **I:** For each formal automaton in this unit: (a) Explain its definition comparing its characteristics with this unit's other automata (b) Using an example, explain step-by-step how the automaton operates on input including whether it accepts the associated input (c) Explain an example of inputs that can and cannot be accepted by the automaton. [Explain] **II:** Given a problem, develop an appropriate automaton that addresses the problem [Create] **III:** Develop a regular expression for a given regular language expressed in natural language [Create] **IV:** Convert among equivalently powerful notations for a language, including among DFAs, NFAs, and regular expressions, and between PDAs and CFGs [Apply] **V:** Apply pumping lemmas, or alternative means, to prove the limitations of Finite State and Pushdown automata [Apply]

Bibliography: [Sipser, 2012, Hopcroft et al., 2013]

5.2 Context-Free Languages and PDA, 16 hr(s)

Competences: AG-C08,AG-C12

Topics: **I:** Formal automata: (a) Finite State (b) Pushdown (c) Linear Bounded (d) Turing Machine **II:** Formal languages, grammars and Chomsky Hierarchy: (a) Regular (Type-3): i. Regular Expressions (b) Context-Free (Type-2) (c) Context-Sensitive (Type-1) (d) Recursively Enumerable (Type-0) **III:** Normal Forms: Chomsky and Greibach **IV:** Pumping Lemma proofs: (a) Proof of Finite State/Regular-Language limitation (b) Pushdown Automata/Context-Free-Language limitation **V:** Relations among formal automata, languages, and grammars

Learning Outcomes **I:** For each formal automaton in this unit: (a) Explain its definition comparing its characteristics with this unit's other automata (b) Using an example, explain step-by-step how the automaton operates on input including whether it accepts the associated input (c) Explain an example of inputs that can and cannot be accepted by the automaton. [Explain] **II:** Given a problem, develop an appropriate automaton that addresses the problem [Create] **III:** Convert among equivalently powerful notations for a language, including among DFAs, NFAs, and regular expressions, and between PDAs and CFGs [Apply] **IV:** Apply pumping lemmas, or alternative means, to prove the limitations of Finite State and Pushdown automata [Apply] **V:** Construct CFGs for formal language descriptions [Assessment]

Bibliography: [Sipser, 2012, Hopcroft et al., 2013]

5.3 Turing Machines and Computability, 16 hr(s)

Competences: AG-C08,AG-C12

Topics: **I:** Formal automata: (a) Finite State (b) Pushdown (c) Linear Bounded (d) Turing Machine **II:** Formal languages, grammars and Chomsky Hierarchy: (a) Regular (Type-3): i. Regular Expressions (b) Context-Free (Type-2) (c) Context-Sensitive (Type-1) (d) Recursively Enumerable (Type-0) **III:** Decidability, (un)computability, and halting **IV:** The Church-Turing thesis **V:** Reducibility and reductions **VI:** Equivalent models of algorithmic computation: (a) Turing Machines and Variations (e.g., multi-tape, non-deterministic) (b) Lambda Calculus (c) Mu-Recursive Functions

Learning Outcomes **I:** For each formal automaton in this unit: (a) Explain its definition comparing its characteristics with this unit's other automata (b) Using an example, explain step-by-step how the automaton operates on input including whether it accepts the associated input (c) Explain an example of inputs that can and cannot be accepted by the automaton. [Explain] **II:** Explain a universal Turing Machine and its operation [Explain] **III:** Present to an audience of coworkers and managers the impossibility of providing them a program that checks all other programs, including some seemingly simple ones, for infinite loops including an explanation of the Halting problem, why it has no algorithmic solution, and its significance for real-world algorithmic computation [Explain] **IV:** Explain the Church-Turing Thesis and its significance for algorithmic computation [Explain] **V:** Apply arithmetization and diagonalization to prove the Halting Problem for Turing Machines is Undecidability [Apply] **VI:** Given a known undecidable language, apply a mapping reduction or computational history to prove that another language is undecidable [Apply]

Bibliography: [Sipser, 2012, Kozen, 2006]

5.4 Chomsky Hierarchy and Equivalent Models, 12 hr(s)

Competences: AG-C08,AG-C12

Topics: **I:** Formal languages, grammars and Chomsky Hierarchy: (a) Regular (Type-3): i. Regular Expressions (b) Context-Free (Type-2) (c) Context-Sensitive (Type-1) (d) Recursively Enumerable (Type-0) **II:** Relations among formal automata, languages, and grammars **III:** Equivalent models of algorithmic computation: (a) Turing Machines and Variations (e.g., multi-tape, non-deterministic) (b) Lambda Calculus (c) Mu-Recursive Functions **IV:** Context-Sensitive Languages (Type-1) and applications

Learning Outcomes **I:** For each formal model in this unit: (a) Explain its definition comparing its characteristics with the others in this unit (b) Explain example inputs that are and cannot be accepted by the language/grammar. [Explain] **II:** Explain the Primitive and General Recursive functions (zero, successor, selection, primitive recursion, composition, and Mu), their significance, and Turing Machine implementations [Explain] **III:** Explain how computation is performed in Lambda Calculus (e.g., Alpha conversion and Beta reduction) [Explain] **IV:** Explain Rice's theorem and its significance [Explain]

Bibliography: [Sipser, 2012, Kozen, 2006]

5.5 Complexity Analysis Framework, 3 hr(s)

Competences: AG-C08,AG-C12

Topics: **I:** Complexity Analysis Framework: (a) Best, average, and worst-case performance of an algorithm (b) Empirical and relative (Order of Growth) measurements (c) Input size and primitive operations (d) Time and space efficiency

Learning Outcomes I: Prepare a presentation that explains to first year students the basic concepts of algorithmic complexity including best, average, and worst-case algorithm behavior, Big-O, Omega, and Theta notations, complexity classes, time and space tradeoffs, empirical measurement, and impact on practical problems [Explain] **II:** Informally evaluate the foundational complexity class of simple algorithms [Evaluate]

Bibliography: [Sipser, 2012, Cormen et al., 2022]

5.6 Asymptotic Notation and Complexity Classes, 4 hr(s)

Competences: AG-C08,AG-C12

Topics: I: Asymptotic complexity analysis (average and worst-case bounds): (a) Big-O, Big-Omega, and Big-Theta formal notations (b) Foundational Complexity Classes and Representative Examples/Problems: i. $O(1)$ Constant (e.g., array access) ii. $O(\log_2 n)$ Logarithmic (e.g., binary search) iii. $O(n)$ Linear (e.g., linear search) iv. $O(n \log_2 n)$ Log Linear (e.g., mergesort) v. $O(n^2)$ Quadratic (e.g., selection sort) vi. $O(n^c)$ Polynomial (e.g., $O(n^3)$ Gaussian elimination) vii. $O(2^n)$ Exponential (e.g., Knapsack, Satisfiability (SAT), Traveling Sales-Person (TSP), all subsets) viii. $O(n!)$ Factorial (e.g., Hamiltonian circuit, all permutations)

Learning Outcomes I: Using examples, explain each of the foundational complexity classes in this unit [Explain] **II:** Explain to a non-technical audience the significance of tractable versus intractable algorithms using an intuitive explanation of Big-O complexity [Explain]

Bibliography: [Sipser, 2012, Cormen et al., 2022]

5.7 Complexity Analysis: Recursion, Amortization, and Tight Bounds, 7 hr(s)

Competences: AG-C08,AG-C12

Topics: I: Little-o, Little-Omega, and Little Theta notations **II:** Formal recursive analysis

Learning Outcomes I: Use recurrence relations to evaluate the time complexity of recursively defined algorithms [Apply] **II:** Apply elementary recurrence relations using a form of the Master Theorem [Apply]

Bibliography: [Sipser, 2012, Cormen et al., 2022]

5.8 Computational Complexity Theory, 6 hr(s)

Competences:

Topics: I: Tractability and intractability: (a) P, NP, and NP-Complete Complexity Classes (b) NP-Complete Problems (e.g., SAT, Knapsack, TSP) (c) Reductions **II:** Turing Machine-based models of complexity: (a) Time complexity: i. P, NP, NP-C, and EXP classes ii. Cook-Levin theorem (b) Space Complexity: i. NSpace and PSpace ii. Savitch's theorem

Learning Outcomes I: Explain the significance of NP-Completeness [Explain] **II:** Explain how NP-Hard is a lower bound and NP is an upper bound for NP-Completeness [Explain] **III:** Explain examples of NP-complete problems [Explain] **IV:** Explain the Cook-Levin Theorem and the NP-Completeness of SAT [Explain] **V:** Explain the classes P and NP [Explain] **VI:** Prove that a problem is NP-Complete by reducing a classic known NP-C problem to it (e.g., 3SAT and Clique) [Create] **VII:** Explain the P-space class and its relation to the EXP class [Explain]

Bibliography: [Sipser, 2012, Cormen et al., 2022]

6. Bibliography

[Cormen et al., 2022] Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2022). *Introduction to Algorithms*. MIT Press, 4th edition.

[Hopcroft et al., 2013] Hopcroft, J. E., Motwani, R., and Ullman, J. D. (2013). *Introduction to Automata Theory, Languages, and Computation*. Pearson, 3rd edition.

[Kozen, 2006] Kozen, D. C. (2006). *Theory of Computation*. Springer.

[Sipser, 2012] Sipser, M. (2012). *Introduction to the Theory of Computation*. Cengage Learning, 3rd edition.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

CS221 Computer Systems Architecture (Mandatory)

1. General information

- **Semester**: 4th Sem. **Credits**: 4
- **Hour of this course**: **Theory**: 2 hours; **Laboratory**: 4 hours;
- **Prerrequisites**:
  - CS1D1 Discrete Structures (2nd Sem-Pg. 273)

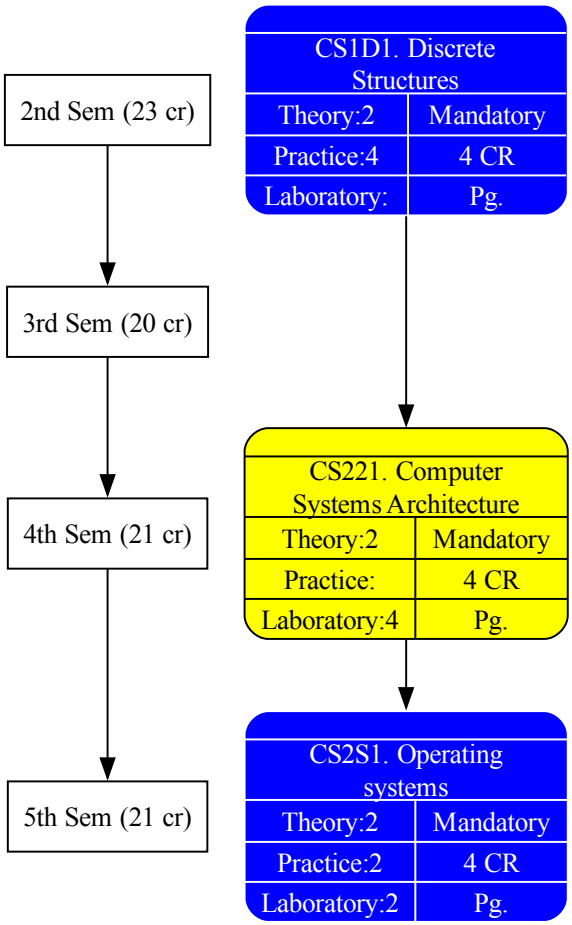

Figure 5.20: Connection Map. CS221 Computer Systems Architecture

2. Course Rationale

A deep understanding of computer architecture is essential for developing high-performance software and understanding the hardware-software interface. This course explores how computers execute instructions, manage memory, and communicate with external devices. By studying the internal organization of processors and the hierarchy of memory systems, students gain the ability to optimize code for specific hardware architectures and understand the trade-offs in modern system design.

3. Goals of the Course

- Understand the principles of digital logic and data representation.
- Understand the organization of the Central Processing Unit (CPU).
- Master the basics of assembly language and instruction set architecture.
- Analyze the memory hierarchy, including cache memory and virtual memory.
- Evaluate performance improvements through pipelining and parallelism.

4. General Competences (*Outcomes*)

AG-C03) Individual and Team Work: Performs effectively as an individual and as a member or leader in diverse teams. (Usage)

AG-C09) Design and Development of Solutions: Designs, implements, and evaluates solutions for complex computing problems. (Usage)

5. Content

5.1 Assembly Level Machine Organization, 20 hr(s)

Competences: AG-C03,AG-C09

Topics: **I:** von Neumann machine architecture **II:** Control unit: instruction fetch, decode, and execution **III:** Introduction to SIMD vs MIMD and the Flynn taxonomy **IV:** Shared memory multiprocessors/multicore organization **V:** Instruction set architecture (ISA) (e.g., x86, ARM and RISC-V): (a) Fixed vs variable-width instruction sets (b) Instruction formats (c) Data manipulation, control, I/O (d) Addressing modes (e) Machine language programming (f) Assembly language programming **VI:** Subroutine call and return mechanisms **VII:** I/O and interrupts **VIII:** Heap, static, stack, and code segments

Learning Outcomes **I:** Discuss how the classical von Neumann functional units are implemented in embedded systems, particularly on-chip and off-chip memory [Debate] **II:** Describe how instructions are executed in a classical von Neumann machine, with extensions for threads, multiprocessor synchronization, and SIMD execution [Describe] **III:** Assess an example diagram with instruction-level parallelism and hazards to describe how they are managed in typical processor pipelines [Assess] **IV:** Discuss how instructions are represented at the machine level and in the context of a symbolic assembler [Debate] **V:** Map an example of high-level language patterns into assembly/machine language notations [Map] **VI:** Contrast different instruction formats considering aspects such as addresses per instruction and variable-length vs fixed-length formats [Contrast] **VII:** Analyze a subroutine diagram to comment on how subroutine calls are handled at the assembly level [Analyze] **VIII:** Describe basic concepts of interrupts and I/O operations [Describe] **IX:** Write a simple assembly language program for string/array processing and manipulation [Write]

Bibliography: [Patterson and Hennessy, 2020, Harris and Harris, 2012]

5.2 Memory Hierarchy, 18 hr(s)

Competences: AG-C03,AG-C09

Topics: **I:** Memory hierarchy: the importance of temporal and spatial locality **II:** Main memory organization and operations **III:** Persistent memory (e.g., SSD, standard disks) **IV:** Latency, cycle time, bandwidth, and interleaving **V:** Cache memories: (a) Address mapping (b) Block size (c) Replacement and store policy (d) Prefetching **VI:** Multiprocessor cache coherence **VII:** Virtual memory (hardware support) **VIII:** Fault handling and reliability **IX:** Reliability: (a) Error coding (b) Data compression (c) Data integrity **X:** Processing In-Memory (PIM)

Learning Outcomes **I:** Using a memory system diagram, identify the main types of memory technology (e.g., SRAM, DRAM) and their relative cost and performance [Analyze] **II:** Measure the effect

of memory latency on running time [Evaluate] **III:** Enumerate the functions of a system with virtual memory management [Enumerate] **IV:** Compute average memory access time under various cache and memory configurations and mixes of instruction and data references [Compute]

Bibliography: [Patterson and Hennessy, 2020, Stallings, 2015]

5.3 Interfacing and Communication, 8 hr(s)

Competences: AG-C03,AG-C09

Topics: **I:** I/O fundamentals: (a) Handshaking and buffering (b) Programmed I/O (c) Interrupt-driven I/O **II:** Interrupt structures: vectored and prioritized, interrupt acknowledgment **III:** I/O devices (e.g., mouse, keyboard, display, camera, sensors, actuators) **IV:** External storage, physical organization, and drives **V:** Buses fundamentals: (a) Bus protocols (b) Arbitration (c) Direct-memory access (DMA)

Learning Outcomes **I:** Analyze an interrupt control diagram to comment on how interrupts are used to implement I/O control and data transfers [Analyze] **II:** Enumerate various types of buses in a computer system [Enumerate] **III:** List the advantages of magnetic disks and contrast them with those of solid-state disks [List]

Bibliography: [Patterson and Hennessy, 2020, Tanenbaum and Austin, 2012]

5.4 Functional Organization, 8 hr(s)

Competences: AG-C03,AG-C09

Topics: **I:** Implementation of simple datapaths, including instruction pipelining, hazard detection, and resolution (e.g., stalls, forwarding) **II:** Control unit: (a) Hardwired implementation (b) Microprogrammed realization **III:** Instruction pipelining **IV:** Introduction to instruction-level parallelism (ILP)

Learning Outcomes **I:** Compare alternative implementation of datapaths in modern computer architectures [Compare] **II:** Produce a set of control signals for adding two integers using hardwired and microprogrammed implementations [Produce] **III:** Discuss instruction-level parallelism using pipelining and significant hazards that may occur [Debate] **IV:** Design a complete processor, including datapath and control [Design] **V:** Compute the average cycles per instruction for a given processor and memory system implementation [Compute]

Bibliography: [Harris and Harris, 2012, Patterson and Hennessy, 2020]

5.5 Heterogeneous Architectures, 10 hr(s)

Competences: AG-C03,AG-C09

Topics: **I:** SIMD and MIMD architectures (e.g., General-Purpose GPUs, TPUs, and NPUs) **II:** Heterogeneous memory systems: (a) Shared memory versus distributed memory (b) Volatile vs non-volatile memory (c) Coherence protocols **III:** Domain-Specific Architectures (DSAs): (a) Machine Learning Accelerator (b) In-networking computing (c) Embedded systems for emerging applications (d) Neuromorphic computing (e) Edge computing devices **IV:** Packaging and integration solutions such as 3DIC and chiplets **V:** Machine learning in architecture design: (a) AI algorithms for workload analysis (b) Optimization of architecture configurations for performance and power efficiency **VI:** Data-Oriented Design (DOD): organizing data in memory (array-of-structures vs. structure-of-arrays) to maximize cache locality and exploit SIMD vectorization; contrast and complementarity with the object-oriented paradigm.

Learning Outcomes I: Analyze a system diagram with alternative parallel architectures, e.g., SIMD and MIMD, and identify the key differences [Assessment] **II:** Discuss what memory-management issues are found in multiprocessors that are not present in uniprocessors and how these issues might be resolved [Assessment] **III:** Indicate the differences between memory backplane, processor memory interconnect, and remote memory via networks, their implications for access latency, and their impact on program performance [Familiarity] **IV:** Discuss how you would determine when to use a domain-specific accelerator instead of a general-purpose CPU [Assessment] **V:** Enumerate key differences in architectural design principles between a vector and scalar-based processing unit [Familiarity] **VI:** List the advantages and disadvantages of a PIM architecture [Familiarity] **VII:** Restructure a collection of objects (AoS) into a DOD layout (SoA) and measure the performance impact [Usage].

Bibliography: [Patterson and Hennessy, 2020, Stallings, 2015]

5.6 Secure Processor Architectures, 8 hr(s)

Competences: AG-C03,AG-C09

Topics: I: Principles of Secure Hardware: (a) Security Risk Analysis, Asset Protection, and Threat Model (b) Cryptographic Acceleration with Hardware (c) Support for virtualization (e.g., OS isolation) **II:** Roots of trust in hardware, Physically Unclonable Functions (PUF) **III:** Hardware Random Number Generators **IV:** Memory protection extensions: (a) Runtime pointer bounds checking (e.g., buffer overflow) (b) Protection at the microarchitectural level (c) Protection at the ISA level **V:** Trusted Execution Environment (TEE): (a) Trusted Computer Base Protections (b) Protecting virtual machines (c) Protecting containers (d) Trusted software modules (Enclaves) **VI:** Homomorphic encryption for privacy-preserving data processing

Learning Outcomes I: Discuss principles of secure hardware, exploring a framework for risk analysis and asset protection [Debate] **II:** Summarize how Physically Unclonable Functions (PUF) can be a unique device identifier in security applications [Summarize] **III:** Distinguish a random number generator with dedicated hardware support from generators without hardware dedicated to generating entropy [Distinguish] **IV:** List the advantages and disadvantages of memory protection at the ISA level [List] **V:** Describe key design issues of a trusted execution environment (TEE) to support virtual machines [Describe]

Bibliography: [Patterson and Hennessy, 2020, Anderson, 2020]

6. Bibliography

[Anderson, 2020] Anderson, R. J. (2020). *Security Engineering: A Guide to Building Dependable Distributed Systems*. Wiley, 3rd edition.

[Harris and Harris, 2012] Harris, D. and Harris, S. (2012). *Digital Design and Computer Architecture*. Morgan Kaufmann, 2nd edition.

[Patterson and Hennessy, 2020] Patterson, D. A. and Hennessy, J. L. (2020). *Computer Organization and Design RISC-V Edition: The Hardware Software Interface*. Morgan Kaufmann, 2nd edition.

[Stallings, 2015] Stallings, W. (2015). *Computer Organization and Architecture: Designing for Performance*. Pearson, 10th edition.

[Tanenbaum and Austin, 2012] Tanenbaum, A. S. and Austin, T. (2012). *Structured Computer Organization*. Pearson, 6th edition.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

CS271 Data Management (Mandatory)

1. General information

- **Semester**: 4th Sem. **Credits**: 4
- **Hour of this course**: **Theory**: 2 hours; **Laboratory**: 4 hours;
- **Prerrequisites**:
  - CS112 Objects-oriented programming I (2nd Sem-Pg. 268)
  - CS1D1 Discrete Structures (2nd Sem-Pg. 273)

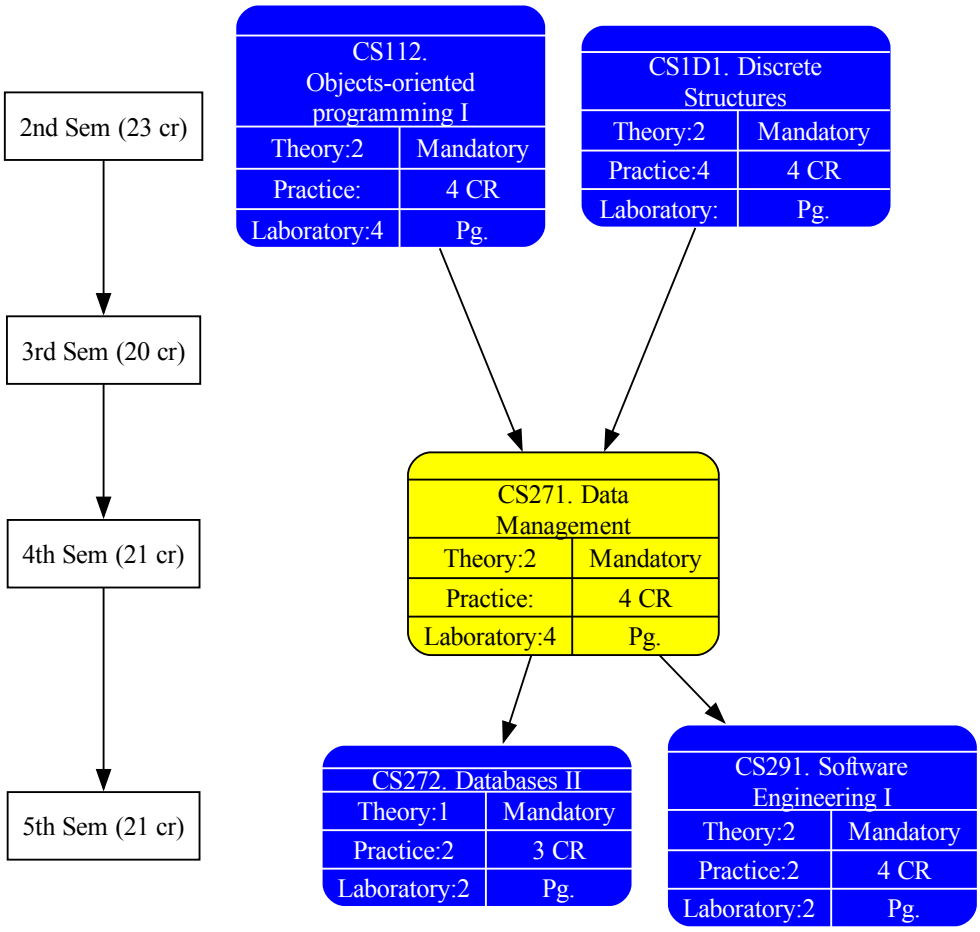

Figure 5.21: Connection Map. CS271 Data Management

2. Course Rationale

This course introduces fundamental concepts and techniques in data management, focusing on relational database systems. Students will learn data modeling using entity-relationship diagrams, relational database design principles, normalization theory, and Structured Query Language (SQL). The course covers core database system components, transaction management, and basic query processing. Emphasis is placed on practical skills for designing, implementing, and querying databases,

as well as understanding the role of data in organizations and the associated ethical considerations in data management.

3. Goals of the Course

- Design and implement normalized relational databases.
- Formulate complex queries using SQL for data retrieval and manipulation.
- Apply data modeling techniques using entity-relationship diagrams.
- Understand database system architecture and transaction management.
- Recognize ethical and security considerations in data management.
- Utilize basic database administration tools and techniques.

4. General Competences (*Outcomes*)

AG-C09) Design and Development of Solutions: Designs, implements, and evaluates solutions for complex computing problems. (Usage)

AG-C12) Applies computer science theory and software development fundamentals to produce computer-based solutions. (Usage)

5. Content

5.1 Core Database System Concepts, 8 hr(s)

Competences: AG-C09,AG-C12

Topics: **I:** Purpose and advantages of database systems **II:** Components of database systems **III:** Design of core DBMS functions (e.g., query mechanisms, transaction management, buffer management, access methods) **IV:** Database architecture, data independence, and data abstraction **V:** Transaction management **VI:** Normalization **VII:** Approaches for managing large volumes of data (e.g., NoSQL database systems, use of MapReduce) **VIII:** How to support CRUD-only applications **IX:** Distributed databases/cloud-based systems **X:** Structured, semi-structured, and unstructured data **XI:** Use of a declarative query language **XII:** Systems supporting structured and/or stream content

Learning Outcomes **I:** Identify at least four advantages that using a database system provides [Familiarity] **II:** Enumerate the components of a (relational) database system [Familiarity] **III:** Explain the advantages of database systems over file systems [Usage]. **IV:** Describe the three-schema architecture and data independence [Familiarity]. **V:** Identify different types of database users and their roles [Familiarity].

Bibliography: [Ramakrishnan and Gehrke, 2002, Silberschatz et al., 2019]

5.2 Data Modeling, 8 hr(s)

Competences: AG-C09,AG-C12

Topics: **I:** Conceptual models (e.g., entity-relationship, UML diagrams) **II:** Entity sets, relationship sets, and attributes **III:** Cardinality constraints: one-to-one, one-to-many, many-to-many **IV:** Weak entity sets and identifying relationships **V:** Extended E-R features: specialization, generalization, aggregation **VI:** UML class diagrams for data modeling

Learning Outcomes **I:** Describe the components of the E-R (or some other non-relational) data model [Familiarity] **II:** Model a given environment using a conceptual data model [Usage] **III:** Design E-R diagrams for real-world scenarios [Usage]. **IV:** Convert E-R diagrams to relational schemas [Usage]. **V:** Apply extended E-R features for complex modeling [Assessment].

Bibliography: [Ramakrishnan and Gehrke, 2002, Elmasri and Navathe, 2015]

5.3 Relational Databases, 8 hr(s)

Competences: AG-C09,AG-C12

Topics: **I:** Entity and referential integrity: Candidate key, superkeys **II:** Relational database design **III:** Relational algebra operations: select, project, join, union, etc. **IV:** Integrity constraints: domain, key, entity, referential **V:** Relational calculus: tuple and domain calculus **VI:** Mapping E-R diagrams to relational schemas

Learning Outcomes **I:** Describe the defining characteristics behind the relational data model [Familiarity] **II:** Comment on the difference between a foreign key and a superkey [Usage] **III:** Express queries using relational algebra [Usage]. **IV:** Design relational schemas with appropriate constraints [Usage]. **V:** Map complex E-R diagrams to normalized relations [Assessment].

Bibliography: [Ramakrishnan and Gehrke, 2002, Elmasri and Navathe, 2015]

5.4 Query Construction, 8 hr(s)

Competences: AG-C09,AG-C12

Topics: **I:** SQL Query Formation: (a) Interactive SQL execution (b) Programmatic execution of an SQL query **II:** Data definition language: CREATE, ALTER, DROP **III:** Data types and domain constraints creation and management **IV:** View definition and materialized views **V:** Database schema design and implementation

Learning Outcomes **I:** Compose SQL queries that incorporate select, project, join, union, intersection, set difference, and set division [Usage] **II:** Define, in SQL, a relation schema, including all integrity constraints and delete/update triggers [Usage] **III:** Create and modify database schemas using SQL DDL [Usage]. **IV:** Implement integrity constraints at schema level [Usage]. **V:** Design and create appropriate indexes for performance [Assessment].

Bibliography: [Ramakrishnan and Gehrke, 2002, Beaulieu, 2009]

5.5 Query Construction, 8 hr(s)

Competences: AG-C09,AG-C12

Topics: **I:** Complex SQL queries: nested subqueries, correlated subqueries **II:** Aggregation functions and GROUP BY operations **III:** Set operations: UNION, INTERSECT, EXCEPT **IV:** JOIN operations: inner, outer, natural, cross joins **V:** Data manipulation: INSERT, UPDATE, DELETE **VI:** Transaction control: COMMIT, ROLLBACK, SAVEPOINT

Learning Outcomes **I:** Determine when a nested SQL query is correlated or not [Usage] **II:** Compose an SQL query to update a tuple in a relation [Usage] **III:** Formulate complex queries with multiple joins and subqueries [Usage]. **IV:** Apply aggregation for data analysis and reporting [Usage]. **V:** Implement data modification operations with transaction control [Assessment].

Bibliography: [Ramakrishnan and Gehrke, 2002, Beaulieu, 2009]

5.6 Relational Databases, 8 hr(s)

Competences: AG-C09,AG-C12

Topics: **I:** Introduction to Functional dependency theory **II:** Normalization Theory: (a) Decomposition of a schema; lossless-join, and dependency-preservation properties of a decomposition (b) Normal forms (BCNF) (c) Denormalization (for efficiency) **III:** Functional dependencies and inference rules **IV:** Normal forms: 1NF, 2NF, 3NF, BCNF **V:** Decomposition algorithms: lossless-join and dependency-preserving **VI:** Practical considerations in normalization

Learning Outcomes **I:** Describe how functional dependency theory generalizes the notion of key [Familiarity] **II:** Detect which normal form a given decomposition yields [Usage] **III:** Identify functional dependencies in relational schemas [Usage]. **IV:** Normalize relations to BCNF [Assessment]. **V:** Evaluate decompositions for lossless-join and dependency preservation [Assessment].

Bibliography: [Ramakrishnan and Gehrke, 2002, Elmasri and Navathe, 2015]

5.7 DBMS Internals, 8 hr(s)

Competences: AG-C09,AG-C12

Topics: **I:** Transaction Management: (a) Isolation Levels (b) ACID (c) Serializability (d) Distributed Transactions **II:** Concurrency Control: (a) 2-Phase Locking (b) Deadlocks handling strategies (c) Quorum-based consistency models **III:** ACID properties: Atomicity, Consistency, Isolation, Durability **IV:** Serializability and conflict serializability **V:** Lock-based concurrency control: two-phase locking **VI:** Deadlock detection and prevention **VII:** Isolation levels and their trade-offs

Learning Outcomes **I:** Describe the four properties for a correct transaction manager [Familiarity] **II:** Outline the principle of serializability [Familiarity] **III:** Explain ACID properties and their importance [Usage]. **IV:** Analyze transaction schedules for serializability [Assessment]. **V:** Apply locking protocols for concurrency control [Usage].

Bibliography: [Ramakrishnan and Gehrke, 2002, Bernstein et al., 1987]

5.8 Data Security and Privacy, 8 hr(s)

Competences: AG-C09,AG-C12

Topics: **I:** Differences between data security and data privacy **II:** Personally identifying information (PII) and its protection **III:** Ethical considerations in ensuring the security and privacy of data **IV:** Database security: authentication, authorization, encryption **V:** SQL injection attacks and prevention **VI:** Introduction to NoSQL databases and big data **VII:** Data governance and compliance

Learning Outcomes **I:** Describe the differences in the goals for data security and data privacy [Familiarity] **II:** Identify and mitigate risks associated with different approaches to protecting data [Usage] **III:** Implement basic database security measures [Usage]. **IV:** Prevent SQL injection vulnerabilities [Assessment]. **V:** Discuss ethical considerations in data collection and use [Assessment].

Bibliography: [Ramakrishnan and Gehrke, 2002, Elmasri and Navathe, 2015]

6. Bibliography

[Beaulieu, 2009] Beaulieu, A. (2009). *Learning SQL*. O'Reilly Media, 2nd edition.

[Bernstein et al., 1987] Bernstein, P. A., Hadzilacos, V., and Goodman, N. (1987). *Concurrency Control and Recovery in Database Systems*. Addison-Wesley.

[Elmasri and Navathe, 2015] Elmasri, R. and Navathe, S. B. (2015). *Fundamentals of Database Systems*. Pearson, 7th edition.

[Ramakrishnan and Gehrke, 2002] Ramakrishnan, R. and Gehrke, J. (2002). *Database Management Systems*. McGraw-Hill, 3rd edition.

[Silberschatz et al., 2019] Silberschatz, A., Korth, H. F., and Sudarshan, S. (2019). *Database System Concepts*. McGraw-Hill, 7th edition.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

CS401 Research Methodology (Mandatory)

1. General information

- **Semester**: 4th Sem. **Credits**: 2
- **Hour of this course**: **Theory**: 1 hours; **Practice**: 2 hours;
- **Prerrequisites**:
  - CS100 Introduction to Computer Science (2nd Sem-Pg. 257)

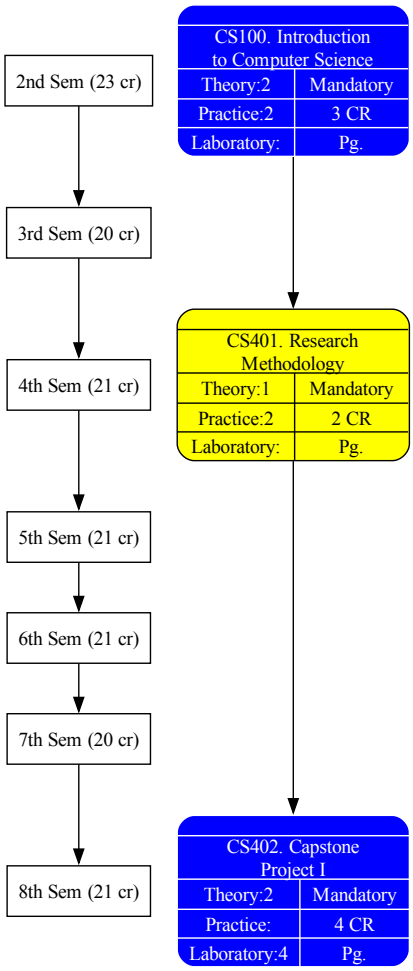

Figure 5.22: Connection Map. CS401 Research Methodology

2. Course Rationale

This course aims to teach students how to conduct scientific research in the field of computer science. The course instructors will determine a specific study area for each student and provide relevant bibliography for analysis. Based on this and additional bibliographic sources researched by the student, the student should be able to construct a survey-type article on the assigned topic.

3. Goals of the Course

- Learn how to initiate scientific research in computer science with methodological rigor.

- Know the main academic bibliographic sources: IEEE Xplore, ACM Digital Library, Google Scholar, and arXiv.
- Critically analyze existing proposals on a topic and relate them coherently in a literature review.
- Write technical documents in computer science using LaTeXwith IEEE and ACM conference templates.
- Reproduce existing results in a topic through implementation and experimentation.
- Build a survey-type article with taxonomy, synthesis, and comparison of existing approaches.
- The deliverables for this course are:

  Partial Progress: Mastery of the article topic and preliminary bibliography in LaTeXarticle format.

  Final: Completed survey-type article, optionally containing experimental results of the studied technique(s).

4. General Competences (*Outcomes*)

AG-C02) Ethics: Applies ethical principles and commits to professional ethics and the standards of professional computing practice. (Assessment)

AG-C04) Communication: Communicates effectively in complex computing activities. (Assessment)

AG-C05) Project Management: Applies management principles in computing to manage projects. (Assessment)

AG-C08) Problem Analysis: Identifies, formulates, and analyzes complex computing problems. (Assessment)

AG-C09) Design and Development of Solutions: Designs, implements, and evaluates solutions for complex computing problems. (Assessment)

5. Content

5.1 Foundations of Scientific Research in Computer Science, 8 hr(s)

Competences: AG-C05,AG-C08

Topics: **I:** Types of research in computer science: empirical, theoretical, constructive, and evaluative. **II:** The scientific method applied to research in computer science. **III:** Main academic databases: IEEE Xplore, ACM Digital Library, Google Scholar, Semantic Scholar, and arXiv. **IV:** Bibliographic search strategies: Boolean operators, citation tracking, and systematic search. **V:** Reference managers: Zotero, Mendeley, and BibTeX integration.

Learning Outcomes **I:** Identify the types of research in computer science and their methodological differences. [Familiarity] **II:** Use the main academic databases to find relevant literature on a given topic. [Usage] **III:** Organize bibliographic references using reference managers and BibTeX format. [Usage]

Bibliography: [IEEE-Computer Society, 2008, ACM, 2008]

5.2 Critical Reading and Analysis of Scientific Literature, 8 hr(s)

Competences: AG-C08,AG-C09

Topics: **I:** Structure of a scientific article: abstract, introduction, related work, methodology, results, and conclusions. **II:** Techniques for efficient reading of scientific articles (three-pass reading). **III:** Critical evaluation: identifying contributions, limitations, and open problems. **IV:** Building a reading matrix and comparative tables of related works. **V:** Classification and taxonomy design of existing approaches on a topic.

Learning Outcomes I: Read and critically evaluate scientific articles identifying their contributions and limitations. [Usage] **II:** Build an organized reading matrix that synthesizes multiple works on a given topic. [Usage] **III:** Design a preliminary taxonomy of approaches for an assigned research area. [Assessment]

Bibliography: [IEEE-Computer Society, 2008, ACM, 2008, CiteSeer.IST, 2008]

5.3 Research Ethics and Academic Integrity, 6 hr(s)

Competences: AG-C02,AG-C04

Topics: I: Academic integrity: plagiarism, self-plagiarism, data fabrication, and falsification. **II:** Ethics in computer science research: privacy, algorithmic bias, and human subjects studies. **III:** Open access, licenses (Creative Commons, GPL), and intellectual property in software. **IV:** Good citation and authorship attribution practices in collaborative work. **V:** The peer review process: role of the reviewer and responsibilities of the author.

Learning Outcomes I: Distinguish forms of academic misconduct and apply correct citation practices. [Usage] **II:** Analyze ethical dilemmas in computational research applying integrity principles. [Assessment] **III:** Explain intellectual property rights and relevant licenses in computer science. [Familiarity]

Bibliography: [IEEE-Computer Society, 2008, ACM, 2008]

5.4 Technical Writing and Scientific Writing Tools, 10 hr(s)

Competences: AG-C04,AG-C09

Topics: I: LaTeX structure: preamble, sections, equations, figures, tables, and cross-references. **II:** Conference and journal templates: IEEE and ACM. **III:** Academic writing style in computer science: clarity, precision, and active voice. **IV:** Writing each section of the article: abstract, introduction, related work, methodology, results, and conclusions. **V:** Document version control with Git for LaTeXprojects. **VI:** Bibliography management with BibTeX and JabRef.

Learning Outcomes I: Write a complete technical article in LaTeXusing IEEE or ACM templates. [Usage] **II:** Apply academic writing conventions of the computer science community. [Usage] **III:** Evaluate the writing quality of an article based on clarity and structure criteria. [Assessment]

Bibliography: [IEEE-Computer Society, 2008, ACM, 2008]

5.5 Systematic Literature Review and Survey Articles, 10 hr(s)

Competences: AG-C04,AG-C08,AG-C09

Topics: I: Types of reviews: narrative, systematic, mapping study, and meta-analysis. **II:** Systematic literature review (SLR) protocol: research questions, search strategy, and selection criteria. **III:** PRISMA guidelines: flowchart for selection of primary studies. **IV:** Synthesis techniques: narrative synthesis, vote counting, and thematic analysis. **V:** Taxonomy design and classification schemes for a survey. **VI:** Structure and writing of a survey-type article.

Learning Outcomes I: Design and execute a systematic literature review protocol for a computer science topic. [Usage] **II:** Apply PRISMA guidelines to document and report the study selection process. [Usage] **III:** Write a survey article with taxonomy, critical synthesis, and comparison of approaches. [Assessment]

Bibliography: [IEEE-Computer Society, 2008, ACM, 2008, CiteSeer.IST, 2008]

5.6 Presentation and Dissemination of Research Results, 6 hr(s)

Competences: AG-C02,AG-C04,AG-C05

Topics: I: Oral presentation techniques for scientific audiences: structure, time management, and Q&A sessions. **II:** Poster design for research conferences and symposia. **III:** Responding to reviewers: addressing comments constructively and with evidence. **IV:** Research project planning: schedule, milestones, and deliverables. **V:** Collaboration and co-authorship practices in scientific research.

Learning Outcomes I: Prepare and deliver an oral presentation of research results to an academic audience. [Usage] **II:** Develop a work plan for a research project with schedule and milestones. [Usage] **III:** Respond constructively to reviewer comments in the publication process. [Assessment]

Bibliography: [IEEE-Computer Society, 2008, ACM, 2008]

6. Bibliography

[ACM, 2008] ACM (2008). *Digital Library.* Association for Computing Machinery. http://portal.acm.org/dl.cfm.

[CiteSeer.IST, 2008] CiteSeer.IST (2008). *Scientific Literature Digital Library.* College of Information Sciences and Technology, Penn State University. http://citeseer.ist.psu.edu.

[IEEE-Computer Society, 2008] IEEE-Computer Society (2008). *Digital Library.* IEEE-Computer Society. http://www.computer.org/publications/dlib.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

MA202 Numerical Methods (Mandatory)

1. General information

- **Semester**: 4th Sem. **Credits**: 3
- **Hour of this course**: **Theory**: 2 hours; **Practice**: 2 hours;
- **Prerrequisites**:
  - BMA103 Integral Calculus (2nd Sem-Pg. 276)

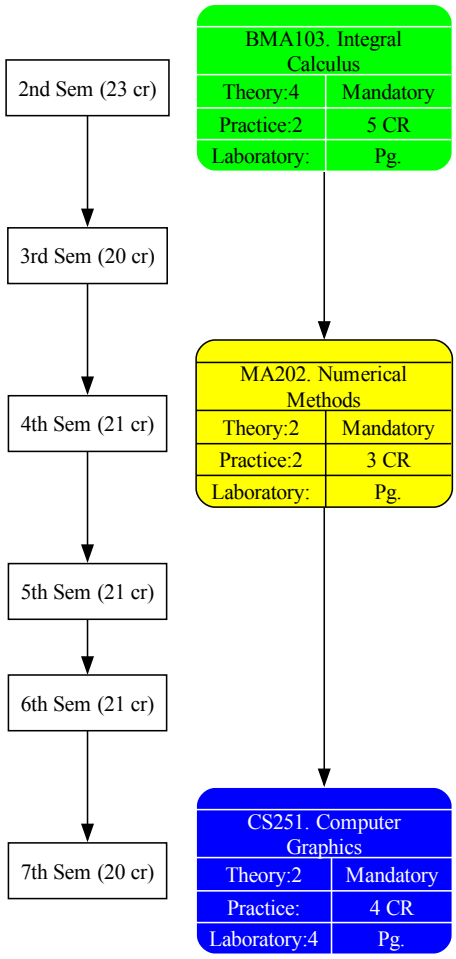

Figure 5.23: Connection Map. MA202 Numerical Methods

2. Course Rationale

Numerical methods are a fundamental tool shared by computer science and engineering disciplines for approximating solutions to mathematical problems that cannot be solved analytically. This course provides an introduction to the most common numerical methods, including error analysis, solution of nonlinear equations, interpolation, numerical integration, and solution of differential equations, in scientific and engineering contexts.

3. Goals of the Course

- Understand the importance of numerical methods in solving scientific and engineering problems.
- Apply different numerical methods to approximate solutions to mathematical problems.
- Analyze the accuracy and efficiency of the numerical methods used.

4. General Competences (*Outcomes*)

AG-C08) Problem Analysis: Identifies, formulates, and analyzes complex computing problems. (Usage)

AG-C09) Design and Development of Solutions: Designs, implements, and evaluates solutions for complex computing problems. (Usage)

AG-C11) Use of Tools: Applies modern computing tools in problem solving. (Usage)

5. Content

5.1 Error Analysis and Floating-Point Arithmetic, 6 hr(s)

Competences: AG-C08,AG-C11

Topics: **I:** IEEE 754 floating-point representation: machine epsilon, overflow, underflow, and rounding modes **II:** Rounding errors, catastrophic cancellation, and loss of significance **III:** Forward and backward error analysis; condition number of a problem **IV:** Numerical stability: stable vs. unstable algorithms; Wilkinson's backward error analysis **V:** Error propagation in arithmetic operations and function evaluation

Learning Outcomes **I:** Explain the IEEE 754 floating-point standard and identify the sources of rounding and cancellation errors [Familiarity] **II:** Estimate the condition number of a problem and predict the accuracy of a numerical result [Usage] **III:** Analyze an algorithm for numerical stability using forward and backward error bounds [Assessment]

Bibliography: [Chapra and Canale, 2015, Burden and Faires, 2010]

5.2 Root-Finding for Nonlinear Equations, 10 hr(s)

Competences: AG-C08,AG-C11

Topics: **I:** Bisection method: interval bracketing, stopping criteria, and error bound **II:** Newton-Raphson method: derivation, quadratic convergence, and failure cases **III:** Secant method as an alternative to Newton's method without an explicit derivative **IV:** Order of convergence and efficiency comparison among iterative methods **V:** Extension of Newton's method to systems of nonlinear equations via the Jacobian matrix, the basis of solvers used in machine learning

Learning Outcomes **I:** Compare the convergence guarantees of the bisection, Newton-Raphson, and secant methods [Familiarity] **II:** Implement the Newton-Raphson method to approximate roots of a nonlinear function [Usage] **III:** Analyze the order of convergence of an iterative method and justify its choice based on computational cost [Assessment]

Bibliography: [Chapra and Canale, 2015, Burden and Faires, 2010]

5.3 Approximation Theory and Interpolation, 10 hr(s)

Competences: AG-C08,AG-C11

Topics: **I:** Polynomial interpolation: Lagrange, Newton divided differences, and Runge's phenomenon **II:** Spline interpolation: cubic splines, B-splines, and piecewise polynomial methods **III:** Best approximation in normed spaces: Chebyshev (minimax) and least squares **IV:** Chebyshev polynomials: properties, orthogonality, and spectral convergence **V:** Trigonometric approximation and the fast Fourier transform (FFT)

Learning Outcomes **I:** Explain Runge's phenomenon and justify the choice of Chebyshev nodes to mitigate it [Familiarity] **II:** Construct cubic spline and Chebyshev interpolants for given data and estimate the interpolation error [Usage] **III:** Apply the FFT to efficiently compute trigonometric approximations of a sampled function [Assessment]

Bibliography: [Chapra and Canale, 2015, Burden and Faires, 2010]

5.4 Numerical Integration and Quadrature, 10 hr(s)

Competences: AG-C08,AG-C11

Topics: **I:** Newton-Cotes rules: trapezoidal, Simpson's, and composite rules; error analysis **II:** Adaptive quadrature and automatic error control **III:** Gaussian quadrature: optimal nodes and weights, orthogonal polynomial connection **IV:** Numerical treatment of improper and singular integrals **V:** Monte Carlo integration and quasi-Monte Carlo methods for high-dimensional integrals

Learning Outcomes **I:** Derive the error formula for composite Simpson's rule and identify its order of accuracy [Familiarity] **II:** Select and apply an appropriate quadrature rule (Gaussian, adaptive) based on integrand regularity [Usage] **III:** Apply Monte Carlo integration to estimate high-dimensional integrals and quantify the statistical error [Assessment]

Bibliography: [Chapra and Canale, 2015, Burden and Faires, 2010]

5.5 Numerical Methods for Differential Equations, 12 hr(s)

Competences: AG-C08,AG-C11

Topics: **I:** Runge-Kutta methods: Euler, RK4, and embedded methods for error control **II:** Linear multistep methods: Adams-Bashforth, Adams-Moulton, and BDF methods **III:** Stability analysis: zero-stability, absolute stability regions, and stiff ODEs **IV:** Finite difference methods for parabolic and elliptic PDEs: stability and convergence

Learning Outcomes **I:** Compare explicit and implicit ODE solvers and explain their relative stability for stiff problems [Familiarity] **II:** Implement a Runge-Kutta method with adaptive step control and apply it to a system of ODEs [Usage] **III:** Analyze the stability and convergence of a finite difference scheme for a parabolic PDE [Assessment]

Bibliography: [Chapra and Canale, 2015, Burden and Faires, 2010]

5.6 Numerical Linear Algebra, 12 hr(s)

Competences: AG-C08,AG-C11

Topics: **I:** Gaussian elimination with partial pivoting, LU factorization, and complexity analysis **II:** QR factorization via Householder and Gram-Schmidt; least squares problems **III:** Eigenvalue algorithms: power iteration, QR algorithm, and the Lanczos method **IV:** Singular value decomposition: computation, truncation, and applications (PCA, pseudoinverse) **V:** Iterative methods for large sparse systems: CG, GMRES, and preconditioning

Learning Outcomes **I:** Compare direct and iterative solvers and select the appropriate method for a given matrix structure [Familiarity] **II:** Apply QR factorization and SVD to solve least squares problems and compute low-rank approximations [Usage] **III:** Analyze the numerical stability and computational cost of eigenvalue algorithms [Assessment]

Bibliography: [Burden and Faires, 2010, Trefethen and III, 1997]

5.7 Finite Element Method, 8 hr(s)

Competences: AG-C08,AG-C09

Topics: **I:** Weak (variational) formulation of boundary value problems and Sobolev spaces **II:** Galerkin method: finite-dimensional approximation spaces and the stiffness matrix **III:** Triangular and quadrilateral finite element spaces: Lagrange elements and conformity **IV:** A priori error estimates: Céa's lemma and interpolation error bounds **V:** A posteriori error estimates and adaptive mesh refinement

Learning Outcomes **I:** Derive the weak formulation of an elliptic BVP and show its equivalence to the strong form [Familiarity] **II:** Assemble the global stiffness matrix and load vector for a linear finite element discretization [Usage] **III:** Estimate the H^1 error of a finite element solution using Céa's lemma and interpolation theory [Assessment]

Bibliography: [Reddy, 2019, Burden and Faires, 2010]

5.8 Optimization Algorithms, 4 hr(s)

Competences: AG-C09,AG-C11

Topics: **I:** Gradient descent and line search methods: Armijo-Wolfe conditions and convergence rates **II:** Newton's method and quasi-Newton methods (BFGS, L-BFGS) **III:** Constrained optimization: KKT conditions, penalty methods, and sequential quadratic programming **IV:** Convex optimization algorithms: interior-point methods and ADMM **V:** Stochastic gradient descent (SGD), variance reduction, and Adam optimizer

Learning Outcomes **I:** Explain the convergence guarantees of gradient descent for smooth convex functions [Familiarity] **II:** Apply Newton and quasi-Newton methods to unconstrained optimization problems [Usage] **III:** Formulate a constrained optimization problem as a KKT system and apply an interior-point method [Assessment]

Bibliography: [Nocedal and Wright, 2006]

5.9 Parallel and High-Performance Computing, 4 hr(s)

Competences: AG-C11

Topics: **I:** Parallel architectures: shared memory (OpenMP), distributed memory (MPI), and GPU (CUDA) **II:** Performance models: Amdahl's law, roofline model, and communication complexity **III:** Parallel algorithms for dense and sparse linear algebra (ScaLAPACK, PETSc) **IV:** Domain decomposition and parallel PDE solvers **V:** Automatic differentiation (forward and reverse mode) and its role in scientific ML

Learning Outcomes **I:** Explain Amdahl's law and identify the bottlenecks limiting parallel speedup in a given algorithm [Familiarity] **II:** Implement a parallel linear algebra routine using MPI or OpenMP and measure parallel efficiency [Usage] **III:** Apply automatic differentiation to compute gradients of complex scientific functions for optimization [Assessment]

Bibliography: [Pacheco and Malensek, 2022]

5.10 Graph Theory, 4 hr(s)

Competences: AG-C08

Topics: **I:** Graphs: definitions, isomorphism, degree sequences, trees, and spanning trees **II:** Connectivity, Menger's theorem, and network flows (max-flow min-cut) **III:** Matchings (Hall's theorem), graph colorings, and the chromatic polynomial **IV:** Planar graphs, Euler's formula, Kuratowski's theorem, and the four-color theorem **V:** Adjacency and Laplacian matrices, eigenvalues, and expander graphs

Learning Outcomes **I:** Identify structural properties of graphs (connectivity, planarity, bipartiteness) and apply Euler's formula [Familiarity] **II:** Apply Hall's marriage theorem and network flow algorithms to matching and routing problems [Usage] **III:** Analyze a graph's spectrum to bound its chromatic number and connectivity properties [Assessment]

Bibliography: [West, 2001]

5.11 Time Series Analysis, 2 hr(s)

Competences: AG-C08,AG-C11

Topics: **I:** Stationarity, autocovariance function, and the autocorrelation function (ACF) **II:** ARMA models: identification, estimation (Yule-Walker, MLE), and diagnostics **III:** Spectral density, the periodogram, and Wiener-Khinchin theorem **IV:** State-space models and the Kalman filter **V:** ARIMA models, seasonal adjustment, and multi-step forecasting

Learning Outcomes **I:** Identify stationarity and determine the order of an ARMA model from ACF and PACF plots [Familiarity] **II:** Fit ARIMA models to time series data, validate residuals, and produce forecasts [Usage] **III:** Apply the Kalman filter to estimate hidden states in a linear Gaussian state-space model [Assessment]

Bibliography: [Shumway and Stoffer, 2017]

6. Bibliography

[Burden and Faires, 2010] Burden, R. L. and Faires, J. D. (2010). *Numerical Analysis*. Cengage Learning.

[Chapra and Canale, 2015] Chapra, S. C. and Canale, R. P. (2015). *Numerical Methods for Engineers*. McGraw-Hill Education.

[Nocedal and Wright, 2006] Nocedal, J. and Wright, S. J. (2006). *Numerical Optimization*. Springer, 2nd edition.

[Pacheco and Malensek, 2022] Pacheco, P. S. and Malensek, M. (2022). *An Introduction to Parallel Programming*. Morgan Kaufmann, 2nd edition.

[Reddy, 2019] Reddy, J. N. (2019). *An Introduction to the Finite Element Method*. McGraw-Hill Education, 4th edition.

[Shumway and Stoffer, 2017] Shumway, R. H. and Stoffer, D. S. (2017). *Time Series Analysis and Its Applications: With R Examples*. Springer, 4th edition.

[Trefethen and III, 1997] Trefethen, L. N. and III, D. B. (1997). *Numerical Linear Algebra*. SIAM.

[West, 2001] West, D. B. (2001). *Introduction to Graph Theory*. Prentice Hall, 2nd edition.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

BEI202 Technical and professional English IV (Elective)

1. General information

- **Semester**: 4th Sem. **Credits**: 2
- **Hour of this course**: **Practice**: 4 hours;
- **Prerrequisites**:
  - BEI201 Technical and professional English III (3rd Sem-Pg. 309)

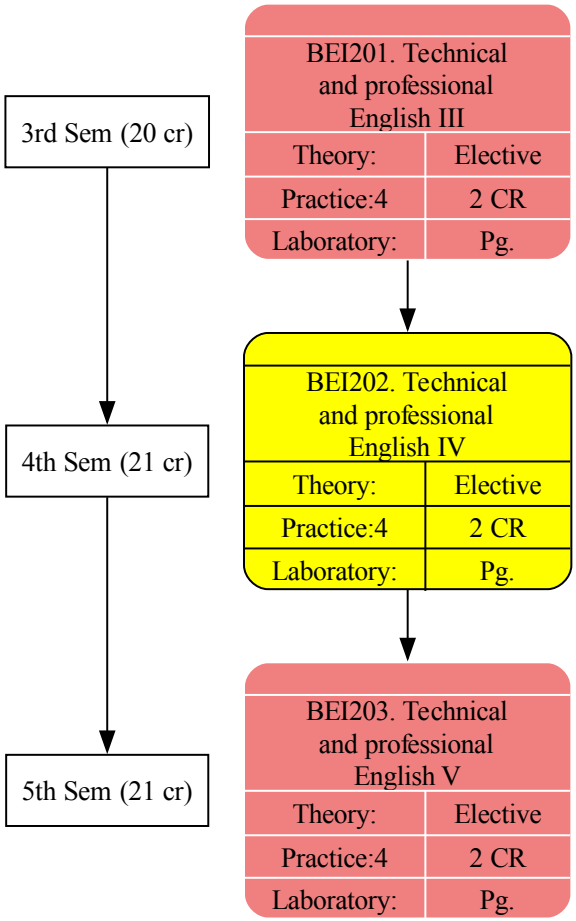

Figure 5.24: Connection Map. BEI202 Technical and professional English IV

2. Course Rationale

At the intermediate level, students apply English to the full range of technical communication required in computing and engineering practice: reading research papers, writing structured technical documents, analyzing case studies, and engaging in professional dialogue. Grammar structures are taught through authentic computing and engineering contexts, reinforcing both linguistic competence and disciplinary knowledge.

3. Goals of the Course

- Develop the ability to write and read structured technical documents in English.
- Bring the student to a more fluent and natural expression in technical and scientific English.
- Master vocabulary for computing, systems engineering, and applied sciences.

4. General Competences (*Outcomes*)

AG-C03) Individual and Team Work: Performs effectively as an individual and as a member or leader in diverse teams. (Familiarity)

AG-C04) Communication: Communicates effectively in complex computing activities. (Usage)

5. Content

5.1 Best Practices and Ethics, 12 hr(s)

Competences: AG-C03,AG-C04

Topics: **I:** Modals of obligation and advice: Should, Must, Have to in engineering codes. **II:** Prohibition and absence of obligation in software licensing and data privacy. **III:** Vocabulary: Professional ethics, intellectual property, responsible AI. **IV:** Writing: Formal technical recommendation and policy documents.

Learning Outcomes **I:** By the end of the unit, students correctly use modal auxiliaries to express engineering standards, ethical obligations, and professional recommendations in formal technical environments.

Bibliography: [? line:47, ? line:47, ? line:47]

5.2 Failures and Root Causes, 12 hr(s)

Competences: AG-C03,AG-C04

Topics: **I:** Past Perfect: Structure and use in root cause analysis and post-mortems. **II:** Contrast between Simple Past, Continuous, and Perfect in failure timelines. **III:** Vocabulary: System outages, security breaches, cascading failures. **IV:** Expressions for reporting information: reported speech in incident briefings.

Learning Outcomes **I:** Students reconstruct complex failure timelines and root cause analyses by establishing clear chronological sequences through the use of past tense contrasts.

Bibliography: [? line:60, ? line:60, ? line:60]

5.3 If-Then Logic, 12 hr(s)

Competences: AG-C03,AG-C04

Topics: **I:** Conditionals Type 0, 1, and 2 applied to computing logic and engineering decisions. **II:** Sentences with "If" and "Unless" for system constraints and requirements. **III:** Vocabulary: Hypothetical scenarios, risk analysis, and optimization trade-offs. **IV:** Expressions of what-if design thinking: "If we used X, the system would..."

Learning Outcomes **I:** Students pose real and hypothetical technical scenarios and their consequences, using conditional structures to reason about design choices and engineering decisions.

Bibliography: [? line:73, ? line:73, ? line:73]

5.4 Ongoing Research, 12 hr(s)

Competences: AG-C03,AG-C04

Topics: **I:** Present Perfect Continuous for describing research in progress. **II:** Contrast between Simple and Continuous Present Perfect in scientific writing. **III:** Vocabulary: Continuous integration, agile development, iterative research. **IV:** Word formation: Prefixes and suffixes in scientific and technical terminology.

Learning Outcomes **I:** Students describe research activities and development processes that began in the past and continue into the present, emphasizing duration and iterative progress.

Bibliography: [**?** line:86, **?** line:86, **?** line:86]

5.5 Manufacturing and Deployment, 12 hr(s)

Competences: AG-C03,AG-C04

Topics: **I:** Passive Voice in all basic tenses for technical process descriptions. **II:** Verbs with two objects in passive: data is sent to the server by the client. **III:** Vocabulary: Hardware manufacturing, cloud deployment, CI/CD pipelines. **IV:** Writing: Technical description of a deployment or production process.

Learning Outcomes **I:** Students use the passive voice across multiple tenses to describe technical processes where the action, system, or result is the focus rather than the agent.

Bibliography: [**?** line:99, **?** line:99, **?** line:99]

5.6 Tech Roadmaps, 12 hr(s)

Competences: AG-C03,AG-C04

Topics: **I:** Future Continuous and Future Perfect for project timelines and milestones. **II:** Adverbs of probability: perhaps, probably, definitely — in tech forecasting. **III:** Vocabulary: Product roadmaps, release planning, emerging research areas. **IV:** Oral presentations: "The state of AI / quantum computing in 5 years".

Learning Outcomes **I:** Students project technical states, completed future milestones, and research outcomes, discussing computing and engineering roadmaps with advanced vocabulary.

Bibliography: [**?** line:112, **?** line:112, **?** line:112]

5.7 Technical Teams, 12 hr(s)

Competences: AG-C03,AG-C04

Topics: **I:** Indirect questions and Tag questions in technical interviews and peer reviews. **II:** Common Phrasal Verbs in engineering and research environments. **III:** Vocabulary: Collaboration tools, code review practices, team roles. **IV:** Writing: Short technical blog posts and engineering team updates.

Learning Outcomes **I:** Students use polite inquiry forms and confirmation strategies in technical interviews and team discussions, improving their fluency and naturalness in professional engineering contexts.

Bibliography: [**?** line:125, **?** line:125, **?** line:125]

5.8 Technical English IV: Networks and Security, 12 hr(s)

Competences: AG-C03,AG-C04

Topics: **I:** Networking terminology: protocols, topologies, OSI model, routing. **II:** Cybersecurity vocabulary: vulnerabilities, threats, encryption, authentication. **III:** Reading and writing network incident and security audit reports. **IV:** Project: Security assessment plan for a computing infrastructure.

Learning Outcomes I: By the end of the course, students command networking and cybersecurity technical terminology to read specialized documentation and communicate effectively with IT and security professionals in English.

Bibliography: [**?** line:138 , **?** line:138 , **?** line:138]

6. Bibliography

line:143 line:143 line:143

Fifth Semester

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

CS212 Analysis and Design of Algorithms (Mandatory)

1. General information

- **Semester**: 5th Sem. **Credits**: 4
- **Hour of this course**: **Theory**: 2 hours; **Practice**: 2 hours; **Laboratory**: 2 hours;
- **Prerrequisites**:
  - CS210 Algorithms and Data Structures (4th Sem-Pg. 313)
  - CS211 Theory of Computation (4th Sem-Pg. 318)

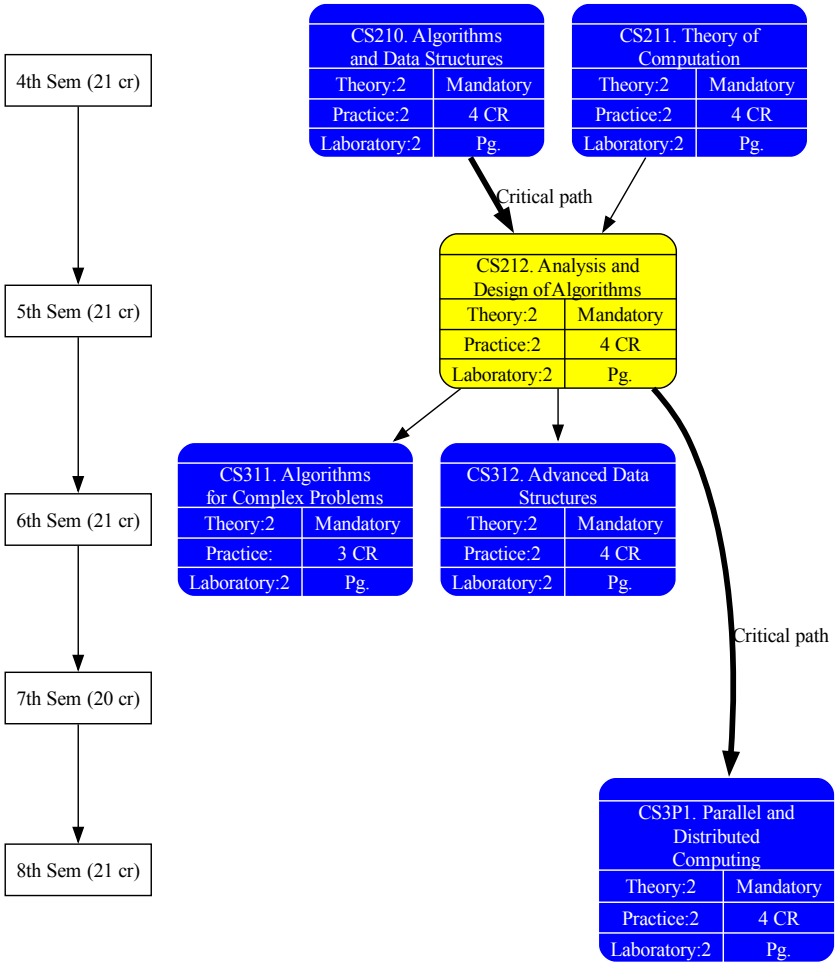

Figure 5.25: Connection Map. CS212 Analysis and Design of Algorithms

2. Course Rationale

Algorithms are the heart of computer science. This course provides the mathematical and analytical tools necessary to evaluate the efficiency of algorithms in terms of time and space. Students will explore fundamental data structures, complexity analysis, and various design paradigms—such

as brute force, greedy, and dynamic programming. Understanding these techniques is essential for building software that is not only functional but also scalable and resource-efficient.

3. Goals of the Course

- Master the use of asymptotic notation to analyze algorithm performance.
- Apply foundational data structures to solve computational problems.
- Use algorithmic strategies like Greedy, Divide and Conquer, and Dynamic Programming.
- Analyze the complexity of algorithms in best, average, and worst cases.
- Develop the ability to select the most appropriate algorithm for a given problem.

4. General Competences (*Outcomes*)

AG-C08) Problem Analysis: Identifies, formulates, and analyzes complex computing problems. (Usage)

AG-C12) Applies computer science theory and software development fundamentals to produce computer-based solutions. (Usage)

5. Content

5.1 Foundational Data Structures, 6 hr(s)

Competences: AG-C08,AG-C12

Topics: **I:** Abstract Data Type (ADT) and operations on an ADT: (a) Dictionary operations (insert, delete, find) **II:** Arrays: (a) Numeric vs non-numeric, character strings (b) Single (vector) vs multidimensional (matrix) **III:** Records/Structs/Tuples and Objects **IV:** Linked lists (for historical reasons): (a) Single vs Double and Linear vs Circular **V:** Stacks **VI:** Queues and deques: (a) Heap-based priority queue **VII:** Hash tables/maps: (a) Collision resolution and complexity (e.g., probing, chaining, rehash) **VIII:** Trees: (a) Binary, n-ary, and search trees (b) Balanced (e.g., AVL, Red-Black, Heap) **IX:** Sets

Learning Outcomes **I:** For each ADT/Data-Structure in this unit: (a) Explain its definition, properties, representation(s), and associated ADT operations. (b) Explain step-by-step how the ADT operations associated with the data structure transform it. [Explain] **II:** Explain how collision avoidance and collision resolution is handled in hash tables [Explain] **III:** Explain the heap property and the use of heaps as an implementation of a priority queue [Explain]

Bibliography: [Cormen et al., 2022, Kleinberg and Tardos, 2005]

5.2 Foundational Algorithms, 6 hr(s)

Competences: AG-C08,AG-C12

Topics: **I:** Graphs (e.g., [un]directed, [a]cyclic, [un]connected, and [un]weighted): (a) Graph representation: adjacency list vs matrix **II:** Search algorithms: (a) $O(n)$ complexity (e.g., linear/sequential array/list search) (b) $O(\log_2 n)$ complexity (e.g., binary search) (c) $O(\log_b n)$ complexity (e.g., uninformed depth/breadth-first tree search) **III:** Sorting algorithms (e.g., stable, unstable): (a) $O(n^2)$ complexity (e.g., insertion, selection) (b) $O(n \log n)$ complexity (e.g., quicksort, merge, timsort) **IV:** Graph algorithms: (a) Shortest path (e.g., Dijkstra's, Floyd's) (b) Minimal spanning tree (e.g., Prim's, Kruskal's) **V:** Sorting algorithms: (a) $O(n \log n)$ complexity heapsort (b) Pseudo $O(n)$ complexity (e.g., bucket, counting, radix) **VI:** Graph algorithms: (a) Transitive closure (e.g., Warshall's) (b) Topological sort **VII:** Matching: (a) Efficient string matching (e.g., Boyer-Moore, Knuth-Morris-Pratt) (b) Longest common subsequence matching (c) Regular expression matching

Learning Outcomes **I:** For each algorithm in this unit explain step-by-step how the algorithm operates [Explain] **II:** For each algorithmic approach (e.g., sorting) in this unit apply a prototypical example of the approach (e.g., merge sort) [Apply] **III:** Given requirements for a problem,

develop multiple solutions using various data structures and algorithms. Subsequently, evaluate the suitability, strengths, and weaknesses selecting an approach that best satisfies the requirements [Create] **IV:** Explain factors beyond computational efficiency that influence the choice of algorithms, such as programming time, maintainability, and the use of application-specific patterns in the input data [Explain] **V:** For each of the algorithms and algorithmic approaches in the KA Core topics: (a) Explain a prototypical example of the algorithm, and (b) Explain step-by-step how the algorithm operates. [Explain]

Bibliography: [Cormen et al., 2022, Kleinberg and Tardos, 2005]

5.3 Advanced Algorithms, 4 hr(s)

Competences: AG-C08,AG-C12

Topics: I: Cryptography algorithms (e.g., SHA-256) **II:** Parallel algorithms **III:** Consensus algorithms (e.g., Blockchain): (a) Proof of work vs proof of stake **IV:** Quantum computing algorithms: (a) Oracle-based (e.g., Deutsch-Jozsa, Bernstein-Vazirani, Simon) (b) Superpolynomial speed-up via QFT (e.g., Shor's) (c) Polynomial speed-up via amplitude amplification (e.g., Grover's) **V:** Fast-Fourier Transform (FFT) algorithm **VI:** Differential evolution algorithm

Learning Outcomes I: An appreciation of quantum computation and its application to certain problems [Explain]

Bibliography: [Cormen et al., 2022, Kleinberg and Tardos, 2005]

5.4 Algorithmic Strategies, 24 hr(s)

Competences: AG-C08,AG-C12

Topics: I: Paradigms: (a) Brute-Force (e.g., linear search, selection sort, traveling salesperson, knapsack) (b) Decrease-and-Conquer: i. By a Constant (e.g., insertion sort, topological sort) ii. By a Constant Factor (e.g., binary search) iii. By a Variable Size (e.g., Euclid's) (c) Divide-and-Conquer (e.g., binary search, quicksort, mergesort, Strassen's) (d) Greedy (e.g., Dijkstra's, Kruskal's, Knapsack) (e) Transform-and-Conquer: i. Instance simplification (e.g., find duplicates via list presort) ii. Representation change (e.g., heapsort) iii. Problem reduction (e.g., least-common-multiple, linear programming) iv. Dynamic programming (e.g., Floyd's, Marshall, Bellman-Ford) (f) Space vs time tradeoffs (e.g., hashing) **II:** Handling exponential growth (e.g., heuristic A*, branch-and-bound, backtracking) **III:** Iteration vs recursion (e.g., factorial, tree search) **IV:** Paradigms: (a) Approximation algorithms (b) Iterative improvement (e.g., Ford-Fulkerson, simplex) (c) Randomized/Stochastic algorithms (e.g., max-cut, balls and bins) **V:** Quantum computing

Learning Outcomes I: For each of the paradigms in this unit: (a) Explain its definitional characteristics (b) Explain an example that demonstrates the paradigm including how this example satisfies the paradigm's characteristics. [Explain] **II:** For each of the algorithms in the Foundational Data Structures, Foundational Algorithms, Advanced Algorithms unit, explain the paradigm used by the algorithm and how it exemplifies this paradigm [Explain] **III:** Given an algorithm, explain the paradigm used by the algorithm and how it exemplifies this paradigm [Explain] **IV:** Give a real-world problem, evaluate appropriate algorithmic paradigms and algorithms from these paradigms that address the problem including evaluating the tradeoffs among the paradigms and algorithms selected [Evaluate] **V:** Give examples of iterative and recursive algorithms that solve the same problem, explain the benefits and disadvantages of each approach [Explain] **VI:** Evaluate whether a greedy approach leads to an optimal solution [Evaluate] **VII:** Explain various approaches for addressing computational problems whose algorithmic solutions are exponential [Explain]

Bibliography: [Cormen et al., 2022, Dasgupta et al., 2006]

5.5 Complexity Analysis Framework, 6 hr(s)

Competences: AG-C08,AG-C12

Topics: **I:** Complexity Analysis Framework: (a) Best, average, and worst-case performance of an algorithm (b) Empirical and relative (Order of Growth) measurements (c) Input size and primitive operations (d) Time and space efficiency **II:** Empirical measurements of performance **III:** Time and space tradeoffs in algorithms

Learning Outcomes **I:** For each algorithm in the Foundational Data Structures, Foundational Algorithms, Advanced Algorithms unit, explain its runtime complexity class and why it belongs to this class [Explain] **II:** Develop empirical studies to determine and validate hypotheses about the runtime complexity of various algorithms by running algorithms on input of various sizes and comparing actual performance to the theoretical analysis [Create] **III:** Explain examples that illustrate time-space tradeoffs of algorithms [Explain] **IV:** Explain how tree balance affects the efficiency of binary search tree operations [Explain]

Bibliography: [Cormen et al., 2022, Sipser, 2012]

5.6 Asymptotic Notation and Complexity Classes, 6 hr(s)

Competences: AG-C08,AG-C12

Topics: **I:** Asymptotic complexity analysis (average and worst-case bounds): (a) Big-O, Big-Omega, and Big-Theta formal notations (b) Foundational Complexity Classes and Representative Examples/Problems: i. $O(1)$ Constant (e.g., array access) ii. $O(\log_2 n)$ Logarithmic (e.g., binary search) iii. $O(n)$ Linear (e.g., linear search) iv. $O(n \log_2 n)$ Log Linear (e.g., mergesort) v. $O(n^2)$ Quadratic (e.g., selection sort) vi. $O(n^c)$ Polynomial (e.g., $O(n^3)$ Gaussian elimination) vii. $O(2^n)$ Exponential (e.g., Knapsack, Satisfiability (SAT), Traveling Sales-Person (TSP), all subsets) viii. $O(n!)$ Factorial (e.g., Hamiltonian circuit, all permutations)

Learning Outcomes **I:** For each foundational complexity class in this unit, explain an algorithm that demonstrates the associated runtime complexity [Explain] **II:** Apply Big-O notation to give upper case bounds on time/space complexity of algorithms [Apply]

Bibliography: [Cormen et al., 2022, Sipser, 2012]

5.7 Complexity Analysis: Recursion, Amortization, and Tight Bounds, 12 hr(s)

Competences: AG-C08,AG-C12

Topics: **I:** Little-o, Little-Omega, and Little Theta notations **II:** Formal recursive analysis **III:** Amortized analysis

Learning Outcomes **I:** Given a problem to program for which there may be several algorithmic approaches, evaluate them and determine which are feasible, and select one that is optimal in implementation and run-time behavior [Evaluate] **II:** Use recurrence relations to evaluate the time complexity of recursively defined algorithms [Apply] **III:** Apply elementary recurrence relations using a form of the Master Theorem [Apply]

Bibliography: [Cormen et al., 2022, Sipser, 2012]

5.8 Computational Complexity Theory, 6 hr(s)

Competences:

Topics: **I:** Tractability and intractability: (a) P, NP, and NP-Complete Complexity Classes (b) NP-Complete Problems (e.g., SAT, Knapsack, TSP) (c) Reductions

Learning Outcomes **I:** Explain the significance of NP-Completeness [Explain] **II:** Explain examples of NP-complete problems [Explain] **III:** Explain the classes P and NP [Explain]

Bibliography: [Sipser, 2012, Cormen et al., 2022]

6. Bibliography

[Cormen et al., 2022] Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2022). *Introduction to Algorithms*. MIT Press, 4th edition.

[Dasgupta et al., 2006] Dasgupta, S., Papadimitriou, C., and Vazirani, U. (2006). *Algorithms*. McGraw-Hill Education.

[Kleinberg and Tardos, 2005] Kleinberg, J. and Tardos, É. (2005). *Algorithm Design*. Pearson.

[Sipser, 2012] Sipser, M. (2012). *Introduction to the Theory of Computation*. Cengage Learning, 3rd edition.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

CS261 Artificial Intelligence (Mandatory)

1. General information

- **Semester**: 5th Sem. **Credits**: 4
- **Hour of this course**: **Theory**: 2 hours; **Practice**: 2 hours; **Laboratory**: 2 hours;
- **Prerrequisites**:
  - STA251 Probability Calculation (3rd Sem-Pg. 304)

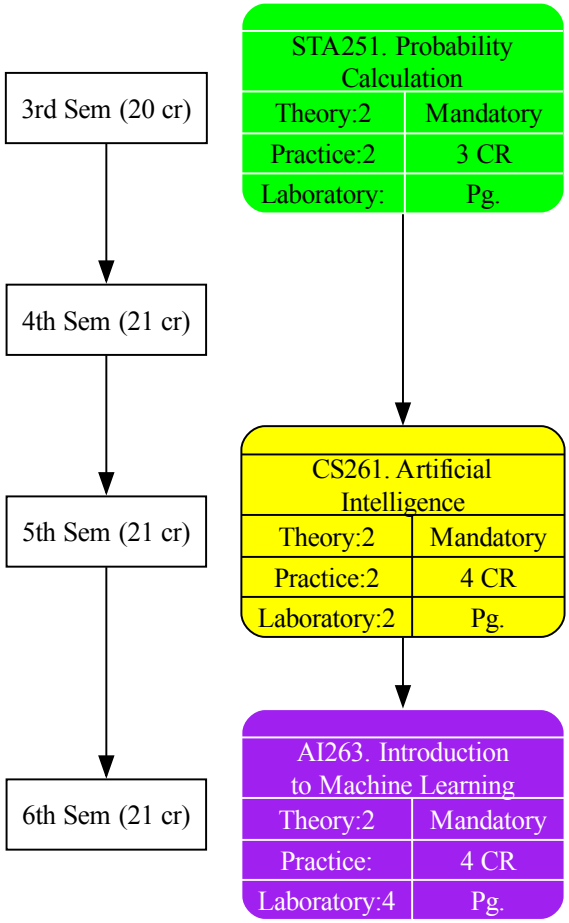

Figure 5.26: Connection Map. CS261 Artificial Intelligence

2. Course Rationale

Artificial Intelligence (AI) is a fundamental field of computer science that seeks to create systems capable of performing tasks that typically require human intelligence. This course provides an introduction to the techniques used to build such systems, including search algorithms, probabilistic reasoning, and machine learning. Students will learn to model complex problems and implement intelligent agents, while considering the social and ethical implications of these technologies.

3. Goals of the Course

- Understand the fundamental concepts and history of Artificial Intelligence.
- Implement and evaluate search algorithms for problem-solving.
- Master the basics of machine learning, including supervised and unsupervised learning.
- Apply probabilistic reasoning to handle uncertainty in intelligent systems.
- Analyze the ethical challenges and social impact of AI.

4. General Competences (*Outcomes*)

AG-C08) Problem Analysis: Identifies, formulates, and analyzes complex computing problems. (Usage)

AG-C09) Design and Development of Solutions: Designs, implements, and evaluates solutions for complex computing problems. (Usage)

5. Content

5.1 Introduction to AI: Concepts and Intelligent Behavior, 4 hr(s)

Competences: AG-C08,AG-C09

Topics: **I:** Overview of AI problems, Examples of successful recent AI applications **II:** Definitions of agents with examples (e.g., reactive, deliberative) **III:** What is intelligent behavior?: (a) The Turing test and its flaws (b) Multimodal input and output (c) Simulation of intelligent behavior (d) Rational versus non-rational reasoning **IV:** Overview of AI Applications, growth, and impact (economic, societal, ethics) **V:** Philosophical issues **VI:** History of AI

Learning Outcomes **I:** Describe the Turing test and the "Chinese Room" thought experiment [Explain] **II:** Differentiate between optimal reasoning/behavior and human-like reasoning/behavior [Evaluate] **III:** Differentiate the terms: AI, machine learning, and deep learning [Explain]

Bibliography: [Russell and Norvig, 2020, Luger, 2008]

5.2 Problem Characteristics and Agent Nature, 4 hr(s)

Competences: AG-C08,AG-C09

Topics: **I:** Problem characteristics: (a) Fully versus partially observable (b) Single versus multi-agent (c) Deterministic versus stochastic (d) Static versus dynamic (e) Discrete versus continuous **II:** Nature of agents: (a) Autonomous, semi-autonomous, mixed-initiative autonomy (b) Reflexive, goal-based, and utility-based (c) Decision making under uncertainty and with incomplete information (d) The importance of perception and environmental interactions (e) Learning-based agents (f) Embodied agents: i. sensors, dynamics, effectors **III:** Practice identifying problem characteristics in example environments **IV:** Additional depth on nature of agents with examples **V:** Additional depth on AI Applications, Growth, and Impact (economic, societal, ethics, security)

Learning Outcomes **I:** Enumerate the characteristics of a specific problem [Apply]

Bibliography: [Russell and Norvig, 2020, Luger, 2008]

5.3 State and Uninformed Search, 5 hr(s)

Competences: AG-C08,AG-C09

Topics: **I:** State space representation of a problem: (a) Specifying states, goals, and operators (b) Factoring states into representations (hypothesis spaces) (c) Problem solving by graph search: i. e.g., Graphs as a space, and tree traversals as exploration of that space ii. Dynamic construction of the graph (not given upfront) **II:** Uninformed graph search for problem solving: (a) Breadth-first

search (b) Depth-first search: i. With iterative deepening (c) Uniform cost search **III:** Space and time complexities of graph search algorithms **IV:** Bidirectional search **V:** Understanding the search space: (a) Constructing search trees (b) Dynamic search spaces (c) Combinatorial explosion of search space (d) Search space topology (e.g., ridges, saddle points, local minima)

Learning Outcomes I: Design the state space representation for a puzzle (e.g., N-queens or 3-jug problem) [Apply] **II:** Select and implement an appropriate uninformed search algorithm for a problem (e.g., tic-tac-toe), and characterize its time and space complexities [Apply]

Bibliography: [Russell and Norvig, 2020, Luger, 2008]

5.4 Heuristic Search and CSP, 6 hr(s)

Competences: AG-C08,AG-C09

Topics: I: Heuristic graph search for problem solving: (a) Heuristic construction and admissibility (b) Hill-climbing (c) Local minima and the search landscape: i. Local vs global solutions (d) Greedy best-first search (e) A* search **II:** Beam search **III:** Implementation of A* search **IV:** Constraint satisfaction **V:** Variations on A* (IDA*, SMA*, RBFS)

Learning Outcomes I: Select and implement an appropriate informed search algorithm for a problem after designing a helpful heuristic function (e.g., a robot navigating a 2D gridworld) [Apply] **II:** Evaluate whether a heuristic for a given problem is admissible/can guarantee an optimal solution [Evaluate] **III:** Design and implement A*/beam search to solve a problem, and compare it against other search algorithms in terms of the solution cost, number of nodes expanded, etc [Apply] **IV:** Compare and contrast various heuristic searches vis-a-vis applicability to a given problem [Evaluate] **V:** Model a logic or Sudoku puzzle as a constraint satisfaction problem, solve it with backtrack search, and determine how much arc consistency can reduce the search space [Apply]

Bibliography: [Russell and Norvig, 2020, Luger, 2008]

5.5 Adversarial Search and Stochastic Optimization, 5 hr(s)

Competences: AG-C08,AG-C09

Topics: I: Two-player adversarial games: (a) Minimax search (b) Alpha-beta pruning: i. Ply cutoff **II:** Local search **III:** Tabu search **IV:** Two-player adversarial games: (a) The horizon effect (b) Opening playbooks/endgame solutions (c) What it means to ”solve” a game (e.g., checkers) **V:** Implementation of minimax search, beam search **VI:** Expectimax search (MDP-solving) and chance nodes **VII:** Stochastic search: (a) Simulated annealing (b) Genetic algorithms (c) Monte-Carlo tree search

Learning Outcomes I: Apply minimax search in a two-player adversarial game (e.g., connect four), using heuristic evaluation at a particular depth to compute the scores to back up [Apply] **II:** Apply minimax search with alpha-beta pruning to prune search space in a two-player adversarial game (e.g., connect four) [Apply] **III:** Compare and contrast genetic algorithms with classic search techniques, explaining when it is most appropriate to use a genetic algorithm to learn a model versus other forms of optimization (e.g., gradient descent) [Evaluate] **IV:** Design and implement a genetic algorithm solution to a problem [Apply] **V:** Design and implement a simulated annealing schedule to avoid local minima in a problem [Apply]

Bibliography: [Russell and Norvig, 2020, Luger, 2008]

5.6 Machine Learning Fundamentals, 7 hr(s)

Competences: AG-C08,AG-C09

Topics: **I:** Definition and examples of a broad variety of machine learning tasks: (a) Supervised learning: i. Classification ii. Regression (b) Reinforcement learning (c) Unsupervised learning: i. Clustering **II:** Fundamental ideas: (a) No free lunch theorem: no one learner can solve all problems; representational design decisions have consequences. (b) Sources of error and undecidability in machine learning **III:** A simple statistical-based supervised learning such as linear regression or decision trees: (a) Focus on how they work without going into mathematical or optimization details; enough to understand and use existing implementations correctly **IV:** The overfitting problem/controlling solution complexity (regularization, pruning - intuition only): (a) The bias (underfitting) - variance (overfitting) tradeoff

Learning Outcomes **I:** Describe the differences among the three main styles of learning (supervised, reinforcement, and unsupervised) and determine which is appropriate to a particular problem domain [Explain] **II:** Differentiate the terms of AI, machine learning, and deep learning [Evaluate] **III:** Frame an application as a classification problem, including the available input features and output to be predicted (e.g., identifying alphabetic characters from pixel grid input) [Apply] **IV:** Identify overfitting in the context of a problem and learning curves and describe solutions to overfitting [Evaluate] **V:** Explain how machine learning works as an optimization/search process [Explain]

Bibliography: [Russell and Norvig, 2020, Alpaydin, 2020]

5.7 Machine Learning Data, Representations and Evaluation, 7 hr(s)

Competences: AG-C08,AG-C09

Topics: **I:** Working with Data: (a) Data preprocessing: i. Importance and pitfalls of preprocessing choices (b) Handling missing values (imputing, flag-as-missing): i. Implications of imputing vs flag-as-missing (c) Encoding categorical variables, encoding real-valued data (d) Normalization/standardization (e) Emphasis on real data, not textbook examples **II:** Representations: (a) Hypothesis spaces and complexity (b) Simple basis feature expansion, such as squaring univariate features (c) Learned feature representations **III:** Machine learning evaluation: (a) Separation of train, validation, and test sets (b) Performance metrics for classifiers (c) Estimation of test performance on held-out data (d) Tuning the parameters of a machine learning model with a validation set (e) Importance of understanding what a model is doing, where its pitfalls/shortcomings are, and the implications of its decisions

Learning Outcomes **I:** Apply two or more simple statistical learning algorithms to a classification task and measure the classifiers' accuracy [Apply] **II:** Implement a statistical learning algorithm and the corresponding optimization process to train the classifier and obtain a prediction on new data [Apply] **III:** Explain proper ML evaluation procedures, including the differences between training and testing performance, and what can go wrong with the evaluation process leading to inaccurate reporting of ML performance [Explain] **IV:** Compare two machine learning algorithms on a dataset, implementing the data preprocessing and evaluation methodology (e.g., metrics and handling of train/test splits) from scratch [Apply]

Bibliography: [Russell and Norvig, 2020, Alpaydin, 2020]

5.8 Neural Networks and Ethics, 6 hr(s)

Competences: AG-C08,AG-C09

Topics: **I:** Basic neural networks: (a) Fundamentals of understanding how neural networks work and their training process, without details of the calculations (b) Basic introduction to generative neural networks (e.g., large language models) **II:** Ethics for Machine Learning: (a) Focus on real data, real scenarios, and case studies (b) Dataset/algorithmic/evaluation bias and unintended consequences

Learning Outcomes **I:** Describe the neural network training process and resulting learned representations [Explain]

Bibliography: [Russell and Norvig, 2020, Alpaydin, 2020]

5.9 Machine Learning Formulation and Evaluation, 3 hr(s)

Competences: AG-C09

Topics: **I:** Formulation of simple machine learning as an optimization problem, such as least squares linear regression or logistic regression: (a) Objective function (b) Gradient descent (c) Regularization to avoid overfitting (mathematical formulation) **II:** Ensembles of models: (a) Simple weighted majority combination **III:** Performance evaluation: (a) Other metrics for classification (e.g., error, precision, recall) (b) Performance metrics for regressors (c) Confusion matrix (d) Cross-validation: i. Parameter tuning (grid/random search, via cross-validation)

Learning Outcomes **I:** Compare and contrast several learning techniques (e.g., decision trees, logistic regression, naive Bayes, neural networks, and belief networks), providing examples of when each strategy is superior [Compare] **II:** Evaluate the performance of a simple learning system on a real-world dataset [Evaluate] **III:** Characterize the state of the art in learning theory, including its achievements and shortcomings [Explain] **IV:** Explain the problem of overfitting, along with techniques for detecting and managing the problem [Explain] **V:** Explain the triple tradeoff among the size of a hypothesis space, the size of the training set, and performance accuracy [Explain]

Bibliography: [Goodfellow et al., 2016]

5.10 Deep Learning, RL and Ethics, 3 hr(s)

Competences: AG-C09

Topics: **I:** Deep learning: (a) Deep feed-forward networks (intuition only, no mathematics) (b) Convolutional neural networks (intuition only, no mathematics) (c) Visualization of learned feature representations from deep nets (d) Other architectures (generative NN, recurrent NN, transformers, etc.) **II:** Overview of reinforcement learning methods **III:** Two or more applications of machine learning algorithms: (a) E.g., medicine and health, economics, vision, natural language, robotics, game play **IV:** Ethics for Machine Learning: (a) Continued focus on real data, real scenarios, and case studies (b) Privacy (c) Fairness (d) Intellectual property (e) Explainability

Learning Outcomes **I:** Visualize the training progress of a neural network through learning curves in a well-established toolkit (e.g., TensorBoard) and visualize the learned features of the network [Apply] **II:** Given a real-world application of machine learning, describe ethical issues regarding the choices of data, preprocessing steps, algorithm selection, and visualization/presentation of results [Apply]

Bibliography: [Goodfellow et al., 2016]

5.11 Knowledge Representation and Probabilistic Reasoning Foundations, 6 hr(s)

Competences: AG-C08,AG-C09

Topics: **I:** Types of representations: (a) Symbolic, logical: i. Creating a representation from a natural language problem statement (b) Learned subsymbolic representations (c) Graphical models (e.g., naive Bayes, Bayesian network) **II:** Review of probabilistic reasoning, Bayes theorem **III:** Bayesian reasoning: (a) Bayesian inference **IV:** Random variables and probability distributions: (a) Axioms of probability (b) Probabilistic inference (c) Bayes' Rule (derivation) (d) Bayesian inference (more complex examples)

Learning Outcomes **I:** Given a natural language problem statement, encode it as a symbolic or logical representation [Apply] **II:** Compute a probabilistic inference in a real-world problem using Bayes' theorem to determine the probability of a hypothesis given evidence [Apply] **III:** Apply Bayes' rule to determine the probability of a hypothesis given evidence [Apply] **IV:** Compute the probability of outcomes and test whether outcomes are independent [Apply]

Bibliography: [Russell and Norvig, 2020]

5.12 Independence, Markov Models and Decision Making, 6 hr(s)

Competences: AG-C08,AG-C09

Topics: I: Independence **II:** Conditional Independence **III:** Markov chains and Markov models **IV:** Utility and decision making

Learning Outcomes I: Explain how we can make decisions under uncertainty, using concepts such as Bayes theorem and utility [Explain]

Bibliography: [Russell and Norvig, 2020]

5.13 AI Applications and Societal Impact, 4 hr(s)

Competences: AG-C08,AG-C09

Topics: I: At least one application of AI to a specific problem and field, such as medicine, health, sustainability, social media, economics, education, robotics, etc. (choose at least one for the CS Core): (a) Formulating and evaluating a specific application as an AI problem: i. How to deal with underspecified or ill-posed problems (b) Data availability/scarcity and cleanliness: i. Basic data cleaning and preprocessing ii. Data set bias (c) Algorithmic bias (d) Evaluation bias (e) Assessment of societal implications of the application **II:** Deployed deep generative models: (a) High-level overview of deep image generative models (e.g., as of 2023, DALL-E, Midjourney, Stable Diffusion, etc.), their uses, and their shortcomings/pitfalls. (b) High-level overview of large language models (e.g., as of 2023, ChatGPT, Bard, etc.), their uses, and their shortcomings/pitfalls. **III:** Overview of societal impact of AI: (a) Ethics (b) Fairness (c) Trust/explainability (d) Privacy and usage of training data (e) Human autonomy and oversight/regulations/legal requirements (f) Sustainability

Learning Outcomes I: Given a real-world application domain and problem, formulate an AI solution to it, identifying proper data/input, preprocessing, representations, AI techniques, and evaluation metrics/methodology [Apply]

Bibliography: [Russell and Norvig, 2020, Alpaydin, 2020]

5.14 Additional AI Applications and Societal Impact, 4 hr(s)

Competences: AG-C08,AG-C09

Topics: I: One or more additional applications of AI to a broad set of problems and diverse fields, such as medicine, health, sustainability, social media, economics, education, robotics, etc. (choose a different area from that chosen for the CS Core): (a) Formulating and evaluating a specific application as an AI problem: i. How to deal with underspecified or ill-posed problems (b) Data availability/scarcity and cleanliness: i. Basic data cleaning and preprocessing ii. Data set bias (c) Algorithmic bias (d) Evaluation bias (e) Assessment of societal implications of the application **II:** Additional depth on deployed deep generative models: (a) Introduction to how deep image generative models work, (e.g., as of 2023, DALL-E, Midjourney, Stable Diffusion) including discussion of attention (b) Introduction to how large language models work, (e.g., as of 2023, ChatGPT, Bard) including discussion of attention (c) Idea of foundational models, how to use them, and the benefits/issues with training them from big data **III:** Analysis and discussion of the societal impact of AI: (a) Ethics (b) Fairness (c) Trust/explainability (d) Privacy and usage of training data (e) Human autonomy and oversight/regulations/legal requirements (f) Sustainability

Learning Outcomes I: Analyze the societal impact of one or more specific real-world AI applications, identifying issues regarding ethics, fairness, bias, trust, and explainability [Evaluate] **II:** Describe some of the failure modes of current deep generative models for language or images, and how this could affect their use in an application [Describe]

Bibliography: [Russell and Norvig, 2020, Alpaydin, 2020]

6. Bibliography

[Alpaydin, 2020] Alpaydin, E. (2020). *Introduction to Machine Learning*. MIT Press, 4th edition.

[Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press.

[Luger, 2008] Luger, G. F. (2008). *Artificial Intelligence: Structures and Strategies for Complex Problem Solving*. Pearson, 6th edition.

[Russell and Norvig, 2020] Russell, S. and Norvig, P. (2020). *Artificial Intelligence: A Modern Approach*. Pearson, 4th edition.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

CS272 Databases II (Mandatory)

1. General information

- **Semester**: 5th Sem. **Credits**: 3
- **Hour of this course**: **Theory**: 1 hours; **Practice**: 2 hours; **Laboratory**: 2 hours;
- **Prerrequisites**:
  - CS271 Data Management (4th Sem-Pg. 327)

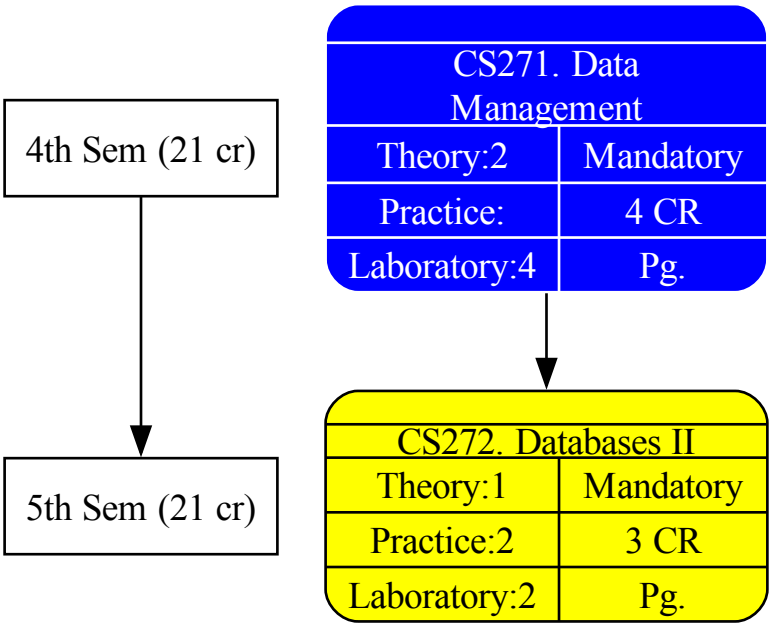

Figure 5.27: Connection Map. CS272 Databases II

2. Course Rationale

This course is a continuation of Data Management I, focusing on the internal mechanisms of database management systems (DBMS) and modern database technologies. It covers query optimization, indexing structures, and transactional integrity in depth. Additionally, it introduces students to non-relational models (NoSQL) and distributed architectures, which are essential for handling large-scale data in contemporary cloud environments.

3. Goals of the Course

- Understand the internal storage structures and indexing techniques.
- Analyze and optimize query execution plans.
- Comprehend advanced transaction protocols and recovery mechanisms.
- Evaluate and implement solutions using NoSQL database models.
- Understand the principles of distributed and parallel databases.

4. General Competences (*Outcomes*)

AG-C08) Problem Analysis: Identifies, formulates, and analyzes complex computing problems. (Usage)

AG-C12) Applies computer science theory and software development fundamentals to produce computer-based solutions. (Usage)

5. Content

5.1 Query Processing, 12 hr(s)

Competences: AG-C08,AG-C12

Topics: **I:** Page structures **II:** Index structures: (a) B+ trees (b) Hash indices: static and dynamic (c) Index creation in SQL **III:** File structures: (a) Heap files (b) Hash files **IV:** Algorithms for query operators: (a) External Sorting (b) Selection (c) Projection; with and without duplicate elimination (d) Natural Joins: Nested loop, Sort-merge, Hash join (e) Analysis of algorithm efficiency **V:** Query transformations **VI:** Query optimization: (a) Access paths (b) Query plan construction (c) Selectivity estimation (d) Index-only plans **VII:** Parallel Query Processing (e.g., parallel scan, parallel join, parallel aggregation) **VIII:** Database tuning/performance: (a) Index selection (b) Impact of indices on query performance (c) Denormalization

Learning Outcomes **I:** Describe the purpose and organization of both B+ tree and hash index structures [Describe] **II:** Compose an SQL command to create an index (any kind) [Compose] **III:** Specify the steps for the various query operator algorithms: external sorting, projection with duplicate elimination, sort-merge join, hash-join, block nested-loop join [Analyze] **IV:** Derive the run-time (in I/O requests) for each of the above algorithms [Analyze] **V:** Transform a query in relational algebra to its equivalent appropriate for a left-deep, pipelined execution [Redesign] **VI:** Compute selectivity estimates for a given selection and/or join operation [Compute] **VII:** Describe how to modify an index structure to facilitate an index-only operation for a given relation [Describe] **VIII:** For a given scenario decide on which indices to support for the efficient execution of a set of queries [Assess] **IX:** Describe how DBMSs leverage parallelism to speed up query processing by dividing the work across multiple processors or nodes [Describe]

Bibliography: [Silberschatz et al., 2019, Ramakrishnan and Gehrke, 2003]

5.2 DBMS Internals, 12 hr(s)

Competences: AG-C08,AG-C12

Topics: **I:** DB Buffer Management **II:** Transaction Management: (a) Isolation Levels (b) ACID (c) Serializability (d) Distributed Transactions **III:** Concurrency Control: (a) 2-Phase Locking (b) Deadlocks handling strategies (c) Quorum-based consistency models **IV:** Recovery Manager: (a) Relation with Buffer Manager **V:** Concurrency Control: (a) Optimistic concurrency control (b) Timestamp concurrency control **VI:** Recovery Manager: (a) Write-Ahead logging (b) ARIES recovery system (Analysis, REDO, UNDO)

Learning Outcomes **I:** Describe how a DBMS manages its Buffer Pool [Describe] **II:** Describe the four properties for a correct transaction manager [Describe] **III:** Outline the principle of serializability [Outline]

Bibliography: [Silberschatz et al., 2019, Garcia-Molina et al., 2008]

5.3 Distributed Databases/Cloud Computing, 12 hr(s)

Competences: AG-C08,AG-C12

Topics: **I:** Distributed DBMS: (a) Distributed data storage (b) Distributed query processing (c) Distributed transaction model (d) Homogeneous and heterogeneous solutions (e) Client-server distributed databases **II:** Parallel DBMS: (a) Parallel DBMS architectures: shared memory, shared disk, shared nothing (b) Speedup and scale-up, e.g., use of the MapReduce processing model (c) Data replication and weak consistency models

Learning Outcomes **I:** Describe the key components of a distributed DBMS, including distributed data storage, query processing, and transaction management [Describe] **II:** Analyze the tradeoffs between parallel DBMS architectures: shared memory, shared disk, and shared nothing [Analyze] **III:** Describe data replication strategies and weak consistency models in distributed database systems [Describe]

Bibliography: [Silberschatz et al., 2019, Özsu and Valduriez, 2020]

5.4 NoSQL Systems, 12 hr(s)

Competences: AG-C08,AG-C12

Topics: **I:** Why NoSQL? (e.g., Impedance mismatch between Application [CRUD] and RDBMS) **II:** Key-Value and Document data model **III:** Storage systems (e.g., Key-Value systems, Data Lakes) **IV:** Distribution Models (Sharding and Replication) **V:** Graph Databases **VI:** Consistency Models (Update and Read, Quorum consistency, CAP theorem) **VII:** Processing model (e.g., Map-Reduce, multi-stage map-reduce, incremental map-reduce) **VIII:** Case Studies: Cloud storage system (e.g., S3); Graph databases; "When not to use NoSQL"

Learning Outcomes **I:** Develop a use case for the use of NoSQL over RDBMS [Create] **II:** Describe the defining characteristics behind Key-Value and Document-based data models [Describe]

Bibliography: [Sadalage and Fowler, 2012, Kleppmann, 2017]

5.5 Semi-structured and Unstructured Databases, 12 hr(s)

Competences: AG-C08,AG-C12

Topics: **I:** Vectorized unstructured data (text, video, audio, etc.) and vector storage: (a) TF-IDF Vectorizer with ngram (b) Word2Vec (c) Array database or array data type handling **II:** Semi-structured databases (e.g., JSON): (a) Storage: i. Encoding and compression of nested data types (b) Indexing: i. Btree, skip index, Bloom filter ii. Inverted index and bitmap compression iii. Space filling curve indexing for semi-structured geo-data (c) Query processing for OLTP and OLAP use cases: i. Insert, Select, update/delete tradeoffs ii. Case studies on Postgres/JSON, MongoDB, and Snowflake/JSON

Learning Outcomes **I:** Describe vectorized representations of unstructured data (e.g., TF-IDF, Word2Vec) and their storage models [Describe] **II:** Apply indexing techniques (e.g., B-tree, inverted index, Bloom filter) to semi-structured data stored in JSON format [Apply] **III:** Analyze query processing tradeoffs for semi-structured databases in OLTP and OLAP use cases [Analyze]

Bibliography: [Kleppmann, 2017, Manning et al., 2008, Mikolov et al., 2013, Agrawal et al., 2023, MongoDB, Inc., 2025]

6. Bibliography

[Agrawal et al., 2023] Agrawal, D. et al. (2023). Toward a systems architecture for ai-enabled applications. *Communications of the ACM*, 66(6):54–63.

[Garcia-Molina et al., 2008] Garcia-Molina, H., Ullman, J. D., and Widom, J. (2008). *Database Systems: The Complete Book*. Pearson, 2nd edition.

[Kleppmann, 2017] Kleppmann, M. (2017). *Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems*. O'Reilly Media.

[Manning et al., 2008] Manning, C. D., Raghavan, P., and Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press.

[Mikolov et al., 2013] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint*. arXiv:1301.3781.

[MongoDB, Inc., 2025] MongoDB, Inc. (2025). Mongodb manual. MongoDB, Inc. `https://www.mongodb.com/docs/`.

[Özsu and Valduriez, 2020] Özsu, M. T. and Valduriez, P. (2020). *Principles of Distributed Database Systems*. Springer, 4th edition.

[Ramakrishnan and Gehrke, 2003] Ramakrishnan, R. and Gehrke, J. (2003). *Database Management Systems*. McGraw-Hill, 3rd edition.

[Sadalage and Fowler, 2012] Sadalage, P. J. and Fowler, M. (2012). *NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence*. Addison-Wesley.

[Silberschatz et al., 2019] Silberschatz, A., Korth, H. F., and Sudarshan, S. (2019). *Database System Concepts*. McGraw-Hill, 7th edition.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

CS291 Software Engineering I (Mandatory)

1. General information

- **Semester**: 5th Sem. **Credits**: 4
- **Hour of this course**: **Theory**: 2 hours; **Practice**: 2 hours; **Laboratory**: 2 hours;
- **Prerrequisites**:
  - CS113 Objects-oriented programming II (3rd Sem-Pg. 287)
  - CS271 Data Management (4th Sem-Pg. 327)

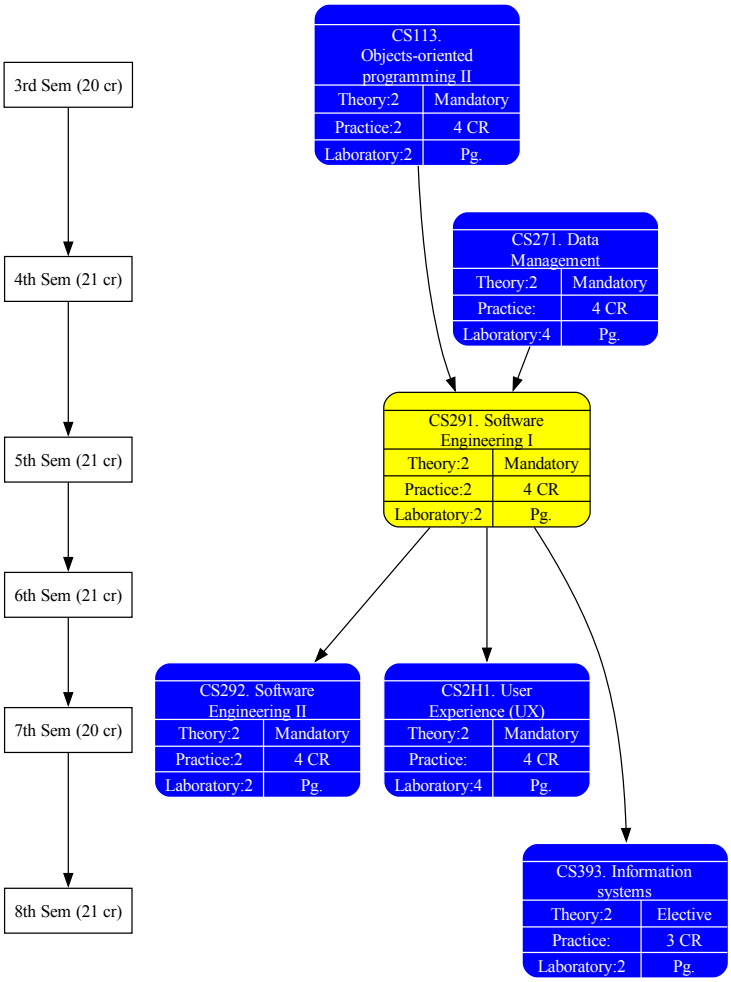

Figure 5.28: Connection Map. CS291 Software Engineering I

2. Course Rationale

Software engineering is a collaborative discipline that applies structured processes to build high-quality systems. This first course focuses on the foundational aspects of the software lifecycle, from initial requirements and human-centric design to systematic testing. By integrating teamwork principles and rigorous development fundamentals, students will learn to create software that is both technically sound and user-oriented.

3. Goals of the Course

- Understand and apply various software process models and configuration management.
- Master requirements elicitation and modeling using standardized notations.
- Develop effective teamwork and collaboration skills for software development.
- Design robust software architectures based on engineering principles.
- Implement systematic testing strategies and understand software quality.

4. General Competences (*Outcomes*)

AG-C03) Individual and Team Work: Performs effectively as an individual and as a member or leader in diverse teams. (Usage)

AG-C09) Design and Development of Solutions: Designs, implements, and evaluates solutions for complex computing problems. (Usage)

5. Content

5.1 Version Control and CI/CD, 12 hr(s)

Competences: AG-C03,AG-C09

Topics: **I:** Software configuration management and version control: Software Development Practices (a) Configuration in version control, reproducible builds/configuration. (b) Version control branching strategies. Development branches vs release branches. Trunk-based development. (c) Merging/rebasing strategies, when relevant. **II:** Release management. **III:** Testing tools including static and dynamic analysis tools. Software Development Practices, Information Flow and Non-Interference, Injection and Input Validation, Memory Safety and Types, Malware Analysis and Advanced Security **IV:** Software process automation: (a) Build systems - the value of fast, hermetic, reproducible builds, compare/contrast approaches to building a project. (b) Continuous Integration (CI) - the use of automation and automated tests to do preliminary validation that the current head/trunk revision builds and passes (basic) tests. (c) Continuous Deployment (CD) - the use of automation to automatically release every change that passes the automated tests to the production environment, ensuring frequent and reliable deliveries. (d) Dependency management - updating external/upstream dependencies, package management, SemVer. **V:** Collaborative review workflow as a quality gate (a) Mandatory pull-request review before merging (b) Code ownership via CODEOWNERS-style rules (c) Conventional commit message conventions (d) Branch-protection rules enforcing checks before merge **VI:** Dependency and supply-chain vulnerability scanning integrated into CI pipelines

Learning Outcomes **I:** Describe the difference between centralized and distributed software configuration management [Describe] **II:** Describe how version control can be used to help manage software release management [Describe] **III:** Identify configuration items and use a source code control tool in a small team-based project [Analyze] **IV:** Describe how available static and dynamic test tools can be integrated into the software development environment [Describe] **V:** Understand the use of CI/CD systems as a ground-truth for the state of the team's shared code (build and test success) [Explain] **VI:** Apply a pull-request review workflow with code-ownership rules and branch protection to enforce quality gates on a team project [Apply] **VII:** Use a dependency/supply-chain vulnerability scanner as part of a CI pipeline to detect vulnerable dependencies [Use]

Bibliography: [Sommerville, 2015, Pressman and Maxim, 2019]

5.2 Requirements Engineering, 14 hr(s)

Competences: AG-C03,AG-C09

Topics: **I:** Describe functional requirements using, for example, use cases or user stories. (a) Using at least one method of documenting and structuring functional requirements. (b) Understanding

how the method supports design and implementation. (c) Strengths and weaknesses of using a specific approach. **II:** Properties of requirements including consistency, validity, completeness, and feasibility. **III:** Requirements elicitation. (a) Sources of requirements, for example, users, administrators, or support personnel. (b) Methods of requirement gathering, for example, surveys, interviews, or behavioral analysis. **IV:** Non-functional requirements, for example, security, usability, or performance, also called as Quality Attributes. Sustainable Design and Pervasive Computing, Environmental Footprint of Computing Systems, Systemic Effects and Social Context **V:** Risk identification and management, including ethical considerations surrounding the proposed product. Ethics Codes and Legal Accountability, Professional Responsibility and Public Policy, Professional Values, Nature and Continuing Development, Ethical Dissent and Cultural Context, Workplace Ethics **VI:** Communicating and/or formalizing requirement specifications.

Learning Outcomes **I:** Compare different methods of eliciting requirements along multiple axes [Compare] **II:** Identify differences between two methods of describing functional requirements (e.g., customer interviews, user studies) and the situations where each would be preferred [Analyze] **III:** Identify which behaviors are required, allowed, or barred from a given set of requirements and a list of candidate behaviors [Analyze] **IV:** Collect a set of requirements for a simple software system [Analyze] **V:** Identify areas of a software system that need to be changed, given a description of the system and a set of new requirements to be implemented [Analyze] **VI:** Identify the functional and non-functional requirements in a set of requirements [Analyze]

Bibliography: [Sommerville, 2015, Wiegers and Beatty, 2013]

5.3 Team Collaboration and Dynamics, 14 hr(s)

Competences: AG-C03,AG-C09

Topics: **I:** Effective communication, including oral and written, as well as formal (email, docs, comments, presentations) and informal (team chat, meetings). Technical and Professional Communication, Team Communication and Culture **II:** Common causes of team conflict, and approaches for conflict resolution. **III:** Cooperative programming: (a) Pair programming or Swarming (b) Code review (c) Collaboration through version control **IV:** Roles and responsibilities in a software team: Ethics Codes and Legal Accountability, Professional Responsibility and Public Policy, Professional Values, Nature and Continuing Development, Ethical Dissent and Cultural Context, Workplace Ethics (a) Advantages of teamwork (b) Risks and complexity of such collaboration **V:** Team processes - responsibilities for tasks, effort estimation, meeting structure, work schedule

Learning Outcomes **I:** Follow effective team communication practices [Analyze] **II:** Articulate the sources of, hazards of, and potential benefits of team conflict - especially focusing on the value of disagreeing about ideas or proposals without insulting people [Articulate] **III:** Facilitate a conflict-resolution and problem-solving strategy in a team setting [Analyze] **IV:** Collaborate effectively in cooperative development/programming [Analyze] **V:** Propose and delegate necessary roles and responsibilities in a software development team [Propose] **VI:** Compose and follow an agenda for a team meeting [Compose] **VII:** Facilitate through involvement in a team project, the central elements of team building, establishing healthy team culture, and team management including creating and executing a team work plan [Create]

Bibliography: [Wigdor and Wixon, 2011, Johnson, 2013]

5.4 Principles and Software Architecture, 12 hr(s)

Competences: AG-C03,AG-C09

Topics: **I:** System design principles. System Reliability (a) Levels of abstraction (e.g., architectural design and detailed design) (b) Separation of concerns (c) Information hiding (d) Coupling and cohesion **II:** Software architecture. System Reliability (a) Design paradigms i. Top-down functional decomposition/layered design ii. Data-oriented architecture iii. Object-oriented

analysis and design iv. Event-driven design (b) Standard architectures (e.g., client-server and microservice architectures including REST discussions, n-layer, pipes-and-filters, Model View Controller) (c) Identifying component boundaries and dependencies **III:** Programming in the large vs programming in the small. System Reliability **IV:** Code smells and other indications of code quality, distinct from correctness. Security Design and Controls Engineering, Threat Analysis and Security Engineering, Trusted Computing and Privacy Engineering

Learning Outcomes **I:** Identify the standard software architecture of a given high-level design [Analyze] **II:** Select and use an appropriate design paradigm to design a simple software system and explain how system design principles have been applied in this design [Evaluate] **III:** Adapt a flawed system design to better follow principles such as separation of concerns or information hiding [Create] **IV:** Identify the dependencies among a set of software components in an architectural design [Analyze]

Bibliography: [Gamma et al., 1994, Pressman and Maxim, 2019]

5.5 Verification and validation Concepts, 12 hr(s)

Competences: AG-C03,AG-C09

Topics: **I:** Verification and validation concepts (a) Verification: Are we building the thing right? (b) Validation: Did we build the right thing? **II:** Why testing matters: Does the component remain functional as the code evolves? **III:** Testing objectives (a) Usability (b) Reliability (c) Conformance to specification (d) Performance (e) Security

Learning Outcomes **I:** Explain why testing is important [Explain] **II:** Distinguish between program validation and verification [Distinguish] **III:** Describe different objectives of testing [Describe]

Bibliography: [Sommerville, 2015, Pressman and Maxim, 2019]

5.6 Testing Tools and Analysis, 4 hr(s)

Competences: AG-C03,AG-C09

Topics: **I:** Verification and validation tooling and automation (a) Static analysis (b) Code coverage (c) Fuzzing (d) Dynamic analysis and fault containment (sanitizers, etc.) (e) Fault logging and fault tracking **II:** Test planning and generation (a) Fault estimation and testing termination including defect seeding (b) Use of random and pseudo random numbers in testing **III:** Testing asynchronous, parallel, and concurrent systems **IV:** Verification and validation of non-code artifacts (documentation, training materials) **V:** Behavior-driven development (BDD) (a) Expressing acceptance criteria as Gherkin Given/When/Then scenarios (b) Executing BDD scenarios as automated end-to-end tests **VI:** Mutation testing and static code-quality analysis (a) Mutation testing to assess test-suite effectiveness (b) Static detection of code smells and excessive cyclomatic complexity

Learning Outcomes **I:** Describe and compare different tools for verification and validation [Describe] **II:** Automate the use of different tools in a small software project [Design] **III:** Explain how and when random numbers should be used in testing [Explain] **IV:** Describe approaches for fault estimation [Describe] **V:** Estimate the number of faults in a small software application based on fault density and fault seeding [Estimate] **VI:** Describe techniques and issues with testing asynchronous, concurrent, and parallel software [Describe] **VII:** Create a test plan for a medium-size code segment which contains asynchronous, concurrent, and/or parallel code, including a test matrix and generation of test data and inputs [Create] **VIII:** Describe techniques for the verification and validation of non-code artifacts [Describe] **IX:** Write Gherkin Given/When/Then scenarios for a feature's acceptance criteria and execute them as automated end-to-end tests [Write] **X:** Analyze a test suite's mutation score and refactor tests to kill surviving mutants [Analyze]

Bibliography: [Sommerville, 2015, Pressman and Maxim, 2019]

5.7 Performance Testing and Benchmarking, 4 hr(s)

Competences: AG-C03,AG-C09

Topics: **I:** Performance testing and benchmarking (a) Throughput and latency (b) Degradation under load (stress testing, FIFO vs LIFO handling of requests) (c) Speedup and scaling i. Amdahl's law ii. Gustafson's law iii. Soft and weak scaling (d) Identifying and measuring figures of merits (e) Common performance bottlenecks i. Compute-bound ii. Memory-bandwidth bound iii. Latency-bound (f) Statistical methods and best practices for benchmarking i. Estimation of uncertainty ii. Confidence intervals (g) Analysis and presentation (graphs, etc.) (h) Timing techniques

Learning Outcomes **I:** Describe throughput and latency and provide examples of each [Describe] **II:** Explain speedup and the different forms of scaling and how they are computed [Explain] **III:** Describe common performance bottlenecks [Describe] **IV:** Describe statistical methods and best practices for benchmarking software [Describe] **V:** Explain techniques for and challenges with measuring time when constructing a benchmark [Explain] **VI:** Identify the figures of merit, construct and run a benchmark, and statistically analyze and visualize the results for a small software project [Analyze]

Bibliography: [Bondi, 2015, Gregg, 2020]

6. Bibliography

[Bondi, 2015] Bondi, A. B. (2015). *Foundations of Software and System Performance Engineering: Process, Performance Modeling, Requirements, Testing, Scalability, and Practice*. Addison-Wesley, Upper Saddle River, NJ.

[Gamma et al., 1994] Gamma, E., Helm, R., Johnson, R., and Vlissides, J. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley.

[Gregg, 2020] Gregg, B. (2020). *Systems Performance: Enterprise and the Cloud*. Addison-Wesley Professional, Boston, MA, 2nd edition.

[Johnson, 2013] Johnson, J. (2013). *Designing with the Mind in Mind*. Morgan Kaufmann, 2nd edition.

[Pressman and Maxim, 2019] Pressman, R. S. and Maxim, B. (2019). *Software Engineering: A Practitioner's Approach*. McGraw-Hill Education, 9th edition.

[Sommerville, 2015] Sommerville, I. (2015). *Software Engineering*. Pearson, 10th edition.

[Wiegers and Beatty, 2013] Wiegers, K. and Beatty, J. (2013). *Software Requirements*. Microsoft Press, 3rd edition.

[Wigdor and Wixon, 2011] Wigdor, D. and Wixon, D. (2011). *Brave NUI World: Designing Natural User Interfaces for Touch and Gesture*. Morgan Kaufmann.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

CS2S1 Operating systems (Mandatory)

1. General information

- **Semester**: 5th Sem. **Credits**: 4
- **Hour of this course**: **Theory**: 2 hours; **Practice**: 2 hours; **Laboratory**: 2 hours;
- **Prerrequisites**:
  - CS221 Computer Systems Architecture (4th Sem-Pg. 323)

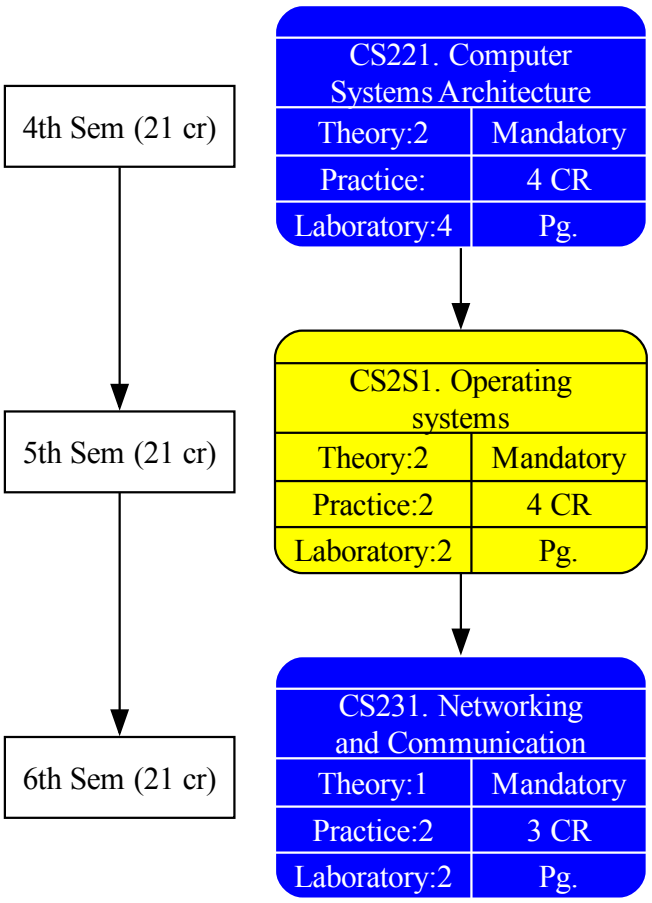

Figure 5.29: Connection Map. CS2S1 Operating systems

2. Course Rationale

The operating system is a collection of services needed to safely interface the hardware with applications. This course explores the mechanisms and policies needed to virtualize computation, memory, and I/O. Students will understand core concepts such as concurrency, persistence, and trust boundaries, which are fundamental for developing robust and efficient software systems.

3. Goals of the Course

- Understand the role and purpose of modern operating systems.

- Master process scheduling and the lifecycle of execution units.
- Analyze memory management, including physical and virtual memory virtualization.
- Evaluate storage systems and file management techniques.
- Comprehend the basics of security and protection within an OS environment.

4. General Competences (*Outcomes*)

AG-C03) Individual and Team Work: Performs effectively as an individual and as a member or leader in diverse teams. (Usage)

AG-C09) Design and Development of Solutions: Designs, implements, and evaluates solutions for complex computing problems. (Usage)

5. Content

5.1 Role and Purpose of Operating Systems, 8 hr(s)

Competences: AG-C03,AG-C09

Topics: **I:** Operating systems mediate between general purpose hardware and application-specific software. **II:** Universal operating system functions (e.g., process, user and device interfaces, persistence of data) **III:** Extended and/or specialized operating system functions (e.g., embedded systems, server types such as file, web, multimedia, boot loaders and boot security) **IV:** Design issues (e.g., efficiency, robustness, flexibility, portability, security, compatibility, power, safety, tradeoffs between error checking and performance, flexibility and performance, and security and performance). See also: Security Design and Controls Engineering, Threat Analysis and Security Engineering, Trusted Computing and Privacy Engineering **V:** Influences of security, networking, multimedia, parallel and distributed computing **VI:** Overarching concern of security/protection: Neglecting to consider security at every layer creates an opportunity to inappropriately access resources. (a) Unauthorized access to files on an unencrypted drive can be achieved by moving the media to another computer. (b) Operating systems enforced security can be defeated by infiltrating the boot layer before the operating system is loaded. (c) Process isolation can be subverted by inadequate authorization checking at API boundaries. (d) Vulnerabilities in system firmware can provide attack vectors that bypass the operating system entirely. (e) Improper isolation of virtual machine memory, computing, and hardware can expose the host system to attacks from guest systems. (f) The operating system may need to mitigate exploitation of hardware and firmware vulnerabilities, leading to potential performance reductions (e.g., Spectre and Meltdown mitigations). **VII:** Exposure of operating systems functions in shells and systems programming. See also: Shell Scripting

Learning Outcomes **I:** Understand the objectives and functions of modern operating systems [Explain] **II:** Evaluate the design issues in different usage scenarios (e.g., real time OS, mobile, server) [Evaluate] **III:** Understand the functions of a contemporary operating system with respect to convenience, efficiency, and the ability to evolve [Explain] **IV:** Understand how evolution and stability are desirable and mutually antagonistic in operating systems function [Explain]

Bibliography: [Silberschatz et al., 2018, Tanenbaum and Bos, 2014]

5.2 Shell Scripting, 8 hr(s)

Competences: AG-C03,AG-C09

Topics: **I:** Error/exception handling **II:** Piping **III:** System commands: (a) Interface with operating systems **IV:** Environment variables **V:** File abstraction and operators **VI:** Data structures, such as arrays and lists. **VII:** Regular expressions **VIII:** Programs and processes **IX:** Workflow

Learning Outcomes **I:** Create and execute automated scripts to manage various system tasks [Create] **II:** Solve various text processing problems through scripting [Solve]

Bibliography: [Robbins and Beebe, 2005, Silberschatz et al., 2018]

5.3 Scheduling, 16 hr(s)

Competences: AG-C03,AG-C09

Topics: **I:** Preemptive and non-preemptive scheduling **II:** Schedulers and policies (e.g., first come, first serve, shortest job first, priority, round robin, multilevel). See also: Resource Management **III:** Concepts of Symmetric Multi-Processor (SMP) scheduling and cache coherence. See also: Memory Hierarchy **IV:** Timers (e.g., building many timers out of finite hardware timers). See also: Assembly Level Machine Organization **V:** Fairness and starvation **VI:** Subtopics of operating systems such as energy-aware scheduling and real-time scheduling. See also: Performance and Energy Efficiency, Embedded Characteristics and Architectures, Real-Time, Memory and Verification, Systems, Programming and Sensors, Mobile Platforms: Capabilities and Asynchronous Computing, Mobile Platforms: Architecture and Constraints, Mobile Platforms: Development Fundamentals **VII:** Cooperative scheduling, such as Linux futexes and userland scheduling.

Learning Outcomes **I:** Compare and contrast the common algorithms used for both preemptive and non-preemptive scheduling of tasks in operating systems, such as priority, performance comparison, and fair-share schemes [Compare] **II:** Explain the relationships between scheduling algorithms and application domains [Explain] **III:** Explain the distinctions among types of processor scheduler such as short-term, medium-term, long-term, and I/O [Explain] **IV:** Evaluate a problem or solution to determine appropriateness for asymmetric and/or symmetric multiprocessing [Evaluate] **V:** Evaluate a problem or solution to determine appropriateness as a process vs threads [Evaluate] **VI:** List some contexts benefitting from preemption and deadline scheduling [List] **VII:** Explain the ways that the logic embodied in scheduling algorithms are applicable to other operating systems mechanisms, such as first come first serve or priority to disk I/O, network scheduling, project scheduling, and problems beyond computing [Explain]

Bibliography: [Silberschatz et al., 2018, Stallings, 2017]

5.4 Memory Management, 16 hr(s)

Competences: AG-C03,AG-C09

Topics: **I:** Review of physical memory, address translation and memory management hardware See also: Memory Hierarchy, Discrete Mathematics **II:** Impact of memory hierarchy including cache concept, cache lookup, and per-CPU caching on operating system mechanisms and policy See also: Memory Hierarchy, Latency, Cache and Memory Hierarchy, Virtualization and Isolation **III:** Logical and physical addressing, address space virtualization. See also: Memory Hierarchy, Discrete Mathematics **IV:** Concepts of paging, page replacement, thrashing and allocation of pages and frames **V:** Allocation/deallocation/storage techniques (algorithms and data structure) performance and flexibility (a) Arenas, slab allocators, free lists, size classes, heterogeneously sized pages (huge pages) **VI:** Memory caching and cache coherence and the effect of flushing the cache to avoid speculative execution vulnerabilities. See also: Functional Organization, Memory Hierarchy, Latency, Cache and Memory Hierarchy, Virtualization and Isolation **VII:** Security mechanisms and concepts in memory management including sandboxing, protection, isolation, and relevant vectors of attack. See also: AI and Hardware Security, Access Control and Applications, Security Mindset and Principles **VIII:** Virtual memory: leveraging virtual memory hardware for OS services and efficiency

Learning Outcomes **I:** Explain memory hierarchy and cost-performance tradeoffs [Explain] **II:** Summarize the principles of virtual memory as applied to caching and paging [Summarize] **III:** Evaluate the tradeoffs in terms of memory size (main memory, cache memory, auxiliary memory) and processor speed [Evaluate] **IV:** Describe the reason for and use of cache memory (performance and proximity, how caches complicate isolation and virtual machine abstraction) [Describe] **V:** Code/Develop efficient programs that consider the effects of page replacement and frame

allocation on the performance of a process and the system in which it executes [Create] **VI:** Explain how hardware is utilized for efficient virtualization [Explain]

Bibliography: [Silberschatz et al., 2018, Arpaci-Dusseau and Arpaci-Dusseau, 2018]

5.5 File Systems API and Implementation, 12 hr(s)

Competences: AG-C03,AG-C09

Topics: **I:** Concept of a file including data, metadata, operations, and access-mode **II:** File system mounting **III:** File access control **IV:** File sharing **V:** Basic file allocation methods, including linked allocation table **VI:** File system structures comprising file allocation including various directory structures and methods for uniquely identifying files (e.g., name, identified or metadata storage location) **VII:** Allocation/deallocation/storage techniques (algorithms and data structure) impact on performance and flexibility (i.e., internal and external fragmentation and compaction) **VIII:** Free space management such as using bit tables vs linking **IX:** Implementation of directories to segment and track file location

Learning Outcomes **I:** Explain the choices to be made in designing file systems [Explain] **II:** Evaluate different approaches to file organization, recognizing the strengths and weaknesses of each [Evaluate] **III:** Apply software constructs appropriately given knowledge of the file system implementation [Apply]

Bibliography: [Silberschatz et al., 2018, Stallings, 2017]

5.6 Protection and Safety, 12 hr(s)

Competences: AG-C03,AG-C09

Topics: **I:** Overview of operating system security mechanisms. See also: AI and Hardware Security, Access Control and Applications, Security Mindset and Principles **II:** Attacks and antagonism (scheduling, etc.). See also: AI and Hardware Security, Access Control and Applications, Security Mindset and Principles **III:** Review of major vulnerabilities in real operating systems. See also: AI and Hardware Security, Access Control and Applications, Security Mindset and Principles **IV:** Operating systems mitigation strategies such as backups. See also: System Reliability **V:** Policy/mechanism separation. See also: Risk and Data Management, Security Governance, Management and Policy **VI:** Security methods and devices. See also: AI and Hardware Security, Access Control and Applications, Security Mindset and Principles (a) Rings of protection (history from Multics to virtualized x86) (b) x86_64 rings -1 and -2 (hypervisor and ME/PSP) **VII:** Protection, access control, and authentication. See also: AI and Hardware Security, Access Control and Applications, Security Mindset and Principles, Protocols, Applications and Advanced Cryptography, Cryptographic Algorithms, Cryptography Fundamentals

Learning Outcomes **I:** Understand the requirement for protection and security mechanisms in operating systems [Explain] **II:** List and describe the attack vectors that leverage OS vulnerabilities [List] **III:** Understand the mechanisms available in an OS to control access to resources [Explain] **IV:** Summarize the features and limitations of an operating system that impact protection and security [Summarize]

Bibliography: [Silberschatz et al., 2018, Tanenbaum and Bos, 2014]

6. Bibliography

[Arpaci-Dusseau and Arpaci-Dusseau, 2018] Arpaci-Dusseau, R. H. and Arpaci-Dusseau, A. C. (2018). *Operating Systems: Three Easy Pieces*. Arpaci-Dusseau Books, 1.0 edition.

[Robbins and Beebe, 2005] Robbins, A. and Beebe, N. H. (2005). *Classic Shell Scripting*. O'Reilly Media.

[Silberschatz et al., 2018] Silberschatz, A., Galvin, P. B., and Gagne, G. (2018). *Operating System Concepts*. Wiley, 10th edition.

[Stallings, 2017] Stallings, W. (2017). *Operating Systems: Internals and Design Principles*. Pearson, 9th edition.

[Tanenbaum and Bos, 2014] Tanenbaum, A. S. and Bos, H. (2014). *Modern Operating Systems*. Pearson, 4th edition.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

FG106 Theater (Mandatory)

1. General information

- **Semester**: 5th Sem. **Credits**: 2
- **Hour of this course**: **Theory**: 1 hours; **Practice**: 2 hours;
- **Prerrequisites**: None

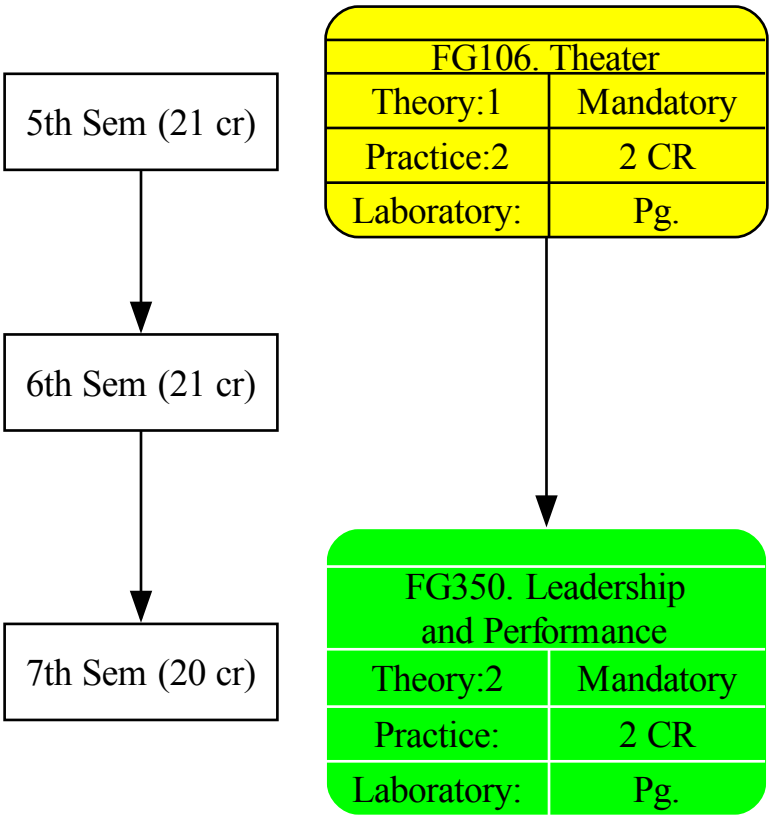

Figure 5.30: Connection Map. FG106 Theater

2. Course Rationale

Theater as a comprehensive performing art invites the student to recognize themselves as a creative and social human being. Through theatrical practice, the student develops their physical, verbal, and emotional expression; strengthens their self-confidence and personal security; and cultivates communication, empathy, and teamwork skills. This course provides practical tools for artistic expression that contribute to the comprehensive education of the university student, stimulating their sensitivity, creativity, and imagination. At the same time, it integrates them into their academic community through shared experiential activities, opening paths of self-discovery and authentic communication with others.

3. Goals of the Course

- Develop the student's body awareness and non-verbal communication skills through theatrical expression techniques.
- Foster creativity, spontaneity, and teamwork through dramatic play and improvisation.
- Learn about the historical trajectory of theater from its origins to its contemporary manifestations, including Peruvian theater.
- Value theatrical art as a tool for social integration, self-discovery, and expression of personal and collective identity.
- Apply the theatrical techniques learned in the creation and public presentation of a group stage proposal.

4. General Competences (*Outcomes*)

AG-C02) Ethics: Applies ethical principles and commits to professional ethics and the standards of professional computing practice. (Usage)

AG-C04) Communication: Communicates effectively in complex computing activities. (Usage)

5. Content

5.1 Art, Creativity, and Theater, 6 hr(s)

Competences: AG-C02,AG-C04

Topics: **I:** What is Art? An experiential and personal experience. **II:** The master key: creativity. **III:** The importance of theater in personal and professional development. **IV:** Utility and approach of theatrical art. **V:** Theater as a pedagogical tool and personal development instrument.

Learning Outcomes **I:** Recognize the relevance of art and creativity in personal and social development [Usage]. **II:** Connect the student with their group, valuing the importance of human communication and the social collective [Usage]. **III:** Identify basic notions of theater and its function in university life [Usage].

Bibliography: [Majorana, 1958, Pavis, 1998, Miravalles, 1998]

5.2 Play: The Actor's Craft, 6 hr(s)

Competences: AG-C02,AG-C04

Topics: **I:** I play, therefore I exist. Play as a universal language. **II:** Child's play and dramatic play. **III:** Group integration games and creativity games. **IV:** Spontaneous improvisation. **V:** The theatrical sequence and body warm-up.

Learning Outcomes **I:** Recognize play as a fundamental tool of theater [Usage]. **II:** Internalize and revalue play as creative learning [Usage]. **III:** Approach the theatrical experience spontaneously and naturally [Usage].

Bibliography: [Pavis, 1998, Miravalles, 1998]

5.3 Body Expression and Dramatic Use of Objects, 9 hr(s)

Competences: AG-C02,AG-C04

Topics: **I:** Body awareness: the body as an instrument. **II:** Awareness of the stage space. **III:** Awareness of time and rhythm. **IV:** Creation of individual and collective sequences: body, space, and time. **V:** The dramatic use of objects: theatrical play. **VI:** Theatrical performances using objects.

Learning Outcomes I: Experiment with new forms of body expression and communication [Usage]. **II:** Develop mechanisms for controlling and managing the body in stage space [Usage]. **III:** Cultivate imagination and the ability to respond to auditory, rhythmic, and visual stimuli [Usage]. **IV:** Explore emotional states and collective atmospheres through movement [Usage].

Bibliography: [Pavis, 1998, Grotowski, 1970]

5.4 Non-Verbal Communication in Theater, 9 hr(s)

Competences: AG-C02,AG-C04

Topics: I: Relaxation, concentration, and breathing. **II:** Disinhibition and interaction with the group. **III:** Improvisation: techniques and exercises. **IV:** Balance, weight, time, and body rhythm. **V:** Movement analysis: types and qualities. **VI:** Stage presence and the actor's energy. **VII:** Dance and theatrical choreography. **VIII:** Expressive gesture and the use of masks.

Learning Outcomes I: Practice non-verbal communication skills [Usage]. **II:** Engage in games and exercises of individual and group body language [Usage]. **III:** Express emotions and feelings freely and creatively through original performances [Usage]. **IV:** Distinguish different types of acting and their technical foundations [Usage].

Bibliography: [Pavis, 1998, Stanislavski, 1936, Grotowski, 1970]

5.5 Traces of Theater Through Time, 6 hr(s)

Competences: AG-C02,AG-C04

Topics: I: The origin of theater: ritual and performance in Greece and Rome. **II:** Medieval theater and Commedia dell'Arte. **III:** Spanish Golden Age theater and English Elizabethan theater. **IV:** Romanticism, Naturalism, and Realism: Ibsen and Chekhov. **V:** Brecht's epic theater and Stanislavski's psychological theater. **VI:** Theater of the absurd, contemporary theater, and total theater. **VII:** Theater in Peru: Yuyachkani, La Tarumba, Pataclaun, and others.

Learning Outcomes I: Understand the influence that society has exerted on theater throughout history [Usage]. **II:** Appreciate the value and contribution of fundamental playwrights and theater directors [Usage]. **III:** Analyze the social and historical context of theatrical art in different eras [Usage]. **IV:** Reflect on Peruvian theater and its contemporary expressions [Usage].

Bibliography: [Pavis, 1998, Campbell, 2001]

5.6 Theatrical Production, 12 hr(s)

Competences: AG-C02,AG-C04

Topics: I: Theatrical appreciation: attendance and critical analysis of plays. **II:** The stage space: types of stages and their dramatic use. **III:** The dramatic text: reading, analysis, and adaptation. **IV:** Character construction: technique and psychology. **V:** Directing and staging: basic fundamentals. **VI:** Production elements: costumes, makeup, set design, and props. **VII:** Public presentation of group theatrical exercises.

Learning Outcomes I: Employ theatrical creation as a manifestation of ideas and feelings before society [Usage]. **II:** Apply the techniques practiced in a concrete and meaningful stage proposal [Usage]. **III:** Exchange experiences and present brief group theatrical performances before an audience [Usage].

Bibliography: [Pavis, 1998, Miravalles, 1998, Brook, 1968, Campbell, 2001]

6. Bibliography

[Brook, 1968] Brook, P. (1968). *The Empty Space: A Book About the Theatre*. Penguin Books.

[Campbell, 2001] Campbell, A. B. (2001). *Historia general del teatro en el Perú*. Universidad de San Martín de Porres.

[Grotowski, 1970] Grotowski, J. (1970). *Towards a Poor Theatre*. Simon and Schuster.

[Majorana, 1958] Majorana, A. (1958). *El arte de hablar en público*. La España Moderna.

[Miravalles, 1998] Miravalles, L. (1998). *Iniciación al teatro: teoría y práctica*. San Pablo.

[Pavis, 1998] Pavis, P. (1998). *Dictionary of the Theatre: Terms, Concepts, and Analysis*. University of Toronto Press.

[Stanislavski, 1936] Stanislavski, C. (1936). *An Actor Prepares*. Theatre Arts Books.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

BEI203 Technical and professional English V (Elective)

1. General information

- **Semester**: 5th Sem. **Credits**: 2
- **Hour of this course**: **Practice**: 4 hours;
- **Prerrequisites**:
  - BEI202 Technical and professional English IV (4th Sem-Pg. 340)

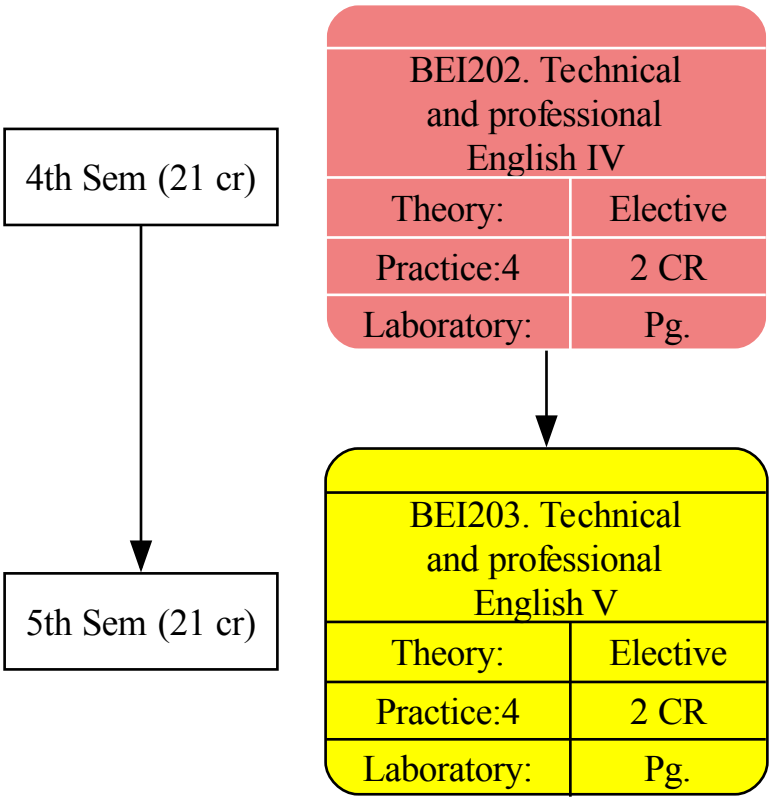

Figure 5.31: Connection Map. BEI203 Technical and professional English V

2. Course Rationale

A fundamental part of the integral formation of a professional is the ability to communicate in a foreign language in addition to the native language itself. In the engineering and computing career, technical English is the most critical tool for accessing the forefront of knowledge and research, writing scientific papers, participating in international conferences, and collaborating in global teams. This course brings students to the level of autonomous technical communicators capable of producing and critically evaluating academic and professional content.

3. Goals of the Course

- Increase the ability and fluency to communicate complex technical ideas in English.

- Ensure that students interact with emphasis on scientific argumentation and advanced technical writing.
- Master vocabulary for software engineering, data science, and systems research.

4. General Competences (*Outcomes*)

AG-C03) Individual and Team Work: Performs effectively as an individual and as a member or leader in diverse teams. (Familiarity)

AG-C04) Communication: Communicates effectively in complex computing activities. (Usage)

5. Content

5.1 Scientific Argumentation, 12 hr(s)

Competences: AG-C03,AG-C04

Topics: **I:** Advanced review of auxiliary verbs and verb tenses in academic writing. **II:** Negative questions and emphatic answers in scientific debate. **III:** Word formation: Prefixes and suffixes common in computing and science. **IV:** Correcting misconceptions: Refuting claims in technical discussions.

Learning Outcomes **I:** By the end of the unit, students use auxiliaries and tenses precisely to construct scientific arguments, challenge erroneous technical claims, and engage in structured academic debate.

Bibliography: [? line:48, ? line:48, ? line:48, ? line:48, ? line:48]

5.2 Research Methodology, 12 hr(s)

Competences: AG-C03,AG-C04

Topics: **I:** Present Simple vs. Present Continuous: Distinguishing general facts from ongoing experiments. **II:** State verbs and action verbs in experimental descriptions. **III:** Vocabulary: Research design, experimental variables, data collection methods. **IV:** Writing: Description of a research method or experimental procedure.

Learning Outcomes **I:** Students differentiate between stative and progressive uses to accurately describe scientific laws, ongoing experiments, and research methodologies in computing and engineering.

Bibliography: [? line:61, ? line:61, ? line:61, ? line:61, ? line:61]

5.3 Case Studies, 12 hr(s)

Competences: AG-C03,AG-C04

Topics: **I:** Narrative tenses: Simple Past, Past Continuous, and Past Perfect in case study analysis. **II:** Structure of "Used to" and "Would" for describing legacy systems and past practices. **III:** Vocabulary: Historical computing cases, paradigm shifts, engineering lessons learned. **IV:** Writing: Structured case study report on a computing incident or innovation.

Learning Outcomes **I:** Students analyze and narrate sequences of technical events and historical computing cases fluently, applying narrative tenses correctly in professional case study reports.

Bibliography: [? line:74, ? line:74, ? line:74, ? line:74, ? line:74]

5.4 Research Proposals, 12 hr(s)

Competences: AG-C03,AG-C04

Topics: **I:** Predictions and plans: Future tense review for academic proposal writing. **II:** Future Continuous and Future Perfect in project scope and outcomes. **III:** Vocabulary: Research objectives, expected contributions, scope and limitations. **IV:** Writing: Short research proposal or technical project abstract.

Learning Outcomes **I:** Students project research goals and anticipated outcomes, and make evidence-based predictions about the evolution of technology using appropriate future forms in formal academic writing.

Bibliography: [? line:87, ? line:87, ? line:87, ? line:87, ? line:87]

5.5 Technical Documentation, 12 hr(s)

Competences: AG-C03,AG-C04

Topics: **I:** Verb patterns (Verbs followed by -ing or infinitive) in instructional writing. **II:** Expressions with "Like" for analogies and technical comparisons. **III:** Vocabulary: API documentation, user manuals, technical specifications. **IV:** Debates: Standards for open-source documentation and reproducibility.

Learning Outcomes **I:** Students use complex verb patterns to write clear, precise technical documentation and use analogical reasoning to explain technical concepts to diverse audiences.

Bibliography: [? line:100, ? line:100, ? line:100, ? line:100, ? line:100]

5.6 Long-term Projects, 12 hr(s)

Competences: AG-C03,AG-C04

Topics: **I:** Present Perfect Continuous vs. Simple Present Perfect in research progress reports. **II:** Modal verbs of deduction (Present and Past) for interpreting experimental results. **III:** Vocabulary: Long-running systems, sustained research programs, startup development. **IV:** Oral presentations: "Progress report on a computing research project".

Learning Outcomes **I:** Students discuss long-term research and development activities, make deductions about technical outcomes, and present progress reports on computing projects with appropriate academic register.

Bibliography: [? line:113, ? line:113, ? line:113, ? line:113, ? line:113]

5.7 Advanced Technical Writing, 12 hr(s)

Competences: AG-C03,AG-C04

Topics: **I:** Relative clauses (Defining vs. Non-defining) in technical paper writing. **II:** Advanced passive voice in formal scientific and engineering prose. **III:** Vocabulary: Peer review, impact factors, citation practices, open access. **IV:** Writing: Abstract and introduction section of a computing research paper.

Learning Outcomes **I:** Students write precise, well-structured technical and scientific texts using relative clauses and advanced passive constructions to add depth and clarity to research writing.

Bibliography: [? line:126, ? line:126, ? line:126, ? line:126, ? line:126]

5.8 English for Software Engineering, 12 hr(s)

Competences: AG-C03,AG-C04

Topics: **I:** Agile Methodologies: Scrum, Kanban, and XP terminology in English. **II:** QA (Quality Assurance) and Testing: Test plans, bug reports, acceptance criteria. **III:** Software requirements documentation: user stories, use cases, specifications. **IV:** Simulation: Technical team meeting (Sprint planning and retrospective).

Learning Outcomes **I:** By the end of the course, students command software engineering and project management terminology, can produce professional technical documents, and actively participate in international technical meetings and reviews in English.

Bibliography: [**?** line:139 , **?** line:139 , **?** line:139 , **?** line:139 , **?** line:139]

6. Bibliography

line:144 line:144 line:144 line:144 line:144

Sixth Semester

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

CS231 Networking and Communication (Mandatory)

1. General information

- **Semester**: 6th Sem. **Credits**: 3
- **Hour of this course**: **Theory**: 1 hours; **Practice**: 2 hours; **Laboratory**: 2 hours;
- **Prerrequisites**:
  - CS2S1 Operating systems (5th Sem-Pg. 365)

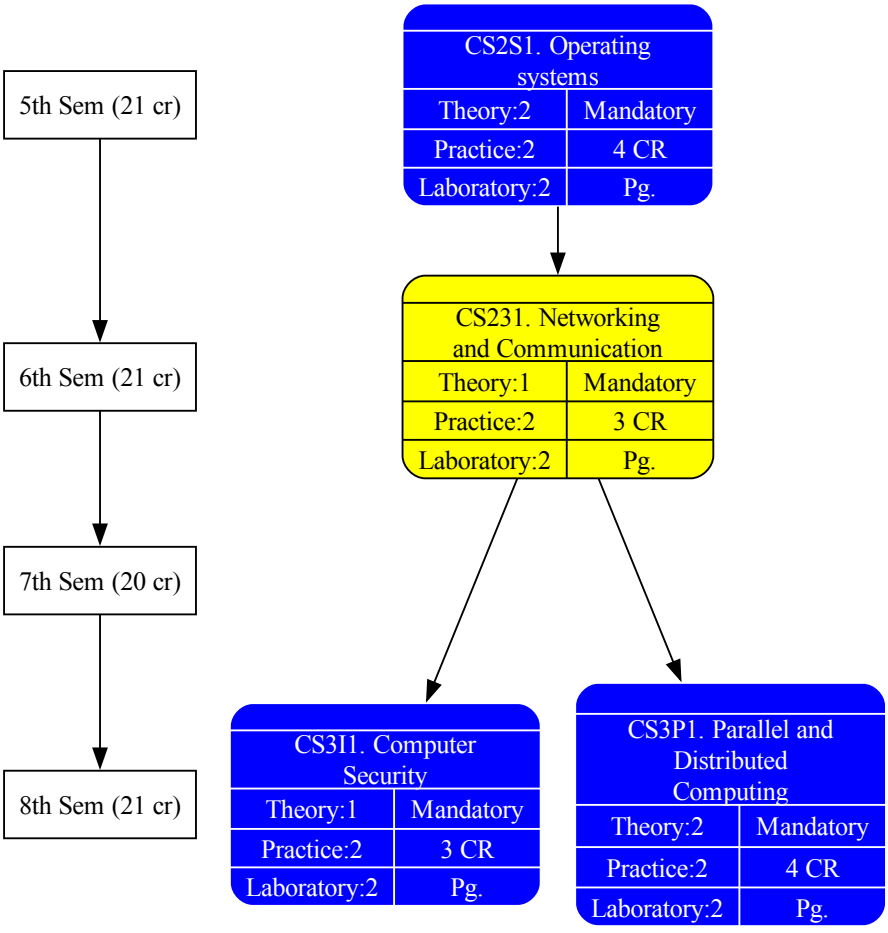

Figure 5.32: Connection Map. CS231 Networking and Communication

2. Course Rationale

Networking and communication are foundational to modern computing systems, distributed applications, and the global internet. This course provides a comprehensive introduction to the full CS2023 Networking and Communication Body of Knowledge—from physical transmission and data link protocols, through reliable transport and routing, to application services, security, mobility, and

emerging paradigms such as SDN and quantum networking. Students will develop both theoretical understanding and practical skills applicable to the design and analysis of modern networked systems.

3. Goals of the Course

- Understand the layered architecture of the TCP/IP and OSI models, including the role and responsibilities of each layer.
- Explain single-hop communication mechanisms including MAC protocols, encoding, Ethernet, WiFi, and LAN topologies.
- Master the principles of reliable data transfer, flow control, and congestion control.
- Analyze routing paradigms, forwarding mechanisms, and IP scalability in local and wide area networks.
- Comprehend fundamental application-layer protocols of the Internet (DNS, HTTP, TCP, UDP) and implement socket-based applications.
- Understand network security threats and the countermeasures used to protect networked systems.
- Describe principles of mobile networking including cellular networks, wireless LANs, and device-to-device communication.
- Identify emerging networking paradigms such as SDN, network virtualization, and quantum networking.

4. General Competences (*Outcomes*)

AG-C03) Individual and Team Work: Performs effectively as an individual and as a member or leader in diverse teams. (Usage)

AG-C09) Design and Development of Solutions: Designs, implements, and evaluates solutions for complex computing problems. (Usage)

5. Content

5.1 Fundamentals of Networks and Communications, 6 hr(s)

Competences: AG-C03,AG-C09

Topics: I: Importance of networking in contemporary computing, and associated challenges. See also: Digital Equity and Accessibility, Social Impact and AI, Civil Liberties and Expression, Data, Surveillance and Privacy Technology, Privacy Philosophical Foundations and Legislation **II:** Organization of the internet (e.g., users, Internet Service Providers, autonomous systems, content providers, content delivery networks) **III:** Switching techniques (e.g., circuit and packet) **IV:** Layers and their roles (application, transport, network, datalink, and physical) **V:** Layering principles (e.g., encapsulation and hourglass model). See also: Digital Systems and Logic, Sequential and Parallel Processing, State Machines **VI:** Network elements (e.g., routers, switches, hubs, access points, and hosts) **VII:** Basic queueing concepts (e.g., relationship with latency, congestion, service levels, etc.)

Learning Outcomes I: Articulate the organization of the internet [Articulate] **II:** List and define the appropriate network terminology [List] **III:** Describe the layered structure of a typical networked architecture [Describe] **IV:** Identify the different types of complexity in a network (edges, core, etc.) [Analyze]

Bibliography: [Kurose and Ross, 2021, Tanenbaum et al., 2021, Peterson and Davie, 2022]

5.2 Single Hop Communication, 6 hr(s)

Competences: AG-C03,AG-C09

Topics: **I:** Introduction to modulation, bandwidth, and communication media **II:** Encoding and Framing **III:** Medium Access Control (MAC) (e.g., random access and scheduled access) **IV:** Ethernet and WiFi **V:** Switching (e.g., spanning trees, VLANS). **VI:** Local Area Network Topologies (e.g., data center, campus networks).

Learning Outcomes **I:** Describe some basic aspects of modulation, bandwidth, and communication media [Describe] **II:** Describe in detail a MAC protocol [Describe] **III:** Demonstrate understanding of encoding and framing solution tradeoffs [Demonstrate] **IV:** Describe details of the implementation of Ethernet [Describe] **V:** Describe how switching works [Describe] **VI:** Describe one kind of a LAN topology [Describe]

Bibliography: [Tanenbaum et al., 2021, Peterson and Davie, 2022]

5.3 Networked Applications, 6 hr(s)

Competences: AG-C03,AG-C09

Topics: **I:** Naming and address schemes (e.g., DNS, and Uniform Resource Identifiers) **II:** Distributed application paradigms (e.g., client/server, peer-to-peer, cloud, edge, and fog). See also: Communication, Coordination **III:** Diversity of networked application demands (e.g., latency, bandwidth, and loss tolerance). See also: Communication, Sustainable Design and Pervasive Computing, Environmental Footprint of Computing Systems, Systemic Effects and Social Context, Digital Equity and Accessibility, Social Impact and AI **IV:** Coverage of application-layer protocols (e.g., HTTP) **V:** Interactions with TCP, UDP, and Socket APIs. See also: Programs: Distribution, Programs: Execution Properties, GPU Programming, Programs: Implementation Mappings, Programs: Declarative Parallelism, Programs: Starting Activities

Learning Outcomes **I:** Define the principles of naming, addressing, resource location [Define] **II:** Analyze the needs of specific networked application demands [Analyze] **III:** Describe the details of one application layer protocol [Describe] **IV:** Implement a simple client-server socket-based application [Implement]

Bibliography: [Kurose and Ross, 2021, Peterson and Davie, 2022]

5.4 Reliability Support, 6 hr(s)

Competences: AG-C03,AG-C09

Topics: **I:** Unreliable delivery (e.g., UDP) **II:** Principles of reliability (e.g., delivery without loss, duplication, or out of order). See also: System Reliability **III:** Error control (e.g., retransmission, error correction) **IV:** Flow control (e.g., stop and wait, window based) **V:** Congestion control (e.g., implicit and explicit congestion notification) **VI:** TCP and performance issues (e.g., Tahoe, Reno, Vegas, Cubic)

Learning Outcomes **I:** Describe the operation of reliable delivery protocols [Describe] **II:** List the factors that affect the performance of reliable delivery protocols [List] **III:** Describe some TCP reliability design issues [Describe] **IV:** Design and implement a simple reliable protocol [Design]

Bibliography: [Kurose and Ross, 2021, Tanenbaum et al., 2021]

5.5 Routing and Forwarding, 6 hr(s)

Competences: AG-C03,AG-C09

Topics: **I:** Routing paradigms and hierarchy (e.g., intra/inter domain, centralized and decentralized, source routing, virtual circuits, QoS) **II:** Forwarding methods (e.g., forwarding tables and matching algorithms) **III:** IP and Scalability issues (e.g., NAT, CIDR, BGP, different versions of IP)

Learning Outcomes **I:** Describe various routing paradigms and hierarchies [Describe] **II:** Describe how packets are forwarded in an IP network [Describe] **III:** Describe how the Internet tackles scalability challenges [Describe]

Bibliography: [Kurose and Ross, 2021, Tanenbaum et al., 2021, Peterson and Davie, 2022]

5.6 Network Security, 6 hr(s)

Competences: AG-C03,AG-C09

Topics: **I:** General intro about security (Threats, vulnerabilities, and countermeasures). See also: Computer Crimes and Attacks, Security Laws, Policies and Responsibility, AI and Hardware Security, Access Control and Applications, Security Mindset and Principles, Security Design and Controls Engineering, Threat Analysis and Security Engineering, Trusted Computing and Privacy Engineering **II:** Network specific threats and attack types (e.g., denial of service, spoofing, sniffing and traffic redirection, attacker-in-the-middle, message integrity attacks, routing attacks, ransomware, and traffic analysis). See also: AI and Hardware Security, Access Control and Applications, Security Mindset and Principles, Security Design and Controls Engineering, Threat Analysis and Security Engineering, Trusted Computing and Privacy Engineering **III:** Countermeasures: (a) Cryptography (e.g. SSL, TLS, symmetric/asymmetric) (b) Architectures for secure networks (e.g., secure channels, secure routing protocols, secure DNS, VPNs, DMZ, Zero Trust Network Access, hyper network security, anonymous communication protocols, isolation) (c) Network monitoring, intrusion detection, firewalls, spoofing and DoS protection, honeypots, tracebacks, BGP Sec, RPKI

Learning Outcomes **I:** Describe some of the threat models of network security [Describe] **II:** Describe specific network-based countermeasures [Describe] **III:** Analyze various aspects of network security from a case study [Analyze]

Bibliography: [Stallings, 2022, Kurose and Ross, 2021]

5.7 Mobility Support, 6 hr(s)

Competences: AG-C03,AG-C09

Topics: **I:** Principles of cellular communication (e.g., 4G, 5G) **II:** Principles of Wireless LANs (mainly 802.11) **III:** Device to device communication (e.g., IoT communication) **IV:** Multi-hop wireless networks (e.g., ad hoc networks, opportunistic, delay tolerant)

Learning Outcomes **I:** Describe some aspects of cellular communication such as registration [Describe] **II:** Describe how 802.11 supports mobile users [Describe] **III:** Describe practical uses of device-to-device communication, as well as multihop [Describe] **IV:** Describe one type of mobile network such as ad hoc [Describe]

Bibliography: [Tanenbaum et al., 2021, Peterson and Davie, 2022]

5.8 Emerging Topics, 6 hr(s)

Competences: AG-C03,AG-C09

Topics: **I:** Middleboxes (e.g., advances in usage of AI, intent-based networking, filtering, deep packet inspection, load balancing, NAT, CDN) **II:** Network Virtualization (e.g., SDN, Data Center Networks) **III:** Quantum Networking (e.g., Intro to the domain, teleportation, security, Quantum Internet) **IV:** Satellite, mmWave, Visible Light

Learning Outcomes **I:** Describe the value of advances in middleboxes in networks [Describe] **II:** Describe the importance of Software Defined Networks [Describe] **III:** Describe some of the added value achieved by using Quantum Networking [Describe]

Bibliography: [Peterson and Davie, 2022, Tanenbaum et al., 2021]

6. Bibliography

[Kurose and Ross, 2021] Kurose, J. and Ross, K. (2021). *Computer Networking: A Top-Down Approach*. Pearson, 8th edition.

[Peterson and Davie, 2022] Peterson, L. L. and Davie, B. S. (2022). *Computer Networks: A Systems Approach*. Morgan Kaufmann, 6th edition.

[Stallings, 2022] Stallings, W. (2022). *Cryptography and Network Security: Principles and Practice*. Pearson, 8th edition.

[Tanenbaum et al., 2021] Tanenbaum, A. S., Feamster, N., and Wetherall, D. J. (2021). *Computer Networks*. Pearson, 6th edition.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

CS311 Algorithms for Complex Problems (Mandatory)

1. General information

- **Semester**: 6th Sem. **Credits**: 3
- **Hour of this course**: **Theory**: 2 hours; **Laboratory**: 2 hours;
- **Prerrequisites**:
  - CS212 Analysis and Design of Algorithms (5th Sem-Pg. 344)

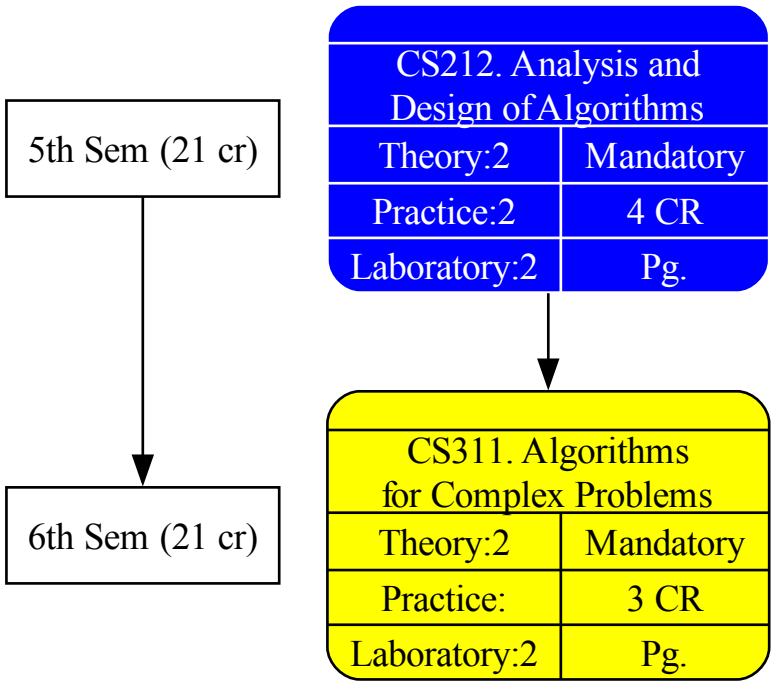

Figure 5.33: Connection Map. CS311 Algorithms for Complex Problems

2. Course Rationale

Competitive Programming trains students to solve complex algorithmic problems under time and resource constraints. This course bridges the gap between theoretical algorithm design and high-performance implementation. It covers advanced techniques in dynamic programming, graph theory, and mathematics, fostering the analytical thinking required for top-tier software engineering interviews and international programming contests like the ICPC.

3. Goals of the Course

- Master advanced competitive programming techniques and paradigms.
- Develop the ability to quickly analyze the complexity of potential solutions.
- Implement efficient data structures for range queries and dynamic updates.
- Apply specialized mathematical and geometric algorithms to solve contests problems.
- Optimize code for execution speed and memory usage in a competitive environment.

4. General Competences (*Outcomes*)

AG-C08) Problem Analysis: Identifies, formulates, and analyzes complex computing problems. (Usage)

AG-C12) Applies computer science theory and software development fundamentals to produce computer-based solutions. (Usage)

5. Content

5.1 Problem Solving Paradigms, 12 hr(s)

Competences: AG-C08,AG-C12

Topics: **I:** Complete Search: Recursive Backtracking and Bitmasking. **II:** Greedy Strategies: Choice property and optimal substructure. **III:** Divide and Conquer: Binary Search on the answer and Ternary Search. **IV:** Dynamic Programming: State reduction, Digit DP, and DP on trees.

Learning Outcomes **I:** Apply search pruning techniques to solve NP-hard problems in small constraints [Usage] **II:** Identify the most efficient paradigm for a given problem statement [Assessment]

Bibliography: [Halim and Halim, 2020, Laaksonen, 2020]

5.2 Advanced Range Queries, 12 hr(s)

Competences: AG-C08,AG-C12

Topics: **I:** Fenwick Trees (Binary Indexed Trees). **II:** Segment Trees: Lazy Propagation and Persistent Segment Trees. **III:** Square Root Decomposition and Mo's Algorithm. **IV:** Sparse Tables for RMQ.

Learning Outcomes **I:** Implement range query structures for real-time data updates [Assessment] **II:** Analyze the trade-offs between different range query data structures [Usage]

Bibliography: [Halim and Halim, 2020, Skiena, 2020]

5.3 Advanced Graph Algorithms, 12 hr(s)

Competences: AG-C08,AG-C12

Topics: **I:** Network Flow: Max Flow Min Cut (Dinic's and Edmonds-Karp). **II:** Strongly Connected Components (Tarjan's and Kosaraju's). **III:** Bipartite Matching and Heavy-Light Decomposition. **IV:** Centroid Decomposition on trees.

Learning Outcomes **I:** For each algorithm in this unit explain step-by-step how the algorithm operates [Explain] **II:** For each algorithmic approach (e.g., sorting) in this unit apply a prototypical example of the approach (e.g., merge sort) [Apply] **III:** Given requirements for a problem, develop multiple solutions using various data structures and algorithms. Subsequently, evaluate the suitability, strengths, and weaknesses selecting an approach that best satisfies the requirements [Create] **IV:** Explain factors beyond computational efficiency that influence the choice of algorithms, such as programming time, maintainability, and the use of application-specific patterns in the input data [Explain] **V:** For each of the algorithms and algorithmic approaches in the KA Core topics: (a) Explain a prototypical example of the algorithm, and (b) Explain step-by-step how the algorithm operates. [Explain]

Bibliography: [Laaksonen, 2020, Halim and Halim, 2020]

5.4 Mathematics and Geometry, 12 hr(s)

Competences: AG-C08,AG-C12

Topics: I: Number Theory: Sieve of Eratosthenes, Extended Euclidean, and Modular Inverse. **II:** Combinatorics: Inclusion-Exclusion and Burnside's Lemma. **III:** Computational Geometry: Cross product, Convex Hull (Monotone Chain), and Line intersection. **IV:** Fast Fourier Transform (FFT) for polynomial multiplication.

Learning Outcomes I: Solve combinatorial problems using modular arithmetic [Usage] **II:** Implement geometric primitives for spatial relationship analysis [Assessment]

Bibliography: [Skiena, 2020, Laaksonen, 2020]

6. Bibliography

[Halim and Halim, 2020] Halim, S. and Halim, F. (2020). *Competitive Programming 4: The Lower Bound of Programming Contests*. Lulu.

[Laaksonen, 2020] Laaksonen, A. (2020). *Competitive Programmer's Handbook*. Draft.

[Skiena, 2020] Skiena, S. S. (2020). *The Algorithm Design Manual*. Springer, 3rd edition.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

CS312 Advanced Data Structures (Mandatory)

1. General information

- **Semester**: 6th Sem. **Credits**: 4
- **Hour of this course**: **Theory**: 2 hours; **Practice**: 2 hours; **Laboratory**: 2 hours;
- **Prerrequisites**:
  - CS212 Analysis and Design of Algorithms (5th Sem-Pg. 344)

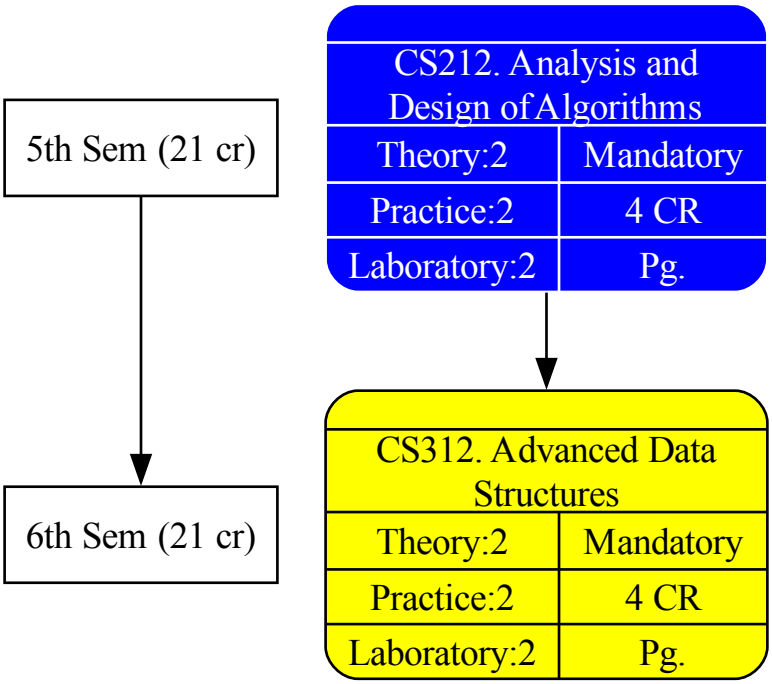

Figure 5.34: Connection Map. CS312 Advanced Data Structures

2. Course Rationale

Algorithms and data structures are a fundamental part of Computer Science that allow us to organize information efficiently. For any professional in the field, a solid foundation in this area is crucial. This course focuses on advanced structures such as Multidimensional Access Methods, Spatio-Temporal Access Methods, and Metric Access Methods, which are essential for high-performance applications in search engines, geographic information systems (GIS), and big data processing.

3. Goals of the Course

- Understand, design, and implement innovative data structures for multidimensional data.
- Apply similarity search techniques and metric indexing to complex data types.
- Analyze the ”Curse of Dimensionality” and its impact on search efficiency.
- Research and present contemporary indexing methods for Big Data in English.

4. General Competences (*Outcomes*)

AG-C08) Problem Analysis: Identifies, formulates, and analyzes complex computing problems. (Usage)

AG-C12) Applies computer science theory and software development fundamentals to produce computer-based solutions. (Usage)

5. Content

5.1 Advanced Implementation Techniques, 12 hr(s)

Competences: AG-C08,AG-C12

Topics: **I:** Static and dynamic implementation techniques. **II:** Data abstraction and encapsulation. **III:** Abstract Data Types (ADT). **IV:** Compile-time vs. Runtime polymorphism. **V:** Generic data structures and C++ Templates. **VI:** Reusable components and design patterns for data structures.

Learning Outcomes **I:** Evaluate the impact of static and dynamic techniques on data structure design [Assessment]. **II:** Implement generic and reusable data structures using advanced programming paradigms [Usage]. **III:** Analyze performance trade-offs between different implementation strategies [Assessment].

Bibliography: [Cuadros-Vargas et al., 2004, Knuth, 1997, Gamma et al., 1994, Vandevoorde et al., 2018]

5.2 Multidimensional Access Methods (Part I), 8 hr(s)

Competences: AG-C08,AG-C12

Topics: **I:** Introduction to multidimensional data. **II:** Vector space representation. **III:** The Curse of Dimensionality: theoretical foundations. **IV:** Impact of high dimensionality on traditional indexing. **V:** Real-world applications in search engines and pattern recognition.

Learning Outcomes **I:** Explain the significance of multidimensional data representation [Familiarity]. **II:** Analyze the complexity and performance degradation in high-dimensional spaces [Usage]. **III:** Discuss real-world scenarios where dimensionality impact is critical [Usage].

Bibliography: [Gaede and Günther, 1998, Samet, 2006]

5.3 Multidimensional Access Methods (Part II), 20 hr(s)

Competences: AG-C08,AG-C12

Topics: **I:** Spatial data structures: Quadtree and Octree. **II:** Point Access Methods: Kd-Tree. **III:** Region Access Methods: R-Tree (Guttman), R+ tree, R* tree. **IV:** Variations of R-trees and their relation to paging and block size. **V:** Advanced structures: X-tree and SS-tree.

Learning Outcomes **I:** Implement spatial data structures for high-volume data indexing [Usage]. **II:** Evaluate the limitations of tree-based spatial structures in specific domains [Assessment]. **III:** Implement range search and Nearest Neighbor (k-NN) strategies [Usage].

Bibliography: [Samet, 2006]

5.4 Metric Access Methods, 12 hr(s)

Competences: AG-C08,AG-C12

Topics: I: Metric Spaces: definitions and properties. **II:** Similarity search vs. Exact search. **III:** Vantage Point Tree (VP-Tree). **IV:** Slim-Tree and M-Tree. **V:** Pivot-based indexing.

Learning Outcomes I: Model similarity search problems using metric space properties [Usage]. **II:** Compare metric access methods against multidimensional methods for non-vector data [Assessment].

Bibliography: [Samet, 2006, Zezula et al., 2007]

5.5 Research Seminars and Final Talks, 12 hr(s)

Competences: AG-C08,AG-C12

Topics: I: Seminar on recent research papers (SIGMOD, VLDB, ICDE). **II:** Indexing for Big Data and streaming data. **III:** Compact data structures and succinct representations. **IV:** Future trends in similarity search.

Learning Outcomes I: Research new methods for indexing large volumes of complex data [Usage]. **II:** Present and lead technical discussions in English on state-of-the-art methods [Assessment]. **III:** Identify potential thesis topics within the advanced data structures domain [Familiarity].

Bibliography: [Navarro, 2016, Traina Jr et al., 2000]

6. Bibliography

[Cuadros-Vargas et al., 2004] Cuadros-Vargas, E., Romero, R. A. F., Mock, M., and Brisaboa, N. (2004). Implementing data structures: An incremental approach. Available at: http://socios.spc.org.pe/ecuadros/cursos/pdfs/.

[Gaede and Günther, 1998] Gaede, V. and Günther, O. (1998). Multidimensional access methods. *ACM Computing Surveys*, 30(2):170–231.

[Gamma et al., 1994] Gamma, E., Helm, R., Johnson, R., and Vlissides, J. M. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional.

[Knuth, 1997] Knuth, D. E. (1997). *The Art of Computer Programming, Volume 1: Fundamental Algorithms*. Addison-Wesley Professional, 3rd edition.

[Navarro, 2016] Navarro, G. (2016). *Compact Data Structures*. Cambridge University Press.

[Samet, 2006] Samet, H. (2006). *Foundations of Multidimensional and Metric Data Structures*. Elsevier/Morgan Kaufmann.

[Traina Jr et al., 2000] Traina Jr, C., Traina, A. J. M., Seeger, B., and Faloutsos, C. (2000). Slim-trees: High performance metric trees minimizing overlap between nodes. In *Advances in Database Technology - EDBT 2000*. Springer.

[Vandevoorde et al., 2018] Vandevoorde, D., Josuttis, N. M., and Gregor, D. (2018). *C++ Templates: The Complete Guide*. Addison-Wesley Professional.

[Zezula et al., 2007] Zezula, P., Amato, G., Dohnal, V., and Batko, M. (2007). *Similarity Search: The Metric Space Approach*. Springer.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación
CS342 Compilers (Mandatory)

1. General information

- **Semester**: 6th Sem. **Credits**: 4
- **Hour of this course**: **Theory**: 2 hours; **Practice**: 2 hours; **Laboratory**: 2 hours;
- **Prerrequisites**:
  - CS211 Theory of Computation (4th Sem-Pg. 318)

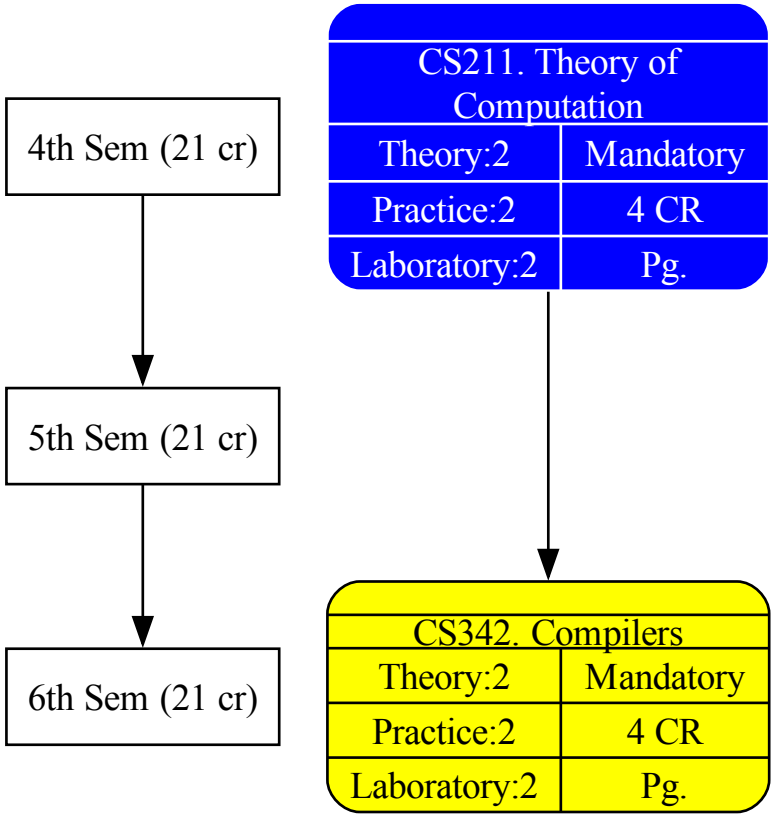

Figure 5.35: Connection Map. CS342 Compilers

2. Course Rationale

This course provides a deep understanding of how high-level languages are transformed into executable machine code. The study of compilers allows students to understand language semantics, memory management, and code optimization, which are fundamental skills for designing domain-specific languages and high-performance software engineering.

3. Goals of the Course

- Understand the phases of a compiler and construction tools.
- Implement lexical and syntactic analyzers based on grammars.
- Generate and optimize intermediate code for a specific architecture.

4. General Competences (*Outcomes*)

AG-C08) Problem Analysis: Identifies, formulates, and analyzes complex computing problems. (Usage)

AG-C12) Applies computer science theory and software development fundamentals to produce computer-based solutions. (Usage)

5. Content

5.1 Program Analysis and Analyzers, 20 hr(s)

Competences: AG-C08,AG-C12

Topics: **I:** Relevant program representations, such as basic blocks, control-flow graphs, def-use chains, and static single assignment **II:** Undecidability and consequences for program analysis **III:** Flow-insensitive analysis, such as type-checking and scalable pointer and alias analysis **IV:** Flow-sensitive analysis, such as forward and backward dataflow analyses **V:** Path-sensitive analysis, such as software model checking and software verification **VI:** Tools and frameworks for implementing analyzers **VII:** Role of static analysis in program optimization and data dependency analysis during exploitation of concurrency **VIII:** Role of program analysis in (partial) verification and bug-finding **IX:** Parallelization: (a) Analysis for auto-parallelization (b) Analysis for detecting concurrency bugs

Learning Outcomes **I:** Explain the difference between dataflow graph and control flow graph [Explain] **II:** Explain why non-trivial sound program analyses must be approximate [Explain] **III:** Argue why an analysis is correct (sound and terminating) [Argue] **IV:** Explain why potential aliasing limits sound program analysis and how alias analysis can help [Explain] **V:** Use the results of a program analysis for program optimization and/or partial program correctness [Use]

Bibliography: [Aho et al., 2006, Appel, 2004]

5.2 Language Translation and Execution, 24 hr(s)

Competences: AG-C08,AG-C12

Topics: **I:** Execution models for JIT (Just-In-Time), compiler, interpreter **II:** Use of intermediate code, e.g., bytecode **III:** Limitations and benefits of JIT, compiler, and interpreter **IV:** Cross compilers/transpilers **V:** BNF and extended BNF representation of context-free grammar **VI:** Parse tree using a simple sentence such as arithmetic expression or if-then-else statement **VII:** Execution as native code or within a virtual machine **VIII:** Language translation pipeline: syntax analysis, parsing, optional type-checking, translation/code generation and optimization, linking, loading, execution **IX:** Run-time representation of core language constructs such as objects (method tables) and functions that can be passed as parameters to and returned from functions (closures) **X:** Secure compiler development

Learning Outcomes **I:** Explain and understand the differences between compiled, JIT, and interpreted language implementations, including the benefits and limitations of each [Explain] **II:** Differentiate syntax and parsing from semantics and evaluation [Differentiate] **III:** Use BNF and extended BNF to specify the syntax of simple constructs such as if-then-else, type declaration and iterative constructs for known languages such as C++ or Python [Use] **IV:** Illustrate the parse tree using a simple sentence/arithmetic expression [Apply] **V:** Illustrate translation of syntax diagrams to BNF/extended BNF for simple constructs such as if-then-else, type declaration, iterative constructs, etc [Apply] **VI:** Illustrate ambiguity in parsing using nested if-then-else/arithmetic expression and show resolution using precedence order [Apply] **VII:** Discuss the benefits and limitations of garbage collection, including the notion of reachability [Debate]

Bibliography: [Aho et al., 2006, Appel, 2004]

5.3 Run-time Behavior and Systems, 20 hr(s)

Competences: AG-C08,AG-C12

Topics: **I:** Process models using stacks and heaps to allocate and deallocate activation records and recovering environments using frame pointers and return addresses during a procedure call including parameter-passing examples **II:** Schematics of code lookup using hash tables for methods in implementations of object-oriented programs **III:** Data layout for objects and activation records **IV:** Object allocation in heap **V:** Implementing virtual entities and virtual methods; virtual method tables and their application **VI:** Run-time behavior of object-oriented programs **VII:** Compare and contrast allocation of memory during information exchange using parameter-passing and non-local variables (using chain of static links). **VIII:** Dynamic memory management approaches and techniques: malloc/free, garbage collection (mark-sweep, copying, reference counting), regions (also known as arenas or zones) **IX:** Just-in-time compilation and dynamic recompilation **X:** Interface to operating system (e.g., for program initialization) **XI:** Interoperability between programming languages including parameter-passing mechanisms and data representation: (a) Big endian, little endian (b) Data layout of composite data types such as arrays **XII:** Other common features of virtual machines, such as class loading, threads, and security checking **XIII:** Sandboxing

Learning Outcomes **I:** Discuss benefits and limitations of automatic memory management [Debate] **II:** Explain the use of metadata in run-time representations of objects and activation records, such as class pointers, array lengths, return addresses, and frame pointers [Explain] **III:** Compare and contrast static allocation vs stack-based allocation vs heap-based allocation of data elements [Compare] **IV:** Explain why some data elements cannot be automatically deallocated at the end of a procedure/method call (need for garbage collection) [Explain] **V:** Discuss advantages, disadvantages, and difficulties of just-in-time and dynamic recompilation [Debate] **VI:** Discuss the use of sandboxing in mobile code [Debate] **VII:** Identify the services provided by modern language run-time systems [Analyze]

Bibliography: [Aho et al., 2006, Cooper and Torczon, 2011]

6. Bibliography

[Aho et al., 2006] Aho, A. V., Lam, M. S., Sethi, R., and Ullman, J. D. (2006). *Compilers: Principles, Techniques, and Tools*. Pearson, 2nd edition.

[Appel, 2004] Appel, A. W. (2004). *Modern Compiler Implementation in Java*. Cambridge University Press, 2nd edition.

[Cooper and Torczon, 2011] Cooper, K. and Torczon, L. (2011). *Engineering a Compiler*. Morgan Kaufmann, 2nd edition.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

AI263 Introduction to Machine Learning (Mandatory)

1. General information

- **Semester**: 6th Sem. **Credits**: 4
- **Hour of this course**: **Theory**: 2 hours; **Laboratory**: 4 hours;
- **Prerrequisites**:
  - CS261 Artificial Intelligence (5th Sem-Pg. 349)

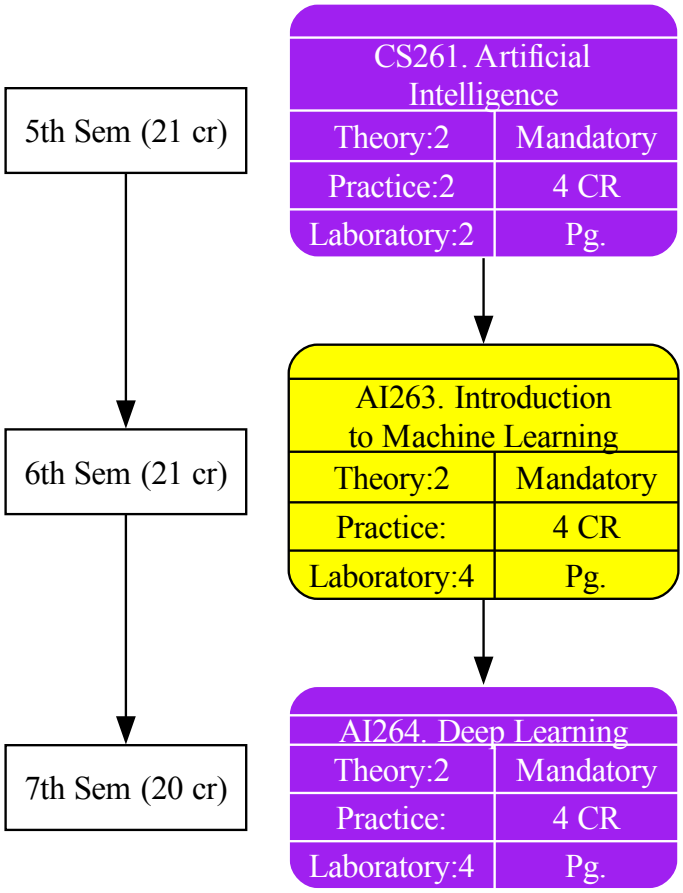

Figure 5.36: Connection Map. AI263 Introduction to Machine Learning

2. Course Rationale

This course introduces the fundamentals of machine learning, covering classical and modern algorithms for classification, regression, and clustering problems. It focuses on practical implementation using scikit-learn and TensorFlow, with applications in computer vision and natural language processing.

3. Goals of the Course

- Understand the mathematical principles behind ML algorithms.

- Implement complete ML pipelines with Python.
- Evaluate and optimize models using standard metrics.

4. General Competences (*Outcomes*)

AG-C08) Problem Analysis: Identifies, formulates, and analyzes complex computing problems. (Usage)

AG-C09) Design and Development of Solutions: Designs, implements, and evaluates solutions for complex computing problems. (Usage)

5. Content

5.1 ML Fundamentals, 15 hr(s)

Competences: AG-C08,AG-C09

Topics: **I:** Supervised vs unsupervised learning **II:** Overfitting and regularization **III:** Cross-validation and learning curves

Learning Outcomes **I:** Explain the bias-variance tradeoff [Familiarity] **II:** Implement k-fold cross-validation [Usage]

Bibliography: [Bishop, 2006, Goodfellow et al., 2016]

5.2 Linear Models, 15 hr(s)

Competences: AG-C08,AG-C09

Topics: **I:** Linear and logistic regression **II:** Support Vector Machines (SVM) **III:** Linear discriminant analysis

Learning Outcomes **I:** Program linear models with scikit-learn [Usage] **II:** Interpret regression coefficients [Assessment]

Bibliography: [Hastie et al., 2009, Murphy, 2012]

5.3 Ensembles and Trees, 15 hr(s)

Competences: AG-C08,AG-C09

Topics: **I:** Decision trees and random forests **II:** Gradient Boosting (XGBoost, LightGBM) **III:** Stacking and voting

Learning Outcomes **I:** Optimize hyperparameters with GridSearch [Usage] **II:** Visualize decision trees [Assessment]

Bibliography: [Hastie et al., 2009, Géron, 2022]

5.4 Basic Neural Networks, 15 hr(s)

Competences: AG-C08,AG-C09

Topics: **I:** Multilayer perceptrons (MLP) **II:** Backpropagation and optimizers **III:** Introduction to Keras/TensorFlow

Learning Outcomes **I:** Build simple neural networks [Usage] **II:** Monitor training with TensorBoard [Assessment]

Bibliography: [Goodfellow et al., 2016, Chollet, 2021]

6. Bibliography

[Bishop, 2006] Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.

[Chollet, 2021] Chollet, F. (2021). *Deep Learning with Python*. Manning, 2nd edition.

[Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press.

[Géron, 2022] Géron, A. (2022). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. O'Reilly, 3rd edition.

[Hastie et al., 2009] Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The Elements of Statistical Learning*. Springer, 2nd edition.

[Murphy, 2012] Murphy, K. P. (2012). *Machine Learning: A Probabilistic Perspective*. MIT Press.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

FI201 Computational Physics (Mandatory)

1. General information

- **Semester**: 6th Sem. **Credits**: 3
- **Hour of this course**: **Theory**: 2 hours; **Laboratory**: 2 hours;
- **Prerrequisites**:
  - BFI101 Physics I (2nd Sem-Pg. 278)

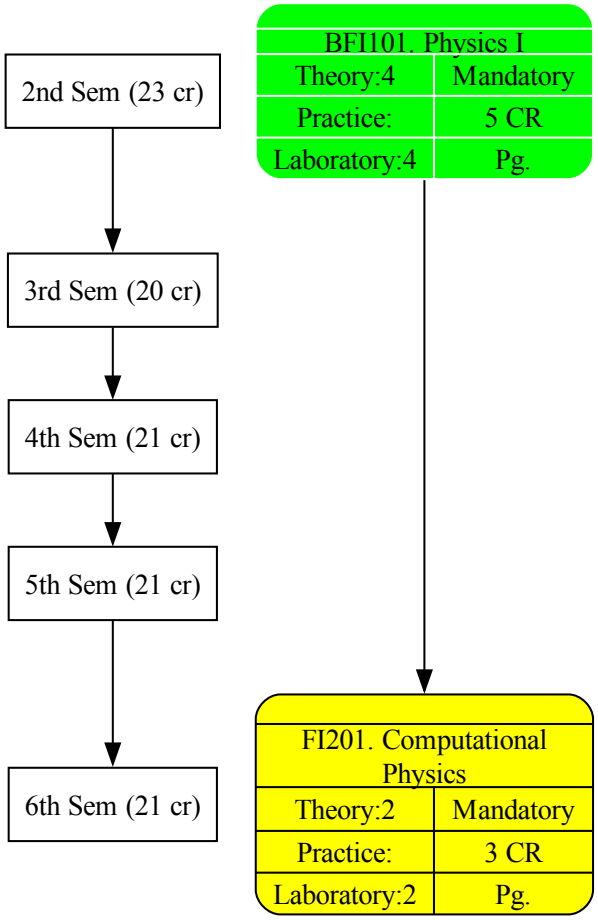

Figure 5.37: Connection Map. FI201 Computational Physics

2. Course Rationale

This course applies the principles of physics to computational problems, with an emphasis on light, wave propagation, collisions, and energy transfer. These concepts are essential in areas such as computer graphics, physical simulations, and video game development.

3. Goals of the Course

- Understand the physical principles relevant to computing.

- Apply these principles to solve specific computational problems.
- Implement physics-based algorithms for simulations and computer graphics.

4. General Competences (*Outcomes*)

AG-C08) Problem Analysis: Identifies, formulates, and analyzes complex computing problems. (Usage)

AG-C11) Use of Tools: Applies modern computing tools in problem solving. (Usage)

5. Content

5.1 Optics and Light Propagation, 10 hr(s)

Competences: AG-C08,AG-C11

Topics: I: Nature of light. **II:** Reflection and refraction. **III:** Lenses and mirrors. **IV:** Interference and diffraction. **V:** Illumination models (e.g., Phong, Blinn-Phong).

Learning Outcomes I: Describe the properties of light and its propagation. [Familiarity] **II:** Apply the laws of reflection and refraction. [Usage] **III:** Implement illumination models in computer graphics. [Assessment]

Bibliography: [Young and Freedman, 2018, Hecht, 2017]

5.2 Collisions and Energy Transfer, 8 hr(s)

Competences: AG-C08,AG-C11

Topics: I: Impulse and linear momentum. **II:** Elastic and inelastic collisions. **III:** Conservation of energy in collisions. **IV:** Deformation of elastic meshes (e.g., mass-spring model).

Learning Outcomes I: Apply the principles of conservation of linear momentum and energy in collisions. [Familiarity] **II:** Model the deformation of elastic meshes due to impact. [Usage] **III:** Implement collision simulations in a computational environment. [Assessment]

Bibliography: [Young and Freedman, 2018, Taylor, 2005]

5.3 Rigid Body Mechanics, 8 hr(s)

Competences: AG-C08,AG-C11

Topics: I: Rotation of rigid bodies. **II:** Moment of inertia. **III:** Torque and rotational kinetic energy.

Learning Outcomes I: Describe the rotation of rigid bodies. [Familiarity] **II:** Calculate the moment of inertia. [Usage] **III:** Apply the equations of rotational dynamics. [Assessment]

Bibliography: [Young and Freedman, 2018, Taylor, 2005]

5.4 Fluid Dynamics, 6 hr(s)

Competences: AG-C08,AG-C11

Topics: I: Basic principles of fluid dynamics. **II:** Viscosity. **III:** Laminar and turbulent flow.

Learning Outcomes I: Describe the properties of fluids. [Familiarity] **II:** Explain the concepts of viscosity and laminar/turbulent flow. [Usage] **III:** Solve simple fluid dynamics problems. [Assessment]

Bibliography: [Young and Freedman, 2018]

5.5 Thermodynamics, 6 hr(s)

Competences: AG-C08,AG-C11

Topics: **I:** Laws of thermodynamics. **II:** Heat transfer.

Learning Outcomes **I:** State the laws of thermodynamics. [Familiarity] **II:** Describe the mechanisms of heat transfer. [Usage]

Bibliography: [Young and Freedman, 2018]

5.6 Physical Simulation, 10 hr(s)

Competences: AG-C08,AG-C11

Topics: **I:** Numerical methods for physical simulation. **II:** Verlet integration. **III:** Collision detection. **IV:** Particle systems. **V:** Constraints and solvers.

Learning Outcomes **I:** Implement basic numerical methods for physical simulation. [Familiarity] **II:** Use Verlet integration to simulate motion. [Usage] **III:** Implement collision detection algorithms. [Assessment]

Bibliography: [Taylor, 2005]

6. Bibliography

[Hecht, 2017] Hecht, E. (2017). *Optics*. Pearson.

[Taylor, 2005] Taylor, J. R. (2005). *Classical Mechanics*. University Science Books.

[Young and Freedman, 2018] Young, H. D. and Freedman, R. A. (2018). *University Physics with Modern Physics*. Pearson.

Seventh Semester

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

CS251 Computer Graphics (Mandatory)

1. General information

- **Semester**: 7th Sem. **Credits**: 4
- **Hour of this course**: **Theory**: 2 hours; **Laboratory**: 4 hours;
- **Prerrequisites**:
  - MA202 Numerical Methods (4th Sem-Pg. 335)

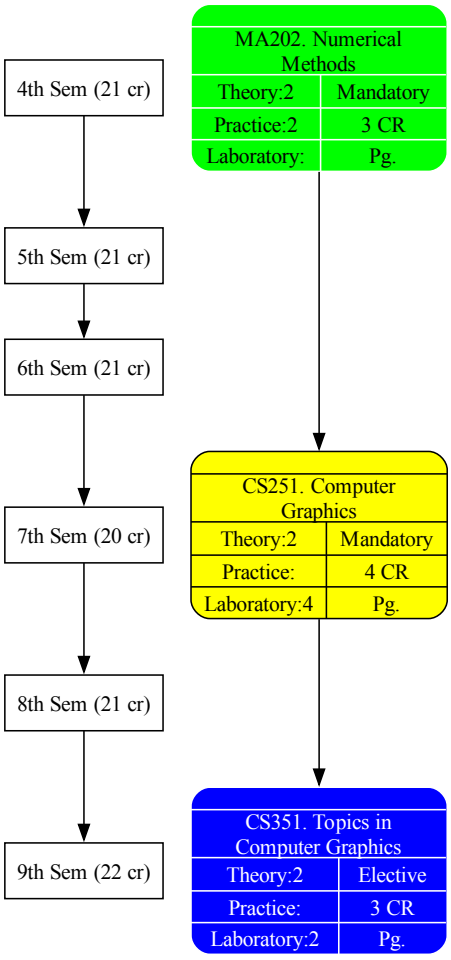

Figure 5.38: Connection Map. CS251 Computer Graphics

2. Course Rationale

Computer Graphics is a fundamental area of Computer Science that focuses on the mathematical and computational techniques for generating and manipulating images. This course provides students with the foundations of 2D and 3D modeling, rendering pipelines, and the physical principles of light

and color, enabling them to develop high-performance visual applications, simulations, and interactive systems.

3. Goals of the Course

- Understand the graphics pipeline and coordinate system transformations.
- Implement rendering algorithms for realistic and real-time visualization.
- Apply mathematical modeling to create and manipulate 3D objects.

4. General Competences (*Outcomes*)

AG-C07) Computing Knowledge: Applies knowledge of mathematics, science, and computing. (Usage)

AG-C08) Problem Analysis: Identifies, formulates, and analyzes complex computing problems. (Usage)

AG-C09) Design and Development of Solutions: Designs, implements, and evaluates solutions for complex computing problems. (Assessment)

5. Content

5.1 Fundamental Concepts of Graphics and Interactive Techniques, 16 hr(s)

Competences: AG-C07,AG-C08

Topics: **I:** Overview of computer graphics pipeline and interactive systems **II:** Basic rendering concepts: rasterization, ray tracing, shading models **III:** Color theory and color spaces **IV:** Coordinate systems and transformations **V:** Basic interaction techniques: input devices, event handling **VI:** Advanced rendering concepts: global illumination, physically based rendering **VII:** Graphics hardware and acceleration techniques **VIII:** Human perception and visual cognition

Learning Outcomes **I:** Explain the basic stages of the computer graphics pipeline [Explain] **II:** Describe basic rendering techniques such as rasterization and ray tracing [Describe] **III:** Apply color theory to design visually effective graphics [Apply] **IV:** Implement coordinate transformations for 2D and 3D graphics [Implement] **V:** Design basic interactive applications using common input devices and event handling [Design] **VI:** Compare advanced rendering techniques such as global illumination and physically based rendering [Compare] **VII:** Analyze the role of graphics hardware in accelerating rendering tasks [Analyze] **VIII:** Explain how human perception influences graphics and interaction design [Explain]

Bibliography: [Marschner and Shirley, 2021, Hearn et al., 2014]

5.2 Applied Rendering and Techniques, 16 hr(s)

Competences: AG-C09

Topics: **I:** Rasterization algorithms: scan conversion, antialiasing **II:** Ray tracing: acceleration structures, sampling, shading **III:** Texture mapping and filtering **IV:** Shading models: Lambertian, Phong, BRDF **V:** Advanced techniques: shadow mapping, environment mapping, deferred rendering

Learning Outcomes **I:** Implement rasterization algorithms for primitive rendering [Implement] **II:** Implement basic ray tracing with acceleration structures [Implement] **III:** Apply texture mapping and filtering techniques to enhance visual realism [Apply] **IV:** Compare different shading models and their applications [Compare] **V:** Implement advanced rendering techniques such as shadow mapping or deferred rendering [Implement]

Bibliography: [Marschner and Shirley, 2021, Hearn et al., 2014, Angel and Shreiner, 2014]

5.3 Geometric Modeling, 16 hr(s)

Competences: AG-C08,AG-C09

Topics: **I:** Geometric representations: polygons, curves, surfaces, implicit forms **II:** Modeling techniques: splines, subdivision surfaces, constructive solid geometry **III:** Procedural modeling and generation **IV:** Geometric algorithms: intersection, distance, decomposition

Learning Outcomes **I:** Describe different geometric representations and their trade-offs [Describe] **II:** Implement modeling techniques such as spline curves or subdivision surfaces [Implement] **III:** Create procedural models using algorithms or rules [Create] **IV:** Apply geometric algorithms to solve problems like intersection testing or mesh processing [Apply]

Bibliography: [Marschner and Shirley, 2021, Angel and Shreiner, 2014]

5.4 Visualization, 16 hr(s)

Competences: AG-C09

Topics: **I:** Principles of data visualization: perception, cognition, and effective encoding **II:** Types of visualizations: scientific, information, geographical **III:** Visualization techniques for multivariate, temporal, and hierarchical data **IV:** Interactive visualization and exploration **V:** Evaluation of visualization effectiveness

Learning Outcomes **I:** Apply principles of perception and cognition to design effective visualizations [Apply] **II:** Select appropriate visualization techniques for different data types and tasks [Evaluate] **III:** Implement interactive visualizations for data exploration [Implement] **IV:** Evaluate the effectiveness of a visualization using empirical methods [Evaluate]

Bibliography: [Marschner and Shirley, 2021]

5.5 Shading and Advanced Rendering, 12 hr(s)

Competences: AG-C09

Topics: **I:** Advanced shading: physically based shading, subsurface scattering **II:** Global illumination: radiosity, photon mapping, path tracing **III:** Real-time rendering techniques: level of detail, culling, occlusion **IV:** GPU programming and shaders

Learning Outcomes **I:** Implement advanced shading models for realistic materials [Implement] **II:** Compare global illumination algorithms and their trade-offs [Compare] **III:** Optimize real-time rendering using techniques like level of detail and culling [Create] **IV:** Develop GPU shaders for custom rendering effects [Create]

Bibliography: [Marschner and Shirley, 2021, Angel and Shreiner, 2014]

6. Bibliography

[Angel and Shreiner, 2014] Angel, E. and Shreiner, D. (2014). *Interactive Computer Graphics: A Top-Down Approach with WebGL*. Pearson, 7th edition.

[Hearn et al., 2014] Hearn, D., Baker, M. P., and Carithers, W. (2014). *Computer Graphics with OpenGL*. Pearson, 4th edition.

[Marschner and Shirley, 2021] Marschner, S. and Shirley, P. (2021). *Fundamentals of Computer Graphics*. A K Peters/CRC Press, 5th edition.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

CS292 Software Engineering II (Mandatory)

1. General information

- **Semester**: 7th Sem. **Credits**: 4
- **Hour of this course**: **Theory**: 2 hours; **Practice**: 2 hours; **Laboratory**: 2 hours;
- **Prerrequisites**:
  - CS291 Software Engineering I (5th Sem-Pg. 360)

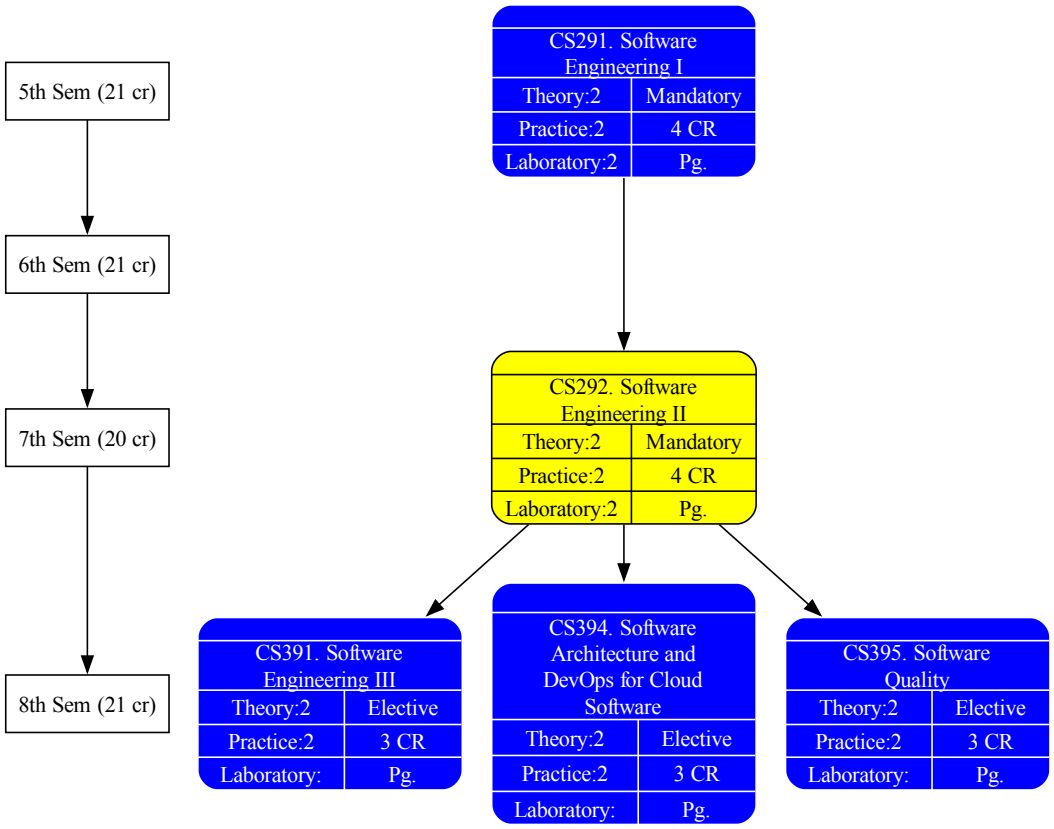

Figure 5.39: Connection Map. CS292 Software Engineering II

2. Course Rationale

Software Engineering II builds upon the foundational principles of the first course, shifting focus towards advanced design, software evolution, and the management of large-scale projects. It explores the complexities of maintaining systems, ensuring software quality through rigorous validation, and the application of professional teamwork and project management techniques in collaborative environments.

3. Goals of the Course

- Apply advanced software design principles and patterns to solve complex architectural problems.
- Master software configuration management and collaborative development practices.
- Understand the principles of software evolution, refactoring, and maintenance.

- Implement systematic testing strategies and understand software reliability.
- Develop professional skills for team-based software development and stakeholder management.

4. General Competences (*Outcomes*)

AG-C03) Individual and Team Work: Performs effectively as an individual and as a member or leader in diverse teams. (Usage)

AG-C09) Design and Development of Solutions: Designs, implements, and evaluates solutions for complex computing problems. (Usage)

5. Content

5.1 Team Collaboration and Dynamics, 14 hr(s)

Competences: AG-C03,AG-C09

Topics: **I:** Effective communication, including oral and written, as well as formal (email, docs, comments, presentations) and informal (team chat, meetings). Technical and Professional Communication, Team Communication and Culture **II:** Common causes of team conflict, and approaches for conflict resolution. **III:** Cooperative programming: (a) Pair programming or Swarming (b) Code review (c) Collaboration through version control **IV:** Roles and responsibilities in a software team: Ethics Codes and Legal Accountability, Professional Responsibility and Public Policy, Professional Values, Nature and Continuing Development, Ethical Dissent and Cultural Context, Workplace Ethics (a) Advantages of teamwork (b) Risks and complexity of such collaboration **V:** Team processes - responsibilities for tasks, effort estimation, meeting structure, work schedule

Learning Outcomes **I:** Follow effective team communication practices [Analyze] **II:** Articulate the sources of, hazards of, and potential benefits of team conflict - especially focusing on the value of disagreeing about ideas or proposals without insulting people [Articulate] **III:** Facilitate a conflict-resolution and problem-solving strategy in a team setting [Analyze] **IV:** Collaborate effectively in cooperative development/programming [Analyze] **V:** Propose and delegate necessary roles and responsibilities in a software development team [Propose] **VI:** Compose and follow an agenda for a team meeting [Compose] **VII:** Facilitate through involvement in a team project, the central elements of team building, establishing healthy team culture, and team management including creating and executing a team work plan [Create]

Bibliography: [Sommerville, 2015, Pressman and Maxim, 2019]

5.2 Principles and Software Architecture, 16 hr(s)

Competences: AG-C03,AG-C09

Topics: **I:** System design principles. System Reliability (a) Levels of abstraction (e.g., architectural design and detailed design) (b) Separation of concerns (c) Information hiding (d) Coupling and cohesion **II:** Software architecture. System Reliability (a) Design paradigms i. Top-down functional decomposition/layered design ii. Data-oriented architecture iii. Object-oriented analysis and design iv. Event-driven design (b) Standard architectures (e.g., client-server and microservice architectures including REST discussions, n-layer, pipes-and-filters, Model View Controller) (c) Identifying component boundaries and dependencies **III:** Programming in the large vs programming in the small. System Reliability **IV:** Code smells and other indications of code quality, distinct from correctness. Security Design and Controls Engineering, Threat Analysis and Security Engineering, Trusted Computing and Privacy Engineering

Learning Outcomes **I:** Identify the standard software architecture of a given high-level design [Analyze] **II:** Select and use an appropriate design paradigm to design a simple software system and explain how system design principles have been applied in this design [Evaluate] **III:** Adapt

a flawed system design to better follow principles such as separation of concerns or information hiding [Create] **IV:** Identify the dependencies among a set of software components in an architectural design [Analyze]

Bibliography: [Gamma et al., 1994, Pressman and Maxim, 2019]

5.3 Coding Practices, 14 hr(s)

Competences: AG-C03,AG-C09

Topics: I: Practical small-scale testing Software Development Practices (a) Unit testing (b) Test-driven development - This is particularly valuable for students psychologically, as it is far easier to engage constructively with the challenge of identifying challenging inputs for a given API (edge cases, corner cases) a priori. If they implement first, the instinct is often to avoid trying to crash their new creation, while a test-first approach gives them the intellectual satisfaction of spotting the problem cases and then watching as more tests pass during the development process. **II:** Documentation Software Development Practices (a) Interface documentation - describe interface requirements, potentially including (formal or informal) contracts, pre and post conditions, invariants. (b) Implementation documentation should focus on tricky and non-obvious pieces of code, whether because the code is using advanced language features, or the behavior of the code is complex. (Do not add comments that re-state common/obvious operations and simple language features.) i. Clarify dataflow, computation, etc., focusing on what the code is. ii. Identify subtle/tricky pieces of code and refactor to be self-explanatory if possible or provide appropriate comments to clarify. **III:** Coding style Software Development Practices (a) Style guides (b) Commenting (c) Naming **IV:** "Best Practices" for coding: techniques, idioms/patterns, mechanisms for building quality programs Software Development Practices, Information Flow and Non-Interference, Injection and Input Validation, Memory Safety and Types, Malware Analysis and Advanced Security (a) Defensive coding practices (b) Secure coding practices and principles (c) Using exception handling mechanisms to make programs more robust, fault-tolerant **V:** Debugging Software Development Practices **VI:** Logging **VII:** Use of libraries and frameworks developed by others Software Development Practices

Learning Outcomes I: Write appropriate unit tests for a small component (several functions, a single type, etc.) [Write] **II:** Write appropriate interface and (if needed) implementation comments for a small component [Write] **III:** Describe techniques, coding idioms and mechanisms for implementing designs to achieve desired properties such as reliability, efficiency, and robustness [Describe] **IV:** Write robust code using exception handling mechanisms [Write] **V:** Describe secure coding and defensive coding practices [Describe] **VI:** Select and use a defined coding standard in a small software project [Use] **VII:** Compare and contrast integration strategies including top-down, bottom-up, and sandwich integration [Compare] **VIII:** Describe the process of analyzing and implementing changes to code base developed for a specific project [Describe] **IX:** Describe the process of analyzing and implementing changes to a large existing code base [Describe]

Bibliography: [Sommerville, 2015]

5.4 Verification and validation Concepts, 10 hr(s)

Competences: AG-C03,AG-C09

Topics: I: Verification and validation concepts (a) Verification: Are we building the thing right? (b) Validation: Did we build the right thing? **II:** Why testing matters: Does the component remain functional as the code evolves? **III:** Testing objectives (a) Usability (b) Reliability (c) Conformance to specification (d) Performance (e) Security

Learning Outcomes I: Explain why testing is important [Explain] **II:** Distinguish between program validation and verification [Distinguish] **III:** Describe different objectives of testing [Describe]

Bibliography: [Pressman and Maxim, 2019]

5.5 Testing Tools and Analysis, 4 hr(s)

Competences: AG-C03,AG-C09

Topics: **I:** Verification and validation tooling and automation (a) Static analysis (b) Code coverage (c) Fuzzing (d) Dynamic analysis and fault containment (sanitizers, etc.) (e) Fault logging and fault tracking **II:** Test planning and generation (a) Fault estimation and testing termination including defect seeding (b) Use of random and pseudo random numbers in testing **III:** Testing asynchronous, parallel, and concurrent systems **IV:** Verification and validation of non-code artifacts (documentation, training materials) **V:** Behavior-driven development (BDD) (a) Expressing acceptance criteria as Gherkin Given/When/Then scenarios (b) Executing BDD scenarios as automated end-to-end tests **VI:** Mutation testing and static code-quality analysis (a) Mutation testing to assess test-suite effectiveness (b) Static detection of code smells and excessive cyclomatic complexity

Learning Outcomes **I:** Describe and compare different tools for verification and validation [Describe] **II:** Automate the use of different tools in a small software project [Design] **III:** Explain how and when random numbers should be used in testing [Explain] **IV:** Describe approaches for fault estimation [Describe] **V:** Estimate the number of faults in a small software application based on fault density and fault seeding [Estimate] **VI:** Describe techniques and issues with testing asynchronous, concurrent, and parallel software [Describe] **VII:** Create a test plan for a medium-size code segment which contains asynchronous, concurrent, and/or parallel code, including a test matrix and generation of test data and inputs [Create] **VIII:** Describe techniques for the verification and validation of non-code artifacts [Describe] **IX:** Write Gherkin Given/When/Then scenarios for a feature's acceptance criteria and execute them as automated end-to-end tests [Write] **X:** Analyze a test suite's mutation score and refactor tests to kill surviving mutants [Analyze]

Bibliography: [Sommerville, 2015, Pressman and Maxim, 2019]

5.6 Performance Testing and Benchmarking, 4 hr(s)

Competences: AG-C03,AG-C09

Topics: **I:** Performance testing and benchmarking (a) Throughput and latency (b) Degradation under load (stress testing, FIFO vs LIFO handling of requests) (c) Speedup and scaling i. Amdahl's law ii. Gustafson's law iii. Soft and weak scaling (d) Identifying and measuring figures of merits (e) Common performance bottlenecks i. Compute-bound ii. Memory-bandwidth bound iii. Latency-bound (f) Statistical methods and best practices for benchmarking i. Estimation of uncertainty ii. Confidence intervals (g) Analysis and presentation (graphs, etc.) (h) Timing techniques

Learning Outcomes **I:** Describe throughput and latency and provide examples of each [Describe] **II:** Explain speedup and the different forms of scaling and how they are computed [Explain] **III:** Describe common performance bottlenecks [Describe] **IV:** Describe statistical methods and best practices for benchmarking software [Describe] **V:** Explain techniques for and challenges with measuring time when constructing a benchmark [Explain] **VI:** Identify the figures of merit, construct and run a benchmark, and statistically analyze and visualize the results for a small software project [Analyze]

Bibliography: [Bondi, 2015, Gregg, 2020]

5.7 Reliability Concepts, 10 hr(s)

Competences: AG-C03,AG-C09

Topics: **I:** Concept of reliability as probability of failure or mean time between failures, and faults as cause of failures **II:** Identifying reliability requirements for different kinds of software **III:** Software failures caused by defects/bugs, and so for high reliability the goal is to have

minimum defects - by injecting fewer defects (better training, education, planning), and by removing most of the injected defects (testing, code review, etc.) **IV:** Software reliability, system reliability and failure behavior **V:** Defect injection and removal cycle, and different approaches for defect removal **VI:** Compare the "error budget" approach to reliability with the "error-free" approach and identify domains where each is relevant.

Learning Outcomes **I:** Describe how to determine the level of reliability required by a software system [Describe] **II:** Explain the problems that exist in achieving very high levels of reliability [Explain] **III:** Understand approaches to minimizing faults that can be applied at each stage of the software lifecycle [Explain]

Bibliography: [Sommerville, 2015]

6. Bibliography

[Bondi, 2015] Bondi, A. B. (2015). *Foundations of Software and System Performance Engineering: Process, Performance Modeling, Requirements, Testing, Scalability, and Practice*. Addison-Wesley, Upper Saddle River, NJ.

[Gamma et al., 1994] Gamma, E., Helm, R., Johnson, R., and Vlissides, J. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley.

[Gregg, 2020] Gregg, B. (2020). *Systems Performance: Enterprise and the Cloud*. Addison-Wesley Professional, Boston, MA, 2nd edition.

[Pressman and Maxim, 2019] Pressman, R. S. and Maxim, B. (2019). *Software Engineering: A Practitioner's Approach*. McGraw-Hill Education, 9th edition.

[Sommerville, 2015] Sommerville, I. (2015). *Software Engineering*. Pearson, 10th edition.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

CS2H1 User Experience (UX) (Mandatory)

1. General information

- **Semester**: 7th Sem. **Credits**: 4
- **Hour of this course**: **Theory**: 2 hours; **Laboratory**: 4 hours;
- **Prerrequisites**:
  - CS291 Software Engineering I (5th Sem-Pg. 360)

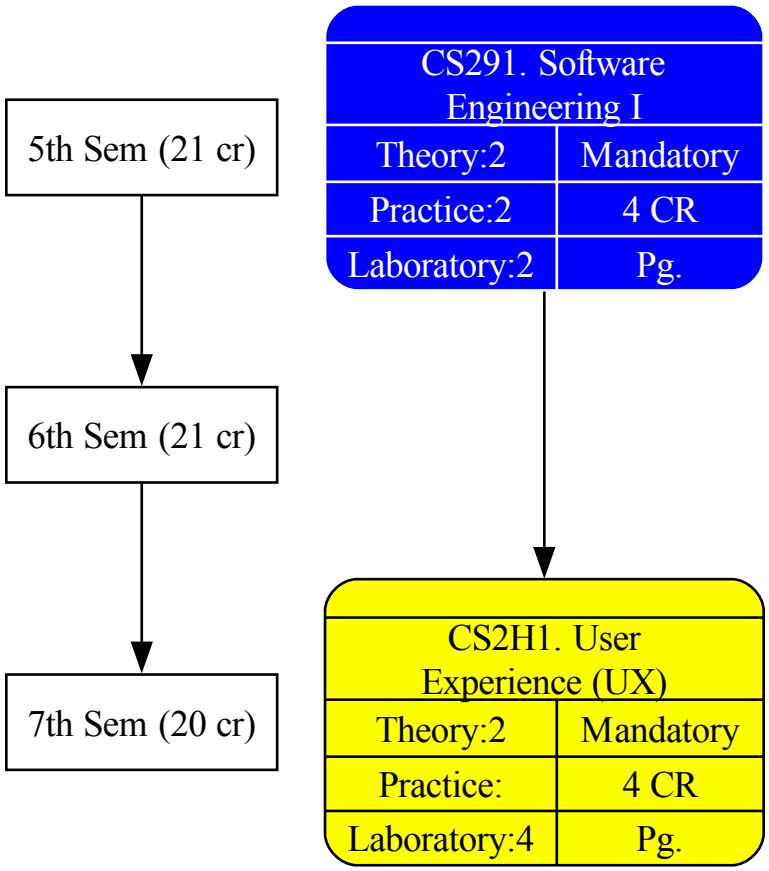

Figure 5.40: Connection Map. CS2H1 User Experience (UX)

2. Course Rationale

This course focuses on the design, evaluation, and implementation of interactive computing systems for human use. Beyond classical HCI principles, it addresses the emerging challenges of interacting with AI systems, intelligent agents, and natural language processing software. Students will develop competencies in user-centered design, accessibility, ethical accountability, and the evaluation of both traditional interfaces and AI-driven conversational systems, acquiring skills essential for designing responsible and human-centered technology.

3. Goals of the Course

- Understand human cognitive, physical, and emotional characteristics relevant to interactive system design.
- Apply ethical accountability and value-sensitive design principles to HCI practice, including algorithmic fairness and bias.
- Apply accessibility standards and inclusive design principles for diverse populations and abilities.
- Execute rigorous usability evaluation methods to iterate and improve user interfaces.
- Design interactive systems using user-centered and iterative design processes across multiple platforms and domains.
- Analyze the societal, ethical, and professional dimensions of interactive technology.
- Understand agent architectures and apply principles of human-agent interaction including AI assistants and autonomous systems.
- Apply natural language processing techniques to design and evaluate human-software interfaces based on language models.

4. General Competences (*Outcomes*)

AG-C03) Individual and Team Work: Performs effectively as an individual and as a member or leader in diverse teams. (Usage)

AG-C09) Design and Development of Solutions: Designs, implements, and evaluates solutions for complex computing problems. (Usage)

5. Content

5.1 Understanding the User: Individual goals and interactions with others, 8 hr(s)

Competences: AG-C03,AG-C09

Topics: **I:** Human capabilities and limitations: perception, cognition, motor skills **II:** User characteristics: demographics, expertise, cultural background **III:** User goals, tasks, and contexts of use **IV:** Social and organizational aspects of interaction **V:** Cognitive models and theories relevant to HCI **VI:** Emotional and affective aspects of interaction **VII:** User research methods: interviews, surveys, observations

Learning Outcomes **I:** Describe human capabilities and limitations that affect interaction design [Describe] **II:** Analyze how user characteristics influence interaction with computing systems [Analyze] **III:** Identify user goals and tasks in specific contexts of use [Analyze] **IV:** Explain social and organizational factors that impact technology adoption and use [Explain] **V:** Apply cognitive models to predict and explain user behavior [Apply] **VI:** Consider emotional and affective factors in interaction design [Analyze] **VII:** Conduct user research using appropriate methods such as interviews or observations [Evaluate]

Bibliography: [Norman, 2013, Johnson, 2020]

5.2 Accountability and Responsibility in Design, 8 hr(s)

Competences: AG-C03,AG-C09

Topics: **I:** Ethical considerations in HCI: fairness, transparency, accountability **II:** Design for values and value-sensitive design **III:** Responsible innovation and design ethics **IV:** Algorithmic accountability and bias in interactive systems **V:** Professional responsibility in HCI practice **VI:** Legal and regulatory considerations for interactive systems

Learning Outcomes **I:** Identify ethical issues in HCI design and implementation [Analyze] **II:** Apply value-sensitive design approaches to technology development [Apply] **III:** Discuss responsible innovation practices in HCI [Debate] **IV:** Analyze algorithmic accountability and bias in interactive systems [Analyze] **V:** Describe professional responsibilities in HCI practice [Describe] **VI:** Explain legal and regulatory considerations relevant to interactive systems [Explain]

Bibliography: [Shneiderman, 2022, Norman, 2013]

5.3 Accessibility and Inclusive Design, 8 hr(s)

Competences: AG-C03,AG-C09

Topics: **I:** Accessibility principles and guidelines **II:** Disabilities and assistive technologies **III:** Inclusive design and universal design principles **IV:** Evaluation methods for accessibility **V:** Designing for diverse abilities and contexts **VI:** Accessibility standards and compliance

Learning Outcomes **I:** Apply accessibility principles to interactive system design [Apply] **II:** Describe how different disabilities affect interaction with technology [Describe] **III:** Design inclusive systems using universal design principles [Design] **IV:** Conduct accessibility evaluations of interactive systems [Evaluate] **V:** Create designs that accommodate diverse abilities and usage contexts [Create] **VI:** Ensure compliance with accessibility standards and regulations [Evaluate]

Bibliography: [Johnson, 2020]

5.4 Evaluating the Design, 8 hr(s)

Competences: AG-C03,AG-C09

Topics: **I:** Usability principles and heuristics **II:** Evaluation methods: heuristic evaluation, cognitive walkthrough **III:** User testing: planning, conducting, analyzing **IV:** Advanced evaluation methods: controlled experiments, field studies **V:** UX metrics and measurement **VI:** Analytical evaluation methods: modeling and simulation

Learning Outcomes **I:** Apply usability principles and heuristics to evaluate designs [Apply] **II:** Conduct heuristic evaluations and cognitive walkthroughs [Evaluate] **III:** Plan, conduct, and analyze user testing sessions [Plan] **IV:** Design and conduct controlled experiments for HCI research [Design] **V:** Measure user experience using appropriate metrics [Evaluate] **VI:** Use analytical methods to model and predict user performance [Use]

Bibliography: [Norman, 2013, Johnson, 2020]

5.5 System Design, 8 hr(s)

Competences: AG-C03,AG-C09

Topics: **I:** Design process: user-centered design, iterative design **II:** Interaction design: concepts, principles, patterns **III:** Prototyping techniques: low-fidelity, high-fidelity, interactive **IV:** Information architecture and navigation design **V:** Advanced interaction techniques: gesture, voice, multimodal **VI:** Design for specific domains: mobile, web, ubiquitous computing **VII:** Collaborative and social computing systems

Learning Outcomes **I:** Apply user-centered design processes to develop interactive systems [Apply] **II:** Design interactions using established principles and patterns [Design] **III:** Create prototypes at different fidelity levels for design exploration [Create] **IV:** Structure information and navigation for effective user interaction [Structure] **V:** Implement advanced interaction techniques such as gesture or voice interfaces [Implement] **VI:** Design interactive systems for specific domains and platforms [Design] **VII:** Design collaborative systems that support group interaction [Design]

Bibliography: [Johnson, 2020, Norman, 2013]

5.6 Society, Ethics, and the Profession, 8 hr(s)

Competences: AG-C03,AG-C09

Topics: **I:** Social impact of interactive technologies **II:** Ethical issues in HCI research and practice **III:** Professional practice in HCI **IV:** Cultural and cross-cultural considerations in HCI **V:** Sustainability and HCI **VI:** Future trends and challenges in HCI

Learning Outcomes **I:** Analyze the social impact of interactive technologies on individuals and society [Analyze] **II:** Address ethical issues in HCI research and design projects [Evaluate] **III:** Describe professional practices and career paths in HCI [Describe] **IV:** Design systems that account for cultural differences and contexts [Design] **V:** Incorporate sustainability considerations into HCI design [Integrate] **VI:** Discuss emerging trends and future challenges in HCI [Debate]

Bibliography: [Shneiderman, 2022, Norman, 2013]

5.7 Agents and Cognitive Systems, 8 hr(s)

Competences: AG-C03,AG-C09

Topics: **I:** Agent architectures (e.g., reactive, layered, cognitive) **II:** Agent theory (including mathematical formalisms) **III:** Rationality, Game Theory: (a) Decision-theoretic agents (b) Markov decision processes (MDP) (c) Bandit algorithms **IV:** Software agents, personal assistants, and information access: (a) Collaborative agents (b) Information-gathering agents (c) Believable agents (synthetic characters, modeling emotions in agents) **V:** Learning agents **VI:** Cognitive systems: (a) Cognitive architectures (e.g., ACT-R, SOAR, ICARUS, FORR) (b) Capabilities (e.g., perception, decision making, prediction, knowledge maintenance) (c) Knowledge representation, organization, utilization, acquisition, and refinement (d) Applications and evaluation of cognitive systems **VII:** Multi-agent systems: (a) Collaborating agents (b) Agent teams (c) Competitive agents (e.g., auctions, voting) (d) Swarm systems and biologically inspired models (e) Multi-agent learning **VIII:** Human-agent interaction: (a) Communication methodologies (verbal and non-verbal) (b) Practical issues (c) Applications: i. Trading agents, supply chain management ii. Ethical issues of AI interactions with humans iii. Regulation and legal requirements of AI systems for interacting with humans

Learning Outcomes **I:** Characterize and contrast the standard agent architectures [Evaluate] **II:** Describe the applications of agent theory to domains such as software agents, personal assistants, and believable agents, and discuss associated ethical implications [Describe] **III:** Describe the primary paradigms used by learning agents [Describe] **IV:** Demonstrate using appropriate examples how multi-agent systems support agent interaction [Demonstrate] **V:** Construct an intelligent agent using a well-established cognitive architecture (ACT-R, SOAR) for solving a specific problem [Apply]

Bibliography: [Russell and Norvig, 2021, Shneiderman, 2022]

5.8 Natural Language Processing, 8 hr(s)

Competences: AG-C03,AG-C09

Topics: **I:** Deterministic and stochastic grammars **II:** Parsing algorithms: (a) CFGs and chart parsers (e.g., CYK) (b) Probabilistic CFGs and weighted CYK **III:** Representing meaning/Semantics: (a) Logic-based knowledge representations (b) Semantic roles (c) Temporal representations (d) Beliefs, desires, and intentions **IV:** Corpus-based methods **V:** N-grams and HMMs **VI:** Smoothing and backoff **VII:** Examples of use: POS tagging and morphology **VIII:** Information retrieval: (a) Vector space model: i. TF & IDF (b) Precision and recall **IX:** Information extraction **X:** Language translation **XI:** Text classification, categorization: (a) Bag of words model **XII:** Deep learning for NLP: (a) RNNs (b) Transformers (c) Multimodal embeddings (e.g., images + text) (d) Generative language models

Learning Outcomes **I:** Define and contrast deterministic and stochastic grammars, providing examples to show the adequacy of each [Define] **II:** Simulate, apply, or implement classic and stochastic algorithms for parsing natural language [Apply] **III:** Identify the challenges of representing meaning [Analyze] **IV:** List the advantages of using standard corpora. Identify examples of current corpora for a variety of NLP tasks [List] **V:** Identify techniques for information retrieval, language translation, and text classification [Analyze] **VI:** Implement a TF/IDF transform, use it to extract features from a corpus, and train an off-the-shelf machine learning algorithm using those features to do text classification [Apply]

Bibliography: [Jurafsky and Martin, 2024, Russell and Norvig, 2021]

6. Bibliography

[Johnson, 2020] Johnson, J. (2020). *Designing with the Mind in Mind: Simple Guide to Understanding User Interface Design Guidelines*. Morgan Kaufmann, 3rd edition.

[Jurafsky and Martin, 2024] Jurafsky, D. and Martin, J. H. (2024). *Speech and Language Processing*. Prentice Hall, 3rd edition.

[Norman, 2013] Norman, D. (2013). *The Design of Everyday Things: Revised and Expanded Edition*. Basic Books, 2nd edition.

[Russell and Norvig, 2021] Russell, S. and Norvig, P. (2021). *Artificial Intelligence: A Modern Approach*. Pearson, 4th edition.

[Shneiderman, 2022] Shneiderman, B. (2022). *Human-Centered AI*. Oxford University Press, 1st edition.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

AI264 Deep Learning (Mandatory)

1. General information

- **Semester**: 7th Sem. **Credits**: 4
- **Hour of this course**: **Theory**: 2 hours; **Laboratory**: 4 hours;
- **Prerrequisites**:
  - AI263 Introduction to Machine Learning (6th Sem-Pg. 392)

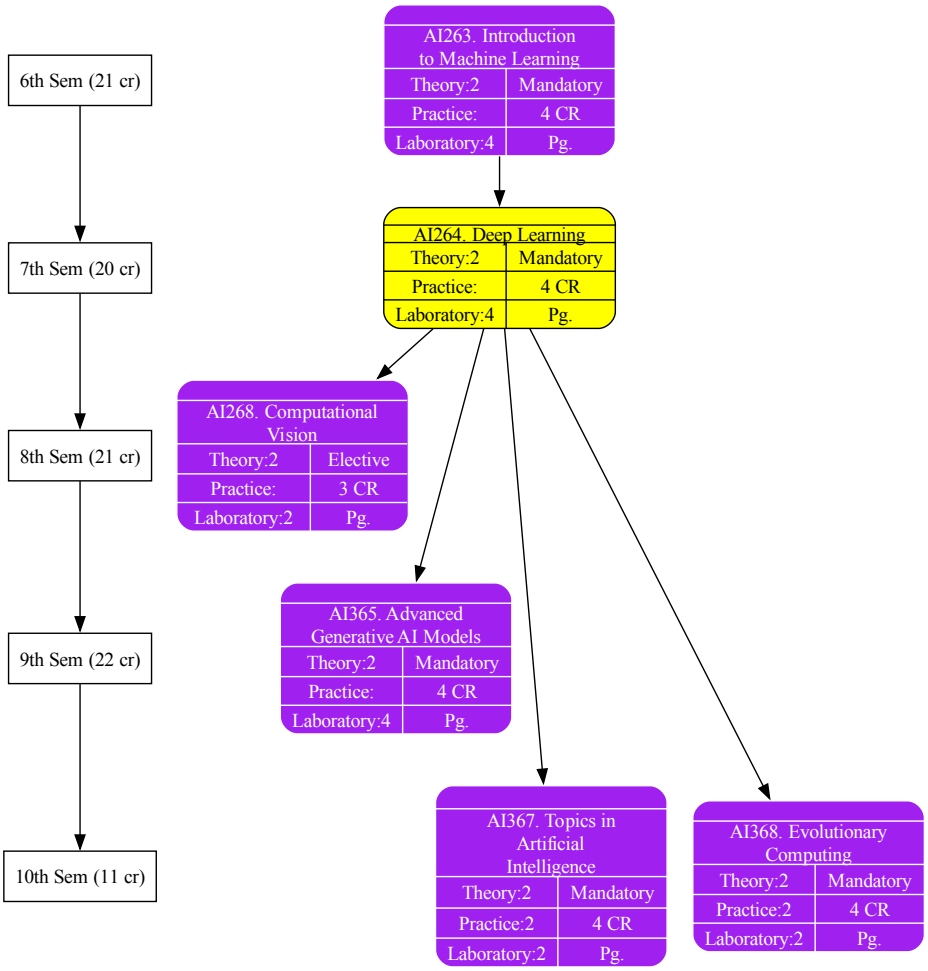

Figure 5.41: Connection Map. AI264 Deep Learning

2. Course Rationale

This course covers modern deep learning fundamentals, including convolutional networks, transformer architectures, and training techniques for advanced AI systems.

3. Goals of the Course

- Implement deep neural networks using modern frameworks (PyTorch/TensorFlow)
- Design architectures for specific domains (vision, language, etc.)
- Analyze ethical limitations of deep models

4. General Competences (*Outcomes*)

AG-C09) Design and Development of Solutions: Designs, implements, and evaluates solutions for complex computing problems. (Usage)

AG-C11) Use of Tools: Applies modern computing tools in problem solving. (Assessment)

5. Content

5.1 Neural Network Training and Optimization, 10 hr(s)

Competences: AG-C09,AG-C11

Topics: **I:** Modern backpropagation: computational graphs and automatic differentiation **II:** Advanced optimizers: momentum, RMSProp, Adam, and learning-rate schedules

Learning Outcomes **I:** Implement a deep neural network from scratch, including its backpropagation pass [Apply] **II:** Select and tune an optimizer and its learning-rate schedule for training stability [Apply]

Bibliography: [Goodfellow et al., 2016]

5.2 Regularization Techniques for Deep Networks, 5 hr(s)

Competences: AG-C09,AG-C11

Topics: **I:** Dropout regularization **II:** Batch normalization

Learning Outcomes **I:** Diagnose overfitting in a training run using regularization diagnostics [Analyze] **II:** Apply dropout and batch normalization to improve a model's generalization [Apply]

Bibliography: [Goodfellow et al., 2016]

5.3 Convolutional and Residual Network Architectures, 9 hr(s)

Competences: AG-C09,AG-C11

Topics: **I:** Deep convolutional neural network architectures **II:** Residual connections and ResNets

Learning Outcomes **I:** Design a deep CNN architecture for a given vision task [Apply] **II:** Adapt a residual architecture to a specific vision domain [Apply]

Bibliography: [He et al., 2016]

5.4 Transformer Architectures and Generative Models, 16 hr(s)

Competences: AG-C09,AG-C11

Topics: **I:** Transformers and the attention mechanism **II:** Generative models: variational autoencoders (VAEs) and generative adversarial networks (GANs)

Learning Outcomes **I:** Train a transformer model on a real dataset [Apply] **II:** Analyze and contrast VAE and GAN generative-model architectures [Analyze]

Bibliography: [Vaswani et al., 2017]

6. Bibliography

[Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press.

[He et al., 2016] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. *CVPR*, pages 770–778.

[Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, 30.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

FG211 Professional Ethics (Mandatory)

1. General information

- **Semester**: 7th Sem. **Credits**: 2
- **Hour of this course**: **Theory**: 1 hours; **Practice**: 2 hours;
- **Prerrequisites**: None

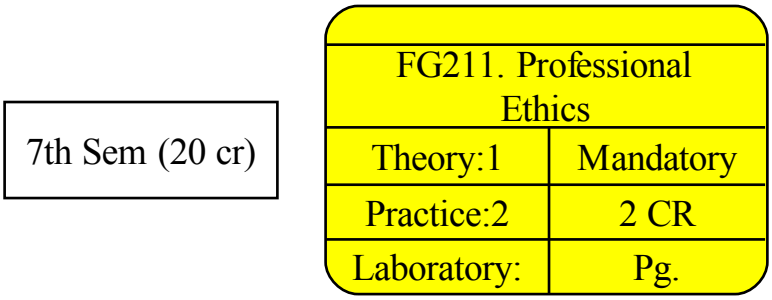

Figure 5.42: Connection Map. FG211 Professional Ethics

2. Course Rationale

This course adapts professional ethical principles to Artificial Intelligence, integrating ACM/IEEE codes with AI-specific challenges. Combines regulatory frameworks (GDPR) with current societal impact cases.

3. Goals of the Course

- Apply ACM/IEEE ethics codes to AI problems.
- Analyze compliance with regulations (GDPR) in autonomous systems.
- Design ethical protocols for AI projects.

4. General Competences (*Outcomes*)

AG-C02) Ethics: Applies ethical principles and commits to professional ethics and the standards of professional computing practice. (Usage)

AG-C04) Communication: Communicates effectively in complex computing activities. (Usage)

5. Content

5.1 Ethical Foundations in AI, 10 hr(s)

Competences: AG-C02,AG-C04

Topics: I: Comparison: ACM vs IEEE ethics codes for AI. **II:** Algorithmic transparency and fairness principles.

Learning Outcomes I: Explain differences between ethical frameworks [Familiarity]. **II:** Evaluate compliance in real cases [Assessment].

Bibliography: [ACM, 2018, IEEE, 2020]

5.2 Privacy and Regulations, 14 hr(s)

Competences: AG-C02,AG-C04

Topics: I: GDPR applied to ML: right to explanation. **II:** Ethical auditing of datasets (e.g., ImageNet).

Learning Outcomes I: Implement GDPR checklists for models [Usage]. **II:** Identify violations in case studies [Assessment].

Bibliography: [Voigt and von dem Bussche, 2018, Müller, 2021]

5.3 Bias and Fairness, 18 hr(s)

Competences: AG-C02,AG-C04

Topics: I: Fairness metrics (demographic parity, equality of opportunity). **II:** Tools: IBM Fairness 360, Google What-If.

Learning Outcomes I: Measure biases in models using Python [Usage]. **II:** Propose mitigation strategies [Assessment].

Bibliography: [Force, 2020, Mehrabi et al., 2021]

5.4 Accountability in Autonomous Systems, 18 hr(s)

Competences: AG-C02,AG-C04

Topics: I: Legal attribution in AI failures (e.g., self-driving cars). **II:** ISO standards for trustworthy AI.

Learning Outcomes I: Draft liability clauses [Assessment]. **II:** Analyze legal conflicts [Familiarity].

Bibliography: [Bostrom, 2014, Committee, 2019]

6. Bibliography

[ACM, 2018] ACM (2018). Acm code of ethics and professional conduct. Technical report, ACM.

[Bostrom, 2014] Bostrom, N. (2014). *Superintelligence: Paths, Dangers, Strategies*. Oxford University Press.

[Committee, 2019] Committee, I. E. (2019). Ethical dilemmas in engineering. Technical report, IEEE.

[Force, 2020] Force, A. E. T. (2020). Case studies in computing and society. Technical report, ACM.

[IEEE, 2020] IEEE (2020). Ieee code of ethics. Technical report, IEEE.

[Mehrabi et al., 2021] Mehrabi, N., Morstatter, F., Saxena, N., Lerman, K., and Galstyan, A. (2021). A survey on bias and fairness in machine learning. *ACM Computing Surveys*, 54(6).

[Müller, 2021] Müller, V. C. (2021). *Ethics of Artificial Intelligence and Robotics*. Cambridge University Press.

[Voigt and von dem Bussche, 2018] Voigt, P. and von dem Bussche, A. (2018). *The EU General Data Protection Regulation (GDPR): A Practical Guide*. Springer.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

FG350 Leadership and Performance (Mandatory)

1. General information

- **Semester**: 7th Sem. **Credits**: 2
- **Hour of this course**: **Theory**: 2 hours;
- **Prerrequisites**:
  - FG106 Theater (5th Sem-Pg. 370)

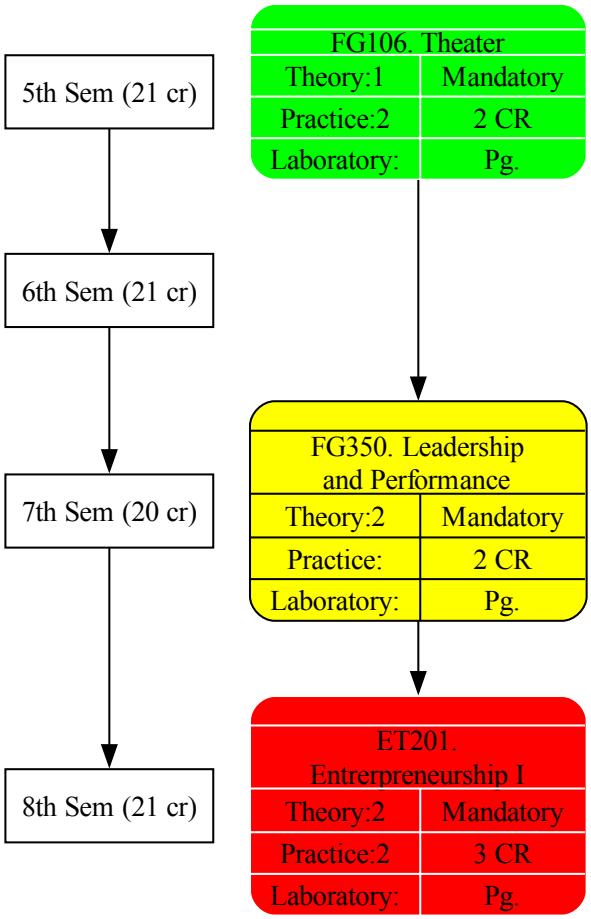

Figure 5.43: Connection Map. FG350 Leadership and Performance

2. Course Rationale

Contemporary organizations demand from their members the effective exercise of leadership: to assume with effectiveness and a spirit of service the challenges entrusted to them, in the pursuit of a more just and reconciled society. This challenge involves forming people with a sound knowledge of themselves, with the ability to judge reality objectively and to propose guidelines that positively modify the environment. The course combines theoretical foundations with the experiential development of competencies, training the student to lead teams with integrity, personal coherence, and genuine service to others.

3. Goals of the Course

- Understand the theoretical foundations of leadership and its ethical dimension, integrating them into a process of self-knowledge.
- Identify and develop the personal competencies and virtues necessary for the effective exercise of leadership.
- Acquire practical skills for the management, motivation, and direction of high-performance teams.
- Contribute to the construction of a more just society through leadership with a sense of service and social responsibility.

4. General Competences (*Outcomes*)

AG-C02) Ethics: Applies ethical principles and commits to professional ethics and the standards of professional computing practice. (Usage)

AG-C04) Communication: Communicates effectively in complex computing activities. (Usage)

5. Content

5.1 Foundations of Leadership, 15 hr(s)

Competences: AG-C02,AG-C04

Topics: **I:** Definition and concept of leadership. **II:** Main leadership theories: transactional, transformational, and situational. **III:** Integral vision of the human being: bodily, affective, and spiritual dimensions. **IV:** Motives of human action: extrinsic, intrinsic, and transcendent. **V:** The practice of virtue in the exercise of leadership. **VI:** The leader as a model: ethics and personal coherence. **VII:** Leadership styles and their application in different contexts.

Learning Outcomes **I:** Analyze and understand the theoretical bases of the exercise of leadership [Familiarity]. **II:** Assume the correct attitude to put leadership into practice [Familiarity]. **III:** Initiate a process of self-knowledge aimed at discovering leadership traits in oneself [Familiarity]. **IV:** Recognize the ethical dimension of leadership and its impact on the personal and social environment [Familiarity].

Bibliography: [Cardona and Lombardi, 2002, Ferreiro and Alcázar, 2009, Dianine-Havard, 2007, D'Souza, 1994]

5.2 Formation of Virtues and Competencies, 15 hr(s)

Competences: AG-C02,AG-C04

Topics: **I:** Theory of competencies and its application to leadership. **II:** Recognition and inventory of personal competencies. **III:** Personal development plan: diagnosis, goals, and follow-up. **IV:** Mental models and their influence on behavior. **V:** Needs and emotional profiles. **VI:** Motivational vices: identification and overcoming. **VII:** Emotional intelligence applied to leadership. **VIII:** Decision-making under pressure and ambiguity.

Learning Outcomes **I:** Know and develop leadership competencies focused on effectiveness and service [Familiarity]. **II:** Recognize the personal and group tendencies necessary for the exercise of leadership [Familiarity]. **III:** Prepare a personal development plan for leadership competencies [Usage].

Bibliography: [Cardona and Wilkinson, 2009, Huete, 2008, Cardona and Lombardi, 2002, Chinchilla and Moragas, 2007, Goleman, 2006]

5.3 Management and Direction of Teams, 18 hr(s)

Competences: AG-C02,AG-C04

Topics: **I:** The personal relationship with the team: trust and communication. **II:** Integral leadership: mission, vision, and shared values. **III:** Delegation and accompaniment: the art of motivating and delegating. **IV:** Coaching and mentoring as development tools. **V:** Conflict resolution and group dynamics. **VI:** The five dysfunctions of a team and how to overcome them. **VII:** Foundations of unity: cohesion and organizational culture. **VIII:** Servant leadership and its impact on the organization.

Learning Outcomes **I:** Develop skills for effective and cohesive teamwork [Familiarity]. **II:** Apply situational leadership techniques to lead teams in different contexts [Usage]. **III:** Design strategies for conflict resolution and improvement of group performance [Usage].

Bibliography: [Goleman, 2006, Cardona and na, 2008, Hersey et al., 2007, Hunsaker, 2001, Hawkins, 2011, Lencioni, 2002]

6. Bibliography

[Cardona and Lombardi, 2002] Cardona, P. and Lombardi, P. G. (2002). *Cómo desarrollar las Competencias de Liderazgo*. PAD, Lima, 3rd edition.

[Cardona and na, 2008] Cardona, P. and na, C. R. P. (2008). *Dirección por misiones: Cómo generar empresas de alto rendimiento*. Deusto.

[Cardona and Wilkinson, 2009] Cardona, P. and Wilkinson, H. (2009). *Creciendo como Líder*. Ediciones Universidad de Navarra EUNSA.

[Chinchilla and Moragas, 2007] Chinchilla, N. and Moragas, M. (2007). *Dueños de Nuestro Destino*. Editorial Ariel.

[Dianine-Havard, 2007] Dianine-Havard, A. (2007). *Virtuous Leadership: An Agenda for Personal Excellence*. Scepter Publishers.

[D'Souza, 1994] D'Souza, A. (1994). *Discover Your Leadership Style*. St. Pauls.

[Ferreiro and Alcázar, 2009] Ferreiro, P. and Alcázar, M. (2009). *Gobierno de Personas en la Empresa*. Ediciones Universidad de Navarra EUNSA.

[Goleman, 2006] Goleman, D. (2006). *Emotional Intelligence: Why It Can Matter More Than IQ*. Bantam Books.

[Hawkins, 2011] Hawkins, P. (2011). *Leadership Team Coaching: Developing Collective Transformational Leadership*. Kogan Page.

[Hersey et al., 2007] Hersey, P., Blanchard, K. H., and Johnson, D. E. (2007). *Management of Organizational Behavior: Leading Human Resources*. Prentice Hall, 9th edition.

[Huete, 2008] Huete, L. M. (2008). *Construye tu Sueño*. LID Editorial Empresarial.

[Hunsaker, 2001] Hunsaker, P. (2001). *Training in Management Skills*. Pearson Prentice Hall.

[Lencioni, 2002] Lencioni, P. (2002). *The Five Dysfunctions of a Team: A Leadership Fable*. Jossey-Bass.

Eighth Semester

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

CS281 Computing in Society (Mandatory)

1. General information

- **Semester**: 8th Sem. **Credits**: 2
- **Hour of this course**: **Theory**: 2 hours;
- **Prerrequisites**: None

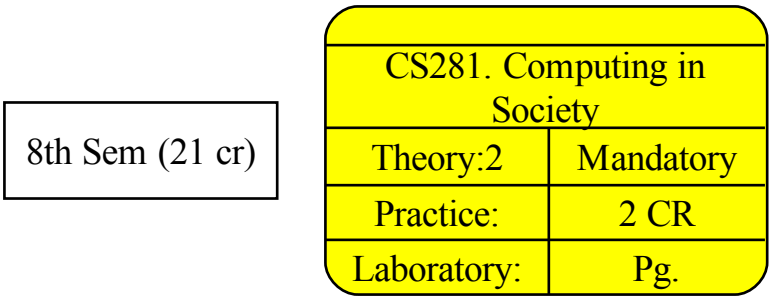

Figure 5.44: Connection Map. CS281 Computing in Society

2. Course Rationale

This course explores the multifaceted relationship between computing technologies and society. It provides a historical perspective on the evolution of the field while equipping students with ethical analysis frameworks to navigate contemporary challenges. Topics include intellectual property, privacy, professional responsibility, sustainability, and the socio-economic impact of automation. Students will develop the communication skills and ethical sensitivity required for professional practice in a diverse and globalized environment.

3. Goals of the Course

- Apply ethical analysis to professional dilemmas in computing.
- Understand the legal frameworks for Intellectual Property and Privacy.
- Evaluate the environmental sustainability and social impact of IT.
- Communicate effectively regarding professional and ethical issues.

4. General Competences (*Outcomes*)

AG-C01) The Professional and the World: Analyzes and evaluates the impact of solutions to complex computing problems on the sustainable development of society. (Usage)

AG-C02) Ethics: Applies ethical principles and commits to professional ethics and the standards of professional computing practice. (Usage)

5. Content

5.1 Foundations and Early History of Computing, 2 hr(s)

Competences: AG-C01,AG-C02

Topics: **I:** The history of computing: hardware, software, and human/organizational. **II:** The role of history in the present including within different social contexts, and the relevance of this history on the future. **III:** Age I (Pre-digital): Ancient analog computing (Stonehenge, Antikythera mechanism, Salisbury Cathedral clock, etc.), human-calculated number tables, Euclid, Lovelace, Babbage, Gödel, Church, Turing, pre-electronic (electro-mechanical and mechanical) hardware. **IV:** Age II (Early modern computing): ENIAC, UNIVAC, Bombes (Bletchley Park and codebreakers), computer companies (e.g., IBM), mainframes, etc.

Learning Outcomes **I:** Understand the relevance and impact of computing history on recent events, present context, and possible future outcomes, from more than one cultural perspective [Assess] **II:** Discuss how perspectives held today have been shaped by history, and that alternative perspectives exist (e.g., fears of AI replacing human workers vs AI augmenting human work, various views on Moore's Law) [Debate] **III:** Identify formative and consequential trends in the history of the computing field [Analyze] **IV:** Identify the contributions of pioneering individuals or organizations (research labs, computer companies, government offices) in the computing field [Analyze]

Bibliography: [Quinn, 2019, Baase and Henry, 2017]

5.2 Social Impact and AI, 2 hr(s)

Competences: AG-C01,AG-C02

Topics: **I:** Social implications (e.g., political and cultural ideologies) in a hyper-networked world where the capabilities and impact of social media, artificial intelligence, and computing in general are rapidly evolving. **II:** Impact of computing applications (e.g., social media, artificial intelligence applications) on individual well-being, and safety of all kinds (e.g., physical, emotional, economic). **III:** Consequences of involving computing technologies, particularly artificial intelligence, biometric technologies, and algorithmic decision-making systems, in civic life (e.g., facial recognition technology, biometric tags, resource distribution algorithms, policing software) and how human agency and oversight is crucial. **IV:** Growth and control of the internet, data, computing, and artificial intelligence.

Learning Outcomes **I:** Describe the different ways that computer technology (networks, mobile computing, artificial intelligence) mediates social interaction at the personal and collective levels [Describe] **II:** Interpret the social context of a given design and its implementation [Interpret] **III:** Analyze the efficacy of a given design and implementation using empirical data [Analyze] **IV:** Understand the implications of technology use (e.g., social media) for different identities, cultures, and communities [Explain] **V:** Describe the internet's role in facilitating communication between citizens, governments, and each other [Describe] **VI:** Analyze the effects of reliance on computing in the implementation of democracy (e.g., delivery of social services, electronic voting) [Analyze] **VII:** Describe the impact of a lack of appropriate representation of people from historically minoritized populations in the computing profession (e.g., industry culture, product diversity) [Describe]

Bibliography: [Quinn, 2019]

5.3 Ethical Foundations and Frameworks, 4 hr(s)

Competences: AG-C01,AG-C02

Topics: **I:** Avoiding fallacies and misrepresentation in argumentation. **II:** Ethical theories and decision-making (philosophical and social frameworks, e.g. [1]). **III:** Recognition of the role culture plays in our understanding, adoption, design, and use of computing technology.

IV: Why ethics is important in computing, and how ethics is similar to, and different from, laws and social norms. **V:** Introduction to ethical frameworks (e.g., consequentialism such as utilitarianism, non-consequentialism such as duty, rights, or justice, agent-centered such as virtue or feminism, contractarianism, ethics of care) and their use for analyzing an ethical dilemma.

Learning Outcomes **I:** Describe how a given cultural context impacts decision making [Describe] **II:** Express the use of example and analogy in ethical argument [Analyze] **III:** Analyze (and avoid) basic logical fallacies in an argument [Analyze] **IV:** Analyze an argument to identify premises and conclusion [Analyze] **V:** Evaluate how and why ethics is so important in computing and how it relates to cultural norms, values, and law [Evaluate] **VI:** Justify a decision made on ethical grounds [Justify] **VII:** Discuss the advantages and disadvantages of traditional ethical frameworks [Debate] **VIII:** Analyze ethical dilemmas related to the creation and use of technology from multiple perspectives using ethical frameworks [Analyze]

Bibliography: [Quinn, 2019, Baase and Henry, 2017]

5.4 Professional Values, Nature and Continuing Development, 4 hr(s)

Competences: AG-C01,AG-C02

Topics: **I:** Community values and the laws by which we live. **II:** The nature of being a professional including care, attention, discipline, fiduciary responsibility, and mentoring. **III:** Keeping up to date as a computing professional in terms of familiarity, tools, skills, legal and professional frameworks as well as the ability and responsibility to self-assess and progress in the computing field.

Learning Outcomes **I:** Identify ethical issues that arise in software design, development practices, and software deployment [Analyze] **II:** Discuss how to address ethical issues in specific situations [Debate] **III:** Describe the mechanisms that typically exist for a professional to keep up to date in ethical matters [Describe]

Bibliography: [Baase and Henry, 2017]

5.5 Intellectual Property Foundations, 4 hr(s)

Competences: AG-C01,AG-C02

Topics: **I:** Intellectual property rights. **II:** Legal foundations for intellectual property protection. **III:** Philosophical foundations of intellectual property. **IV:** Forms of intellectual property (e.g., copyrights, patents, trade secrets, trademarks) and the rights they protect.

Learning Outcomes **I:** Defend legal and ethical uses of copyrighted materials [Defend] **II:** Discuss the philosophical bases of intellectual property in an appropriate context (e.g., country) [Debate] **III:** Contrast the protections and obligations of copyright, patent, trade secret, and trademarks [Contrast] **IV:** Describe the rationale for the legal protection of intellectual property in the appropriate context (e.g., country) [Describe]

Bibliography: [Quinn, 2019]

5.6 Digital IP Rights and Copyright, 6 hr(s)

Competences:

Topics: **I:** Intangible digital intellectual property (IDIP). **II:** Limitations on copyright protections, including fair use and the first sale doctrine. **III:** Intellectual property laws and treaties that impact the enforcement of copyrights.

Learning Outcomes **I:** Describe and critique legislation and precedent aimed at digital copyright infringements [Describe] **II:** Identify contemporary examples of intangible digital intellectual property [Analyze] **III:** Discuss whether a use of copyrighted material is likely to be fair use [Debate] **IV:** Analyze the use of copyrighted work under the concepts of fair use and the first sale doctrine [Analyze] **V:** Discuss the nature of anti-circumvention laws in the context of copyright protection [Debate]

Bibliography: [Tavani, 2013]

5.7 Data, Surveillance and Privacy Technology, 4 hr(s)

Competences: AG-C01,AG-C02

Topics: **I:** Privacy implications of widespread data collection including but not limited to transactional databases, data warehouses, surveillance systems, cloud computing, and artificial intelligence. **II:** Conceptions of anonymity, pseudonymity, and identity. **III:** Technology-based solutions for privacy protection (e.g., end-to-end encryption and differential privacy).

Learning Outcomes **I:** Evaluate solutions to privacy threats in transactional databases and data warehouses [Evaluate] **II:** Describe the role of data collection in the implementation of pervasive surveillance systems (e.g., RFID, face recognition, toll collection, mobile computing) [Describe] **III:** Distinguish the concepts and goals of anonymity and pseudonymity [Distinguish] **IV:** Describe the ramifications of technology-based privacy protections, including differential privacy and end-to-end encryption [Describe]

Bibliography: [Baase and Henry, 2017]

5.8 Environmental Footprint of Computing Systems, 2 hr(s)

Competences: AG-C01,AG-C02

Topics: **I:** Environmental, social, and cultural impacts of implementation decisions (e.g., sustainability goals, algorithmic bias/outcomes, economic viability, and resource consumption). **II:** Local/regional/global social and environmental impacts of computing systems and their use (e.g., carbon footprints, resource usage, e-waste) due to hardware (e.g., e-waste, data centers, rare element and resource utilization, recycling) and software (e.g., cloud-based services, blockchain, AI model training and use). This includes everyday use of hardware (cheap hardware replaced frequently) and software (web-browsing, email, and other services with hidden/remote computational demands).

Learning Outcomes **I:** Assess the environmental impacts of a given project's deployment (e.g., energy consumption, contribution to e-waste, impact of manufacturing) [Assess] **II:** Describe global social and environmental impacts of computer use and disposal [Describe] **III:** Describe the environmental impacts of design choices within the field of computing that relate to algorithm design, operating system design, networking design, database design, etc [Describe] **IV:** Assess computing applications in respect to environmental issues (e.g., energy, pollution, resource usage, recycling and reuse, food management and production) [Assess]

Bibliography: [Quinn, 2019]

5.9 Economic Models and Strategies, 2 hr(s)

Competences: AG-C01,AG-C02

Topics: **I:** Economic models: regulated and unregulated, monopolies, network effects, and open market; knowledge and attention economies. **II:** Pricing and deployment strategies: planned obsolescence, subscriptions, freemium, software licensing, open-source, free software, adware. **III:** Impacts of differences in access to computing resources, and the effect of skilled labor supply and demand on the quality of computing products.

Learning Outcomes I: Summarize the social effects of economic models (e.g., the knowledge and attention economies) [Summarize] **II:** Describe the differences and similarities of competing strategies (e.g., subscription vs freemium vs free) [Describe] **III:** Discuss examples of digital divides [Debate]

Bibliography: [Baase and Henry, 2017]

5.10 Computer Crimes and Attacks, 2 hr(s)

Competences: AG-C01,AG-C02

Topics: I: Computer crimes, legal redress for computer criminals and impact on victims and society. **II:** Social engineering, computing-enabled fraud, identity theft and recovery from these. **III:** Cyber terrorism, criminal hacking, and hacktivism. **IV:** Malware, viruses, worms. **V:** Attacks on critical infrastructure such as electrical grids and pipelines.

Learning Outcomes I: List classic examples of computer crimes and social engineering incidents with societal impact [List] **II:** Identify issues with laws that apply to computer crimes [Analyze] **III:** Describe the motivation and ramifications of cyber terrorism, data theft, hacktivism, ransomware, and other attacks [Describe] **IV:** Examine the ethical and legal issues surrounding the misuse of access and various breaches of security [Examine]

Bibliography: [Baase and Henry, 2017]

5.11 Identity and Diversity in Computing, 2 hr(s)

Competences: AG-C01,AG-C02

Topics: I: How identity impacts and is impacted by computing technologies and environments (academic and professional). **II:** The benefits of diverse development teams and the impacts of teams that are not diverse.

Learning Outcomes I: Define and distinguish equity, equality, diversity, inclusion, and accessibility [Define] **II:** Identify examples of the benefits that diverse teams can bring to software products, and how a lack of diversity has costs [Analyze] **III:** Assess the impact of power and privilege in the computing profession as it relates to culture, industry, products, and society [Assess]

Bibliography: [Quinn, 2019]

5.12 Technical and Professional Communication, 2 hr(s)

Competences: AG-C01,AG-C02

Topics: I: Oral, written, and electronic team and group communication. **II:** Technical communication materials (e.g., source code, and documentation, tutorials, reference materials, API documentation). **III:** Communicating with different stakeholders such as customers, leadership, or the public. **IV:** Tradeoffs in competing factors that affect communication channels and choices.

Learning Outcomes I: Understand the importance of writing concise and accurate technical documents following well-defined standards for format and for including appropriate tables, figures, and references [Produce] **II:** Analyze written technical documentation for technical accuracy, concision, lack of ambiguity, and awareness of audience [Analyze] **III:** Compose and deliver an audience-aware, accessible, and organized formal presentation [Compose] **IV:** Identify and describe qualities of effective communication (e.g., virtual, face-to-face, intragroup, shared documents) [Analyze] **V:** Choose an appropriate way to communicate delicate ethical concerns [Assess]

Bibliography: [Quinn, 2019]

6. Bibliography

[Baase and Henry, 2017] Baase, S. and Henry, T. (2017). *A Gift of Fire: Social, Legal, and Ethical Issues for Computing Technology*. Pearson, 5th edition.

[Quinn, 2019] Quinn, M. J. (2019). *Ethics for the Information Age*. Pearson, 8th edition.

[Tavani, 2013] Tavani, H. T. (2013). *Ethics and Technology: Controversies, Questions, and Strategies for Ethical Computing*. Wiley, Hoboken, NJ, 4th edition.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

CS3I1 Computer Security (Mandatory)

1. General information

- **Semester**: 8th Sem. **Credits**: 3
- **Hour of this course**: **Theory**: 1 hours; **Practice**: 2 hours; **Laboratory**: 2 hours;
- **Prerrequisites**:
  - CS231 Networking and Communication (6th Sem-Pg. 378)

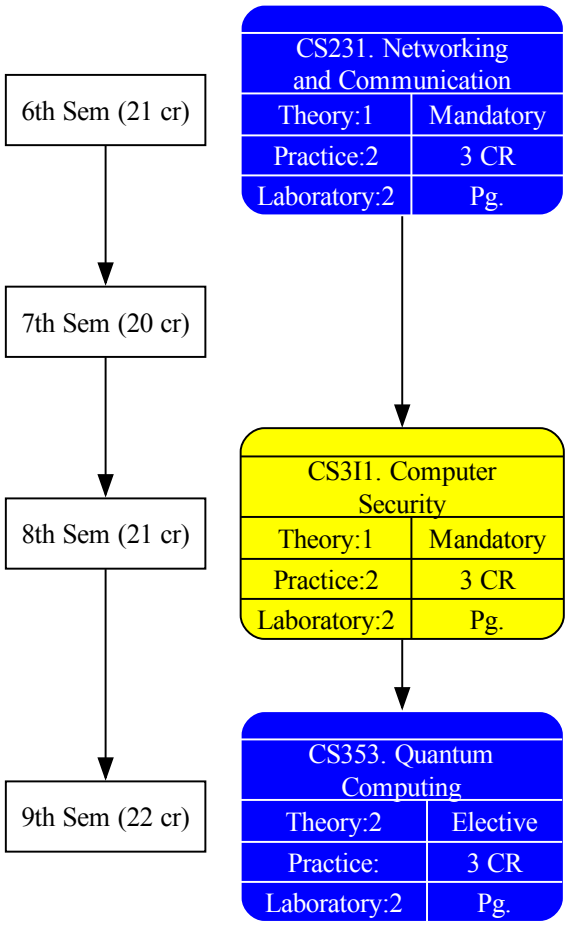

Figure 5.45: Connection Map. CS3I1 Computer Security

2. Course Rationale

In an increasingly digital world, security is no longer an optional feature but a fundamental requirement. This course provides a comprehensive introduction to the principles and practices of cybersecurity, covering everything from theoretical foundations to secure coding and digital forensics. Students will gain the skills necessary to design, implement, and govern secure systems while understanding the ethical and professional responsibilities involved in protecting information and infrastructure.

3. Goals of the Course

- Understand the fundamental principles and mathematical bases of security.
- Apply secure coding techniques and cryptographic methods to software.
- Analyze system vulnerabilities and perform forensic investigations.
- Evaluate security governance and professional ethics in cybersecurity.

4. General Competences (*Outcomes*)

AG-C09) Design and Development of Solutions: Designs, implements, and evaluates solutions for complex computing problems. (Usage)

AG-C11) Use of Tools: Applies modern computing tools in problem solving. (Usage)

5. Content

5.1 Security Mindset and Principles, 6 hr(s)

Competences: AG-C09,AG-C11

Topics: **I:** Developing a security mindset incorporating crosscutting concepts: confidentiality, integrity, availability, risk assessment, systems thinking, adversarial thinking, human-centered thinking. **II:** Vulnerabilities, threats, attack surfaces, and attack vectors Protection and Safety. **III:** Denial of Service (DoS) and Distributed Denial of Service (DDoS) Protection and Safety. **IV:** Principles and practices of protection, e.g., least privilege, open design, fail-safe defaults, defense in depth, and zero trust; and how they can be implemented Principles of Operating System, Protection and Safety, Coding Practices, Large-Scale Construction and Process, Computer Crimes and Attacks, Security Laws, Policies and Responsibility. **V:** Optimization considerations between security, privacy, performance, and other design goals System Design, Software Development Practices, Performance Testing and Benchmarking, Testing Tools and Analysis, Verification and validation Concepts, Test Planning and Types, Advanced Testing Practices. **VI:** Concepts of trust and trustworthiness.

Learning Outcomes **I:** Evaluate a system for possible attacks that can be launched by an adversary [Evaluate] **II:** Design and develop approaches to protect a system from a set of identified threats [Design] **III:** Describe how harm to user privacy can be avoided [Describe] **IV:** Develop a system that incorporates various principles of security and privacy [Create]

Bibliography: [Pfleeger et al., 2015, Stallings, 2017]

5.2 Cryptography Fundamentals, 8 hr(s)

Competences: AG-C09,AG-C11

Topics: **I:** Differences between algorithmic, applied, and mathematical views of cryptography. **II:** Mathematical preliminaries: modular arithmetic, Euclidean algorithm, probabilistic independence, linear algebra basics, number theory, finite fields, complexity, asymptotic analysis Discrete Mathematics, Linear Algebra. **III:** Basic cryptography: symmetric key and public key cryptography Foundational Data Structures, Foundational Algorithms, Advanced Algorithms, Discrete Mathematics. **IV:** Classical cryptosystems, such as shift, substitution, transposition ciphers, code books, and machines Discrete Mathematics. **V:** Kerckhoff's principle and use of vetted libraries Coding Practices, Large-Scale Construction and Process.

Learning Outcomes **I:** Explain the role of cryptography in supporting security and privacy [Explain] **II:** Discuss the risks of inventing one's own cryptographic methods [Debate] **III:** Discuss the importance of prime numbers in cryptography and explain their use in cryptographic algorithms [Debate] **IV:** Implement and cryptanalyze classical ciphers [Implement]

Bibliography: [Stallings, 2017]

5.3 Injection and Input Validation, 8 hr(s)

Competences: AG-C09,AG-C11

Topics: **I:** Common vulnerabilities and weaknesses. **II:** SQL injection and other injection attacks. **III:** Cross-site scripting techniques and mitigations. **IV:** Input validation and data sanitization Protection and Safety, Basic Programming Concepts and Constructs, Programming Practices: Testing, Debugging, and Documentation, Performance Testing and Benchmarking, Testing Tools and Analysis, Verification and validation Concepts, Test Planning and Types, Advanced Testing Practices.

Learning Outcomes **I:** Identify underlying problems in given examples of an enumeration of common weaknesses and explain how they can be circumvented [Analyze] **II:** Apply input validation and data sanitization techniques to enhance security of a program [Apply] **III:** Evaluate and prevent SQL injections attacks on a database application [Evaluate] **IV:** Evaluate and prevent cross-site scripting attacks against a website [Evaluate] **V:** Construct a program that tests for all input handling errors [Create]

Bibliography: [Pfleeger et al., 2015]

5.4 Malware Analysis and Advanced Security, 6 hr(s)

Competences: AG-C09

Topics: **I:** Offensive security techniques as a defense. **II:** AI-assisted malware detection techniques. **III:** Ransomware: creation, prevention, and mitigation. **IV:** Secure use of third-party components Coding Practices, Large-Scale Construction and Process, Performance Testing and Benchmarking, Testing Tools and Analysis, Verification and validation Concepts, Test Planning and Types, Advanced Testing Practices. **V:** Malware: varieties, creation, reverse engineering, and defense against them Language Translation and Execution, Systems Execution and Memory Model. **VI:** Assurance: testing (including fuzzing and penetration testing), verification, and validation Protection and Safety, Basic Programming Concepts and Constructs, Programming Practices: Testing, Debugging, and Documentation, Coding Practices, Large-Scale Construction and Process, Performance Testing and Benchmarking, Testing Tools and Analysis, Verification and validation Concepts, Test Planning and Types, Advanced Testing Practices. **VII:** Static and dynamic analyses Program Analysis and Analyzers, Protection and Safety, Evaluation, Performance Testing and Benchmarking, Testing Tools and Analysis, Verification and validation Concepts, Test Planning and Types, Advanced Testing Practices. **VIII:** Secure compilers and secure code generation Language Translation and Execution, Run-time Behavior and Systems.

Learning Outcomes **I:** Describe different kinds of malicious software [Describe] **II:** Explain the risks of misusing interfaces with third-party code and how to correctly use third-party code [Explain] **III:** Discuss the need to update software to fix security vulnerabilities and the lifecycle management of the fix [Debate] **IV:** Apply static and dynamic tools to identify programming faults [Apply] **V:** Evaluate a system for the existence of malware and remove it [Evaluate] **VI:** Implement preventive techniques to reduce the occurrence of ransomware [Implement]

Bibliography: [Sikorski and Honig, 2012, Eilam, 2005, National Institute of Standards and Technology, 2020]

5.5 Threat Analysis and Security Engineering, 8 hr(s)

Competences: AG-C09,AG-C11

Topics: **I:** Security engineering goals: building systems that remain dependable despite errors, accidents, or malicious adversaries Coding Practices, Large-Scale Construction and Process, Performance Testing and Benchmarking, Testing Tools and Analysis, Verification and validation Concepts, Test Planning and Types, Advanced Testing Practices, Computer Crimes and Attacks, Security Laws, Policies and Responsibility. **II:** Problem analysis and situational analysis to address system security Performance Testing and Benchmarking, Testing Tools and Analysis,

Verification and validation Concepts, Test Planning and Types, Advanced Testing Practices. **III:** Security analysis, covering security requirements analysis; security controls analysis; threat analysis; and vulnerability analysis Program Analysis and Analyzers, Evaluation. **IV:** Security attack domains and attack surfaces, e.g., communications and networking, hardware, physical, social engineering, software, and supply chain Network Security. **V:** Security attack modes, techniques, and tactics, e.g., authentication abuse; brute force; buffer manipulation; code injection; content insertion; denial of service; eavesdropping; function bypass; impersonation; integrity attack; interception; phishing; protocol analysis; privilege abuse; spoofing; and traffic injection Network Security, Protection and Safety, Performance Testing and Benchmarking, Testing Tools and Analysis, Verification and validation Concepts, Test Planning and Types, Advanced Testing Practices.

Learning Outcomes I: Create a threat model for a system or system design [Create] **II:** Apply situational analysis to develop secure solutions under a specified scenario [Apply] **III:** Identify and mitigate security vulnerabilities and weaknesses in a system [Analyze]

Bibliography: [Anderson, 2020]

5.6 Digital Forensics Techniques, 6 hr(s)

Competences: AG-C09,AG-C11

Topics: I: Basic principles and methodologies for digital forensics. **II:** System design for forensics. **III:** Forensics in different situations: operating systems, file systems, application forensics, web forensics, network forensics, mobile device forensics, use of database auditing Network Security. **IV:** Attacks on forensics and preventing such attacks.

Learning Outcomes I: Explain what a digital investigation is and how it can be implemented Computer Crimes and Attacks, Security Laws, Policies and Responsibility [Explain] **II:** Design and implement software to support forensics [Design] **III:** Extract data from a hard drive to comply with the law Computer Crimes and Attacks, Security Laws, Policies and Responsibility [Extract] **IV:** Recover data based on a given search term from an imaged system [Analyze] **V:** Reconstruct data and events from an application history, or a web artifact, or a cloud database, or a mobile device. Mobile Platforms: Capabilities and Asynchronous Computing, Mobile Platforms: Architecture and Constraints, Mobile Platforms: Development Fundamentals, Cloud Architecture and Data Management, Frontend, Frameworks and Web Standards [Apply] **VI:** Capture and analyze network traffic. Network Security [Analyze] **VII:** Develop approaches to address the challenges associated with mobile device forensics [Create] **VIII:** Apply forensics tools to investigate security breaches [Apply] **IX:** Identify and mitigate anti-forensic methods [Analyze]

Bibliography: [Pfleeger et al., 2015]

5.7 Security Governance, Management and Policy, 6 hr(s)

Competences: AG-C09,AG-C11

Topics: I: Protecting critical assets from threats. **II:** Security governance: organizational objectives and general risk assessment. **III:** Security management: achieve and maintain appropriate levels of confidentiality, integrity, availability, accountability, authenticity, and reliability Performance Testing and Benchmarking, Testing Tools and Analysis, Verification and validation Concepts, Test Planning and Types, Advanced Testing Practices. **IV:** Security policy: organizational policies, issue-specific policies, system-specific policies.

Learning Outcomes I: Describe critical assets and how they can be protected [Describe] **II:** Differentiate between security governance, management, and controls, giving examples of each [Differentiate] **III:** Describe a technical control and implement it to mitigate specific threats [Describe]

Bibliography: [Anderson, 2020]

5.8 Security Ethics, Law and Society, 6 hr(s)

Competences: AG-C09,AG-C11

Topics: **I:** Societal impacts on breakdowns in security and privacy Digital Equity and Accessibility, Social Impact and AI, Civil Liberties and Expression, Data, Surveillance and Privacy Technology, Privacy Philosophical Foundations and Legislation, Computer Crimes and Attacks, Security Laws, Policies and Responsibility. **II:** Applicability of laws and regulations on security and privacy Computer Crimes and Attacks, Security Laws, Policies and Responsibility. **III:** Professional ethical considerations when designing secure systems and maintaining privacy; ethical hacking Civil Liberties and Expression, Data, Surveillance and Privacy Technology, Privacy Philosophical Foundations and Legislation, Ethics Codes and Legal Accountability, Professional Responsibility and Public Policy, Professional Values, Nature and Continuing Development, Ethical Dissent and Cultural Context, Workplace Ethics, Computer Crimes and Attacks, Security Laws, Policies and Responsibility. **IV:** Security and privacy implications of malicious AI/machine learning actors, e.g., identifying deep fakes Machine Learning Formulation and Evaluation, Deep Learning, RL and Ethics, Introduction to AI: Concepts and Intelligent Behavior, Problem Characteristics and Agent Nature, Machine Learning Fundamentals, Machine Learning Data, Representations and Evaluation, Neural Networks and Ethics, Civil Liberties and Expression, Data, Surveillance and Privacy Technology, Privacy Philosophical Foundations and Legislation, Computer Crimes and Attacks, Security Laws, Policies and Responsibility. **V:** Societal impacts of Internet of Things (IoT) devices and other emerging technologies on security and privacy Civil Liberties and Expression, Data, Surveillance and Privacy Technology, Privacy Philosophical Foundations and Legislation, Computer Crimes and Attacks, Security Laws, Policies and Responsibility.

Learning Outcomes **I:** Calculate the impact of a breakdown in security of a given system [Calculate] **II:** Construct a system that conforms to security laws [Create] **III:** Apply a set of privacy regulations to design a system that protects privacy [Apply] **IV:** Evaluate the legal ramifications of a system not corresponding to applicable laws and regulations [Evaluate]

Bibliography: [Anderson, 2020, Pfleeger et al., 2015]

6. Bibliography

[Anderson, 2020] Anderson, R. (2020). *Security Engineering: A Guide to Building Dependable Distributed Systems*. Wiley, 3rd edition.

[Eilam, 2005] Eilam, E. (2005). *Reversing: Secrets of Reverse Engineering*. Wiley, Indianapolis, IN.

[National Institute of Standards and Technology, 2020] National Institute of Standards and Technology (2020). Security and privacy controls for information systems and organizations. Technical Report NIST SP 800-53 Rev. 5, National Institute of Standards and Technology, Gaithersburg, MD. Disponible en: `https://csrc.nist.gov/publications/detail/sp/800-53/rev-5/final`.

[Pfleeger et al., 2015] Pfleeger, C. P., Pfleeger, S. L., and Margulies, J. (2015). *Security in Computing*. Prentice Hall, 5th edition.

[Sikorski and Honig, 2012] Sikorski, M. and Honig, A. (2012). *Practical Malware Analysis: The Hands-On Guide to Dissecting Malicious Software*. No Starch Press, San Francisco, CA.

[Stallings, 2017] Stallings, W. (2017). *Computer Security: Principles and Practice*. Pearson, 4th edition.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

CS3P1 Parallel and Distributed Computing (Mandatory)

1. General information

- **Semester**: 8th Sem. **Credits**: 4
- **Hour of this course**: **Theory**: 2 hours; **Practice**: 2 hours; **Laboratory**: 2 hours;
- **Prerrequisites**:
  - CS212 Analysis and Design of Algorithms (5th Sem-Pg. 344)
  - CS231 Networking and Communication (6th Sem-Pg. 378)

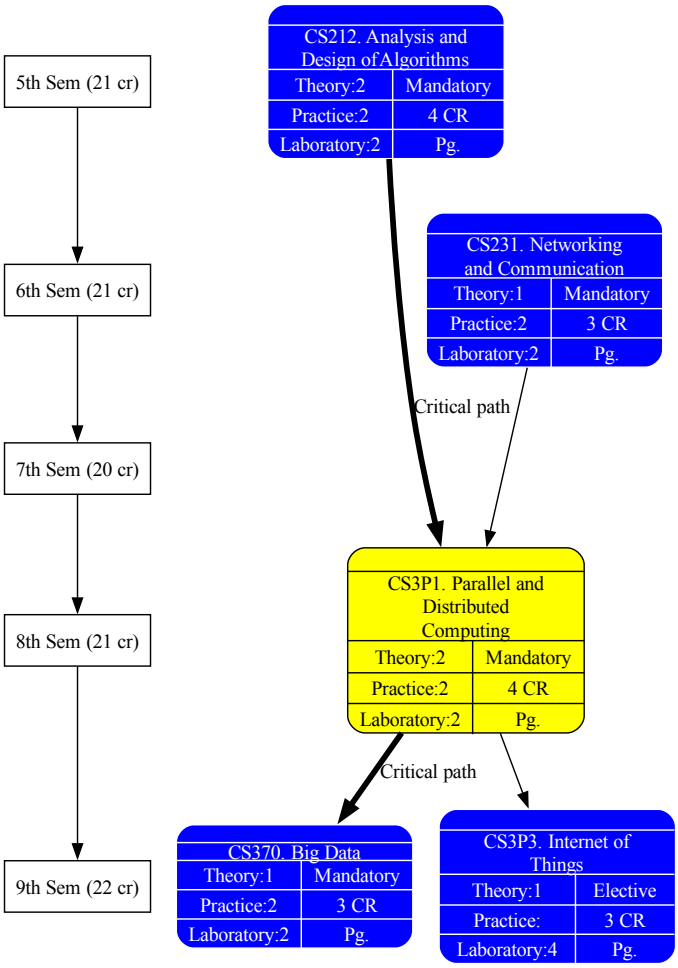

Figure 5.46: Connection Map. CS3P1 Parallel and Distributed Computing

2. Course Rationale

With the end of frequency scaling in single-core processors, parallel and distributed computing has become essential for high-performance software development. This course introduces students to the programming models, communication patterns, and coordination mechanisms required to exploit multi-core systems and distributed clusters. Students will learn to design scalable algorithms and evaluate their performance under different computational constraints.

3. Goals of the Course

- Design and implement parallel programs using shared and distributed memory.
- Master communication and synchronization protocols in distributed systems.
- Evaluate performance and scalability through formal metrics.
- Apply parallel algorithms to solve complex computational problems.

4. General Competences (*Outcomes*)

AG-C09) Design and Development of Solutions: Designs, implements, and evaluates solutions for complex computing problems. (Usage)

AG-C11) Use of Tools: Applies modern computing tools in problem solving. (Usage)

5. Content

5.1 Programs: Declarative Parallelism, 4 hr(s)

Competences: AG-C09,AG-C11

Topics: I: Parallelism (a) Declarative parallelism: Determining which actions can, or must not, be performed in parallel, at the level of instructions, functions, closures, compound actions, sessions, tasks, and services is the core idea underlying PDC algorithms; failing to do so is the primary source of errors. See also: Algorithms (b) Defining order: for example, using "happens-before" relationships or series-parallel directed acyclic graphs (DAGs) representing programs. (c) Independence: determining when order does not matter, in terms of commutativity, dependencies, preconditions. (d) Guaranteeing order between otherwise parallel actions when necessary, including locks, safe publication; and enforcing communication - sending a message happens before receiving it; and relaxing it when not necessary.

Learning Outcomes I: Graphically represent (as a Directed Acyclic Graph - DAG) how to parallelize a compound numeric expression; for example, $a = (b + c) * (d + e)$ [Design] **II:** Explain why consistency and fault tolerance concepts do not arise in purely sequential programs [Explain]

Bibliography: [Pacheco and Malensek, 2021]

5.2 Programs: Starting Activities, 3 hr(s)

Competences: AG-C09,AG-C11

Topics: I: Starting activities (a) Options allowing actions to be performed (eventually) at locations range from hardwired to configuration scripts; also establishing communication and resource management; these are expressed differently in languages and contexts, typically relying on automated provisioning and management by platforms. See also: Resource Management (b) Procedural: Allowing multiple actions to start at a given program point; for example, starting new threads, possibly delimiting their scope or organizing them into hierarchical groups. (c) Reactive: Enabling upon event occurrence by installing an event handler, with less control over when actions start or finish, and may apply even on single processors. (d) Dependent: Enabling upon completion of others; for example, sequencing sets of parallel actions. See also: Coordination. (e) Granularity: The execution cost of action bodies should outweigh the overhead of organizing them.

Learning Outcomes I: Write a service that creates a thread (or other form of procedural activation) to return a requested web page to each new client [Write]

Bibliography: [Pacheco and Malensek, 2021]

5.3 Programs: Execution Properties, 3 hr(s)

Competences: AG-C09,AG-C11

Topics: I: Execution Properties (a) Non-deterministic execution of actions without order. (b) Consistency: Ensuring agreement among parties on values and predicates when necessary to avoid races, maintain safety and atomicity, or reach consensus. (c) Fault tolerance: Handling failures in parts or communication, including (Byzantine) misbehavior due to unreliable parties and protocols, when necessary to maintain progress or availability. See also: System Reliability. (d) Trade-offs are a focus of evaluation. See also: Evaluation.

Learning Outcomes I: Write a function that counts events, such as network packet receptions, efficiently [Write]

Bibliography: [van Steen and Tanenbaum, 2023]

5.4 Programs: Distribution, 2 hr(s)

Competences: AG-C09,AG-C11

Topics: I: Distribution (a) Defining locations as devices that execute actions, including hardware components, remote hosts; may also include external uncontrolled devices, hosts, and users. See also: Interfacing and Communication (b) A device may time-share or emulate multiple parallel actions using fewer processors through scheduling and virtualization. See also: Scheduling (c) Naming or identifying locations (e.g., device IDs) and actions as parts (e.g., thread IDs). (d) Activities across locations may communicate through media. See also: Communication

Learning Outcomes I: Write a filter/map/reduce program in multiple styles [Write]

Bibliography: [van Steen and Tanenbaum, 2023]

5.5 Programs: Implementation Mappings, 2 hr(s)

Competences: AG-C09,AG-C11

Topics: I: One or more of the following mappings and mechanisms across layered systems: (a) Instruction-level and data parallelism in the CPU. See also: Functional Organization. (b) SIMD and heterogeneous data parallelism. See also: Heterogeneous Architectures. (c) Scheduled concurrency on multi-core, tasks, actors. See also: Scheduling. (d) Clusters, clouds; elastic provisioning. See also: Common Aspects: Platforms, APIs and Constraints, Common Aspects: Platform Languages and Web Programming Patterns. (e) Networked distributed systems. See also: Networked Applications. (f) Emerging technologies such as quantum computing and molecular computing.

Learning Outcomes I: Explain the trade-offs between different mapping strategies in terms of performance, cost, and implementation complexity [Explain]

Bibliography: [Pacheco and Malensek, 2021, mei W. Hwu et al., 2022]

5.6 GPU Programming, 4 hr(s)

Competences: AG-C09

Topics: I: General-Purpose GPU Computing (GPGPU) (a) GPU architecture overview: streaming multiprocessors (SMs), CUDA cores, and warp execution model. (b) SIMT (Single Instruction, Multiple Threads) execution model and its differences from CPU SIMD. (c) Host-device interaction: data transfer between CPU (host) and GPU (device) memory via PCIe. (d) Use cases: scientific computing, machine learning, image processing, and large-scale simulation.
II: CUDA Programming Model (a) CUDA execution hierarchy: grids, thread blocks, and individual threads; mapping to GPU hardware. (b) Kernel definition and launch syntax; configuration parameters (grid and block dimensions). (c) Thread indexing: threadIdx, blockIdx,

blockDim, gridDim; computing global indices in 1D, 2D, and 3D. (d) Warp divergence: performance implications of branching within a warp. (e) Intra-block synchronization primitive: __syncthreads(). **III:** CUDA Memory Hierarchy (a) Global memory: large, high-latency, accessible by all threads; coalesced access patterns for performance. (b) Shared memory: low-latency on-chip memory shared within a thread block; bank conflicts. (c) Registers and local memory: per-thread private storage. (d) Constant and texture memory: read-only caches optimized for specific access patterns. (e) Unified Memory: simplified programming model with automatic host-device data migration.

Learning Outcomes I: Write a CUDA kernel that performs a data-parallel operation (e.g., vector addition, matrix multiplication) and correctly configure grid and block dimensions [Write] **II:** Analyze the impact of global memory access patterns (coalesced vs. strided) and shared memory usage on GPU kernel performance [Analyze] **III:** Design a parallel algorithm using the CUDA programming model, including thread hierarchy, memory allocation strategy, and host-device data transfers [Design]

Bibliography: [Kirk and mei W. Hwu, 2016, Corporation, 2024]

5.7 Communication, 8 hr(s)

Competences: AG-C09,AG-C11

Topics: I: Media (a) Varieties: channels (message passing or I/O), shared memory, heterogeneous, data stores (b) Reliance on the availability and nature of underlying hardware, connectivity, and protocols; language support, emulation. See also: Interfacing and Communication **II:** Channels (a) Explicit (usually named) party-to-party communication media (b) APIs: Sockets, architectural, language-based, and toolkit constructs, such as Message Passing Interface (MPI), and layered constructs such as Remote Procedure Call (RPC). See also: Fundamentals of Networks and Communications (c) I/O channel APIs **III:** Memory (a) Shared memory architectures in which parties directly communicate only with memory at given addresses, with extensions to heterogeneous memory supporting multiple memory stores with explicit data transfer across them; for example, GPU local and shared memory, Direct Memory Access (DMA) (b) Memory hierarchies: Multiple layers of sharing domains, scopes, and caches; locality: latency, false-sharing (c) Consistency properties: Bitwise atomicity limits, coherence, local ordering **IV:** Data Stores (a) Cooperatively maintained data structures implementing maps and related ADTs (b) Varieties: Owned, shared, sharded, replicated, immutable, versioned

Learning Outcomes I: Explain the similarities and differences among: (1) Party A sends a message on channel X with contents 1 received by party B (2) A sets shared variable X to 1, read by B (3) A sets "X=1" in a distributed shared map accessed by B [Explain] **II:** Write a program that distributes different segments of a data set to multiple workers, and collects results (for the simplest example, summing segments of an array) [Write] **III:** Write a parallel program that requests data from multiple sites and summarizes them using some form of reduction [Write] **IV:** Compare the performance of buffered versus unbuffered versions of a producer-consumer program [Compare]

Bibliography: [van Steen and Tanenbaum, 2023, Kleppmann, 2017]

5.8 Communication: Properties and Extensions, 3 hr(s)

Competences: AG-C09,AG-C11

Topics: I: One or more of the following properties and extensions (a) Topologies: Unicast, Multicast, Mailboxes, Switches; Routing via hardware and software interconnection networks (b) Media concurrency properties: Ordering, consistency, idempotency, overlapping communication with computation (c) Media performance: Latency, bandwidth (throughput) contention (congestion), responsiveness (liveness), reliability (error and drop rates), protocol-based progress (acks, timeouts, mediation) (d) Media security properties: integrity, privacy, authentication, authorization.

See also: Information Flow and Non-Interference, Injection and Input Validation, Memory Safety and Types, Malware Analysis and Advanced Security (e) Data formats: Marshaling, validation, encryption, compression (f) Channel policies: Endpoints, sessions, buffering, saturation response (waiting vs dropping), rate control (g) Multiplexing and demultiplexing many relatively slow I/O devices or parties; completion-based and scheduler-based techniques; async-await, select and polling APIs (h) Formalization and analysis of channel communication; for example, CSP (i) Applications of queuing theory to model and predict performance.

Learning Outcomes **I:** Determine whether a given communication scheme provides sufficient security properties for a given usage [Determine] **II:** Give an example of a scenario in which blocking message sends can deadlock [Create] **III:** Describe at least one design technique for avoiding liveness failures in programs using multiple locks [Describe]

Bibliography: [van Steen and Tanenbaum, 2023, Kleppmann, 2017]

5.9 Memory and Consistency, 3 hr(s)

Competences: AG-C09,AG-C11

Topics: **I:** Memory models, data consistency, and fault tolerance (a) Memory models: sequential and release/acquire consistency (b) Memory management; including reclamation of shared data; reference counts and alternatives (c) Bulk data placement and transfer; reducing message traffic and improving locality; overlapping data transfer and computation; impact of data layout such as array-of-structs vs struct-of-arrays (d) Emulating shared memory: distributed shared memory, Remote Direct Memory Access (RDMA) (e) Data store consistency: Atomicity, linearizability, transactionality, coherence, causal ordering, conflict resolution, eventual consistency, blockchains (f) Faults, partitioning, and partial failures; voting; protocols such as Paxos and Raft. (g) Design tradeoffs among consistency, availability, partition (fault) tolerance; impossibility of meeting all at once (h) Security and trust: Byzantine failures, proof of work and alternatives

Learning Outcomes **I:** Give an example of an ordering of accesses among concurrent activities (e.g., program with a data race) that is not sequentially consistent [Create] **II:** Write a program that illustrates memory-access or message reordering [Write] **III:** Describe the relative merits of optimistic versus conservative concurrency control under different rates of contention among updates [Describe] **IV:** Give an example of a scenario in which an attempted optimistic update may never complete [Create] **V:** Modify a concurrent system to use a more scalable, reliable, or available data store [Create] **VI:** Using an existing platform supporting replicated data stores, write a program that maintains a key-value mapping even when one or more hosts fail [Write]

Bibliography: [van Steen and Tanenbaum, 2023, Kleppmann, 2017]

5.10 Coordination, 7 hr(s)

Competences: AG-C09,AG-C11

Topics: **I:** Dependencies (a) Initiation or progress of one activity may be dependent on other activities, so as to avoid race conditions, ensure termination, or meet other requirements (b) Ensuring progress by avoiding dependency cycles, using monotonic conditions, removing inessential dependencies **II:** Control constructs and design patterns (a) Completion-based: Barriers, joins, including termination control (b) Data-enabled: Queues, producer-consumer designs (c) Condition-based: Polling, retrying, backoffs, helping, suspension, signaling, timeouts (d) Reactive: Enabling and triggering continuations

Learning Outcomes **I:** Show how to ensure that a program correctly terminates when all of a set of concurrent tasks have completed [Design] **II:** Write a function that efficiently counts events such as sensor inputs or networking packet receptions [Write] **III:** Write a filter/map/reduce program in multiple styles [Write] **IV:** Write a program in which the termination of one set of parallel actions is followed by another [Write] **V:** Write a service that creates a thread (or other procedural form of activation) to return a requested web page to each new client [Write]

Bibliography: [Herlihy et al., 2020, van Steen and Tanenbaum, 2023]

5.11 Coordination: Synchronization and Atomicity, 3 hr(s)

Competences: AG-C09,AG-C11

Topics: I: Atomicity (a) Atomic instructions, enforced local access orderings (b) Locks and mutual exclusion; lock granularity (c) Using locks in a specific language; maintaining liveness without introducing races (d) Deadlock avoidance: Ordering, coarsening, randomized retries; backoffs, encapsulation via lock managers (e) Common errors: Failing to lock or unlock when necessary, holding locks while invoking unknown operations (f) Avoiding locks: replication, read-only, ownership, and non-blocking constructions

Learning Outcomes I: Show how to avoid or repair a race error in a given program [Analyze]

Bibliography: [Herlihy et al., 2020]

5.12 Coordination: Advanced Properties, 4 hr(s)

Competences: AG-C09,AG-C11

Topics: I: One or more of the following properties and extensions (a) Progress properties including lock-free, wait-free, fairness, priority scheduling, interactions with consistency, reliability (b) Performance with respect to contention, granularity, convoying, scaling (c) Non-blocking data structures and algorithms (d) Ownership and resource control (e) Lock variants and alternatives: sequence locks, read-write locks; Read-Copy-Update (RCU), reentrancy; tickets; controlling spinning versus blocking (f) Transaction-based control: Optimistic and conservative (g) Distributed locking: reliability (h) Alternatives to barriers: Clocks; counters, virtual clocks; dataflow and continuations; futures and RPC; consensus-based, gathering results with reducers and collectors (i) Speculation, selection, cancellation; observability and security consequences (j) Resource control using semaphores and condition variables (k) Control flow: Scheduling computations, series-parallel loops with (possibly elected) leaders, pipelines and streams, nested parallelism (l) Exceptions and failures. Handlers, detection, timeouts, fault tolerance, voting

Learning Outcomes I: Write a program that speculatively searches for a solution by multiple activities, terminating others when one is found [Write] **II:** Write a program in which a numerical exception (such as divide by zero) in one activity causes termination of others [Write] **III:** Write a program for multiple parties to agree upon the current time of day; discuss its limitations compared to protocols such as network transfer protocol (NTP) [Write]

Bibliography: [Herlihy et al., 2020]

5.13 Evaluation, 7 hr(s)

Competences: AG-C09,AG-C11

Topics: I: Safety and liveness requirements in terms of temporal logic constructs to express "always" and "eventually" See also: Parallel and Distributed Computing **II:** Identifying, testing for, and repairing violations, including common forms of errors such as failure to ensure necessary ordering (race errors), atomicity (including check-then-act errors), and termination (livelock) **III:** Performance requirements metrics for throughput, responsiveness, latency, availability, energy consumption, scalability, resource usage, communication costs, waiting and rate control, fairness; service level agreements. See also: Latency, Cache and Memory Hierarchy, Virtualization and Isolation **IV:** Performance impact of design and implementation choices, including granularity, overhead, consensus costs, and energy consumption. See also: Sustainable Design and Pervasive Computing, Environmental Footprint of Computing Systems, Systemic Effects and Social Context **V:** Estimating scalability limitations, for example using Amdahl's Law or Universal Scalability Law. See also: Events, Tools and Experimentation, Performance Metrics and Benchmarks, Analytical Performance Models

Learning Outcomes I: Revise a specification to enable parallelism and distribution without violating other essential properties or features [Redesign] **II:** Explain how concurrent notions of safety and liveness extend their sequential counterparts [Explain] **III:** Specify a set of invariants that must hold at each bulk-parallel step of a computation [Analyze] **IV:** Write a test program that can reveal a data race error; for example, missing an update when two activities both try to increment a variable [Write] **V:** In a given context, explain the extent to which introducing parallelism in an otherwise sequential program would be expected to improve throughput and/or reduce latency, and how it may impact energy efficiency [Explain] **VI:** Show how scaling and efficiency change for sample problems without and with the assumption of problem size changing with the number of processors; further explain whether and how scalability would change under relaxations of sequential dependencies [Design]

Bibliography: [Sterling et al., 2024, Pacheco and Malensek, 2021]

5.14 Evaluation: Formal Methods, 3 hr(s)

Competences: AG-C09,AG-C11

Topics: I: Formal verification and analysis methods (a) Extensions to formal sequential requirements such as linearizability (b) Protocol, session, and transactional specifications (c) Use of tools such as Unified Modelling Language (UML), Temporal Logic of Actions (TLA), program logics (d) Security analysis: safety and liveness in the presence of hostile or buggy behaviors by other parties; required properties of communication mechanisms (for example lack of cross-layer leakage), input screening, rate limiting. See also: AI and Hardware Security, Access Control and Applications, Security Mindset and Principles (e) Static analysis applied to correctness, throughput, latency, resources, energy. See also: Sustainable Design and Pervasive Computing, Environmental Footprint of Computing Systems, Systemic Effects and Social Context (f) Directed Acyclic Graph (DAG) model analysis of algorithmic efficiency (work, span, critical paths)

Learning Outcomes I: Specify and measure behavior when a service is requested by unexpectedly many clients [Analyze] **II:** Identify and repair a performance problem due to sequential bottlenecks [Analyze] **III:** Empirically compare throughput of two implementations of a common design (perhaps using an existing test harness framework) [Compare]

Bibliography: [Sterling et al., 2024, Pacheco and Malensek, 2021]

5.15 Evaluation: Testing and Measurement, 2 hr(s)

Competences: AG-C09,AG-C11

Topics: I: Testing tools and measurement techniques (a) Testing and debugging; tools such as race detectors, fuzzers, lock dependency checkers, unit/stress/torture tests, visualizations, continuous integration, continuous deployment, and test generators (b) Measuring and comparing throughput, overhead, waiting, contention, communication, data movement, locality, resource usage, behavior in the presence of excessive numbers of events, clients, or threads. See also: Events, Tools and Experimentation, Performance Metrics and Benchmarks, Analytical Performance Models (c) Application domain specific analyses and evaluation techniques

Learning Outcomes I: Identify and repair a performance problem due to communication or data latency [Analyze] **II:** Identify and repair a performance problem due to resource management overhead [Analyze] **III:** Identify and repair a reliability or availability problem [Analyze]

Bibliography: [Sterling et al., 2024, Pacheco and Malensek, 2021]

5.16 Algorithms, 4 hr(s)

Competences: AG-C09,AG-C11

Topics: I: Expressing and implementing algorithms in given languages and frameworks, to initiate activities (for example threads), use shared memory constructs, and channel, socket, and/or remote procedure call APIs. See also: Parallel and Distributed Computing. (a) Data parallel examples including map/reduce. (b) Using channel, socket, and/or RPC APIs in a given language, with program control for sending (usually procedural) vs receiving. (usually reactive or RPC-based). (c) Using locks, barriers, and/or synchronizers to maintain liveness without introducing races.

Learning Outcomes I: Implement a parallel/distributed component based on a known algorithm [Implement] **II:** Write a data-parallel program that for example computes the average of an array of numbers [Write] **III:** Write a producer-consumer program in which one component generates numbers, and another computes their average. Measure speedups when the numbers are small scalars versus large multi-precision values [Write]

Bibliography: [Pacheco and Malensek, 2021, Herlihy et al., 2020]

5.17 Algorithms: Application Domains Survey, 4 hr(s)

Competences: AG-C09,AG-C11

Topics: I: Survey of common application domains across multicore, reactive, data parallel, cluster, cloud, open distributed systems, and frameworks (with reference to the following table). (a) Multicore: Typical Execution agents: Threads. Typical Communication mechanisms: Shared memory, Atomics, locks. Typical Algorithmic domains: Resource management, data processing. Typical Engineering goals: Throughput, latency, energy. (b) Reactive: Typical Execution agents: Handlers, threads. Typical Communication mechanisms: I/O Channels. Typical Algorithmic domains: Services, real-time. Typical Engineering goals: Latency. (c) Data parallel: Typical Execution agents: GPU, SIMD, accelerators, hybrid. Typical Communication mechanisms: Heterogeneous memory. Typical Algorithmic domains: Linear algebra, graphics, data analysis. Typical Engineering goals: Throughput, energy. (d) Cluster: Typical Execution agents: Managed hosts. Typical Communication mechanisms: Sockets, channels. Typical Algorithmic domains: Simulation, data analysis. Typical Engineering goals: Throughput. (e) Cloud: Typical Execution agents: Provisioned hosts. Typical Communication mechanisms: Service APIs. Typical Algorithmic domains: Web applications. Typical Engineering goals: Scalability. (f) Open distributed: Typical Execution agents: Autonomous hosts. Typical Communication mechanisms: Sockets, Data stores. Typical Algorithmic domains: Fault tolerant data stores and services. Typical Engineering goals: Reliability.

Learning Outcomes I: Extend an event-driven sequential program by establishing a new activity in an event handler (for example a new thread in a GUI action handler) [Design] **II:** Improve the performance of a sequential component by introducing parallelism and/or distribution [Create] **III:** Choose among different parallel/distributed designs for components of a given system [Assess]

Bibliography: [Pacheco and Malensek, 2021, Herlihy et al., 2020]

5.18 Algorithms: Algorithmic Domains, 4 hr(s)

Competences: AG-C09,AG-C11

Topics: I: One of more of the following algorithmic domains. See also: Algorithmic Strategies: (a) Linear algebra: Vector and matrix operations, numerical precision/stability, applications in data analytics and machine learning. (b) Data processing: sorting, searching and retrieval, concurrent data structures. (c) Graphs, search, and combinatorics: Marking, edge-parallelization, bounding, speculation, network-based analytics. (d) Modeling and simulation: differential equations; randomization, N-body problems, genetic algorithms. (e) Computational logic: satisfiability (SAT), concurrent logic programming. (f) Graphics and computational geometry: Transforms, rendering, ray-tracing. (g) Resource management: Allocating, placing, recycling and scheduling

processors, memory, channels, and hosts; exclusive vs shared resources; static, dynamic and elastic algorithms; Real-time constraints; Batching, prioritization, partitioning; decentralization via work-stealing and related techniques. (h) Services: Implementing web APIs, electronic currency, transaction systems, multiplayer games.

Learning Outcomes I: Design, implement, analyze, and evaluate a component or application for X operating in a given context, where X is in one of the listed domains, for example, a genetic algorithm for factory floor design [Design] **II:** Critique the design and implementation of an existing component or application, or one developed by classmates [Critique] **III:** Compare the performance and energy efficiency of multiple implementations of a similar design, for example, multicore versus clustered versus GPU [Compare]

Bibliography: [Pacheco and Malensek, 2021, Herlihy et al., 2020]

6. Bibliography

[Corporation, 2024] Corporation, N. (2024). Cuda c++ programming guide. `https://docs.nvidia.com/cuda/cuda-c-programming-guide/`. Documentación oficial de NVIDIA.

[Herlihy et al., 2020] Herlihy, M., Shavit, N., Luchangco, V., and Spear, M. (2020). *The Art of Multiprocessor Programming*. Morgan Kaufmann, 2nd edition.

[Kirk and mei W. Hwu, 2016] Kirk, D. B. and mei W. Hwu, W. (2016). *Programming Massively Parallel Processors: A Hands-on Approach*. Morgan Kaufmann, Cambridge, MA, 3rd edition.

[Kleppmann, 2017] Kleppmann, M. (2017). *Designing Data-Intensive Applications*. O'Reilly Media, 1st edition.

[mei W. Hwu et al., 2022] mei W. Hwu, W., Kirk, D. B., and Hajj, I. E. (2022). *Programming Massively Parallel Processors: A Hands-on Approach*. Morgan Kaufmann, 4th edition.

[Pacheco and Malensek, 2021] Pacheco, P. S. and Malensek, M. (2021). *An Introduction to Parallel Programming*. Morgan Kaufmann, 2nd edition.

[Sterling et al., 2024] Sterling, T., Brodowicz, M., and Anderson, M. (2024). *High Performance Computing: Modern Systems and Practices*. Morgan Kaufmann, 2nd edition.

[van Steen and Tanenbaum, 2023] van Steen, M. and Tanenbaum, A. S. (2023). *Distributed Systems*. Maarten van Steen, 4th edition.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

CS402 Capstone Project I (Mandatory)

1. General information

- **Semester**: 8th Sem. **Credits**: 4
- **Hour of this course**: **Theory**: 2 hours; **Laboratory**: 4 hours;
- **Prerrequisites**:
  - CS401 Research Methodology (4th Sem-Pg. 331)

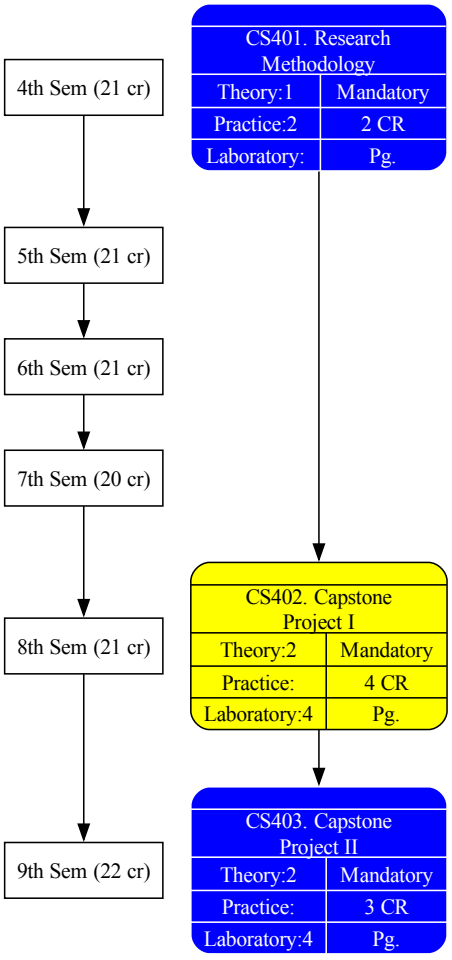

Figure 5.47: Connection Map. CS402 Capstone Project I

2. Course Rationale

This course aims to enable students to conduct a deep state-of-the-art study on a specific topic chosen for their thesis project. Students will produce a technical report with preliminary comparative experiments and formulate a research proposal, culminating this part alongside with a faculty advisor who specializes in the chosen research area.

3. Goals of the Course

- Conduct a deep state-of-the-art study on a specific topic in computer science.

- Demonstrate mastery of the relevant literature through critical and comparative analysis of existing approaches.
- Design and execute preliminary experiments that reproduce and compare state-of-the-art techniques.
- Write a high-quality technical report with theoretical framework, methodology, and experimental results.
- Formulate a research proposal with research question, hypothesis, objectives, and schedule.
- Select a thesis advisor who specializes in the chosen research area.
- The deliverables for this course are:

  Midterm Progress: Solid bibliography and draft of the theoretical framework in LaTeXformat.

  Final: Technical Report with preliminary comparative experiments and research proposal approved by the advisor.

4. General Competences (*Outcomes*)

AG-C02) Ethics: Applies ethical principles and commits to professional ethics and the standards of professional computing practice. (Assessment)

AG-C04) Communication: Communicates effectively in complex computing activities. (Assessment)

AG-C05) Project Management: Applies management principles in computing to manage projects. (Assessment)

AG-C08) Problem Analysis: Identifies, formulates, and analyzes complex computing problems. (Assessment)

AG-C09) Design and Development of Solutions: Designs, implements, and evaluates solutions for complex computing problems. (Assessment)

5. Content

5.1 Deep State-of-the-Art Study on a Specific Topic, 10 hr(s)

Competences: AG-C08,AG-C09

Topics: I: Selection and refinement of the research topic in coordination with the advisor. **II:** Exhaustive multi-source search: IEEE Xplore, ACM Digital Library, Google Scholar, arXiv, and specialized databases. **III:** Identification of seminal works and most-cited papers in the area. **IV:** Construction of the citation graph and temporal evolution analysis of the topic. **V:** Identification of sub-areas, variants, and main approaches in the field.

Learning Outcomes I: Build a comprehensive literature map for a specific topic identifying the foundational works. [Assessment] **II:** Analyze the temporal evolution of a research area identifying trends and gaps. [Assessment] **III:** Organize the literature into coherent subcategories that reflect the structure of the field. [Assessment]

Bibliography: [IEEE-Computer Society, 2008, ACM, 2008, CiteSeer.IST, 2008]

5.2 Critical and Comparative Analysis of Approaches, 10 hr(s)

Competences: AG-C08,AG-C09

Topics: I: Critical analysis of proposals: methodology, datasets, metrics, and limitations of each approach. **II:** Construction of comparative tables of existing proposals using objective criteria. **III:** Identification of unresolved research gaps in the area. **IV:** Formulation of the student's own research question based on the state-of-the-art analysis. **V:** Theoretical framework: structured and coherent synthesis of relevant knowledge in the area.

Learning Outcomes **I:** Systematically compare multiple approaches using objective and quantifiable criteria. [Assessment] **II:** Identify unresolved research gaps in a specific area of computer science. [Assessment] **III:** Formulate an original research question grounded in the state-of-the-art analysis. [Assessment]

Bibliography: [IEEE-Computer Society, 2008, ACM, 2008, CiteSeer.IST, 2008]

5.3 Design and Execution of Preliminary Experiments, 12 hr(s)

Competences: AG-C05,AG-C08,AG-C09

Topics: **I:** Experimental design: variables, hypotheses, metrics, and benchmarks of the area. **II:** Reproduction of results from relevant state-of-the-art works. **III:** Implementation of baselines for objective comparison of approaches. **IV:** Basic statistical analysis: significance tests and confidence intervals. **V:** Experimental data management: datasets, configurations, versions, and results logging.

Learning Outcomes **I:** Reproduce the results of at least two relevant works from the state of the art of the area. [Usage] **II:** Design and execute a preliminary comparative experiment with defined baselines. [Usage] **III:** Interpret and report experimental results with appropriate statistical rigor. [Assessment]

Bibliography: [IEEE-Computer Society, 2008, ACM, 2008]

5.4 Technical Report Writing, 12 hr(s)

Competences: AG-C04,AG-C09

Topics: **I:** Structure of the technical report: introduction, theoretical framework, methodology, experiments, results, and conclusions. **II:** Writing the theoretical framework: thematic organization and critical synthesis of the literature. **III:** Documentation of methodology and experimental configuration in LaTeX. **IV:** Presentation of results: high-quality tables, charts, and comparative analyses. **V:** Revision cycles: writing, advisor review, and incorporation of feedback.

Learning Outcomes **I:** Write a complete technical report including theoretical framework and preliminary experiments. [Usage] **II:** Structure and present experimental results with clear and precise visualizations. [Usage] **III:** Apply revision cycles to improve the quality, coherence, and rigor of the document. [Assessment]

Bibliography: [IEEE-Computer Society, 2008, ACM, 2008]

5.5 Research Proposal and Advisor Selection, 8 hr(s)

Competences: AG-C02,AG-C05,AG-C09

Topics: **I:** Structure of a research proposal: problem, motivation, hypothesis, and objectives. **II:** Definition of thesis project scope: methodology, expected contribution, and schedule. **III:** Feasibility analysis: resources, timeline, risks, and contingencies. **IV:** Criteria for selecting a thesis advisor: expertise, availability, and thematic alignment. **V:** Formalization of the researcher-advisor relationship within the academic program.

Learning Outcomes **I:** Produce a research proposal defining the problem, hypothesis, objectives, and methodology. [Assessment] **II:** Select an advisor with demonstrated expertise in the chosen research area. [Usage] **III:** Define a viable thesis plan with objectives, scope, schedule, and risk analysis. [Assessment]

Bibliography: [IEEE-Computer Society, 2008, ACM, 2008]

5.6 Presentation and Defense of the State of the Art, 8 hr(s)

Competences: AG-C02,AG-C04,AG-C05

Topics: **I:** Preparation of the oral presentation of the technical report before the evaluation committee. **II:** Defense techniques for handling questions and challenges from an academic committee. **III:** Design of a technical poster of the state of the art for symposium presentation. **IV:** Incorporation of evaluating committee feedback into the final revision of the report. **V:** Planning the transition to CS403: objectives, expected contribution, and next steps.

Learning Outcomes **I:** Present and orally defend the technical report before an academic committee with rigor. [Assessment] **II:** Constructively incorporate committee feedback into a revision of the report. [Usage] **III:** Plan the transition to the thesis project (CS403) with clear objectives and next steps. [Usage]

Bibliography: [IEEE-Computer Society, 2008, ACM, 2008]

6. Bibliography

[ACM, 2008] ACM (2008). *Digital Library.* Association for Computing Machinery. http://portal.acm.org/dl.cfm.

[CiteSeer.IST, 2008] CiteSeer.IST (2008). *Scientific Literature Digital Library.* College of Information Sciences and Technology, Penn State University. http://citeseer.ist.psu.edu.

[IEEE-Computer Society, 2008] IEEE-Computer Society (2008). *Digital Library.* IEEE-Computer Society. http://www.computer.org/publications/dlib.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

EX301 Extracurricular Activities (Mandatory)

1. General information

- **Semester**: 8th Sem. **Credits**: 2
- **Hour of this course**: **Theory**: 1 hours; **Practice**: 2 hours;
- **Prerrequisites**: None

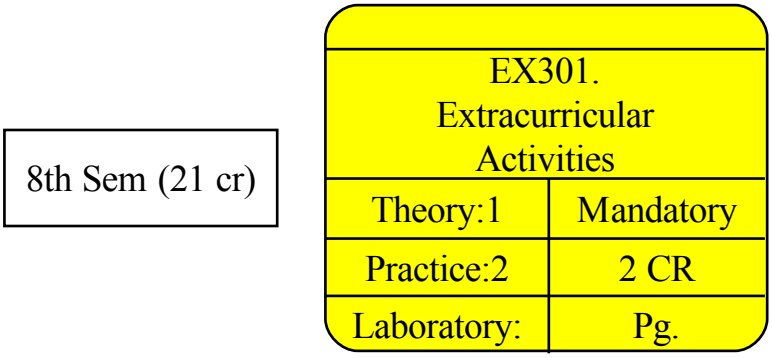

Figure 5.48: Connection Map. EX301 Extracurricular Activities

2. Course Rationale

This course records the student's participation in activities that complement their academic training, such as workshops, conferences, social outreach, volunteering, or technical skills. Its objective is to promote comprehensive development, soft skills, and social commitment.

3. Goals of the Course

- Participate in activities that complement their professional training.
- Develop communication and teamwork skills.
- Contribute to initiatives with a social or technical impact.

4. General Competences (*Outcomes*)

AG-C01) The Professional and the World: Analyzes and evaluates the impact of solutions to complex computing problems on the sustainable development of society. (Familiarity)

AG-C06) Life-long Learning: Recognizes the importance of continuous learning and adaptation to new technologies. (Familiarity)

5. Content

5.1 Extracurricular Activities, 48 hr(s)

Competences: AG-C01,AG-C06

Topics: **I:** Technical or interdisciplinary workshops. **II:** Attendance at conferences, seminars, or talks. **III:** Social outreach (volunteering, mentoring). **IV:** Academic skills (hackathons, science fairs).

Learning Outcomes **I:** Demonstrate participation in at least three activities [Familiarity]. **II:** Prepare reflective reports on experiences [Usage]. **III:** Collaborate in teams for social or technical projects [Assessment].

Bibliography: [ACM, 2019, ABET Computing Accreditation Commission, 2022]

6. Bibliography

[ABET Computing Accreditation Commission, 2022] ABET Computing Accreditation Commission (2022). Abet criteria for student professional development. Technical report, ABET.

[ACM, 2019] ACM (2019). Guidelines for extracurricular activities in computing. Technical report, ACM.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

ET201 Entrerpreneurship I (Mandatory)

1. General information

- **Semester**: 8th Sem. **Credits**: 3
- **Hour of this course**: **Theory**: 2 hours; **Practice**: 2 hours;
- **Prerrequisites**:
  - FG350 Leadership and Performance (7th Sem-Pg. 416)

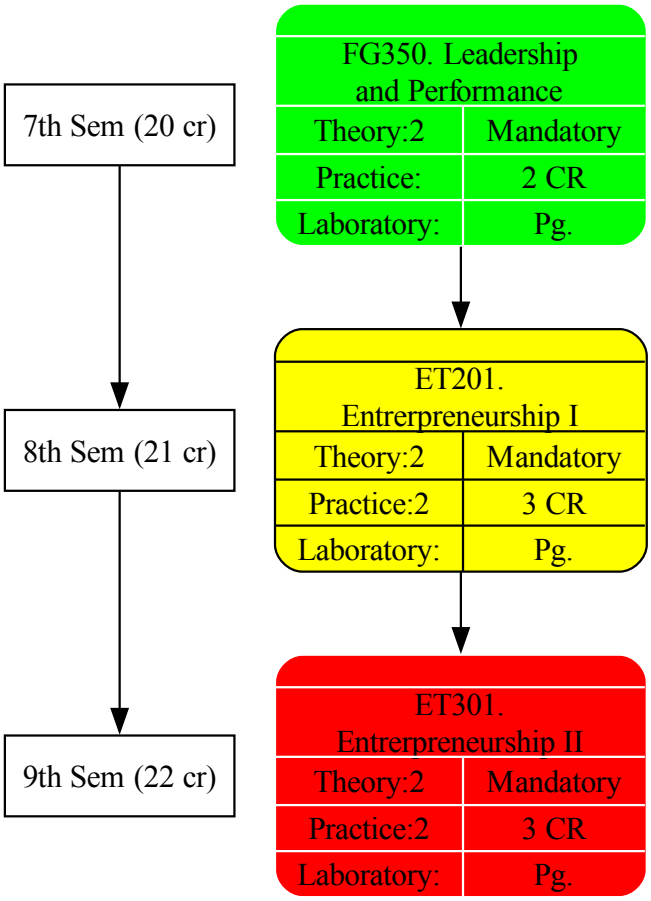

Figure 5.49: Connection Map. ET201 Entrerpreneurship I

2. Course Rationale

This is the first course in the area of technology-based business formation. Its objective is to equip future professionals with the knowledge, attitudes, and skills necessary to develop a business plan for a technology-based company. The course is divided into the following units: Introduction, Creativity, From Idea to Opportunity, the Canvas Model, Customer Development and Lean Startup, Legal and Marketing Aspects, Business Finance, and Presentation.

The course seeks to leverage students' creative and innovative potential and their efforts in the creation of new businesses.

3. Goals of the Course

- Students will learn how to develop a business plan to launch a technology-based company.
- Students will be able to design and present a business proposal using business models.

4. General Competences (*Outcomes*)

No items for this table

5. Content

5.1 Introduction, 5 hr(s)

Competences:

Topics: **I:** Entrepreneur, entrepreneurship, and technological innovation **II:** Business models **III:** Team formation

Learning Outcomes **I:** Identify the characteristics of entrepreneurs [Familiarity] **II:** Introduce business models [Familiarity]

Bibliography: [Byers et al., 2010, Osterwalder and Pigneur, 2010, Garzozi-Pincay et al., 2014]

5.2 Creativity, 5 hr(s)

Competences:

Topics: **I:** Vision **II:** Mission **III:** Value proposition **IV:** Creativity and invention **V:** Types and sources of innovation **VI:** Strategy and technology **VII:** Scale and scope

Learning Outcomes **I:** Correctly formulate a company's vision and mission [Usage] **II:** Characterize an innovative value proposition [Assessment] **III:** Identify different types and sources of innovation [Familiarity]

Bibliography: [Byers et al., 2010, Blank and Dorf, 2012, Garzozi-Pincay et al., 2014]

5.3 From Idea to Opportunity, 5 hr(s)

Competences:

Topics: **I:** Business strategy **II:** Barriers **III:** Sustainable competitive advantage **IV:** Alliances **V:** Organizational learning **VI:** Product development and design

Learning Outcomes **I:** Understand business strategies [Familiarity] **II:** Characterize barriers and competitive advantages [Familiarity]

Bibliography: [Byers et al., 2010, Osterwalder and Pigneur, 2010, Ries, 2011, Garzozi-Pincay et al., 2014]

5.4 The Canvas Model, 20 hr(s)

Competences:

Topics: **I:** Creating a new business **II:** The business plan **III:** Canvas **IV:** Elements of the Canvas

Learning Outcomes **I:** Understand the elements of the Canvas model [Usage] **II:** Develop a business plan based on the Canvas model [Usage]

Bibliography: [Osterwalder and Pigneur, 2010, Blank and Dorf, 2012, Garzozi-Pincay et al., 2014]

5.5 Customer Development and Lean Startup, 20 hr(s)

Competences:

Topics: **I:** Acceleration vs. incubation **II:** Customer Development **III:** Lean Startup

Learning Outcomes **I:** Understand and apply the Customer Development model [Usage] **II:** Understand and apply the Lean Startup model [Usage]

Bibliography: [Blank and Dorf, 2012, Ries, 2011, Garzozi-Pincay et al., 2014]

5.6 Legal and Marketing Aspects, 5 hr(s)

Competences:

Topics: **I:** Legal and tax aspects for company formation **II:** Intellectual property **III:** Patents **IV:** Copyrights and trademarks **V:** Marketing objectives and market segments **VI:** Market research and customer acquisition

Learning Outcomes **I:** Understand the legal aspects necessary for forming a technology company [Familiarity] **II:** Identify market segments and marketing objectives [Familiarity]

Bibliography: [Byers et al., 2010, Ries, 2011, Congreso de la Republica del Perú, 1996, de la Republica del Peru, 1997, Garzozi-Pincay et al., 2014]

5.7 Business Finance, 5 hr(s)

Competences:

Topics: **I:** Cost model **II:** Profit model **III:** Pricing **IV:** Financial plan **V:** Financing methods **VI:** Sources of capital **VII:** Venture capital

Learning Outcomes **I:** Define a cost and profit model [Assessment] **II:** Understand various financing sources [Familiarity]

Bibliography: [Byers et al., 2010, Blank and Dorf, 2012, Garzozi-Pincay et al., 2014]

5.8 Presentation, 5 hr(s)

Competences:

Topics: **I:** The Elevator Pitch **II:** Presentation **III:** Negotiation

Learning Outcomes **I:** Understand various methods for presenting business proposals [Familiarity] **II:** Deliver a business proposal presentation [Usage]

Bibliography: [Byers et al., 2010, Blank and Dorf, 2012, Garzozi-Pincay et al., 2014]

6. Bibliography

[Blank and Dorf, 2012] Blank, S. and Dorf, B. (2012). *The Startup Owner's Manual: The Step-By-Step Guide for Building a Great Company*. K and S Ranch.

[Byers et al., 2010] Byers, T., Dorf, R., and Nelson, A. (2010). *Technology Ventures: From Idea to Enterprise*. McGraw-Hill Science.

[Congreso de la Republica del Perú, 1996] Congreso de la Republica del Perú (1996). *Decreto Legislativo Nº823. Ley de la Propiedad Industrial*. El Peruano.

[de la Republica del Peru, 1997] de la Republica del Peru, C. (1997). *Ley Nº26887. Ley General de Sociedades*. El Peruano.

[Garzozi-Pincay et al., 2014] Garzozi-Pincay, R., Messina-Scolaro, M., Moncada-Marino, C., Ochoa-Luna, J., Ilabel-Pérez, G., and Zambrano-Segura, R. (2014). *Planes de Negocios para Emprendedores*. Iniciativa Latinoamericana de Libros de Texto Abiertos (LATIn).

[Osterwalder and Pigneur, 2010] Osterwalder, A. and Pigneur, Y. (2010). *Business Model Generation: A Handbook for Visionaries, Game Changers, and Challengers*. Wiley.

[Ries, 2011] Ries, E. (2011). *The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses*. Crown Business.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

AI268 Computational Vision (Elective)

1. General information

- **Semester**: 8th Sem. **Credits**: 3
- **Hour of this course**: **Theory**: 2 hours; **Laboratory**: 2 hours;
- **Prerrequisites**:
  - AI264 Deep Learning (7th Sem-Pg. 411)

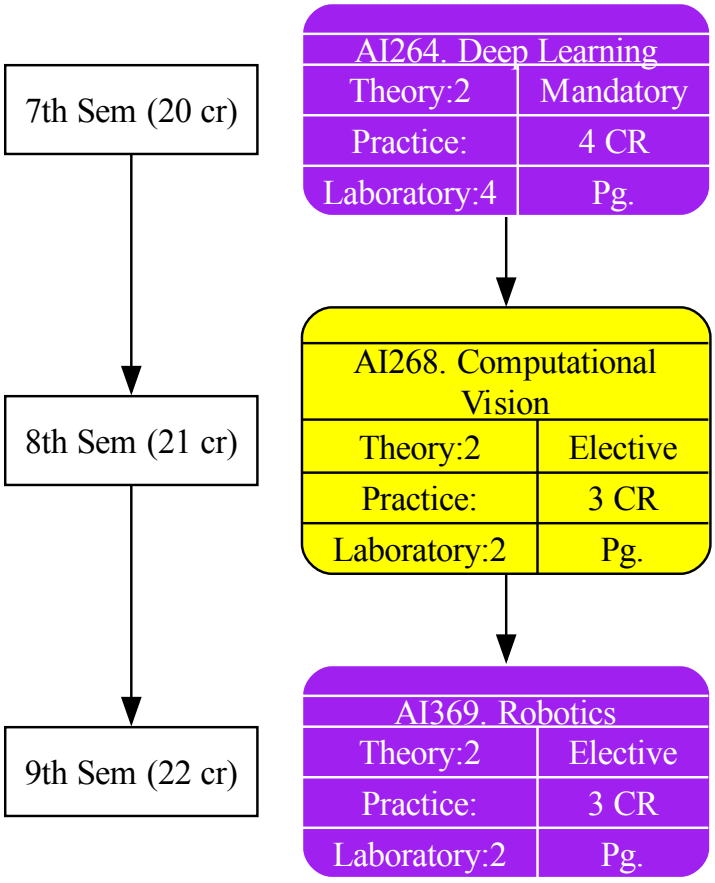

Figure 5.50: Connection Map. AI268 Computational Vision

2. Course Rationale

Computer Vision is a key area of Artificial Intelligence that enables machines to interpret and understand the visual world. This course covers fundamental techniques for image processing, feature extraction, and object recognition. With the advent of deep learning, computer vision has become essential for autonomous vehicles, medical imaging, and augmented reality. Students will learn the mathematical foundations and algorithmic approaches to solve real-world visual perception problems.

3. Goals of the Course

- Understand the fundamentals of image formation and digital representation.
- Master image processing techniques such as filtering and edge detection.
- Apply feature extraction methods to identify key patterns in images.
- Implement object detection and recognition using deep learning models.
- Analyze 3D vision and motion estimation techniques.

4. General Competences (*Outcomes*)

AG-C08) Problem Analysis: Identifies, formulates, and analyzes complex computing problems. (Usage)

AG-C09) Design and Development of Solutions: Designs, implements, and evaluates solutions for complex computing problems. (Assessment)

AG-C12) Applies computer science theory and software development fundamentals to produce computer-based solutions. (Usage)

5. Content

5.1 Introduction to AI: Concepts and Intelligent Behavior, 5 hr(s)

Competences: AG-C08

Topics: **I:** Overview of AI problems, Examples of successful recent AI applications **II:** Definitions of agents with examples (e.g., reactive, deliberative) **III:** What is intelligent behavior?: (a) The Turing test and its flaws (b) Multimodal input and output (c) Simulation of intelligent behavior (d) Rational versus non-rational reasoning **IV:** Overview of AI Applications, growth, and impact (economic, societal, ethics) **V:** Philosophical issues **VI:** History of AI

Learning Outcomes **I:** Describe the Turing test and the "Chinese Room" thought experiment [Explain] **II:** Differentiate between optimal reasoning/behavior and human-like reasoning/behavior [Evaluate] **III:** Differentiate the terms: AI, machine learning, and deep learning [Explain]

Bibliography: [Szeliski, 2022, Gonzalez and Woods, 2018]

5.2 Problem Characteristics and Agent Nature, 5 hr(s)

Competences: AG-C08

Topics: **I:** Problem characteristics: (a) Fully versus partially observable (b) Single versus multi-agent (c) Deterministic versus stochastic (d) Static versus dynamic (e) Discrete versus continuous **II:** Nature of agents: (a) Autonomous, semi-autonomous, mixed-initiative autonomy (b) Reflexive, goal-based, and utility-based (c) Decision making under uncertainty and with incomplete information (d) The importance of perception and environmental interactions (e) Learning-based agents (f) Embodied agents: i. sensors, dynamics, effectors **III:** Practice identifying problem characteristics in example environments **IV:** Additional depth on nature of agents with examples **V:** Additional depth on AI Applications, Growth, and Impact (economic, societal, ethics, security)

Learning Outcomes **I:** Enumerate the characteristics of a specific problem [Apply]

Bibliography: [Szeliski, 2022, Gonzalez and Woods, 2018]

5.3 Perception and Computer Vision, 12 hr(s)

Competences: AG-C08

Topics: **I:** Computer vision: (a) Image acquisition, representation, processing, and properties (b) Shape representation, object recognition, and segmentation (c) Motion analysis (d) Generative models **II:** Audio and speech recognition **III:** Touch and proprioception **IV:** Other modalities (e.g., olfaction) **V:** Modularity in recognition **VI:** Approaches to pattern recognition: (a) Classification algorithms and measures of classification quality (b) Statistical techniques (c) Deep learning techniques

Learning Outcomes **I:** Summarize the importance of image and object recognition in AI and indicate several significant applications of this technology [Summarize] **II:** List at least three image-segmentation approaches, such as thresholding, edge-based and region-based algorithms, along with their defining characteristics, strengths, and weaknesses [List] **III:** Implement 2d object recognition based on contour-based and/or region-based shape representations [Apply] **IV:** Distinguish the goals of sound-recognition, speech-recognition, and speaker-recognition and identify how the raw audio signal will be handled differently in each of these cases [Distinguish] **V:** Provide at least two examples of a transformation of a data source from one sensory domain to another, e.g., tactile data interpreted as single-band 2d images [Generate] **VI:** Implement a feature-extraction algorithm on real data, e.g., an edge or corner detector for images or vectors of Fourier coefficients describing a short slice of audio signal [Apply] **VII:** Implement an algorithm combining features into higher-level percepts, e.g., a contour or polygon from visual primitives or phoneme hypotheses from an audio signal [Apply] **VIII:** Implement a classification algorithm that segments input percepts into output categories and quantitatively evaluates the resulting classification [Apply] **IX:** Evaluate the performance of the underlying feature-extraction, relative to at least one alternative possible approach (whether implemented or not) in its contribution to the classification task (8), above [Evaluate] **X:** Describe at least three classification approaches, their prerequisites for applicability, their strengths, and their shortcomings [Describe] **XI:** Implement and evaluate a deep learning solution to problems in computer vision, such as object or scene recognition [Apply]

Bibliography: [Szeliski, 2022, Prince, 2012]

5.4 Machine Learning Fundamentals, 5 hr(s)

Competences: AG-C08

Topics: **I:** Definition and examples of a broad variety of machine learning tasks: (a) Supervised learning: i. Classification ii. Regression (b) Reinforcement learning (c) Unsupervised learning: i. Clustering **II:** Fundamental ideas: (a) No free lunch theorem: no one learner can solve all problems; representational design decisions have consequences. (b) Sources of error and undecidability in machine learning **III:** A simple statistical-based supervised learning such as linear regression or decision trees: (a) Focus on how they work without going into mathematical or optimization details; enough to understand and use existing implementations correctly **IV:** The overfitting problem/controlling solution complexity (regularization, pruning - intuition only): (a) The bias (underfitting) - variance (overfitting) tradeoff

Learning Outcomes **I:** Describe the differences among the three main styles of learning (supervised, reinforcement, and unsupervised) and determine which is appropriate to a particular problem domain [Explain] **II:** Differentiate the terms of AI, machine learning, and deep learning [Evaluate] **III:** Frame an application as a classification problem, including the available input features and output to be predicted (e.g., identifying alphabetic characters from pixel grid input) [Apply] **IV:** Identify overfitting in the context of a problem and learning curves and describe solutions to overfitting [Evaluate] **V:** Explain how machine learning works as an optimization/search process [Explain]

Bibliography: [Goodfellow et al., 2016, Russell and Norvig, 2020]

5.5 Machine Learning Data, Representations and Evaluation, 5 hr(s)

Competences: AG-C08

Topics: **I:** Working with Data: (a) Data preprocessing: i. Importance and pitfalls of preprocessing choices (b) Handling missing values (imputing, flag-as-missing): i. Implications of imputing vs flag-as-missing (c) Encoding categorical variables, encoding real-valued data (d) Normalization/standardization (e) Emphasis on real data, not textbook examples **II:** Representations: (a) Hypothesis spaces and complexity (b) Simple basis feature expansion, such as squaring univariate features (c) Learned feature representations **III:** Machine learning evaluation: (a) Separation of train, validation, and test sets (b) Performance metrics for classifiers (c) Estimation of test performance on held-out data (d) Tuning the parameters of a machine learning model with a validation set (e) Importance of understanding what a model is doing, where its pitfalls/shortcomings are, and the implications of its decisions

Learning Outcomes **I:** Apply two or more simple statistical learning algorithms to a classification task and measure the classifiers' accuracy [Apply] **II:** Implement a statistical learning algorithm and the corresponding optimization process to train the classifier and obtain a prediction on new data [Apply] **III:** Explain proper ML evaluation procedures, including the differences between training and testing performance, and what can go wrong with the evaluation process leading to inaccurate reporting of ML performance [Explain] **IV:** Compare two machine learning algorithms on a dataset, implementing the data preprocessing and evaluation methodology (e.g., metrics and handling of train/test splits) from scratch [Apply]

Bibliography: [Goodfellow et al., 2016, Russell and Norvig, 2020]

5.6 Neural Networks and Ethics, 4 hr(s)

Competences: AG-C08

Topics: **I:** Basic neural networks: (a) Fundamentals of understanding how neural networks work and their training process, without details of the calculations (b) Basic introduction to generative neural networks (e.g., large language models) **II:** Ethics for Machine Learning: (a) Focus on real data, real scenarios, and case studies (b) Dataset/algorithmic/evaluation bias and unintended consequences

Learning Outcomes **I:** Describe the neural network training process and resulting learned representations [Explain]

Bibliography: [Goodfellow et al., 2016, Russell and Norvig, 2020]

5.7 Machine Learning Formulation and Evaluation, 3 hr(s)

Competences: AG-C09

Topics: **I:** Formulation of simple machine learning as an optimization problem, such as least squares linear regression or logistic regression: (a) Objective function (b) Gradient descent (c) Regularization to avoid overfitting (mathematical formulation) **II:** Ensembles of models: (a) Simple weighted majority combination **III:** Performance evaluation: (a) Other metrics for classification (e.g., error, precision, recall) (b) Performance metrics for regressors (c) Confusion matrix (d) Cross-validation: i. Parameter tuning (grid/random search, via cross-validation)

Learning Outcomes **I:** Compare and contrast several learning techniques (e.g., decision trees, logistic regression, naive Bayes, neural networks, and belief networks), providing examples of when each strategy is superior [Compare] **II:** Evaluate the performance of a simple learning system on a real-world dataset [Evaluate] **III:** Characterize the state of the art in learning theory, including its achievements and shortcomings [Explain] **IV:** Explain the problem of overfitting, along with techniques for detecting and managing the problem [Explain] **V:** Explain the triple tradeoff among the size of a hypothesis space, the size of the training set, and performance accuracy [Explain]

Bibliography: [Goodfellow et al., 2016]

5.8 Deep Learning, RL and Ethics, 3 hr(s)

Competences: AG-C09

Topics: **I:** Deep learning: (a) Deep feed-forward networks (intuition only, no mathematics) (b) Convolutional neural networks (intuition only, no mathematics) (c) Visualization of learned feature representations from deep nets (d) Other architectures (generative NN, recurrent NN, transformers, etc.) **II:** Overview of reinforcement learning methods **III:** Two or more applications of machine learning algorithms: (a) E.g., medicine and health, economics, vision, natural language, robotics, game play **IV:** Ethics for Machine Learning: (a) Continued focus on real data, real scenarios, and case studies (b) Privacy (c) Fairness (d) Intellectual property (e) Explainability

Learning Outcomes **I:** Visualize the training progress of a neural network through learning curves in a well-established toolkit (e.g., TensorBoard) and visualize the learned features of the network [Apply] **II:** Given a real-world application of machine learning, describe ethical issues regarding the choices of data, preprocessing steps, algorithm selection, and visualization/presentation of results [Apply]

Bibliography: [Goodfellow et al., 2016]

5.9 Perception and Computer Vision, 12 hr(s)

Competences: AG-C08

Topics: **I:** Computer vision: (a) Image acquisition, representation, processing, and properties (b) Shape representation, object recognition, and segmentation (c) Motion analysis (d) Generative models **II:** Audio and speech recognition **III:** Touch and proprioception **IV:** Other modalities (e.g., olfaction) **V:** Modularity in recognition **VI:** Approaches to pattern recognition: (a) Classification algorithms and measures of classification quality (b) Statistical techniques (c) Deep learning techniques

Learning Outcomes **I:** Summarize the importance of image and object recognition in AI and indicate several significant applications of this technology [Summarize] **II:** List at least three image-segmentation approaches, such as thresholding, edge-based and region-based algorithms, along with their defining characteristics, strengths, and weaknesses [List] **III:** Implement 2d object recognition based on contour-based and/or region-based shape representations [Apply] **IV:** Distinguish the goals of sound-recognition, speech-recognition, and speaker-recognition and identify how the raw audio signal will be handled differently in each of these cases [Distinguish] **V:** Provide at least two examples of a transformation of a data source from one sensory domain to another, e.g., tactile data interpreted as single-band 2d images [Generate] **VI:** Implement a feature-extraction algorithm on real data, e.g., an edge or corner detector for images or vectors of Fourier coefficients describing a short slice of audio signal [Apply] **VII:** Implement an algorithm combining features into higher-level percepts, e.g., a contour or polygon from visual primitives or phoneme hypotheses from an audio signal [Apply] **VIII:** Implement a classification algorithm that segments input percepts into output categories and quantitatively evaluates the resulting classification [Apply] **IX:** Evaluate the performance of the underlying feature-extraction, relative to at least one alternative possible approach (whether implemented or not) in its contribution to the classification task (8), above [Evaluate] **X:** Describe at least three classification approaches, their prerequisites for applicability, their strengths, and their shortcomings [Describe] **XI:** Implement and evaluate a deep learning solution to problems in computer vision, such as object or scene recognition [Apply]

Bibliography: [Szeliski, 2022]

6. Bibliography

[Gonzalez and Woods, 2018] Gonzalez, R. C. and Woods, R. E. (2018). *Digital Image Processing*. Pearson, 4th edition.

[Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press.

[Prince, 2012] Prince, S. J. D. (2012). *Computer Vision: Models, Learning, and Inference*. Cambridge University Press.

[Russell and Norvig, 2020] Russell, S. and Norvig, P. (2020). *Artificial Intelligence: A Modern Approach*. Pearson, 4th edition.

[Szeliski, 2022] Szeliski, R. (2022). *Computer Vision: Algorithms and Applications*. Springer, 2nd edition.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

CS391 Software Engineering III (Elective)

1. General information

- **Semester**: 8th Sem. **Credits**: 3
- **Hour of this course**: **Theory**: 2 hours; **Practice**: 2 hours;
- **Prerrequisites**:
  - CS292 Software Engineering II (7th Sem-Pg. 401)

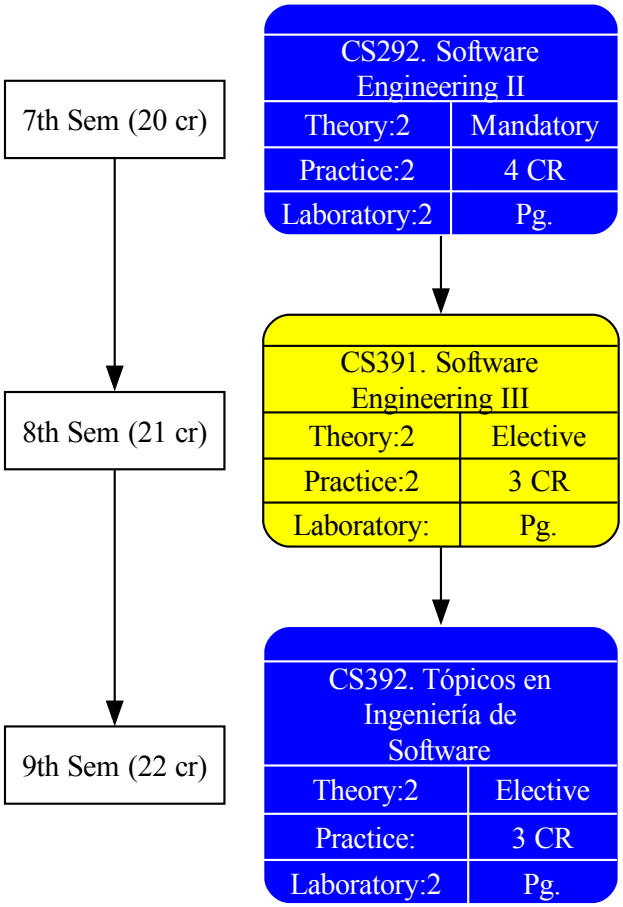

Figure 5.51: Connection Map. CS391 Software Engineering III

2. Course Rationale

Software Engineering III focuses on advanced topics of the software lifecycle, emphasizing software quality, evolution, and deployment in modern architectures. This course prepares students to handle the complexities of maintaining systems, ensuring software quality through rigorous validation, and the application of professional tools and design practices for resilient and maintainable software products.

3. Goals of the Course

- Implement advanced quality assurance and reliability metrics.
- Understand software evolution, refactoring, and technical debt management.
- Integrate development and operations tools in collaborative environments.
- Design cloud-native applications using advanced software architectures.
- Analyze and apply systematic testing and validation strategies.

4. General Competences (*Outcomes*)

AG-C03) Individual and Team Work: Performs effectively as an individual and as a member or leader in diverse teams. (Usage)

AG-C09) Design and Development of Solutions: Designs, implements, and evaluates solutions for complex computing problems. (Usage)

5. Content

5.1 Verification and validation Concepts, 12 hr(s)

Competences: AG-C03,AG-C09

Topics: I: Verification and validation concepts (a) Verification: Are we building the thing right? (b) Validation: Did we build the right thing? **II:** Why testing matters: Does the component remain functional as the code evolves? **III:** Testing objectives (a) Usability (b) Reliability (c) Conformance to specification (d) Performance (e) Security

Learning Outcomes I: Explain why testing is important [Explain] **II:** Distinguish between program validation and verification [Distinguish] **III:** Describe different objectives of testing [Describe]

Bibliography: [Sommerville, 2015, Pressman and Maxim, 2019]

5.2 Testing Tools and Analysis, 3 hr(s)

Competences: AG-C03,AG-C09

Topics: I: Verification and validation tooling and automation (a) Static analysis (b) Code coverage (c) Fuzzing (d) Dynamic analysis and fault containment (sanitizers, etc.) (e) Fault logging and fault tracking **II:** Test planning and generation (a) Fault estimation and testing termination including defect seeding (b) Use of random and pseudo random numbers in testing **III:** Testing asynchronous, parallel, and concurrent systems **IV:** Verification and validation of non-code artifacts (documentation, training materials) **V:** Behavior-driven development (BDD) (a) Expressing acceptance criteria as Gherkin Given/When/Then scenarios (b) Executing BDD scenarios as automated end-to-end tests **VI:** Mutation testing and static code-quality analysis (a) Mutation testing to assess test-suite effectiveness (b) Static detection of code smells and excessive cyclomatic complexity

Learning Outcomes I: Describe and compare different tools for verification and validation [Describe] **II:** Automate the use of different tools in a small software project [Design] **III:** Explain how and when random numbers should be used in testing [Explain] **IV:** Describe approaches for fault estimation [Describe] **V:** Estimate the number of faults in a small software application based on fault density and fault seeding [Estimate] **VI:** Describe techniques and issues with testing asynchronous, concurrent, and parallel software [Describe] **VII:** Create a test plan for a medium-size code segment which contains asynchronous, concurrent, and/or parallel code, including a test matrix and generation of test data and inputs [Create] **VIII:** Describe techniques for the verification and validation of non-code artifacts [Describe] **IX:** Write Gherkin Given/When/Then scenarios for a feature's acceptance criteria and execute them as automated end-to-end tests [Write] **X:** Analyze a test suite's mutation score and refactor tests to kill surviving mutants [Analyze]

Bibliography: [Pressman and Maxim, 2019]

5.3 Performance Testing and Benchmarking, 3 hr(s)

Competences: AG-C03,AG-C09

Topics: **I:** Performance testing and benchmarking (a) Throughput and latency (b) Degradation under load (stress testing, FIFO vs LIFO handling of requests) (c) Speedup and scaling i. Amdahl's law ii. Gustafson's law iii. Soft and weak scaling (d) Identifying and measuring figures of merits (e) Common performance bottlenecks i. Compute-bound ii. Memory-bandwidth bound iii. Latency-bound (f) Statistical methods and best practices for benchmarking i. Estimation of uncertainty ii. Confidence intervals (g) Analysis and presentation (graphs, etc.) (h) Timing techniques

Learning Outcomes **I:** Describe throughput and latency and provide examples of each [Describe] **II:** Explain speedup and the different forms of scaling and how they are computed [Explain] **III:** Describe common performance bottlenecks [Describe] **IV:** Describe statistical methods and best practices for benchmarking software [Describe] **V:** Explain techniques for and challenges with measuring time when constructing a benchmark [Explain] **VI:** Identify the figures of merit, construct and run a benchmark, and statistically analyze and visualize the results for a small software project [Analyze]

Bibliography: [Bondi, 2015, Gregg, 2020]

5.4 API Compatibility and Versioning, 12 hr(s)

Competences: AG-C03,AG-C09

Topics: **I:** Hyrum's Law/The Law of Implicit Interfaces **II:** Backward compatibility (a) Compatibility is not a property of a single entity, it's a property of a relationship. (b) Backward compatibility needs to be evaluated in terms of provider + consumer(s) or with a well-specified model of what forms of compatibility a provider aspires to/promises. **III:** Versioning (a) Semantic Versioning (SemVer) (b) Trunk-based development

Learning Outcomes **I:** Identify both explicit and implicit behavior of an interface and identify potential risks from Hyrum's Law [Analyze] **II:** Identify changes that can be broadly considered "backward compatible," potentially with explicit statements about what usage is or is not supported [Analyze] **III:** Evaluate whether a proposed change is sufficiently safe given the versioning methodology in use for a given project [Evaluate]

Bibliography: [Sommerville, 2015]

5.5 Version Control and CI/CD, 12 hr(s)

Competences: AG-C03,AG-C09

Topics: **I:** Software configuration management and version control: Software Development Practices (a) Configuration in version control, reproducible builds/configuration. (b) Version control branching strategies. Development branches vs release branches. Trunk-based development. (c) Merging/rebasing strategies, when relevant. **II:** Release management. **III:** Testing tools including static and dynamic analysis tools. Software Development Practices, Information Flow and Non-Interference, Injection and Input Validation, Memory Safety and Types, Malware Analysis and Advanced Security **IV:** Software process automation: (a) Build systems - the value of fast, hermetic, reproducible builds, compare/contrast approaches to building a project. (b) Continuous Integration (CI) - the use of automation and automated tests to do preliminary validation that the current head/trunk revision builds and passes (basic) tests. (c) Continuous Deployment (CD) - the use of automation to automatically release every change that passes the automated tests to the production environment, ensuring frequent and reliable deliveries. (d) Dependency management - updating external/upstream dependencies, package management, SemVer.
V: Collaborative review workflow as a quality gate (a) Mandatory pull-request review before merging (b) Code ownership via CODEOWNERS-style rules (c) Conventional commit message

conventions (d) Branch-protection rules enforcing checks before merge **VI:** Dependency and supply-chain vulnerability scanning integrated into CI pipelines

Learning Outcomes I: Describe the difference between centralized and distributed software configuration management [Describe] **II:** Describe how version control can be used to help manage software release management [Describe] **III:** Identify configuration items and use a source code control tool in a small team-based project [Analyze] **IV:** Describe how available static and dynamic test tools can be integrated into the software development environment [Describe] **V:** Understand the use of CI/CD systems as a ground-truth for the state of the team's shared code (build and test success) [Explain] **VI:** Apply a pull-request review workflow with code-ownership rules and branch protection to enforce quality gates on a team project [Apply] **VII:** Use a dependency/supply-chain vulnerability scanner as part of a CI pipeline to detect vulnerable dependencies [Use]

Bibliography: [Kim et al., 2016]

5.6 Principles and Software Architecture, 12 hr(s)

Competences: AG-C03,AG-C09

Topics: I: System design principles. System Reliability (a) Levels of abstraction (e.g., architectural design and detailed design) (b) Separation of concerns (c) Information hiding (d) Coupling and cohesion **II:** Software architecture. System Reliability (a) Design paradigms i. Top-down functional decomposition/layered design ii. Data-oriented architecture iii. Object-oriented analysis and design iv. Event-driven design (b) Standard architectures (e.g., client-server and microservice architectures including REST discussions, n-layer, pipes-and-filters, Model View Controller) (c) Identifying component boundaries and dependencies **III:** Programming in the large vs programming in the small. System Reliability **IV:** Code smells and other indications of code quality, distinct from correctness. Security Design and Controls Engineering, Threat Analysis and Security Engineering, Trusted Computing and Privacy Engineering

Learning Outcomes I: Identify the standard software architecture of a given high-level design [Analyze] **II:** Select and use an appropriate design paradigm to design a simple software system and explain how system design principles have been applied in this design [Evaluate] **III:** Adapt a flawed system design to better follow principles such as separation of concerns or information hiding [Create] **IV:** Identify the dependencies among a set of software components in an architectural design [Analyze]

Bibliography: [Fowler, 2017]

6. Bibliography

[Bondi, 2015] Bondi, A. B. (2015). *Foundations of Software and System Performance Engineering: Process, Performance Modeling, Requirements, Testing, Scalability, and Practice*. Addison-Wesley, Upper Saddle River, NJ.

[Fowler, 2017] Fowler, M. (2017). *Refactoring: Improving the Design of Existing Code*. Addison-Wesley, 2nd edition.

[Gregg, 2020] Gregg, B. (2020). *Systems Performance: Enterprise and the Cloud*. Addison-Wesley Professional, Boston, MA, 2nd edition.

[Kim et al., 2016] Kim, G., Humble, J., Debois, P., and Willis, J. (2016). *The DevOps Handbook*. IT Revolution Press.

[Pressman and Maxim, 2019] Pressman, R. S. and Maxim, B. (2019). *Software Engineering: A Practitioner's Approach*. McGraw-Hill Education, 9th edition.

[Sommerville, 2015] Sommerville, I. (2015). *Software Engineering*. Pearson, 10th edition.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

CS393 Information systems (Elective)

1. General information

- **Semester**: 8th Sem. **Credits**: 3
- **Hour of this course**: **Theory**: 2 hours; **Laboratory**: 2 hours;
- **Prerrequisites**:
  - CS291 Software Engineering I (5th Sem-Pg. 360)

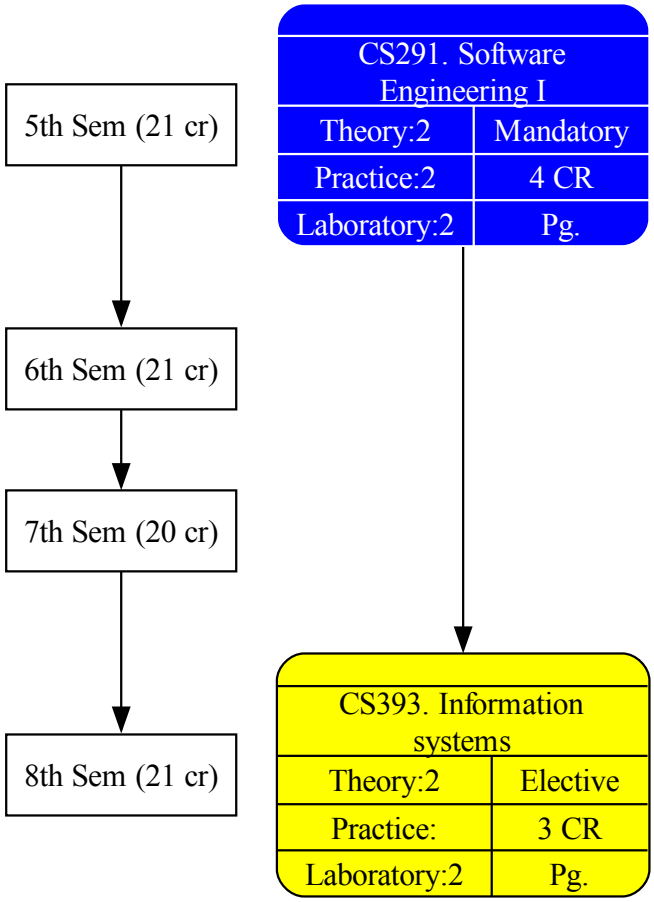

Figure 5.52: Connection Map. CS393 Information systems

2. Course Rationale

This course provides a comprehensive overview of how information systems support business processes and decision-making within organizations. It covers the strategic role of information technology, the lifecycle of information systems, data management strategies, and the integration of enterprise applications to achieve competitive advantages in a globalized digital economy.

3. Goals of the Course

- Understand the fundamental components and types of Information Systems.

- Analyze the role of IS in organizational strategy and decision-making.
- Evaluate emerging technologies and their impact on business models.

4. General Competences (*Outcomes*)

AG-C03) Individual and Team Work: Performs effectively as an individual and as a member or leader in diverse teams. (Usage)

AG-C09) Design and Development of Solutions: Designs, implements, and evaluates solutions for complex computing problems. (Usage)

5. Content

5.1 Foundations of Information Systems, 12 hr(s)

Competences: AG-C03,AG-C09

Topics: **I:** Components of Information Systems: Hardware, Software, Data, Networks. **II:** Types of IS: TPS, MIS, DSS, and ESS. **III:** Competitive advantage through Information Technology. **IV:** Business processes and Information Systems integration.

Learning Outcomes **I:** Identify the core components of an information system [Familiarity]. **II:** Explain how IS supports various levels of organizational hierarchy [Usage]. **III:** Analyze a business case to identify strategic uses of IT [Assessment].

Bibliography: [Laudon and Laudon, 2021, Stair and Reynolds, 2017]

5.2 Data Management and Business Intelligence, 12 hr(s)

Competences: AG-C03,AG-C09

Topics: **I:** Database management systems and data warehousing. **II:** Big Data analytics in a business context. **III:** Business Intelligence (BI) and Data Visualization tools. **IV:** Knowledge Management Systems.

Learning Outcomes **I:** Design a basic data architecture for a business requirement [Usage]. **II:** Evaluate the importance of data quality and governance [Assessment].

Bibliography: [Laudon and Laudon, 2021, Valacich and Schneider, 2021]

5.3 Enterprise Systems and E-Commerce, 12 hr(s)

Competences: AG-C03,AG-C09

Topics: **I:** Enterprise Resource Planning (ERP) systems. **II:** Supply Chain Management (SCM) and Customer Relationship Management (CRM). **III:** E-commerce models: B2B, B2C, and C2C. **IV:** Mobile commerce and digital payment systems.

Learning Outcomes **I:** Describe the functional areas of an ERP system [Familiarity]. **II:** Assess the impact of e-commerce on traditional business models [Assessment].

Bibliography: [Stair and Reynolds, 2017, Valacich and Schneider, 2021]

5.4 IS Development and Security, 12 hr(s)

Competences: AG-C03,AG-C09

Topics: **I:** Systems Development Life Cycle (SDLC) and Agile methodologies. **II:** Outsourcing and cloud-based IS services. **III:** Information security threats and risk management. **IV:** Ethical and social issues in Information Systems.

Learning Outcomes I: Compare different system development methodologies [Usage]. **II:** Formulate ethical guidelines for data privacy and security [Assessment].

Bibliography: [Laudon and Laudon, 2021, Stair and Reynolds, 2017]

6. Bibliography

[Laudon and Laudon, 2021] Laudon, K. C. and Laudon, J. P. (2021). *Management Information Systems: Managing the Digital Firm*. Pearson, 17th edition.

[Stair and Reynolds, 2017] Stair, R. and Reynolds, G. (2017). *Principles of Information Systems*. Cengage Learning, 13th edition.

[Valacich and Schneider, 2021] Valacich, J. and Schneider, C. (2021). *Information Systems Today: Managing in the Digital World*. Pearson, 9th edition.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

CS394 Software Architecture and DevOps for Cloud Software (Elective)

1. General information

- **Semester**: 8th Sem. **Credits**: 3
- **Hour of this course**: **Theory**: 2 hours; **Practice**: 2 hours;
- **Prerrequisites**:
  - CS292 Software Engineering II (7th Sem-Pg. 401)

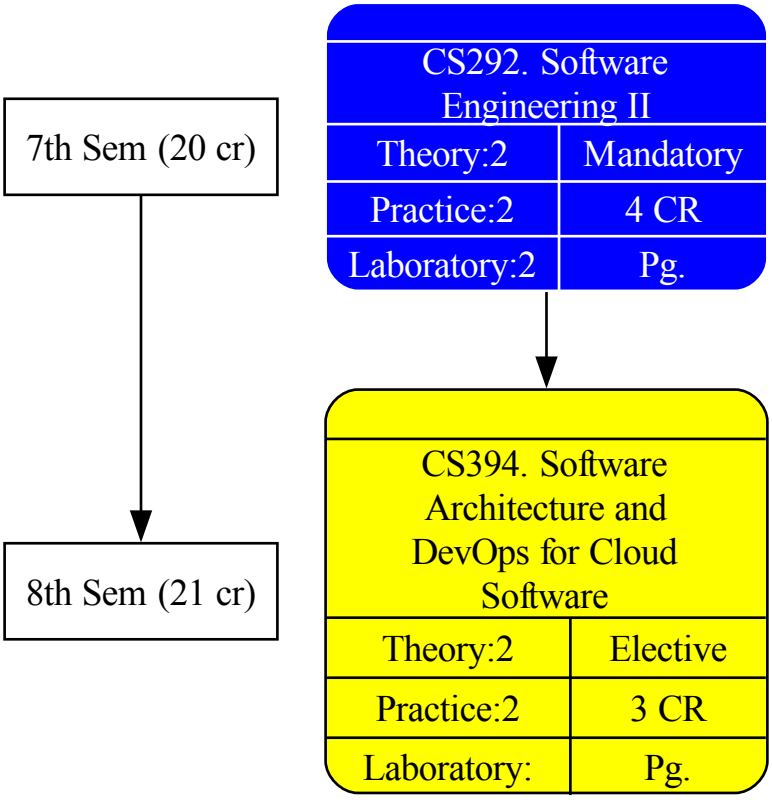

Figure 5.53: Connection Map. CS394 Software Architecture and DevOps for Cloud Software

2. Course Rationale

Modern software design requires combining solid architectural principles with agile continuous delivery practices and cloud operations. This course integrates three areas that are inseparable in today's industry: software architecture, DevOps, and cloud computing.

Students will learn to design systems at scale using well-established architectural patterns, to automate build, test, and deployment cycles through CI/CD pipelines, and to provision and operate infrastructure on public cloud platforms using containers, orchestration, and infrastructure-as-code tools.

The course places special emphasis on the reliability and observability of distributed systems, covering metrics, distributed traces, error budgets, and site reliability engineering (SRE) practices as applied in high-performing technology organizations.

3. Goals of the Course

- Design scalable and maintainable software systems by applying modern architectural patterns.
- Implement continuous integration and continuous delivery (CI/CD) pipelines to accelerate and secure the software lifecycle.
- Deploy and manage containerized applications on public cloud platforms using industry-standard tools.
- Provision cloud infrastructure in a reproducible and versioned manner through infrastructure as code (IaC).
- Apply observability and SRE practices to ensure the reliability of systems in production.

4. General Competences (*Outcomes*)

AG-C09) Design and Development of Solutions: Designs, implements, and evaluates solutions for complex computing problems. (Assessment)

AG-C11) Use of Tools: Applies modern computing tools in problem solving. (Assessment)

AG-C12) Applies computer science theory and software development fundamentals to produce computer-based solutions. (Usage)

5. Content

5.1 Principles and Software Architecture, 8 hr(s)

Competences: AG-C09,AG-C12

Topics: **I:** System design principles. System Reliability (a) Levels of abstraction (e.g., architectural design and detailed design) (b) Separation of concerns (c) Information hiding (d) Coupling and cohesion **II:** Software architecture. System Reliability (a) Design paradigms i. Top-down functional decomposition/layered design ii. Data-oriented architecture iii. Object-oriented analysis and design iv. Event-driven design (b) Standard architectures (e.g., client-server and microservice architectures including REST discussions, n-layer, pipes-and-filters, Model View Controller) (c) Identifying component boundaries and dependencies **III:** Programming in the large vs programming in the small. System Reliability **IV:** Structural and behavioral models of software designs **V:** Design modeling, for instance with class diagrams, entity relationship diagrams, or sequence diagrams **VI:** API design principles (a) Consistency i. Consistent APIs are easier to learn and less error-prone ii. Consistency is both internal (between different portions of the API) and external (following common API patterns) (b) Composability (c) Documenting contracts i. API operations should describe their effect on the system, but not generally their implementation ii. Preconditions, postconditions, and invariants (d) Expandability (e) Error reporting i. Errors should be clear, predictable, and actionable ii. Input that does not match the contract should produce an error iii. Errors that can be reliably managed without reporting should be managed

Learning Outcomes **I:** Identify the standard software architecture of a given high-level design [Analyze] **II:** Identify the dependencies among a set of software components in an architectural design [Analyze] **III:** Select and use an appropriate design paradigm to design a simple software system and explain how system design principles have been applied in this design [Evaluate] **IV:** Translate a natural language software design into class diagrams [Translate]

Bibliography: [Bass et al., 2021, Richards and Ford, 2020]

5.2 Architectural Patterns and Styles, 8 hr(s)

Competences: AG-C09,AG-C12

Topics: **I:** Evaluating design tradeoffs (e.g., efficiency vs reliability, security vs usability) **II:** Architectural styles: layered, event-driven, microkernel, service-oriented, pipeline. **III:** Quality attributes: availability, scalability, performance, security, maintainability. **IV:** Requirement traceability (a) Understanding which requirements are satisfied by a design **V:** Measurement and analysis of design quality

Learning Outcomes **I:** Contrast two software designs across different qualities, such as efficiency or usability [Contrast] **II:** Identify which requirements are satisfied by a provided software design [Analyze] **III:** Adapt a flawed system design to better follow principles such as separation of concerns or information hiding [Create] **IV:** Select the most appropriate architectural style given the quality attributes required by a system. [Assessment]

Bibliography: [Richards and Ford, 2020]

5.3 Microservices Architecture, 6 hr(s)

Competences: AG-C09,AG-C11

Topics: **I:** Microservices principles: cohesion, loose coupling, and service boundaries. **II:** Monolith decomposition: domain-driven design (DDD). **III:** API design principles (a) Consistency i. Consistent APIs are easier to learn and less error-prone ii. Consistency is both internal (between different portions of the API) and external (following common API patterns) (b) Composability (c) Documenting contracts i. API operations should describe their effect on the system, but not generally their implementation ii. Preconditions, postconditions, and invariants (d) Expandability (e) Error reporting i. Errors should be clear, predictable, and actionable ii. Input that does not match the contract should produce an error iii. Errors that can be reliably managed without reporting should be managed **IV:** API Gateway and service mesh. **V:** Resilience patterns: Circuit Breaker, Retry, Bulkhead, Timeout. **VI:** Data Stores (a) Cooperatively maintained data structures implementing maps and related ADTs (b) Varieties: Owned, shared, sharded, replicated, immutable, versioned

Learning Outcomes **I:** Design an API for a single component of a large software system, including identifying and documenting each operation's invariants, contract, and error conditions [Design] **II:** Evaluate an API description in terms of consistency, composability, and expandability [Evaluate] **III:** Design a microservices-based system by decomposing a monolithic application into independent services. [Usage] **IV:** Apply resilience patterns to tolerate failures in distributed systems. [Usage]

Bibliography: [Newman, 2021]

5.4 Containers and Docker, 6 hr(s)

Competences: AG-C11,AG-C12

Topics: **I:** Containers and their comparison to virtual machines. **II:** Types of virtualizations (including hardware/software, OS, server, service, network). See also: Latency, Cache and Memory Hierarchy, Virtualization and Isolation **III:** Portable virtualization; emulation vs isolation. See also: Latency, Cache and Memory Hierarchy, Virtualization and Isolation **IV:** Virtual machines and container escapes, dangers from a security perspective. See also: Security Design and Controls Engineering, Threat Analysis and Security Engineering, Trusted Computing and Privacy Engineering **V:** Dockerfile: best practices for building efficient and secure images. **VI:** Multi-container application composition with docker-compose.

Learning Outcomes **I:** Explain the difference between emulation and isolation [Explain] **II:** Evaluate virtualization tradeoffs [Evaluate] **III:** Build and deploy containerized applications using Docker. [Usage] **IV:** Apply security best practices when building Docker images. [Usage]

Bibliography: [Burns et al., 2022]

5.5 Container Orchestration with Kubernetes, 6 hr(s)

Competences: AG-C09,AG-C11

Topics: **I:** Kubernetes architecture: control plane and worker nodes. **II:** Core objects: Pods, Deployments, Services, ConfigMaps, Secrets. **III:** Horizontal Pod Autoscaling (HPA) and self-healing. **IV:** Dependencies (a) Initiation or progress of one activity may be dependent on other activities, so as to avoid race conditions, ensure termination, or meet other requirements (b) Ensuring progress by avoiding dependency cycles, using monotonic conditions, removing inessential dependencies **V:** Package management with Helm.

Learning Outcomes **I:** Deploy and manage containerized applications on a Kubernetes cluster. [Usage] **II:** Configure auto-scaling and self-healing for cloud-native workloads. [Usage] **III:** Modify a concurrent system to use a more scalable, reliable, or available data store [Create]

Bibliography: [Burns et al., 2022]

5.6 Version Control and CI/CD, 6 hr(s)

Competences: AG-C11,AG-C12

Topics: **I:** Software configuration management and version control: Software Development Practices (a) Configuration in version control, reproducible builds/configuration. (b) Version control branching strategies. Development branches vs release branches. Trunk-based development. (c) Merging/rebasing strategies, when relevant. **II:** Release management. **III:** Software process automation: (a) Build systems - the value of fast, hermetic, reproducible builds, compare/contrast approaches to building a project. (b) Continuous Integration (CI) - the use of automation and automated tests to do preliminary validation that the current head/trunk revision builds and passes (basic) tests. (c) Continuous Deployment (CD) - the use of automation to automatically release every change that passes the automated tests to the production environment, ensuring frequent and reliable deliveries. (d) Dependency management - updating external/upstream dependencies, package management, SemVer. **IV:** Testing tools including static and dynamic analysis tools. Software Development Practices, Information Flow and Non-Interference, Injection and Input Validation, Memory Safety and Types, Malware Analysis and Advanced Security **V:** Tool integration concepts and mechanisms. Software Development Practices **VI:** DevOps culture: flow, feedback, and continuous learning.

Learning Outcomes **I:** Describe the difference between centralized and distributed software configuration management [Describe] **II:** Describe how version control can be used to help manage software release management [Describe] **III:** Understand the use of CI/CD systems as a ground-truth for the state of the team's shared code (build and test success) [Explain] **IV:** Demonstrate the capability to use software tools in support of the development of a software product of medium size [Demonstrate]

Bibliography: [Kim et al., 2021]

5.7 Infrastructure as Code (IaC), 6 hr(s)

Competences: AG-C11,AG-C12

Topics: **I:** IaC principles: reproducibility, idempotency, and infrastructure versioning. **II:** Declarative vs. imperative approaches. **III:** Terraform: providers, resources, modules, and state management. **IV:** Cloud-native IaC tools: AWS CloudFormation, Azure ARM Templates, GCP Deployment Manager. **V:** Software configuration management and version control: Software Development Practices (a) Configuration in version control, reproducible builds/configuration. (b) Version control branching strategies. Development branches vs release branches. Trunk-based development. (c) Merging/rebasing strategies, when relevant.

Learning Outcomes I: Write and apply Terraform configurations to provision cloud infrastructure. [Usage] **II:** Identify configuration items and use a source code control tool in a small team-based project [Analyze] **III:** Compare IaC tools based on their state management and modularity capabilities. [Assessment]

Bibliography: [Morris, 2021, Kim et al., 2021]

5.8 Cloud Platforms and Native Services, 6 hr(s)

Competences: AG-C09,AG-C11

Topics: I: Cloud service models: IaaS, PaaS, SaaS, FaaS. **II:** Distributed application paradigms (e.g., client/server, peer-to-peer, cloud, edge, and fog). See also: Communication, Coordination **III:** Major providers: AWS, Google Cloud, Azure — core services. **IV:** Compute services: virtual machines, serverless functions, managed containers. **V:** Storage services: object, block, and managed databases. **VI:** Cloud networking: VPCs, load balancers, CDN.

Learning Outcomes I: Analyze the needs of specific networked application demands [Analyze] **II:** Compare IaaS, PaaS, SaaS, and FaaS models to select the most appropriate one for a given architecture. [Assessment] **III:** Design a cloud-native application using managed services from a major cloud provider. [Usage]

Bibliography: [Kim et al., 2021, Burns et al., 2022]

5.9 Observability: Metrics, Logs, and Traces, 6 hr(s)

Competences: AG-C11,AG-C12

Topics: I: Measurement-based analysis of software reliability - telemetry, monitoring and alerting, dashboards, release qualification metrics, etc. **II:** The three pillars of observability: metrics, logs, and distributed traces. **III:** Distributed tracing and correlation identifiers. **IV:** Software reliability engineering practices - including reviews, testing, practical model checking **V:** Alerting strategies and on-call practices.

Learning Outcomes I: Describe how to determine the level of reliability required by a software system [Describe] **II:** Identify methods that will lead to the realization of a software architecture that achieves a specified level of reliability [Analyze] **III:** Implement observability in a cloud application using metrics, logs, and distributed traces. [Usage]

Bibliography: [Kim et al., 2021]

5.10 Site Reliability Engineering (SRE), 6 hr(s)

Competences: AG-C09,AG-C12

Topics: I: Concept of reliability as probability of failure or mean time between failures, and faults as cause of failures **II:** Compare the "error budget" approach to reliability with the "error-free" approach and identify domains where each is relevant. **III:** Software fault tolerance techniques and models (a) Contextual differences in fault tolerance (e.g., crashing a flight critical system is strongly avoided, crashing a data processing system before corrupt data is written to storage is highly valuable) **IV:** Reliability through redundancy: (a) check and retry Reliability Support, Fault tolerance (b) redundant encoding (error correction codes, CRC, FEC, RAID) Memory Hierarchy, Distributed Databases/Cloud Computing, Reliability Support (c) duplication/mirroring/replicas Distributed Databases/Cloud Computing **V:** Software reliability, system reliability and failure behavior **VI:** Service Level Indicators (SLI), Objectives (SLO), and Agreements (SLA).

Learning Outcomes I: Identify ways to apply redundancy to achieve fault tolerance [Analyze] **II:** Identify single-point-of-failure (SPF) dependencies in a system design [Analyze] **III:** Explain the problems that exist in achieving very high levels of reliability [Explain] **IV:** Define SLIs, SLOs, and error budgets for a service in production. [Usage]

Bibliography: [Kim et al., 2021]

6. Bibliography

[Bass et al., 2021] Bass, L., Clements, P., and Kazman, R. (2021). *Software Architecture in Practice*. Addison-Wesley Professional, 4th edition.

[Burns et al., 2022] Burns, B., Beda, J., Hightower, K., and Evenson, L. (2022). *Kubernetes: Up and Running*. O'Reilly Media, 3rd edition.

[Kim et al., 2021] Kim, G., Humble, J., Debois, P., Willis, J., and Forsgren, N. (2021). *The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations*. IT Revolution Press, 2nd edition.

[Morris, 2021] Morris, K. (2021). *Infrastructure as Code: Dynamic Systems for the Cloud Age*. O'Reilly Media, 2nd edition.

[Newman, 2021] Newman, S. (2021). *Building Microservices: Designing Fine-Grained Systems*. O'Reilly Media, 2nd edition.

[Richards and Ford, 2020] Richards, M. and Ford, N. (2020). *Fundamentals of Software Architecture: An Engineering Approach*. O'Reilly Media, 1st edition.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

CS395 Software Quality (Elective)

1. General information

- **Semester**: 8th Sem. **Credits**: 3
- **Hour of this course**: **Theory**: 2 hours; **Practice**: 2 hours;
- **Prerrequisites**:
  - CS292 Software Engineering II (7th Sem-Pg. 401)

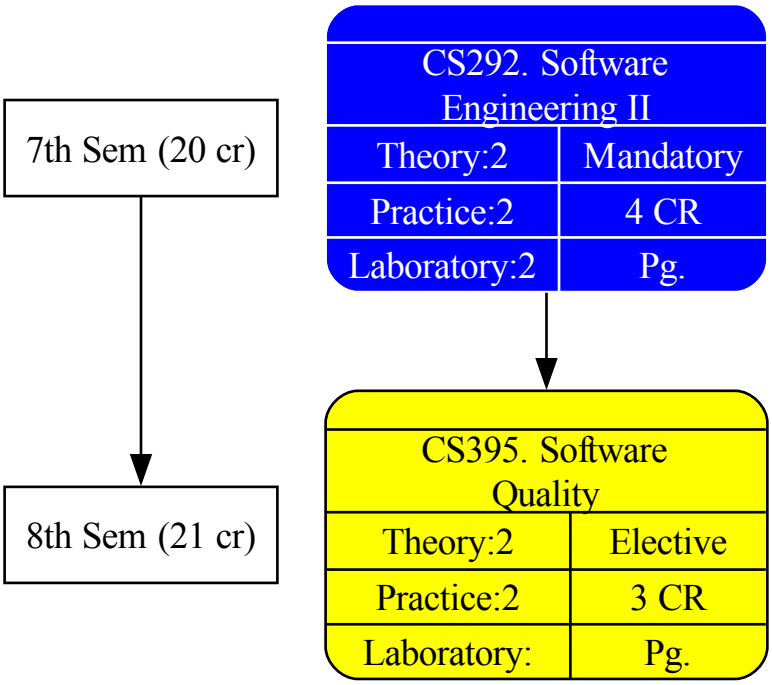

Figure 5.54: Connection Map. CS395 Software Quality

2. Course Rationale

Delivering software that is merely "functional" is no longer sufficient for industry: modern organizations demand systems that are reliable, secure, maintainable, and built through disciplined, auditable engineering processes. This course teaches software quality as a first-class, hands-on engineering discipline practiced through the same tools and workflows used by professional teams on GitHub: branch protection, mandatory code review, automated testing at every level, continuous integration/continuous delivery (CI/CD), static analysis, and production observability.

Students progress from foundational quality models through advanced collaborative-development workflows, automated testing strategies (unit, integration, end-to-end, and behavior-driven), CI/CD pipelines, static analysis and refactoring, and production-quality concerns such as supply-chain security, performance, and reliability engineering. The course closes with an emphasis on DevOps culture – the shared team practices and values (Definition of Done, blameless postmortems, continuous learning) that sustain quality at scale.

3. Goals of the Course

- Evaluate software quality using an internationally recognized quality-characteristics model and reason about the cost of poor quality and technical debt.
- Apply a professional, review-gated Git/GitHub workflow, including branch protection, CODEOWNERS, and conventional commits.
- Design and implement unit, integration, end-to-end, and behavior-driven test suites, including test-driven development.
- Build and operate CI/CD pipelines that enforce automated quality gates, dependency scanning, and semantic versioning.
- Apply static analysis, mutation testing, and refactoring techniques to identify and remove code smells and technical debt.
- Analyze production-quality concerns, including performance testing and reliability/observability practices.
- Adopt DevOps culture practices – Definition of Done, blameless postmortems, and continuous team learning – to sustain quality.

4. General Competences (*Outcomes*)

AG-C09) Design and Development of Solutions: Designs, implements, and evaluates solutions for complex computing problems. (Assessment)

AG-C11) Use of Tools: Applies modern computing tools in problem solving. (Assessment)

AG-C12) Applies computer science theory and software development fundamentals to produce computer-based solutions. (Usage)

5. Content

5.1 Software Design Quality and Evaluation, 4 hr(s)

Competences: AG-C09,AG-C12

Topics: **I:** Data design Data Modeling (a) Data structures (b) Storage systems **II:** Requirement traceability (a) Understanding which requirements are satisfied by a design **III:** Design modeling, for instance with class diagrams, entity relationship diagrams, or sequence diagrams **IV:** Measurement and analysis of design quality **V:** Principles of secure design and coding Security Design and Controls Engineering, Threat Analysis and Security Engineering, Trusted Computing and Privacy Engineering (a) Principle of least privilege (b) Principle of fail-safe defaults (c) Principle of psychological acceptability **VI:** Evaluating design tradeoffs (e.g., efficiency vs reliability, security vs usability) **VII:** ISO/IEC 25010 quality characteristics model and cost of poor quality (a) Quality characteristics: functional suitability, performance efficiency, compatibility, usability, reliability, security, maintainability, portability (b) Cost of poor quality and technical debt as a business/economic concern, not just a technical one

Learning Outcomes **I:** Design a set of data structures to implement a provided API surface [Design] **II:** Identify which requirements are satisfied by a provided software design [Analyze] **III:** Translate a natural language software design into class diagrams [Translate] **IV:** Adapt a flawed system design to better follow the principles of least privilege and fail-safe defaults [Create] **V:** Contrast two software designs across different qualities, such as efficiency or usability [Contrast] **VI:** Evaluate a software system against the ISO/IEC 25010 quality characteristics and estimate the business cost of poor quality/technical debt [Evaluate]

Bibliography: [International Organization for Standardization, 2011, Sommerville, 2015]

5.2 Version Control and CI/CD, 12 hr(s)

Competences: AG-C11,AG-C12

Topics: **I:** Software configuration management and version control: Software Development Practices (a) Configuration in version control, reproducible builds/configuration. (b) Version control branching strategies. Development branches vs release branches. Trunk-based development. (c) Merging/rebasing strategies, when relevant. **II:** Release management. **III:** Testing tools including static and dynamic analysis tools. Software Development Practices, Information Flow and Non-Interference, Injection and Input Validation, Memory Safety and Types, Malware Analysis and Advanced Security **IV:** Software process automation: (a) Build systems - the value of fast, hermetic, reproducible builds, compare/contrast approaches to building a project. (b) Continuous Integration (CI) - the use of automation and automated tests to do preliminary validation that the current head/trunk revision builds and passes (basic) tests. (c) Continuous Deployment (CD) - the use of automation to automatically release every change that passes the automated tests to the production environment, ensuring frequent and reliable deliveries. (d) Dependency management - updating external/upstream dependencies, package management, SemVer. **V:** Collaborative review workflow as a quality gate (a) Mandatory pull-request review before merging (b) Code ownership via CODEOWNERS-style rules (c) Conventional commit message conventions (d) Branch-protection rules enforcing checks before merge **VI:** Dependency and supply-chain vulnerability scanning integrated into CI pipelines

Learning Outcomes **I:** Describe the difference between centralized and distributed software configuration management [Describe] **II:** Describe how version control can be used to help manage software release management [Describe] **III:** Identify configuration items and use a source code control tool in a small team-based project [Analyze] **IV:** Describe how available static and dynamic test tools can be integrated into the software development environment [Describe] **V:** Understand the use of CI/CD systems as a ground-truth for the state of the team's shared code (build and test success) [Explain] **VI:** Apply a pull-request review workflow with code-ownership rules and branch protection to enforce quality gates on a team project [Apply] **VII:** Use a dependency/supply-chain vulnerability scanner as part of a CI pipeline to detect vulnerable dependencies [Use]

Bibliography: [Kim et al., 2021]

5.3 Test Planning and Types, 8 hr(s)

Competences: AG-C09

Topics: **I:** Test kinds (a) Unit (b) Integration (c) Validation (d) System **II:** Stylistic differences between tests and production code: DAMP vs DRY - more duplication is warranted in test code. **III:** Test planning and generation (a) Test case generation, from formal models, specifications, etc. (b) Test coverage i. Test matrices ii. Code coverage - how much of the code is tested? iii. Environment coverage - how many hardware architectures, operating systems, browsers, etc. are tested? (c) Test data and inputs

Learning Outcomes **I:** Compare and contrast the different types and levels of testing (regression, unit, integration, systems, and acceptance) [Compare] **II:** Describe techniques for creating a test plan and generating test cases [Describe] **III:** Create a test plan for a medium-size code segment which includes a test matrix and generation of test data and inputs [Create] **IV:** Implement a test plan for a medium-size code segment [Implement]

Bibliography: [Sommerville, 2015]

5.4 Advanced Testing Practices, 4 hr(s)

Competences: AG-C11

Topics: **I:** Test development (a) Test-driven development (b) Object oriented testing, mocking, and dependency injection (c) Opaque-box (previously, black-box) and transparent-box (previously, white-box) testing techniques (d) Test tooling, including code coverage, static analysis, and fuzzing **II:** Verification and validation in the development cycle (a) Code reviews (b) Test

automation, including automation of tooling (c) Pre-commit and post-commit testing (d) Trade-offs between test coverage and throughput/latency of testing (e) Defect tracking and prioritization: reproducibility of reported defects **III:** Domain specific verification and validation challenges (a) Performance testing and benchmarking (b) Asynchrony, parallelism, and concurrency (c) Safety-critical (d) Numeric

Learning Outcomes **I:** Identify the fundamental principles of test-driven development methods and explain the role of automated testing in these methods [Analyze] **II:** Discuss issues involving the testing of object-oriented software [Debate] **III:** Describe mocking and dependency injection and their application [Describe] **IV:** Undertake, as part of a team activity, a code review of a medium-size code segment [Evaluate] **V:** Describe the role that tools can play in the validation of software [Describe] **VI:** Automate the testing in a small software project [Design] **VII:** Explain the roles, pros, and cons of pre-commit and post-commit testing [Explain] **VIII:** Discuss the tradeoffs between test coverage and test throughput/latency and how this can impact verification [Debate] **IX:** Use a defect tracking tool to manage software defects in a small software project [Use] **X:** Discuss the limitations of testing in certain domains [Debate]

Bibliography: [Beck, 2002]

5.5 Testing Tools and Analysis, 8 hr(s)

Competences: AG-C11

Topics: **I:** Verification and validation tooling and automation (a) Static analysis (b) Code coverage (c) Fuzzing (d) Dynamic analysis and fault containment (sanitizers, etc.) (e) Fault logging and fault tracking **II:** Test planning and generation (a) Fault estimation and testing termination including defect seeding (b) Use of random and pseudo random numbers in testing **III:** Testing asynchronous, parallel, and concurrent systems **IV:** Verification and validation of non-code artifacts (documentation, training materials) **V:** Behavior-driven development (BDD) (a) Expressing acceptance criteria as Gherkin Given/When/Then scenarios (b) Executing BDD scenarios as automated end-to-end tests **VI:** Mutation testing and static code-quality analysis (a) Mutation testing to assess test-suite effectiveness (b) Static detection of code smells and excessive cyclomatic complexity

Learning Outcomes **I:** Describe and compare different tools for verification and validation [Describe] **II:** Automate the use of different tools in a small software project [Design] **III:** Explain how and when random numbers should be used in testing [Explain] **IV:** Describe approaches for fault estimation [Describe] **V:** Estimate the number of faults in a small software application based on fault density and fault seeding [Estimate] **VI:** Describe techniques and issues with testing asynchronous, concurrent, and parallel software [Describe] **VII:** Create a test plan for a medium-size code segment which contains asynchronous, concurrent, and/or parallel code, including a test matrix and generation of test data and inputs [Create] **VIII:** Describe techniques for the verification and validation of non-code artifacts [Describe] **IX:** Write Gherkin Given/When/Then scenarios for a feature's acceptance criteria and execute them as automated end-to-end tests [Write] **X:** Analyze a test suite's mutation score and refactor tests to kill surviving mutants [Analyze]

Bibliography: [Wynne and Hellesøy, 2012]

5.6 API Compatibility and Versioning, 4 hr(s)

Competences: AG-C11,AG-C12

Topics: **I:** Hyrum's Law/The Law of Implicit Interfaces **II:** Backward compatibility (a) Compatibility is not a property of a single entity, it's a property of a relationship. (b) Backward compatibility needs to be evaluated in terms of provider + consumer(s) or with a well-specified model of what forms of compatibility a provider aspires to/promises. **III:** Versioning (a) Semantic Versioning (SemVer) (b) Trunk-based development

Learning Outcomes **I:** Identify both explicit and implicit behavior of an interface and identify potential risks from Hyrum's Law [Analyze] **II:** Identify changes that can be broadly considered "backward compatible," potentially with explicit statements about what usage is or is not supported [Analyze] **III:** Evaluate whether a proposed change is sufficiently safe given the versioning methodology in use for a given project [Evaluate]

Bibliography: [Sommerville, 2015]

5.7 Refactoring Techniques, 4 hr(s)

Competences: AG-C09,AG-C12

Topics: **I:** Refactoring (a) Standard refactoring patterns (rename, inline, outline, etc.) (b) Use of refactoring tools in IDE (c) Application of static-analysis tools (to identify code in need of refactoring, generate changes, etc.) (d) Value of refactoring as a remedy for technical debt **II:** "Large Scale" Refactoring - techniques when a refactoring change is too large to commit safely (large projects), or when it is impossible to synchronize change between provider + all consumers (multiple repositories, consumers with private code). (a) Express both old and new APIs so that they can co-exist. (b) Minimize the size of behavior changes. (c) Why these techniques are required, (e.g., "API consumers I can see" vs "consumers I can't see").

Learning Outcomes **I:** Consider inputs from static analysis tools and/or Software Design principles to identify code in need of refactoring [Analyze] **II:** Refactor the implementation of an interface to improve design, clarity, etc. with minimal/zero impact on existing users [Create] **III:** Plan a complex multi-step refactoring to change default behavior of an API safely [Plan]

Bibliography: [Fowler, 2017]

5.8 Performance Testing and Benchmarking, 4 hr(s)

Competences:

Topics: **I:** Performance testing and benchmarking (a) Throughput and latency (b) Degradation under load (stress testing, FIFO vs LIFO handling of requests) (c) Speedup and scaling i. Amdahl's law ii. Gustafson's law iii. Soft and weak scaling (d) Identifying and measuring figures of merits (e) Common performance bottlenecks i. Compute-bound ii. Memory-bandwidth bound iii. Latency-bound (f) Statistical methods and best practices for benchmarking i. Estimation of uncertainty ii. Confidence intervals (g) Analysis and presentation (graphs, etc.) (h) Timing techniques

Learning Outcomes **I:** Describe throughput and latency and provide examples of each [Describe] **II:** Explain speedup and the different forms of scaling and how they are computed [Explain] **III:** Describe common performance bottlenecks [Describe] **IV:** Describe statistical methods and best practices for benchmarking software [Describe] **V:** Explain techniques for and challenges with measuring time when constructing a benchmark [Explain] **VI:** Identify the figures of merit, construct and run a benchmark, and statistically analyze and visualize the results for a small software project [Analyze]

Bibliography: [Bondi, 2015, Gregg, 2020]

5.9 Reliability Engineering, 4 hr(s)

Competences: AG-C11,AG-C12

Topics: **I:** Software reliability models **II:** Software fault tolerance techniques and models (a) Contextual differences in fault tolerance (e.g., crashing a flight critical system is strongly avoided, crashing a data processing system before corrupt data is written to storage is highly valuable) **III:** Software reliability engineering practices - including reviews, testing, practical model checking **IV:** Identification of dependent and independent failure domains, and their impact on

system reliability **V:** Measurement-based analysis of software reliability - telemetry, monitoring and alerting, dashboards, release qualification metrics, etc.

Learning Outcomes **I:** Demonstrate the ability to apply multiple methods to develop reliability estimates for a software system [Demonstrate] **II:** Identify methods that will lead to the realization of a software architecture that achieves a specified level of reliability [Analyze] **III:** Identify ways to apply redundancy to achieve fault tolerance [Analyze] **IV:** Identify single-point-of-failure (SPF) dependencies in a system design [Analyze]

Bibliography: [Kim et al., 2021, Forsgren et al., 2018]

5.10 Large-Scale Construction and Process, 4 hr(s)

Competences: AG-C12

Topics: **I:** Larger-scale testing (a) Test doubles (stubs, mocks, fakes) (b) Dependency injection **II:** Work sequencing, including dependency identification, milestones, and risk retirement (a) Dependency identification: Identifying the dependencies between different tasks (b) Milestones: A collection of tasks that serve as a marker of progress when completed. Ideally, the milestone encompasses a useful unit of functionality. (c) Risk retirement: Identifying what elements of a project are risky and prioritizing completing tasks that address those risks. **III:** Potential security problems in programs Information Flow and Non-Interference, Injection and Input Validation, Memory Safety and Types, Malware Analysis and Advanced Security (a) Buffer and other types of overflows (b) Race conditions (c) Improper initialization, including choice of privileges (d) Input validation **IV:** Documentation (autogenerated) **V:** Development context: "green field" vs existing code base (a) Change impact analysis (b) Change actualization **VI:** Release management **VII:** DevOps practices (a) Definition of Done as a shared quality bar for completed work (b) Blameless postmortems for learning from incidents/failures (c) Team quality culture: shared ownership, feedback, and continuous learning

Learning Outcomes **I:** Rewrite a simple program to remove common vulnerabilities, such as buffer overflows, integer overflows and race conditions [Apply] **II:** Write a software component that performs some non-trivial task and is resilient to input and run-time errors [Write]

Bibliography: [Forsgren et al., 2018]

5.11 Capstone: End-to-End Quality Pipeline Demonstration, 8 hr(s)

Competences: AG-C09

Topics: **I:** Integration of all course practices into a single live repository: branch protection, mandatory PR review, automated test suites (unit, integration, E2E/BDD), CI/CD pipeline with quality gates, dependency scanning, static analysis, and observability. **II:** Live repository review: demonstration and defense of the team's quality pipeline in front of peers/instructor. **III:** Retrospective and blameless postmortem of the project's quality journey.

Learning Outcomes **I:** Integrate version control, automated testing, CI/CD, static analysis, and observability into a single, end-to-end quality pipeline for a real repository. [Usage] **II:** Defend the design and effectiveness of a team's quality pipeline through a live repository review. [Usage] **III:** Conduct a blameless postmortem to identify process improvements for future projects. [Assessment]

Bibliography: [Forsgren et al., 2018]

6. Bibliography

[Beck, 2002] Beck, K. (2002). *Test-Driven Development: By Example.* Addison-Wesley.

[Bondi, 2015] Bondi, A. B. (2015). *Foundations of Software and System Performance Engineering: Process, Performance Modeling, Requirements, Testing, Scalability, and Practice*. Addison-Wesley, Upper Saddle River, NJ.

[Forsgren et al., 2018] Forsgren, N., Humble, J., and Kim, G. (2018). *Accelerate: The Science of Lean Software and DevOps: Building and Scaling High Performing Technology Organizations*. IT Revolution Press.

[Fowler, 2017] Fowler, M. (2017). *Refactoring: Improving the Design of Existing Code*. Addison-Wesley, 2nd edition.

[Gregg, 2020] Gregg, B. (2020). *Systems Performance: Enterprise and the Cloud*. Addison-Wesley Professional, Boston, MA, 2nd edition.

[International Organization for Standardization, 2011] International Organization for Standardization (2011). Iso/iec 25010:2011 – systems and software engineering – systems and software quality requirements and evaluation (square) – system and software quality models. ISO/IEC.

[Kim et al., 2021] Kim, G., Humble, J., Debois, P., Willis, J., and Forsgren, N. (2021). *The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations*. IT Revolution Press, 2nd edition.

[Sommerville, 2015] Sommerville, I. (2015). *Software Engineering*. Pearson, 10th edition.

[Wynne and Hellesøy, 2012] Wynne, M. and Hellesøy, A. (2012). *The Cucumber Book: Behaviour-Driven Development for Testers and Developers*. Pragmatic Bookshelf.

Ninth Semester

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

CS370 Big Data (Mandatory)

1. General information

- **Semester**: 9th Sem. **Credits**: 3
- **Hour of this course**: **Theory**: 1 hours; **Practice**: 2 hours; **Laboratory**: 2 hours;
- **Prerrequisites**:
  - CS3P1 Parallel and Distributed Computing (8th Sem-Pg. 430)

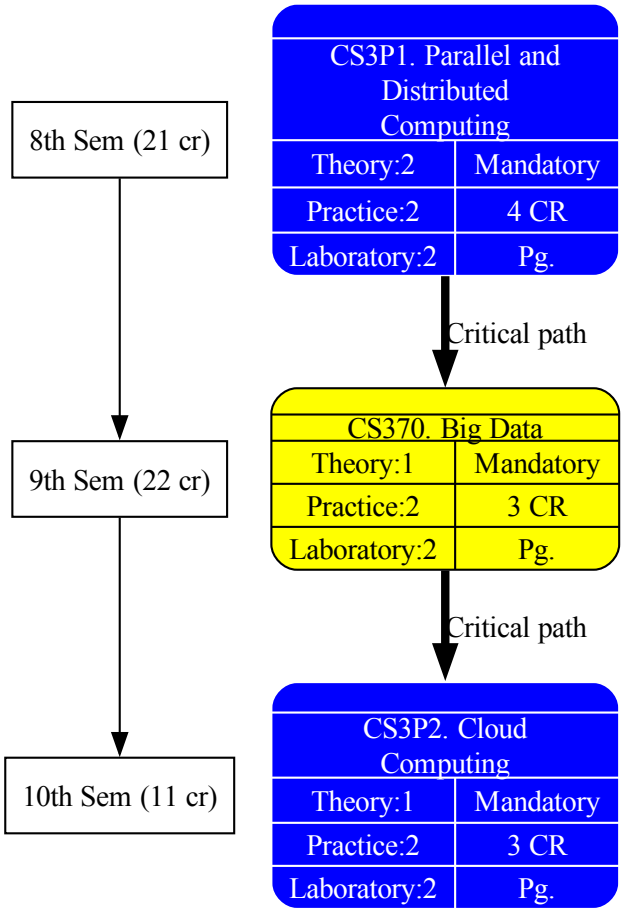

Figure 5.55: Connection Map. CS370 Big Data

2. Course Rationale

In the current era, the volume, velocity, and variety of data exceed the capabilities of traditional storage and processing systems. This course introduces the fundamental concepts and technologies of Big Data. It covers distributed file systems, parallel processing frameworks, NoSQL databases, and

real-time data streaming. Students will learn to design scalable architectures to extract value from massive datasets.

3. Goals of the Course

- Understand the 5 V's of Big Data and their impact on system design.
- Design and implement scalable solutions using the MapReduce paradigm.
- Master the use of distributed file systems like HDFS.
- Analyze different architectural patterns for Big Data (Lambda and Kappa).
- Apply stream processing techniques for real-time data analysis.

4. General Competences (*Outcomes*)

AG-C09) Design and Development of Solutions: Designs, implements, and evaluates solutions for complex computing problems. (Usage)

AG-C11) Use of Tools: Applies modern computing tools in problem solving. (Usage)

5. Content

5.1 Core Database System Concepts, 12 hr(s)

Competences: AG-C09,AG-C11

Topics: **I:** Purpose and advantages of database systems **II:** Components of database systems **III:** Design of core DBMS functions (e.g., query mechanisms, transaction management, buffer management, access methods) **IV:** Database architecture, data independence, and data abstraction **V:** Transaction management **VI:** Normalization **VII:** Approaches for managing large volumes of data (e.g., NoSQL database systems, use of MapReduce) **VIII:** How to support CRUD-only applications **IX:** Distributed databases/cloud-based systems **X:** Structured, semi-structured, and unstructured data **XI:** Use of a declarative query language **XII:** Systems supporting structured and/or stream content

Learning Outcomes **I:** Identify at least four advantages that using a database system provides [Analyze] **II:** Enumerate the components of a (relational) database system [Enumerate] **III:** Follow a query as it is processed by the components of a (relational) database system [Analyze] **IV:** Defend the value of data independence [Defend] **V:** Compose a simple select-project-join query in SQL [Compose] **VI:** Enumerate the four properties of a correct transaction manager [Enumerate] **VII:** Describe the advantages for eliminating duplicate repeated data [Describe] **VIII:** Outline how MapReduce uses parallelism to process data efficiently [Outline] **IX:** Evaluate the differences between structured and semi/unstructured databases [Evaluate]

Bibliography: [White, 2015, Karau et al., 2017]

5.2 Distributed Databases/Cloud Computing, 12 hr(s)

Competences: AG-C09,AG-C11

Topics: **I:** Distributed DBMS: (a) Distributed data storage (b) Distributed query processing (c) Distributed transaction model (d) Homogeneous and heterogeneous solutions (e) Client-server distributed databases **II:** Parallel DBMS: (a) Parallel DBMS architectures: shared memory, shared disk, shared nothing (b) Speedup and scale-up, e.g., use of the MapReduce processing model (c) Data replication and weak consistency models

Learning Outcomes **I:** Describe the key components of a distributed DBMS, including distributed data storage, query processing, and transaction management [Describe] **II:** Analyze the trade-offs between parallel DBMS architectures: shared memory, shared disk, and shared nothing [Analyze] **III:** Describe data replication strategies and weak consistency models in distributed database systems [Describe]

Bibliography: [White, 2015, Lin and Dyer, 2010]

5.3 NoSQL Systems, 12 hr(s)

Competences: AG-C09,AG-C11

Topics: **I:** Why NoSQL? (e.g., Impedance mismatch between Application [CRUD] and RDBMS) **II:** Key-Value and Document data model **III:** Storage systems (e.g., Key-Value systems, Data Lakes) **IV:** Distribution Models (Sharding and Replication) **V:** Graph Databases **VI:** Consistency Models (Update and Read, Quorum consistency, CAP theorem) **VII:** Processing model (e.g., Map-Reduce, multi-stage map-reduce, incremental map-reduce) **VIII:** Case Studies: Cloud storage system (e.g., S3); Graph databases; "When not to use NoSQL"

Learning Outcomes **I:** Develop a use case for the use of NoSQL over RDBMS [Create] **II:** Describe the defining characteristics behind Key-Value and Document-based data models [Describe]

Bibliography: [Sadalage and Fowler, 2012, Kleppmann, 2017]

5.4 Data Analytics, 12 hr(s)

Competences: AG-C09,AG-C11

Topics: **I:** Exploratory data techniques (motivation, representation, descriptive statistics, visualizations) **II:** Data science lifecycle: business understanding, data understanding, data preparation, modeling, evaluation, deployment, and user acceptance **III:** Data mining and machine learning algorithms: e.g., classification, clustering, association, regression **IV:** Data acquisition and governance **V:** Data security and privacy considerations **VI:** Data fairness and bias **VII:** Data visualization techniques and their use in data analytics **VIII:** Entity Resolution

Learning Outcomes **I:** Describe several data exploration approaches, including visualization, to understanding unfamiliar datasets [Describe] **II:** Apply several data exploration approaches to understanding unfamiliar datasets [Apply] **III:** Describe basic machine learning/data mining algorithms and when they are appropriate for use [Describe] **IV:** Apply several machine learning/data mining algorithms [Apply] **V:** Describe legal and ethical considerations in acquiring, using, and modifying datasets [Describe] **VI:** Describe issues of fairness and bias in data collection and usage [Describe]

Bibliography: [Matei et al., 2012, Bill and Matei, 2018, Kleppmann, 2017, Psaltis, 2017]

6. Bibliography

[Bill and Matei, 2018] Bill, C. and Matei, Z. (2018). *Spark: The Definitive Guide: Big Data Processing Made Simple*. O'Reilly Media, Sebastopol, CA.

[Karau et al., 2017] Karau, H., Konwinski, A., Wendell, P., and Zaharia, M. (2017). *Learning Spark: Lightning-Fast Big Data Analysis*. O'Reilly Media.

[Kleppmann, 2017] Kleppmann, M. (2017). *Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems*. O'Reilly Media.

[Lin and Dyer, 2010] Lin, J. and Dyer, C. (2010). *Data-Intensive Text Processing with MapReduce*. Morgan and Claypool Publishers.

[Matei et al., 2012] Matei, Z., Mosharaf, C., Tathagata, D., Ankur, D., Justin, M., Murphy, M., J, F. M., Scott, S., and Ion, S. (2012). Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In *Proceedings of the 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, San Jose, CA. USENIX Association.

[Psaltis, 2017] Psaltis, A. (2017). *Streaming Data: Understanding the real-time pipeline*. Manning Publications.

[Sadalage and Fowler, 2012] Sadalage, P. J. and Fowler, M. (2012). *NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence*. Addison-Wesley.

[White, 2015] White, T. (2015). *Hadoop: The Definitive Guide*. O'Reilly Media, 4th edition.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

CS400 Pre-professional internships (Mandatory)

1. General information

- **Semester**: 9th Sem. **Credits**: 2
- **Hour of this course**: **Theory**: 1 hours; **Practice**: 2 hours;
- **Prerrequisites**:
  - 110Cr

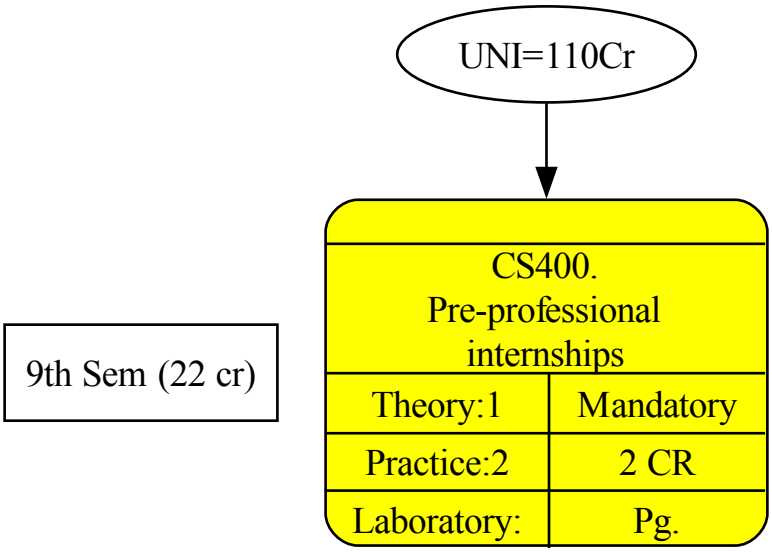

Figure 5.56: Connection Map. CS400 Pre-professional internships

2. Course Rationale

This course allows students to apply the knowledge acquired during their academic training in a real work environment, under the supervision of a company and the university. Pre-professional internships are essential to develop professional, ethical, and technical competencies, ensuring that the student gains experience in projects related to the degree program.

3. Goals of the Course

- Evaluate student performance in a real work environment, applying Computer Science knowledge.
- Develop professional, ethical, and teamwork skills in a business context.
- Ensure that the activities performed are aligned with the degree program objectives.

4. General Competences (*Outcomes*)

AG-C01) The Professional and the World: Analyzes and evaluates the impact of solutions to complex computing problems on the sustainable development of society. (Usage)

AG-C02) Ethics: Applies ethical principles and commits to professional ethics and the standards of professional computing practice. (Assessment)

AG-C03) Individual and Team Work: Performs effectively as an individual and as a member or leader in diverse teams. (Assessment)

AG-C04) Communication: Communicates effectively in complex computing activities. (Assessment)

AG-C05) Project Management: Applies management principles in computing to manage projects. (Usage)

AG-C06) Life-long Learning: Recognizes the importance of continuous learning and adaptation to new technologies. (Assessment)

AG-C09) Design and Development of Solutions: Designs, implements, and evaluates solutions for complex computing problems. (Assessment)

5. Content

5.1 Performance Evaluation in Work Environment, 48 hr(s)

Competences: AG-C02,AG-C04,AG-C05,AG-C09

Topics: **I:** Student integration into a professional work team. **II:** Application of technical knowledge in real projects. **III:** Evaluation of ethical and professional competencies in the work environment. **IV:** Presentation of reports and feedback from the company and university.

Learning Outcomes **I:** Demonstrate technical skills in a real work environment [Assessment]. **II:** Apply ethical and professional principles in their activities [Assessment]. **III:** Work effectively in teams and communicate professionally [Assessment].

Bibliography: [ACM Education Committee, 2022, IEEE Computer Society, 2021]

6. Bibliography

[ACM Education Committee, 2022] ACM Education Committee (2022). Guide to successful pre-professional internships in computer science. Technical report, Association for Computing Machinery. Official document with standards for internships in computing.

[IEEE Computer Society, 2021] IEEE Computer Society (2021). *Professional Practices in Computing: Ethics and Management*. IEEE Press. Best-practices manual for IT work environments.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

CS403 Capstone Project II (Mandatory)

1. General information

- **Semester**: 9th Sem. **Credits**: 3
- **Hour of this course**: **Theory**: 2 hours; **Laboratory**: 4 hours;
- **Prerrequisites**:
  - CS402 Capstone Project I (8th Sem-Pg. 439)

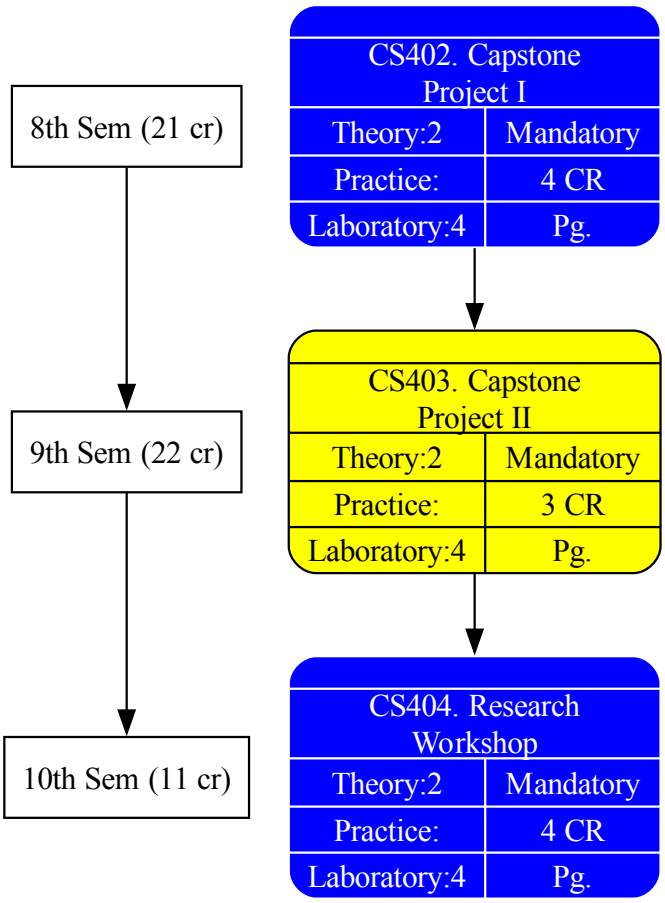

Figure 5.57: Connection Map. CS403 Capstone Project II

2. Course Rationale

This course aims to enable students to develop their research proposal and execute formal experiments, producing the core chapters of their thesis under the supervision of their faculty advisor.

3. Goals of the Course

- Formalize the thesis plan in the institutional format with objectives, schedule, and feasibility analysis.

- Write the theoretical framework and related works chapter with critical synthesis of the literature.
- Design the original methodology and implement a functional prototype of the research proposal.
- Execute preliminary comparative experiments with statistical rigor and well-grounded interpretation of results.
- Present and defend the thesis progress before the evaluation committee incorporating their feedback.
- The deliverables for this course are:

  Midterm Progress: Thesis plan in institutional format, theoretical framework chapter and related works in LaTeX.

  Final: Methodology, proposal, and preliminary experiments chapters approved by the advisor.

4. General Competences (*Outcomes*)

AG-C02) Ethics: Applies ethical principles and commits to professional ethics and the standards of professional computing practice. (Assessment)

AG-C04) Communication: Communicates effectively in complex computing activities. (Assessment)

AG-C05) Project Management: Applies management principles in computing to manage projects. (Assessment)

AG-C08) Problem Analysis: Identifies, formulates, and analyzes complex computing problems. (Assessment)

AG-C09) Design and Development of Solutions: Designs, implements, and evaluates solutions for complex computing problems. (Assessment)

5. Content

5.1 Thesis Plan Formalization, 10 hr(s)

Competences: AG-C05,AG-C08,AG-C09

Topics: I: Structure of the thesis plan following the institutional format. **II:** Precise definition of the research problem and central thesis question. **III:** Formulation of hypothesis, general and specific verifiable objectives. **IV:** Activity schedule with milestones through the final thesis defense. **V:** Feasibility analysis: resources, timeline, risks, and contingencies of the project.

Learning Outcomes I: Produce the thesis plan in the format required by the academic institution. [Assessment] **II:** Define an original research problem with verifiable hypothesis and objectives. [Assessment] **III:** Develop a detailed schedule with milestones and risk analysis through the thesis defense. [Assessment]

Bibliography: [IEEE-Computer Society, 2008, ACM, 2008]

5.2 Theoretical Framework and Advanced State of the Art, 10 hr(s)

Competences: AG-C04,AG-C08,AG-C09

Topics: I: Thematic organization and critical synthesis of the literature in the thesis area. **II:** Writing the theoretical framework chapter in LaTeXwith coherent structure. **III:** Identification and justification of research gaps specific to the problem. **IV:** Comparative analysis of approaches: classification tables and diagrams of the literature. **V:** Reference integration with BibTeX and version control with Git.

Learning Outcomes I: Write the theoretical framework chapter with structured and critical synthesis of the literature. [Assessment] **II:** Justify the originality of the problem in relation to the state of the art. [Assessment] **III:** Build comparative tables that evidence the research gaps of the area. [Assessment]

Bibliography: [IEEE-Computer Society, 2008, ACM, 2008, CiteSeer.IST, 2008]

5.3 Original Methodology Design, 10 hr(s)

Competences: AG-C05,AG-C08,AG-C09

Topics: **I:** Definition of the methodological approach: experimental, analytical, or constructive. **II:** Specification of the original contribution: algorithm, system, model, or methodology. **III:** Experimental protocol design: variables, metrics, datasets, and benchmarks. **IV:** Validation criteria and generalization of the proposal's results. **V:** Alignment of methodology with the objectives and hypothesis of the thesis plan.

Learning Outcomes **I:** Design the research methodology aligned with the hypothesis and objectives. [Assessment] **II:** Specify the original contribution of the thesis project with verifiable criteria. [Assessment] **III:** Define the complete experimental protocol with variables, metrics, and benchmarks. [Assessment]

Bibliography: [IEEE-Computer Society, 2008, ACM, 2008]

5.4 Implementation and Preliminary Experiments, 12 hr(s)

Competences: AG-C05,AG-C08,AG-C09

Topics: **I:** Implementation of the original proposal: prototype, algorithm, or thesis model. **II:** Reproduction of relevant baselines from the state of the art for comparison. **III:** Execution of preliminary experiments with basic statistical analysis. **IV:** Recording and documentation of configurations, versions, and results. **V:** Identification of limitations and iterative refinement of the proposal.

Learning Outcomes **I:** Implement a functional prototype of the original thesis proposal. [Usage] **II:** Execute preliminary comparative experiments with defined baselines. [Usage] **III:** Analyze and interpret results with appropriate statistical rigor. [Assessment]

Bibliography: [IEEE-Computer Society, 2008, ACM, 2008]

5.5 Writing the Core Thesis Chapters, 10 hr(s)

Competences: AG-C02,AG-C04,AG-C09

Topics: **I:** Writing the methodology and experimental design chapter in LaTeX. **II:** Writing the related works chapter with critical comparative analysis. **III:** Presentation of preliminary results: tables, figures, and comparative analysis. **IV:** Revision cycles with the advisor: structured incorporation of feedback. **V:** Narrative coherence and consistency across thesis chapters.

Learning Outcomes **I:** Write the methodology and related works chapters with academic rigor. [Assessment] **II:** Present preliminary results with clear visualizations and comparative analysis. [Usage] **III:** Apply revision cycles with the advisor to improve document quality and coherence. [Assessment]

Bibliography: [IEEE-Computer Society, 2008, ACM, 2008]

5.6 Progress Presentation and Final Planning, 8 hr(s)

Competences: AG-C02,AG-C04,AG-C05

Topics: **I:** Oral presentation of thesis progress before the evaluation committee. **II:** Defense of methodology and preliminary results before committee questions. **III:** Incorporation of committee feedback into the document revision. **IV:** Planning of remaining activities toward the final thesis defense. **V:** Research ethics management: authorship, academic integrity, and data confidentiality.

Learning Outcomes I: Present and orally defend the thesis progress before an academic committee. [Assessment] **II:** Incorporate committee feedback into a well-grounded revision of the document. [Assessment] **III:** Plan the remaining activities with clear objectives toward the final defense. [Usage]

Bibliography: [IEEE-Computer Society, 2008, ACM, 2008]

6. Bibliography

[ACM, 2008] ACM (2008). *Digital Library*. Association for Computing Machinery. http://portal.acm.org/dl.cfm.

[CiteSeer.IST, 2008] CiteSeer.IST (2008). *Scientific Literature Digital Library*. College of Information Sciences and Technology, Penn State University. http://citeseer.ist.psu.edu.

[IEEE-Computer Society, 2008] IEEE-Computer Society (2008). *Digital Library*. IEEE-Computer Society. http://www.computer.org/publications/dlib.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

AI365 Advanced Generative AI Models (Mandatory)

1. General information

- **Semester**: 9th Sem. **Credits**: 4
- **Hour of this course**: **Theory**: 2 hours; **Laboratory**: 4 hours;
- **Prerrequisites**:
  - AI264 Deep Learning (7th Sem-Pg. 411)

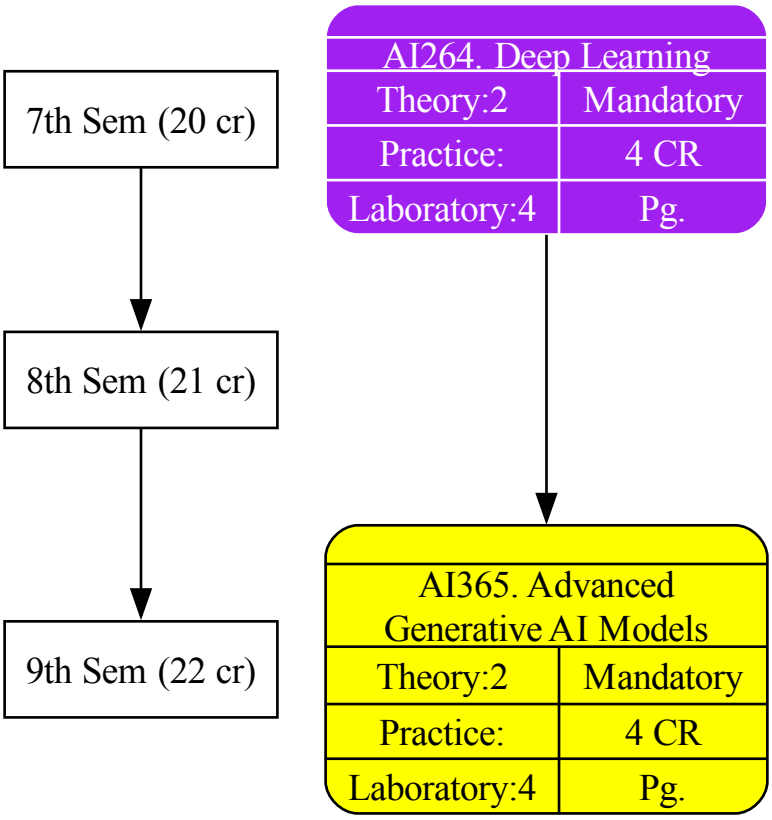

Figure 5.58: Connection Map. AI365 Advanced Generative AI Models

2. Course Rationale

This course explores modern generative models (GANs, Diffusion Models, LLMs) for multimodal content creation, covering both mathematical foundations and practical applications with ethical considerations for synthetic media.

3. Goals of the Course

- Implement image/text/video generation pipelines
- Design AI-assisted editing systems
- Evaluate synthetic content risks

4. General Competences (*Outcomes*)

AG-C09) Design and Development of Solutions: Designs, implements, and evaluates solutions for complex computing problems. (Assessment)

AG-C11) Use of Tools: Applies modern computing tools in problem solving. (Assessment)

5. Content

5.1 Generative Fundamentals, 12 hr(s)

Competences: AG-C09,AG-C11

Topics: **I:** GANs theory (minimax, Wasserstein) **II:** Autoregressive models (PixelCNN, Transformers) **III:** Evaluation metrics (FID, Inception Score)

Learning Outcomes **I:** Train basic GANs [Usage] **II:** Analyze generation failure modes [Assessment]

Bibliography: [Goodfellow et al., 2014]

5.2 Diffusion Models, 24 hr(s)

Competences: AG-C09,AG-C11

Topics: **I:** Diffusion processes (forward/reverse) **II:** Stable Diffusion and latent architectures **III:** Control techniques (ControlNet, LoRA)

Learning Outcomes **I:** Implement diffusion samplers [Usage] **II:** Fine-tune for domain-specific generation [Assessment]

Bibliography: [Sohl-Dickstein et al., 2023, Rombach et al., 2022]

5.3 Synthetic Media Ethics, 8 hr(s)

Competences: AG-C09,AG-C11

Topics: **I:** Deepfakes and misinformation **II:** Copyright in generated content **III:** Authenticity marking (C2PA, watermarking)

Learning Outcomes **I:** Detect synthetic content [Familiarity] **II:** Implement traceability systems [Usage]

Bibliography: [Sohl-Dickstein et al., 2023]

6. Bibliography

[Goodfellow et al., 2014] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial networks. In *NeurIPS*.

[Rombach et al., 2022] Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. (2022). High-resolution image synthesis with latent diffusion models. *CVPR*.

[Sohl-Dickstein et al., 2023] Sohl-Dickstein, J., Song, Y., and Ermon, S. (2023). *Diffusion Models: A Comprehensive Practical Guide*. AI Press.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

CB309 Bioinformatics (Mandatory)

1. General information

- **Semester**: 9th Sem. **Credits**: 4
- **Hour of this course**: **Theory**: 2 hours; **Practice**: 4 hours;
- **Prerrequisites**: None

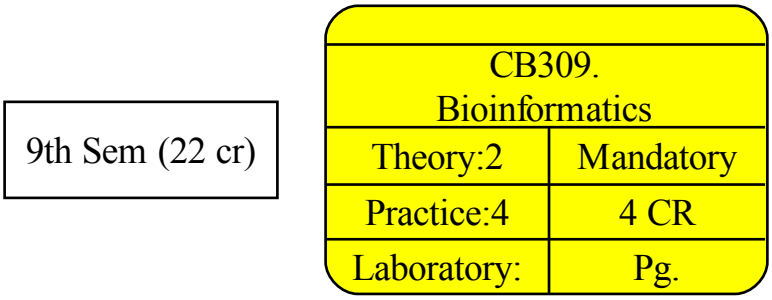

Figure 5.59: Connection Map. CB309 Bioinformatics

2. Course Rationale

The use of computational methods in the biological sciences has become one of the key tools for the field of molecular biology, being a fundamental part of research in this area.
In Molecular Biology, there are several applications that involve both DNA, protein analysis or sequencing of the human genome, which depend on computational methods. Many of these problems are really complex and deal with large data sets.
This course can be used to see concrete use cases of several areas of knowledge of Computer Science such as Programming Languages (PL), Algorithms and Complexity (AL), Probabilities and Statistics, Information Management (IM), Intelligent Systems (IS).

3. Goals of the Course

- That the student has a solid knowledge of molecular biological problems that challenge computing.
- That the student is able to abstract the essence of the various biological problems to pose solutions using their knowledge of Computer Science

4. General Competences (*Outcomes*)

AG-C10) Inquiry: Studies complex computing problems using information science methods. (Usage)

AG-C11) Use of Tools: Applies modern computing tools in problem solving. (Usage)

5. Content

5.1 Introduction to Molecular Biology, 4 hr(s)

Competences: AG-C10,AG-C11

Topics: **I:** Review of organic chemistry: molecules and macromolecules, sugars, nucleic acids, nucleotides, RNA, DNA, proteins, amino acids and levels of structure in proteins. **II:** The Dogma of Life: From DNA to Proteins, Transcription, Translation, Protein Synthesis. **III:** Genome study: Maps and sequences, specific techniques

Learning Outcomes **I:** Achive a general knowledge of the most important topics in Molecular Biology. [Familiarity] **II:** Understand that biological problems are a challenge to the computational world. [Assessment]

Bibliography: [Clote and Backofen, 2000, Setubal and Meidanis, 1997]

5.2 Sequence Comparison, 4 hr(s)

Competences: AG-C10,AG-C11

Topics: **I:** Sequences of nucleotides and amino acid sequences. **II:** Sequence alignment, paired alignment problem, exhaustive search, Dynamic programming, global alignment, local alignment, gaps penalty **III:** Comparison of multiple sequences: sum of pairs, complexity analysis by dynamic programming, alignment heuristics, star algorithm, progressive alignment algorithms.

Learning Outcomes **I:** Understand and solve the problem of aligning a pair of sequences. [Usage] **II:** Understand and solve the problem of multiple sequence alignment. [Usage] **III:** Know the various algorithms for aligning existing sequences in the literature . [Familiarity]

Bibliography: [Clote and Backofen, 2000, Setubal and Meidanis, 1997, Pevzner, 2000]

5.3 Phylogenetic Trees, 4 hr(s)

Competences: AG-C10,AG-C11

Topics: **I:** Phylogeny: Introduction and phylogenetic relations **II:** Phylogenetic trees: definition, type of trees, problem of search and reconstruction of trees **III:** Reconstruction methods: parsimony methods, distance methods, maximum likelihood methods, confidence of reconstructed trees

Learning Outcomes **I:** Understand the concept of phylogeny, phylogenetic trees and the methodological difference between biology and molecular biology. [Familiarity] **II:** Understand the problem of the reconstruction of phylogenetic trees, to know and apply the main algorithms for the reconstruction of phylogenetic trees. [Assessment]

Bibliography: [Clote and Backofen, 2000, Setubal and Meidanis, 1997, Pevzner, 2000]

5.4 DNA Sequence Assembling, 4 hr(s)

Competences: AG-C10,AG-C11

Topics: **I:** Biological basis: ideal case, difficulties, alternative methods for DNA sequencing **II:** Formal Assembly Models: Shortest Common Superstring, Reconstruction, Multicontig **III:** Algorithms for sequence assembly: representation of overlaps, paths to create superstrings, voracious algorithm, acyclic graphs. **IV:** Assembly heuristics: search for overlays, ordering fragments, alignments and consensus.

Learning Outcomes **I:** Understand the computational challenge of the Sequence Assembly problem. [Familiarity] **II:** Understand the principle of formal model for assembly. [Assessment] **III:** Know the main heuristics for the problem of assembjale of DNA sequences[Usage]

Bibliography: [Setubal and Meidanis, 1997, Aluru, 2006]

5.5 Secondary and tertiary structures, 4 hr(s)

Competences: AG-C10,AG-C11

Topics: **I:** Molecular structures: primary, secondary, tertiary, quaternary. **II:** Prediction of secondary structures of RNA: formal model, pair energy, structures with independent bases, solution with Dynamic Programming, structures with loops. **III:** *Protein folding*: Estructuras en proteinas, problema de protein folding. **IV:** *Protein Threading*: Definitions, Branch Bound Algorithm, Branch Bound for protein threading. **V:** *Structural Alignment*: Definitions, DALI algorithm

Learning Outcomes **I:** Know the protein structures and the necessity of computational methods for the prediction of the geometry. [Familiarity] **II:** Know the algorithms for solving prediction problems of secondary structures RNA, and structures in proteins. [Assessment]

Bibliography: [Setubal and Meidanis, 1997, Clote and Backofen, 2000, Aluru, 2006]

5.6 Probabilistic Models in Molecular Biology, 4 hr(s)

Competences: AG-C10,AG-C11

Topics: **I:** Probability: Random Variables, Markov Chains, Metropoli-Hasting Algorithm, Markov Random Fields, and Gibbs Sampler, Maximum Likelihood. **II:** Hidden Markov Models (HMM), parameter estimation, Viterbi algorithm and Baul-Welch method, Application in paired and multiple alignments, Motifs detection in proteins, in eukaryotic DNA, in sequences families. **III:** Probabilistic phylogeny: probabilistic models of evolution, likelihood of alignments, likelihood for inference, comparison of probailistic and non-probabilistic methods

Learning Outcomes **I:** Review concepts of Probabilistic Models and understand their importance in Computational Molecular Biology. [Assessment] **II:** Know and apply Hidden Markov Models for various analyzes in Molecular Biology.. [Usage] **III:** Know the application of probabilistic models in Phylogeny and to compare them with non-probabilistic models[Assessment]

Bibliography: [R et al., 1998, Clote and Backofen, 2000, Aluru, 2006, Krogh et al., 1994]

6. Bibliography

[Aluru, 2006] Aluru, S., editor (2006). *Handbook of Computational Molecular Biology*. Computer and Information Science Series. Chapman & Hall, CRC, Boca Raton, FL.

[Clote and Backofen, 2000] Clote, P. and Backofen, R. (2000). *Computational Molecular Biology: An Introduction*. John Wiley & Sons Ltd. 279 pages.

[Krogh et al., 1994] Krogh, A., Brown, M., Mian, I. S., Sjölander, K., and Haussler, D. (1994). Hidden markov models in computational biology, applications to protein modeling. *J Mol. Biol*, 235:1501–1531.

[Pevzner, 2000] Pevzner, P. A. (2000). *Computational Molecular Biology: an Algorithmic Approach*. The MIT Press, Cambridge, Massachusetts.

[R et al., 1998] R, D., S.R, E., A, K., and G, M. (1998). *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press.

[Setubal and Meidanis, 1997] Setubal, J. C. and Meidanis, J. (1997). *Introduction to computational molecular biology*. Boston: PWS Publishing Company.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

ET301 Entrerpreneurship II (Mandatory)

1. General information

- **Semester**: 9th Sem. **Credits**: 3
- **Hour of this course**: **Theory**: 2 hours; **Practice**: 2 hours;
- **Prerrequisites**:
  - ET201 Entrerpreneurship I (8th Sem-Pg. 445)

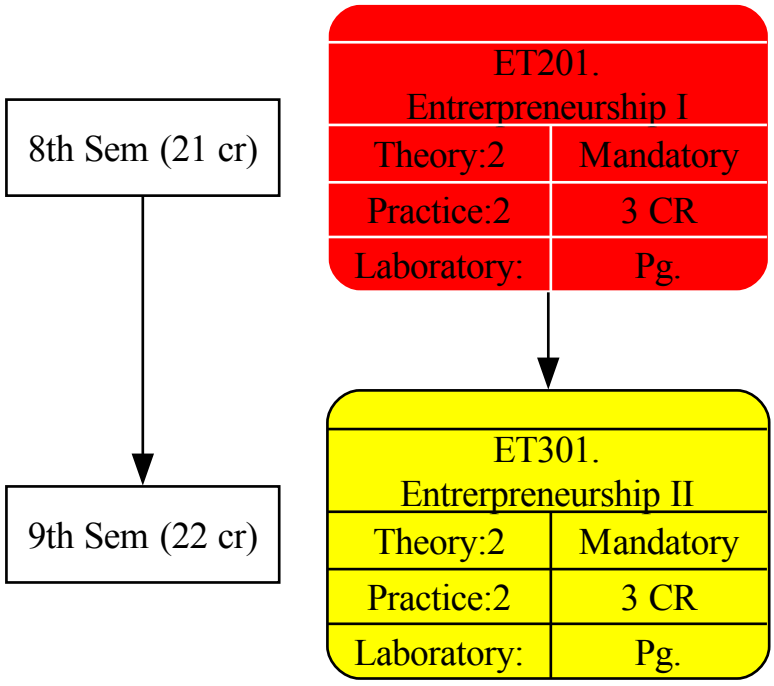

Figure 5.60: Connection Map. ET301 Entrerpreneurship II

2. Course Rationale

Este curso tiene como objetivo dotar al futuro profesional de conocimientos, actitudes y aptitudes que le permitan formar su propia empresa de desarrollo de software y/o consultoría en informática. El curso está dividido en tres unidades: Valorización de Proyectos, Marketing de Servicios y Negociaciones. En la primera unidad se busca que el alumno pueda analizar y tomar decisiones en relación a la viabilidad de un proyecto y/o negocio.

En la segunda unidad se busca preparar al alumno para que este pueda llevar a cabo un plan de marketing satisfactorio del bien o servicio que su empresa pueda ofrecer al mercado. La tercera unidad busca desarrollar la capacidad negociadora de los participantes a través del entrenamiento vivencial y práctico y de los conocimientos teóricos que le permitan cerrar contrataciones donde tanto el cliente como el proveedor resulten ganadores. Consideramos estos temas sumamente críticos en las etapas de lanzamiento, consolidación y eventual relanzamiento de una empresa de base tecnológica.

3. Goals of the Course

- Que el alumno comprenda y aplique la terminología y conceptos fundamentales de ingeniería económica que le permitan valorizar un proyecto para tomar la mejor decisión económica.
- Que el alumno adquiera las bases para formar su propia empresa de base tecnológica.

4. General Competences (*Outcomes*)

No items for this table

5. Content

5.1 Valorización de Proyectos, 20 hr(s)

Competences:

Topics: **I:** Introducción **II:** Proceso de toma de decisiones **III:** El valor del dinero en el tiempo **IV:** Tasa de interés y tasa de rendimiento **V:** Interés simple e interés compuesto **VI:** Identificación de costos **VII:** Flujo de Caja Neto **VIII:** Tasa de Retorno de Inversión (TIR) **IX:** Valor Presente Neto (VPN) **X:** Valorización de Proyectos

Learning Outcomes **I:** Permitir al alumno tomar decisiones sobre como invertir mejor los fondos disponibles, fundamentadas en el análisis de los factores tanto económicos como no económicos que determinen la viabilidad de un emprendimiento. [Assessment]

Bibliography: [Blank and Tarkin, 2006]

5.2 Marketing de Servicios, 30 hr(s)

Competences:

Topics: **I:** Introducción **II:** Importancia del marketing en las empresas de servicios **III:** El Proceso estratégico. **IV:** El Plan de Marketing **V:** Marketing estratégico y marketing operativo **VI:** Segmentación, targeting y posicionamiento de servicios en mercados competitivos **VII:** Ciclo de vida del producto **VIII:** Aspectos a considerar en la fijación de precios en servicios **IX:** El rol de la publicidad, las ventas y otras formas de comunicación **X:** El comportamiento del consumidor en servicios **XI:** Fundamentos de marketing de servicios **XII:** Creación del modelo de servicio **XIII:** Gestión de la calidad de servicio

Learning Outcomes **I:** Brindar las herramientas al alumno para que pueda identificar, analizar y aprovechar las oportunidades de marketing que generan valor en un emprendimiento. [Usage] **II:** Lograr que el alumno conozca, entienda e identifique criterios, habilidades, métodos y procedimientos que permitan una adecuada formulación de estrategias de marketing en sectores y medios específicos como lo es una empresa de base tecnológica. [Usage]

Bibliography: [Kotler and Keller, 2006, Lovelock and Wirtz, 2009]

5.3 Negociaciones, 10 hr(s)

Competences:

Topics: **I:** Introducción. ¿Qué es una negociación? **II:** Teoría de las necesidades de la negociación **III:** La proceso de la negociación **IV:** Estilos de negociación **V:** Teoría de juegos **VI:** El método Harvard de negociación

Learning Outcomes **I:** Conocer los puntos clave en el proceso de negociación. [Usage] **II:** Establecer una metodología de negociación eficaz. [Usage] **III:** Desarrollar destrezas y habilidades que permitan llevar a cabo una negociación exitosa. [Usage]

Bibliography: [Fisher et al., 1996, de Manuel Dasí and Martínez, 2006]

6. Bibliography

[Blank and Tarkin, 2006] Blank, L. and Tarkin, A. (2006). *Ingeniería Económica*. McGraw Hill, México D.F., México.

[de Manuel Dasí and Martínez, 2006] de Manuel Dasí, F. and Martínez, R. M.-V. (2006). *Técnicas de Negociación. Un método práctico*. Esic, Madrid.

[Fisher et al., 1996] Fisher, R., Ury, W., and Patton, B. (1996). *Si... ¡de acuerdo! Cómo negociar sin ceder*. Norma, Barcelona.

[Kotler and Keller, 2006] Kotler, P. and Keller, K. L. (2006). *Dirección de Marketing*. Prentice Hall, México.

[Lovelock and Wirtz, 2009] Lovelock, C. and Wirtz, J. (2009). *Marketing de servicios. Personal, tecnología y estrategia*. Prentice Hall, México.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

AI369 Robotics (Elective)

1. General information

- **Semester**: 9th Sem. **Credits**: 3
- **Hour of this course**: **Theory**: 2 hours; **Laboratory**: 2 hours;
- **Prerrequisites**:
  - AI268 Computational Vision (8th Sem-Pg. 449)

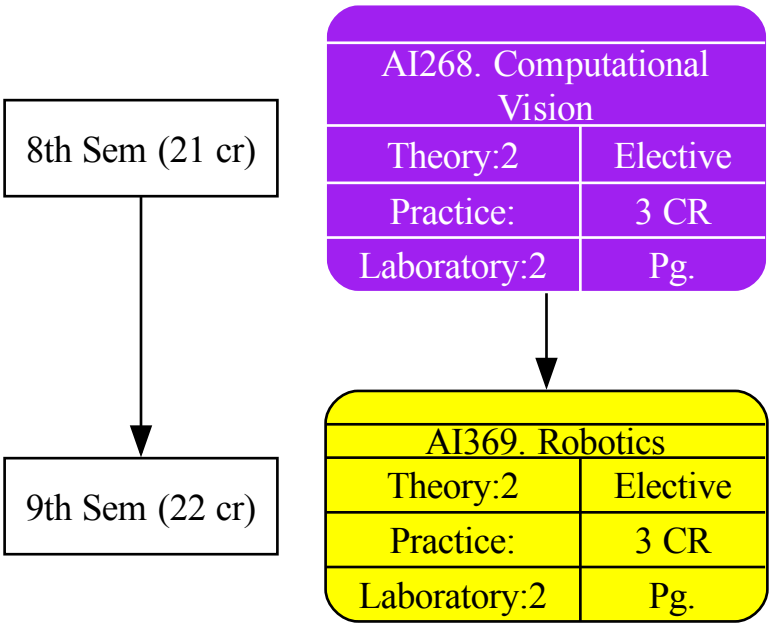

Figure 5.61: Connection Map. AI369 Robotics

2. Course Rationale

Robotics is the physical manifestation of Artificial Intelligence. This course introduces the fundamental concepts of robot design, including kinematics, perception, and control. Students will explore how robots interact with the physical world through sensors and actuators, and how they make autonomous decisions using probabilistic mapping and localization techniques. The course bridges the gap between abstract algorithms and physical execution.

3. Goals of the Course

- Understand the mechanical and mathematical foundations of robot kinematics.
- Master sensor fusion techniques for robot perception.
- Implement navigation and path planning algorithms.
- Analyze the integration of machine learning in robotic control.
- Design and simulate autonomous agents for specific environments.

4. General Competences (*Outcomes*)

AG-C09) Design and Development of Solutions: Designs, implements, and evaluates solutions for complex computing problems. (Usage)

AG-C11) Use of Tools: Applies modern computing tools in problem solving. (Assessment)

5. Content

5.1 Robotics, 12 hr(s)

Competences: AG-C09,AG-C11

Topics: **I:** Overview: problems and progress: (a) State-of-the-art robot systems, including their sensors and an overview of their sensor processing (b) Robot control architectures, e.g., deliberative vs reactive control and Braitenberg vehicles (c) World modeling and world models (d) Inherent uncertainty in sensing and in control **II:** Sensors and effectors: (a) Sensors: e.g., LIDAR, sonar, vision, depth, stereoscopic, event cameras, microphones, haptics (b) Effectors: e.g., wheels, arms, grippers **III:** Coordinate frames, translation, and rotation (2D and 3D) **IV:** Configuration space and environmental maps **V:** Interpreting uncertain sensor data **VI:** Localization and mapping **VII:** Navigation and control **VIII:** Forward and inverse kinematics **IX:** Motion path planning and trajectory optimization **X:** Manipulation and grasping **XI:** Joint control and dynamics **XII:** Vision-based control **XIII:** Multiple-robot coordination and collaboration **XIV:** Human-robot interaction: (a) Shared workspaces (b) Human-robot teaming and physical HRI (c) Social assistive robots (d) Motion/task/goal prediction (e) Collaboration and communication (explicit vs implicit, verbal or symbolic vs non-verbal or visual) (f) Trust **XV:** Applications and Societal, Economic, and Ethical Issues: (a) Societal, economic, right-to-work implications (b) Ethical and privacy implications of robotic applications (c) Liability in autonomous robotics (d) Autonomous weapons and ethics (e) Human oversight and control

Learning Outcomes **I:** List capabilities and limitations of today's state-of-the-art robot systems, including their sensors and the crucial sensor processing that informs those systems [List] **II:** Integrate sensors, actuators, and software into a robot designed to undertake a specific task [Apply] **III:** Program a robot to accomplish simple tasks using deliberative, reactive, and/or hybrid control architectures [Apply] **IV:** Implement fundamental motion planning algorithms within a robot configuration space [Apply] **V:** Characterize the uncertainties associated with common robot sensors and actuators; articulate strategies for mitigating these uncertainties [Characterize] **VI:** List the differences among robots' representations of their external environment, including their strengths and shortcomings [List] **VII:** Compare and contrast at least three strategies for robot navigation within known and/or unknown environments, including their strengths and shortcomings [Compare] **VIII:** Describe at least one approach for coordinating the actions and sensing of several robots to accomplish a single task [Describe] **IX:** Compare and contrast a multi-robot coordination and a human-robot collaboration approach and attribute their differences to differences between the problem settings [Compare] **X:** Analyze the societal, economic, and ethical issues of a real-world robotics application [Analyze]

Bibliography: [Siegwart et al., 2011, Craig, 2017]

5.2 Perception and Computer Vision, 12 hr(s)

Competences: AG-C09,AG-C11

Topics: **I:** Computer vision: (a) Image acquisition, representation, processing, and properties (b) Shape representation, object recognition, and segmentation (c) Motion analysis (d) Generative models **II:** Audio and speech recognition **III:** Touch and proprioception **IV:** Other modalities (e.g., olfaction) **V:** Modularity in recognition **VI:** Approaches to pattern recognition: (a) Classification algorithms and measures of classification quality (b) Statistical techniques (c) Deep learning techniques

Learning Outcomes **I:** Summarize the importance of image and object recognition in AI and indicate several significant applications of this technology [Summarize] **II:** List at least three image-segmentation approaches, such as thresholding, edge-based and region-based algorithms, along with their defining characteristics, strengths, and weaknesses [List] **III:** Implement 2d object recognition based on contour-based and/or region-based shape representations [Apply] **IV:** Distinguish the goals of sound-recognition, speech-recognition, and speaker-recognition and identify how the raw audio signal will be handled differently in each of these cases [Distinguish] **V:** Provide at least two examples of a transformation of a data source from one sensory domain to another, e.g., tactile data interpreted as single-band 2d images [Generate] **VI:** Implement a feature-extraction algorithm on real data, e.g., an edge or corner detector for images or vectors of Fourier coefficients describing a short slice of audio signal [Apply] **VII:** Implement an algorithm combining features into higher-level percepts, e.g., a contour or polygon from visual primitives or phoneme hypotheses from an audio signal [Apply] **VIII:** Implement a classification algorithm that segments input percepts into output categories and quantitatively evaluates the resulting classification [Apply] **IX:** Evaluate the performance of the underlying feature-extraction, relative to at least one alternative possible approach (whether implemented or not) in its contribution to the classification task (8), above [Evaluate] **X:** Describe at least three classification approaches, their prerequisites for applicability, their strengths, and their shortcomings [Describe] **XI:** Implement and evaluate a deep learning solution to problems in computer vision, such as object or scene recognition [Apply]

Bibliography: [Thrun et al., 2005, Siegwart et al., 2011]

5.3 Robotics, 24 hr(s)

Competences: AG-C09,AG-C11

Topics: **I:** Overview: problems and progress: (a) State-of-the-art robot systems, including their sensors and an overview of their sensor processing (b) Robot control architectures, e.g., deliberative vs reactive control and Braitenberg vehicles (c) World modeling and world models (d) Inherent uncertainty in sensing and in control **II:** Sensors and effectors: (a) Sensors: e.g., LIDAR, sonar, vision, depth, stereoscopic, event cameras, microphones, haptics (b) Effectors: e.g., wheels, arms, grippers **III:** Coordinate frames, translation, and rotation (2D and 3D) **IV:** Configuration space and environmental maps **V:** Interpreting uncertain sensor data **VI:** Localization and mapping **VII:** Navigation and control **VIII:** Forward and inverse kinematics **IX:** Motion path planning and trajectory optimization **X:** Manipulation and grasping **XI:** Joint control and dynamics **XII:** Vision-based control **XIII:** Multiple-robot coordination and collaboration **XIV:** Human-robot interaction: (a) Shared workspaces (b) Human-robot teaming and physical HRI (c) Social assistive robots (d) Motion/task/goal prediction (e) Collaboration and communication (explicit vs implicit, verbal or symbolic vs non-verbal or visual) (f) Trust **XV:** Applications and Societal, Economic, and Ethical Issues: (a) Societal, economic, right-to-work implications (b) Ethical and privacy implications of robotic applications (c) Liability in autonomous robotics (d) Autonomous weapons and ethics (e) Human oversight and control

Learning Outcomes **I:** List capabilities and limitations of today's state-of-the-art robot systems, including their sensors and the crucial sensor processing that informs those systems [List] **II:** Integrate sensors, actuators, and software into a robot designed to undertake a specific task [Apply] **III:** Program a robot to accomplish simple tasks using deliberative, reactive, and/or hybrid control architectures [Apply] **IV:** Implement fundamental motion planning algorithms within a robot configuration space [Apply] **V:** Characterize the uncertainties associated with common robot sensors and actuators; articulate strategies for mitigating these uncertainties [Characterize] **VI:** List the differences among robots' representations of their external environment, including their strengths and shortcomings [List] **VII:** Compare and contrast at least three strategies for robot navigation within known and/or unknown environments, including their strengths and shortcomings [Compare] **VIII:** Describe at least one approach for coordinating the actions and sensing of several robots to accomplish a single task [Describe] **IX:** Compare and contrast a multi-robot coordination and a human-robot collaboration approach and

attribute their differences to differences between the problem settings [Compare] **X:** Analyze the societal, economic, and ethical issues of a real-world robotics application [Analyze]

Bibliography: [Thrun et al., 2005, Siegwart et al., 2011]

6. Bibliography

[Craig, 2017] Craig, J. J. (2017). *Introduction to Robotics: Mechanics and Control*. Pearson, 4th edition.

[Siegwart et al., 2011] Siegwart, R., Nourbakhsh, I. R., and Scaramuzza, D. (2011). *Introduction to Autonomous Mobile Robots*. MIT Press, 2nd edition.

[Thrun et al., 2005] Thrun, S., Burgard, W., and Fox, D. (2005). *Probabilistic Robotics*. MIT Press.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

CS351 Topics in Computer Graphics (Elective)

1. General information

- **Semester**: 9th Sem. **Credits**: 3
- **Hour of this course**: **Theory**: 2 hours; **Laboratory**: 2 hours;
- **Prerrequisites**:
  - CS251 Computer Graphics (7th Sem-Pg. 398)

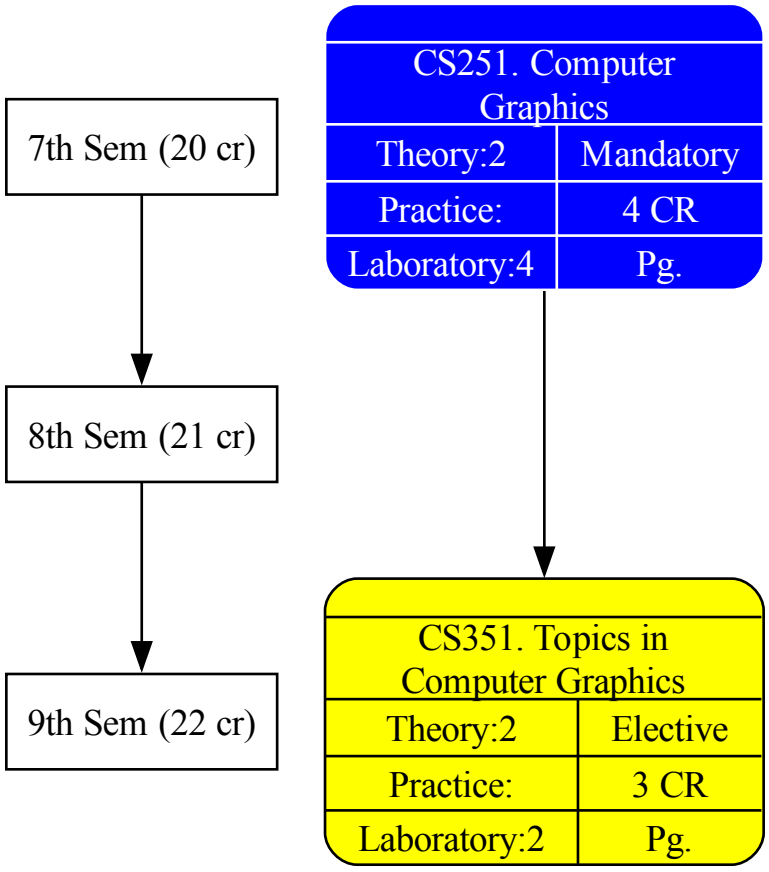

Figure 5.62: Connection Map. CS351 Topics in Computer Graphics

2. Course Rationale

Modern computer graphics go beyond simple image generation, encompassing real-time interaction, physical simulation, and immersive environments. This course explores advanced topics in the graphics pipeline, focusing on programmable shading, complex geometric modeling, and computer animation. Additionally, it addresses the intersection of graphics with physical computing and the ethical implications of synthetic content generation.

3. Goals of the Course

- Master advanced shader programming and non-photorealistic rendering.
- Implement physically-based simulations and complex animations.
- Design immersive and interactive experiences using tangible computing.
- Evaluate the ethical and societal impact of computer-generated imagery.

4. General Competences (*Outcomes*)

AG-C09) Design and Development of Solutions: Designs, implements, and evaluates solutions for complex computing problems. (Usage)

AG-C11) Use of Tools: Applies modern computing tools in problem solving. (Usage)

5. Content

5.1 Shading and Advanced Rendering, 12 hr(s)

Competences: AG-C09,AG-C11

Topics: **I:** Advanced shading: physically based shading, subsurface scattering **II:** Global illumination: radiosity, photon mapping, path tracing **III:** Real-time rendering techniques: level of detail, culling, occlusion **IV:** GPU programming and shaders

Learning Outcomes **I:** Implement advanced shading models for realistic materials [Implement] **II:** Compare global illumination algorithms and their trade-offs [Compare] **III:** Optimize real-time rendering using techniques like level of detail and culling [Create] **IV:** Develop GPU shaders for custom rendering effects [Create]

Bibliography: [Marschner and Shirley, 2021a, Marschner and Shirley, 2021b]

5.2 Geometric Modeling, 10 hr(s)

Competences: AG-C09,AG-C11

Topics: **I:** Geometric representations: polygons, curves, surfaces, implicit forms **II:** Modeling techniques: splines, subdivision surfaces, constructive solid geometry **III:** Procedural modeling and generation **IV:** Geometric algorithms: intersection, distance, decomposition

Learning Outcomes **I:** Describe different geometric representations and their trade-offs [Describe] **II:** Implement modeling techniques such as spline curves or subdivision surfaces [Implement] **III:** Create procedural models using algorithms or rules [Create] **IV:** Apply geometric algorithms to solve problems like intersection testing or mesh processing [Apply]

Bibliography: [Marschner and Shirley, 2021a]

5.3 Computer Animation, 8 hr(s)

Competences: AG-C09,AG-C11

Topics: **I:** Animation principles: timing, spacing, anticipation, follow-through **II:** Keyframe animation and interpolation **III:** Procedural animation: particle systems, flocking, physics-based **IV:** Character animation: skeletal animation, skinning, inverse kinematics **V:** Motion capture and retargeting

Learning Outcomes **I:** Apply animation principles to create believable motion [Apply] **II:** Implement keyframe animation with interpolation techniques [Implement] **III:** Create procedural animations such as particle systems or physics-based simulations [Create] **IV:** Implement character animation using skeletal systems and skinning [Implement] **V:** Process motion capture data for animation retargeting [Implement]

Bibliography: [Parent, 2012]

5.4 Simulation, 8 hr(s)

Competences: AG-C09,AG-C11

Topics: **I:** Physics-based simulation: rigid bodies, fluids, cloth **II:** Numerical methods for simulation: integration, collision detection **III:** Real-time simulation constraints and optimizations **IV:** Applications in games, visual effects, and scientific visualization

Learning Outcomes **I:** Implement physics-based simulations for rigid bodies or fluids [Implement] **II:** Apply numerical integration methods for dynamic simulations [Apply] **III:** Optimize simulations for real-time performance [Create] **IV:** Develop simulations for specific applications such as games or scientific visualization [Create]

Bibliography: [Parent, 2012]

5.5 Immersion, 5 hr(s)

Competences: AG-C09,AG-C11

Topics: **I:** Virtual reality (VR) and augmented reality (AR) concepts **II:** Immersive display technologies: head-mounted displays, CAVEs **III:** Tracking and input for immersive environments **IV:** Presence and comfort in immersive systems

Learning Outcomes **I:** Explain the core concepts and differences between VR and AR [Explain] **II:** Evaluate immersive display technologies and their applications [Evaluate] **III:** Implement tracking and input solutions for immersive environments [Implement] **IV:** Design immersive experiences that maximize presence and minimize discomfort [Design]

Bibliography: [LaValle, 2023]

5.6 Interaction, 5 hr(s)

Competences: AG-C09,AG-C11

Topics: **I:** Interaction paradigms: direct manipulation, gesture-based, tangible **II:** User interface design for graphics applications **III:** Multimodal interaction: combining visual, auditory, haptic feedback **IV:** Evaluation of interaction techniques: usability, user experience

Learning Outcomes **I:** Compare different interaction paradigms and their suitability for various tasks [Compare] **II:** Design user interfaces for graphics-intensive applications [Design] **III:** Implement multimodal interaction systems incorporating multiple feedback channels [Implement] **IV:** Conduct usability evaluations of interaction techniques [Evaluate]

Bibliography: [LaValle, 2023]

5.7 Tangible/Physical Computing, 8 hr(s)

Competences: AG-C09,AG-C11

Topics: **I:** Tangible interfaces and physical computing **II:** Sensors and actuators for interactive systems **III:** Embedded systems for interactive art and design **IV:** Interactive installations and environments

Learning Outcomes **I:** Design tangible interfaces that bridge physical and digital interactions [Design] **II:** Integrate sensors and actuators to create responsive systems [Integrate] **III:** Develop embedded systems for interactive art or design projects [Create] **IV:** Create interactive installations that engage users in physical spaces [Create]

Bibliography: [Noble, 2012]

5.8 Society, Ethics, and the Profession, 8 hr(s)

Competences: AG-C09,AG-C11

Topics: I: Ethical considerations in graphics and interaction: representation, accessibility, privacy **II:** Social impact of immersive technologies and visualization **III:** Professional practice in graphics and interactive industries

Learning Outcomes I: Analyze ethical issues related to representation, accessibility, and privacy in graphics and interaction [Analyze] **II:** Discuss the social implications of immersive technologies and data visualization [Debate] **III:** Describe professional practices and standards in graphics and interactive industries [Describe]

Bibliography: [Marschner and Shirley, 2021a]

6. Bibliography

[LaValle, 2023] LaValle, S. M. (2023). *Virtual Reality*. Cambridge University Press.

[Marschner and Shirley, 2021a] Marschner, S. and Shirley, P. (2021a). *Fundamentals of Computer Graphics*. CRC Press, 5th edition.

[Marschner and Shirley, 2021b] Marschner, S. and Shirley, P. (2021b). *Fundamentals of Computer Graphics*. CRC Press, 5th edition. Same book as Marschner2021; alternate citation key using lead author Shirley.

[Noble, 2012] Noble, J. (2012). *Programming Interactivity: A Designer's Guide to Processing, Arduino, and OpenFrameworks*. O'Reilly Media, 2nd edition.

[Parent, 2012] Parent, R. (2012). *Computer Animation: Algorithms and Techniques*. Morgan Kaufmann, 3rd edition.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

CS353 Quantum Computing (Elective)

1. General information

- **Semester**: 9th Sem. **Credits**: 3
- **Hour of this course**: **Theory**: 2 hours; **Laboratory**: 2 hours;
- **Prerrequisites**:
  - CS3I1 Computer Security (8th Sem-Pg. 425)

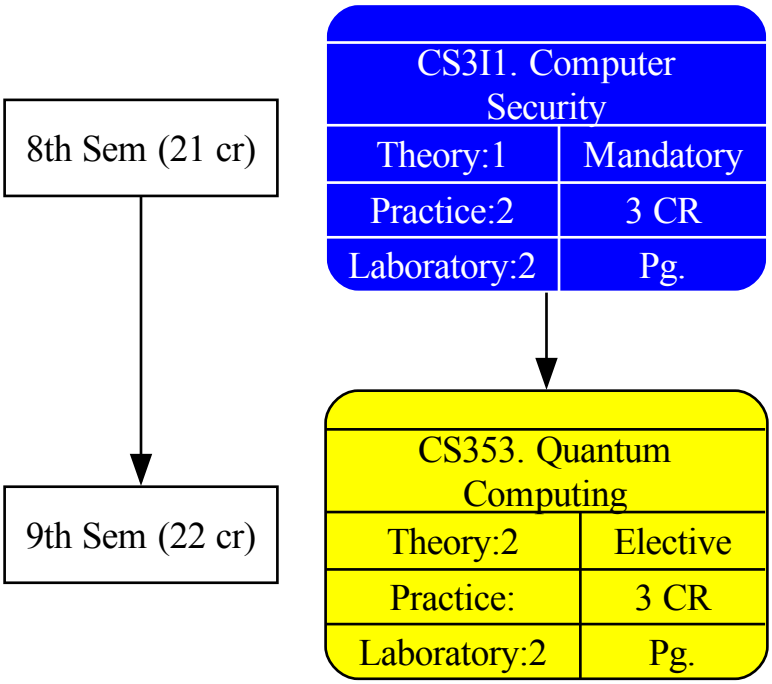

Figure 5.63: Connection Map. CS353 Quantum Computing

2. Course Rationale

This course introduces the fundamental principles of quantum computing, including qubits, superposition, entanglement, quantum algorithms, and hardware. Students will contrast classical and quantum models, exploring applications in cryptography, optimization, physical simulation, and quantum machine learning, using frameworks like Qiskit or Cirq. Includes a practical module on hardware architectures (superconductors, ion traps) and their impact on algorithm design.

3. Goals of the Course

- Understand the postulates of quantum mechanics applied to computing.
- Implement basic quantum circuits and optimization algorithms.
- Analyze the ethical, technical, and commercial impact of quantum computing.
- Evaluate hardware limitations in real-world applications.

4. General Competences (*Outcomes*)

AG-C10) Inquiry: Studies complex computing problems using information science methods. (Usage)

AG-C11) Use of Tools: Applies modern computing tools in problem solving. (Usage)

5. Content

5.1 Foundational concepts, 10 hr(s)

Competences: AG-C10,AG-C11

Topics: **I:** Qubits and Bloch sphere representation. **II:** Postulates of quantum mechanics. **III:** Quantum gates (Hadamard, CNOT, Toffoli).

Learning Outcomes **I:** Mathematically model a qubit [Usage]. **II:** Simulate basic operations in Qiskit [Assessment].

Bibliography: [Nielsen and Chuang, 2010, Team, 2023]

5.2 Quantum Architectures, 10 hr(s)

Competences: AG-C10,AG-C11

Topics: **I:** Principles: (a) The wave-particle duality principle (b) The uncertainty principle in the double-slit experiment (c) What is a Qubit? Superposition, interference, and measurement. Photons as qubits (d) Systems of two qubits, Entanglement, Bell states, The No-Signaling theorem **II:** Axioms of QM: superposition principle, measurement axiom, unitary evolution **III:** Single qubit gates for the circuit model of quantum computation: X, Z, H **IV:** Two qubit gates and tensor products, working with matrices **V:** The No-Cloning Theorem. The Quantum Teleportation protocol **VI:** Algorithms: (a) Simple quantum algorithms: Bernstein-Vazirani, Simon's algorithm (b) Implementing Deutsch-Josza with Mach-Zehnder Interferometers (c) Quantum factoring (Shor's Algorithm) (d) Quantum search (Grover's Algorithm) **VII:** Implementation aspects: (a) The physical implementation of qubits (b) Classical control of a Quantum Processing Unit (QPU) (c) Error mitigation and control, NISQ and beyond (d) Measurement approaches **VIII:** Emerging Applications: (a) Post-quantum encryption (b) The Quantum Internet (c) Adiabatic quantum computation (AQC) and quantum annealing

Learning Outcomes **I:** Discuss how a quantum object produced as a particle propagates like a wave and is detected as a particle with a probability distribution corresponding to the wave [Debate] **II:** Discuss the quantum-level nature that is inherently probabilistic [Debate] **III:** Express your view on entanglement that can be used to create non-classical correlations, but there is no way to use quantum entanglement to send messages faster than the speed of light [Evaluate] **IV:** Describe quantum parallelism and the role of constructive vs destructive interference in quantum algorithms given the probabilistic nature of measurement(s) [Describe] **V:** Analyze a code snippet providing the role of quantum Fourier transform (QFT) in Shor's algorithm [Analyze] **VI:** Write a program to implement Shor's algorithm in a simulator, highlighting the classical components and aspects of Shor's algorithm [Write] **VII:** Enumerate the specifics of each qubit modality (e.g., trapped ion, superconducting, silicon spin, photonic, quantum dot, neutral atom, topological, color center, electron-on-helium) [Enumerate] **VIII:** Contrast AQC with the gate model of quantum computation and the problems each is better suited to solve [Contrast]

Bibliography: [Möttönen and Vartiainen, 2023, IBM Research, 2023]

5.3 Algorithmic Strategies, 12 hr(s)

Competences: AG-C10,AG-C11

Topics: **I:** Quantum computing **II:** Deutsch-Jozsa algorithm. **III:** Grover's search (applied to combinatorial optimization). **IV:** Shor's factorization and its impact on RSA.

Learning Outcomes **I:** Implement Grover for SAT problems [Usage].

Bibliography: [Preskill, 2018, Team, 2023]

5.4 Formal Languages and Automata, 10 hr(s)

Competences: AG-C10,AG-C11

Topics: **I:** Quantum computation: (a) Postulates of quantum mechanics: i. State space ii. State evolution iii. State composition iv. State measurement (b) Column vector representations of qubits (c) Matrix representations of quantum operations (d) Simple quantum gates (e.g., XNOT, CNOT) **II:** Quantum Approximate Optimization Algorithm (QAOA). **III:** Logistics/finance applications (Portfolio Optimization).

Learning Outcomes **I:** For a quantum system give examples that explain the following postulates: (a) State Space - system state represented as a unit vector in Hilbert space (b) State Evolution - the use of unitary operators to evolve system state (c) State Composition - the use of tensor product to compose systems states (d) State Measurement - the probabilistic output of measuring a system state. [Familiarity]. **II:** Design a QAOA for routing problems [Assessment].

Bibliography: [Schuld and Petruccione, 2021, Bravyi et al., 2022]

5.5 Ethics and Post-Quantum Security, 6 hr(s)

Competences: AG-C10,AG-C11

Topics: **I:** Post-quantum cryptography (NIST PQC Standardization). **II:** Quantum supremacy: technical/ethical debate. **III:** Patents and quantum geopolitics.

Learning Outcomes **I:** Propose post-quantum migration strategies [Familiarity].

Bibliography: [Team, 2019]

6. Bibliography

[Bravyi et al., 2022] Bravyi, S., Dial, O., Gambetta, J. M., Gil, D., and Nazario, Z. (2022). Quantum algorithms for fixed qubit architectures. *Nature Reviews Physics*, 4(8):499–512. Survey on quantum algorithms for optimization and simulation with near-term devices.

[IBM Research, 2023] IBM Research (2023). Ibm quantum hardware roadmap. Technical report, IBM Research. Accessed: 2026-02-13. The original page has been moved or updated.

[Möttönen and Vartiainen, 2023] Möttönen, M. and Vartiainen, J. (2023). *Quantum Computing: From Qubits to Quantum Machines*. Springer.

[Nielsen and Chuang, 2010] Nielsen, M. A. and Chuang, I. L. (2010). *Quantum Computation and Quantum Information*. Cambridge University Press.

[Preskill, 2018] Preskill, J. (2018). Lecture notes on quantum computation. Ph219/CS219 course, California Institute of Technology. Material available online. The original URL `http://theory.caltech.edu/~preskill/ph219/` is no longer active (February 2026). Check the author's website for the current version.

[Schuld and Petruccione, 2021] Schuld, M. and Petruccione, F. (2021). *Machine Learning with Quantum Computers*. Springer, Cham, Switzerland, 2nd edition. Comprehensive introduction to quantum machine learning algorithms and implementations.

[Team, 2019] Team, G. A. Q. (2019). Quantum supremacy using a programmable superconducting processor. *Nature*, 574.

[Team, 2023] Team, I. Q. (2023). *Qiskit Textbook*. IBM.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

CS392 Tópicos en Ingeniería de Software (Elective)

1. General information

- **Semester**: 9th Sem. **Credits**: 3
- **Hour of this course**: **Theory**: 2 hours; **Laboratory**: 2 hours;
- **Prerrequisites**:
  - CS391 Software Engineering III (8th Sem-Pg. 455)

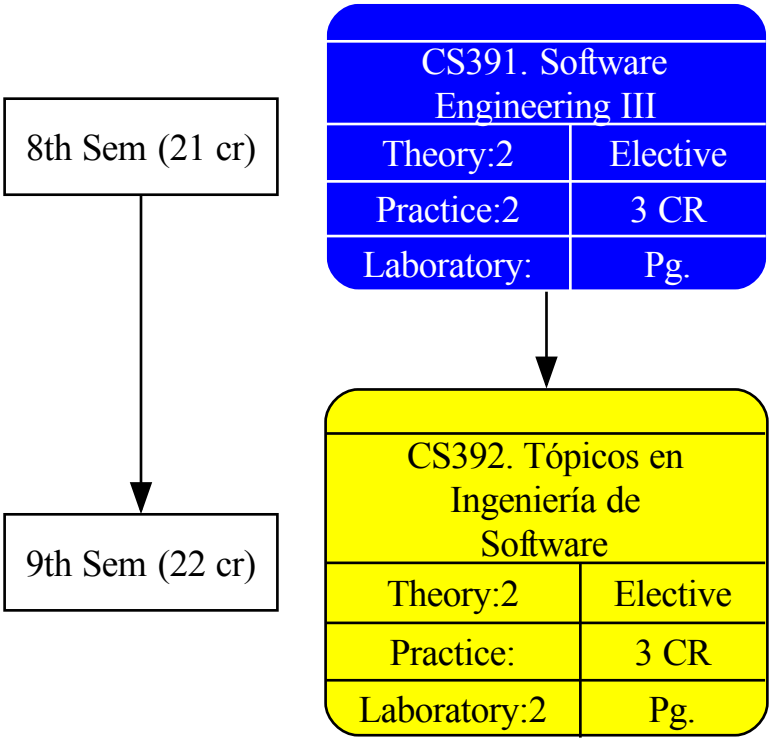

Figure 5.64: Connection Map. CS392 Tópicos en Ingeniería de Software

2. Course Rationale

Modern software development requires rigorous engineering practices to ensure quality, maintainability, and reliability. This course explores advanced topics in the software lifecycle, focusing on team collaboration, formal verification, and advanced refactoring. Students will master the use of professional tools to design, construct, and validate complex systems while ensuring high standards of reliability and formal correctness.

3. Goals of the Course

- Apply advanced software construction and refactoring techniques to improve code quality.
- Use formal methods and validation strategies to ensure system reliability.
- Manage complex requirements and design architectures that meet stakeholder needs.
- Lead software teams using professional tools and collaborative workflows.

4. General Competences (*Outcomes*)

AG-C03) Individual and Team Work: Performs effectively as an individual and as a member or leader in diverse teams. (Usage)

AG-C09) Design and Development of Solutions: Designs, implements, and evaluates solutions for complex computing problems. (Usage)

5. Content

5.1 Team Collaboration and Dynamics, 4 hr(s)

Competences: AG-C03,AG-C09

Topics: **I:** Effective communication, including oral and written, as well as formal (email, docs, comments, presentations) and informal (team chat, meetings). Technical and Professional Communication, Team Communication and Culture **II:** Common causes of team conflict, and approaches for conflict resolution. **III:** Cooperative programming: (a) Pair programming or Swarming (b) Code review (c) Collaboration through version control **IV:** Roles and responsibilities in a software team: Ethics Codes and Legal Accountability, Professional Responsibility and Public Policy, Professional Values, Nature and Continuing Development, Ethical Dissent and Cultural Context, Workplace Ethics (a) Advantages of teamwork (b) Risks and complexity of such collaboration **V:** Team processes - responsibilities for tasks, effort estimation, meeting structure, work schedule

Learning Outcomes **I:** Follow effective team communication practices [Analyze] **II:** Articulate the sources of, hazards of, and potential benefits of team conflict - especially focusing on the value of disagreeing about ideas or proposals without insulting people [Articulate] **III:** Facilitate a conflict-resolution and problem-solving strategy in a team setting [Analyze] **IV:** Collaborate effectively in cooperative development/programming [Analyze] **V:** Propose and delegate necessary roles and responsibilities in a software development team [Propose] **VI:** Compose and follow an agenda for a team meeting [Compose] **VII:** Facilitate through involvement in a team project, the central elements of team building, establishing healthy team culture, and team management including creating and executing a team work plan [Create]

Bibliography: [Sommerville, 2015]

5.2 Version Control and CI/CD, 4 hr(s)

Competences: AG-C03,AG-C09

Topics: **I:** Software configuration management and version control: Software Development Practices (a) Configuration in version control, reproducible builds/configuration. (b) Version control branching strategies. Development branches vs release branches. Trunk-based development. (c) Merging/rebasing strategies, when relevant. **II:** Release management. **III:** Testing tools including static and dynamic analysis tools. Software Development Practices, Information Flow and Non-Interference, Injection and Input Validation, Memory Safety and Types, Malware Analysis and Advanced Security **IV:** Software process automation: (a) Build systems - the value of fast, hermetic, reproducible builds, compare/contrast approaches to building a project. (b) Continuous Integration (CI) - the use of automation and automated tests to do preliminary validation that the current head/trunk revision builds and passes (basic) tests. (c) Continuous Deployment (CD) - the use of automation to automatically release every change that passes the automated tests to the production environment, ensuring frequent and reliable deliveries. (d) Dependency management - updating external/upstream dependencies, package management, SemVer.
V: Collaborative review workflow as a quality gate (a) Mandatory pull-request review before merging (b) Code ownership via CODEOWNERS-style rules (c) Conventional commit message conventions (d) Branch-protection rules enforcing checks before merge **VI:** Dependency and supply-chain vulnerability scanning integrated into CI pipelines

Learning Outcomes I: Describe the difference between centralized and distributed software configuration management [Describe] **II:** Describe how version control can be used to help manage software release management [Describe] **III:** Identify configuration items and use a source code control tool in a small team-based project [Analyze] **IV:** Describe how available static and dynamic test tools can be integrated into the software development environment [Describe] **V:** Understand the use of CI/CD systems as a ground-truth for the state of the team's shared code (build and test success) [Explain] **VI:** Apply a pull-request review workflow with code-ownership rules and branch protection to enforce quality gates on a team project [Apply] **VII:** Use a dependency/supply-chain vulnerability scanner as part of a CI pipeline to detect vulnerable dependencies [Use]

Bibliography: [Pressman and Maxim, 2014]

5.3 Requirements Engineering, 6 hr(s)

Competences: AG-C03,AG-C09

Topics: I: Describe functional requirements using, for example, use cases or user stories. (a) Using at least one method of documenting and structuring functional requirements. (b) Understanding how the method supports design and implementation. (c) Strengths and weaknesses of using a specific approach. **II:** Properties of requirements including consistency, validity, completeness, and feasibility. **III:** Requirements elicitation. (a) Sources of requirements, for example, users, administrators, or support personnel. (b) Methods of requirement gathering, for example, surveys, interviews, or behavioral analysis. **IV:** Non-functional requirements, for example, security, usability, or performance, also called as Quality Attributes. Sustainable Design and Pervasive Computing, Environmental Footprint of Computing Systems, Systemic Effects and Social Context **V:** Risk identification and management, including ethical considerations surrounding the proposed product. Ethics Codes and Legal Accountability, Professional Responsibility and Public Policy, Professional Values, Nature and Continuing Development, Ethical Dissent and Cultural Context, Workplace Ethics **VI:** Communicating and/or formalizing requirement specifications.

Learning Outcomes I: Compare different methods of eliciting requirements along multiple axes [Compare] **II:** Identify differences between two methods of describing functional requirements (e.g., customer interviews, user studies) and the situations where each would be preferred [Analyze] **III:** Identify which behaviors are required, allowed, or barred from a given set of requirements and a list of candidate behaviors [Analyze] **IV:** Collect a set of requirements for a simple software system [Analyze] **V:** Identify areas of a software system that need to be changed, given a description of the system and a set of new requirements to be implemented [Analyze] **VI:** Identify the functional and non-functional requirements in a set of requirements [Analyze]

Bibliography: [Sommerville, 2015]

5.4 Principles and Software Architecture, 6 hr(s)

Competences: AG-C03,AG-C09

Topics: I: System design principles. System Reliability (a) Levels of abstraction (e.g., architectural design and detailed design) (b) Separation of concerns (c) Information hiding (d) Coupling and cohesion **II:** Software architecture. System Reliability (a) Design paradigms i. Top-down functional decomposition/layered design ii. Data-oriented architecture iii. Object-oriented analysis and design iv. Event-driven design (b) Standard architectures (e.g., client-server and microservice architectures including REST discussions, n-layer, pipes-and-filters, Model View Controller) (c) Identifying component boundaries and dependencies **III:** Programming in the large vs programming in the small. System Reliability **IV:** Code smells and other indications of code quality, distinct from correctness. Security Design and Controls Engineering, Threat Analysis and Security Engineering, Trusted Computing and Privacy Engineering

Learning Outcomes **I:** Identify the standard software architecture of a given high-level design [Analyze] **II:** Select and use an appropriate design paradigm to design a simple software system and explain how system design principles have been applied in this design [Evaluate] **III:** Adapt a flawed system design to better follow principles such as separation of concerns or information hiding [Create] **IV:** Identify the dependencies among a set of software components in an architectural design [Analyze]

Bibliography: [Sommerville, 2015]

5.5 Coding Practices, 6 hr(s)

Competences: AG-C03,AG-C09

Topics: **I:** Practical small-scale testing Software Development Practices (a) Unit testing (b) Test-driven development - This is particularly valuable for students psychologically, as it is far easier to engage constructively with the challenge of identifying challenging inputs for a given API (edge cases, corner cases) a priori. If they implement first, the instinct is often to avoid trying to crash their new creation, while a test-first approach gives them the intellectual satisfaction of spotting the problem cases and then watching as more tests pass during the development process. **II:** Documentation Software Development Practices (a) Interface documentation - describe interface requirements, potentially including (formal or informal) contracts, pre and post conditions, invariants. (b) Implementation documentation should focus on tricky and non-obvious pieces of code, whether because the code is using advanced language features, or the behavior of the code is complex. (Do not add comments that re-state common/obvious operations and simple language features.) i. Clarify dataflow, computation, etc., focusing on what the code is. ii. Identify subtle/tricky pieces of code and refactor to be self-explanatory if possible or provide appropriate comments to clarify. **III:** Coding style Software Development Practices (a) Style guides (b) Commenting (c) Naming **IV:** "Best Practices" for coding: techniques, idioms/patterns, mechanisms for building quality programs Software Development Practices, Information Flow and Non-Interference, Injection and Input Validation, Memory Safety and Types, Malware Analysis and Advanced Security (a) Defensive coding practices (b) Secure coding practices and principles (c) Using exception handling mechanisms to make programs more robust, fault-tolerant **V:** Debugging Software Development Practices **VI:** Logging **VII:** Use of libraries and frameworks developed by others Software Development Practices

Learning Outcomes **I:** Write appropriate unit tests for a small component (several functions, a single type, etc.) [Write] **II:** Write appropriate interface and (if needed) implementation comments for a small component [Write] **III:** Describe techniques, coding idioms and mechanisms for implementing designs to achieve desired properties such as reliability, efficiency, and robustness [Describe] **IV:** Write robust code using exception handling mechanisms [Write] **V:** Describe secure coding and defensive coding practices [Describe] **VI:** Select and use a defined coding standard in a small software project [Use] **VII:** Compare and contrast integration strategies including top-down, bottom-up, and sandwich integration [Compare] **VIII:** Describe the process of analyzing and implementing changes to code base developed for a specific project [Describe] **IX:** Describe the process of analyzing and implementing changes to a large existing code base [Describe]

Bibliography: [Fowler, 2018]

5.6 API Compatibility and Versioning, 6 hr(s)

Competences: AG-C03,AG-C09

Topics: **I:** Hyrum's Law/The Law of Implicit Interfaces **II:** Backward compatibility (a) Compatibility is not a property of a single entity, it's a property of a relationship. (b) Backward compatibility needs to be evaluated in terms of provider + consumer(s) or with a well-specified model of what forms of compatibility a provider aspires to/promises. **III:** Versioning (a) Semantic Versioning (SemVer) (b) Trunk-based development

Learning Outcomes **I:** Identify both explicit and implicit behavior of an interface and identify potential risks from Hyrum's Law [Analyze] **II:** Identify changes that can be broadly considered "backward compatible," potentially with explicit statements about what usage is or is not supported [Analyze] **III:** Evaluate whether a proposed change is sufficiently safe given the versioning methodology in use for a given project [Evaluate]

Bibliography: [Fowler, 2018]

5.7 Verification and validation Concepts, 6 hr(s)

Competences: AG-C03,AG-C09

Topics: **I:** Verification and validation concepts (a) Verification: Are we building the thing right? (b) Validation: Did we build the right thing? **II:** Why testing matters: Does the component remain functional as the code evolves? **III:** Testing objectives (a) Usability (b) Reliability (c) Conformance to specification (d) Performance (e) Security

Learning Outcomes **I:** Explain why testing is important [Explain] **II:** Distinguish between program validation and verification [Distinguish] **III:** Describe different objectives of testing [Describe]

Bibliography: [Pressman and Maxim, 2014]

5.8 Testing Tools and Analysis, 3 hr(s)

Competences: AG-C03,AG-C09

Topics: **I:** Verification and validation tooling and automation (a) Static analysis (b) Code coverage (c) Fuzzing (d) Dynamic analysis and fault containment (sanitizers, etc.) (e) Fault logging and fault tracking **II:** Test planning and generation (a) Fault estimation and testing termination including defect seeding (b) Use of random and pseudo random numbers in testing **III:** Testing asynchronous, parallel, and concurrent systems **IV:** Verification and validation of non-code artifacts (documentation, training materials) **V:** Behavior-driven development (BDD) (a) Expressing acceptance criteria as Gherkin Given/When/Then scenarios (b) Executing BDD scenarios as automated end-to-end tests **VI:** Mutation testing and static code-quality analysis (a) Mutation testing to assess test-suite effectiveness (b) Static detection of code smells and excessive cyclomatic complexity

Learning Outcomes **I:** Describe and compare different tools for verification and validation [Describe] **II:** Automate the use of different tools in a small software project [Design] **III:** Explain how and when random numbers should be used in testing [Explain] **IV:** Describe approaches for fault estimation [Describe] **V:** Estimate the number of faults in a small software application based on fault density and fault seeding [Estimate] **VI:** Describe techniques and issues with testing asynchronous, concurrent, and parallel software [Describe] **VII:** Create a test plan for a medium-size code segment which contains asynchronous, concurrent, and/or parallel code, including a test matrix and generation of test data and inputs [Create] **VIII:** Describe techniques for the verification and validation of non-code artifacts [Describe] **IX:** Write Gherkin Given/When/Then scenarios for a feature's acceptance criteria and execute them as automated end-to-end tests [Write] **X:** Analyze a test suite's mutation score and refactor tests to kill surviving mutants [Analyze]

Bibliography: [Pressman and Maxim, 2014]

5.9 Performance Testing and Benchmarking, 3 hr(s)

Competences: AG-C03,AG-C09

Topics: **I:** Performance testing and benchmarking (a) Throughput and latency (b) Degradation under load (stress testing, FIFO vs LIFO handling of requests) (c) Speedup and scaling i. Amdahl's law ii. Gustafson's law iii. Soft and weak scaling (d) Identifying and measuring figures of

merits (e) Common performance bottlenecks i. Compute-bound ii. Memory-bandwidth bound iii. Latency-bound (f) Statistical methods and best practices for benchmarking i. Estimation of uncertainty ii. Confidence intervals (g) Analysis and presentation (graphs, etc.) (h) Timing techniques

Learning Outcomes **I:** Describe throughput and latency and provide examples of each [Describe] **II:** Explain speedup and the different forms of scaling and how they are computed [Explain] **III:** Describe common performance bottlenecks [Describe] **IV:** Describe statistical methods and best practices for benchmarking software [Describe] **V:** Explain techniques for and challenges with measuring time when constructing a benchmark [Explain] **VI:** Identify the figures of merit, construct and run a benchmark, and statistically analyze and visualize the results for a small software project [Analyze]

Bibliography: [Bondi, 2015, Gregg, 2020]

5.10 Reliability Concepts, 5 hr(s)

Competences: AG-C03,AG-C09

Topics: **I:** Concept of reliability as probability of failure or mean time between failures, and faults as cause of failures **II:** Identifying reliability requirements for different kinds of software **III:** Software failures caused by defects/bugs, and so for high reliability the goal is to have minimum defects - by injecting fewer defects (better training, education, planning), and by removing most of the injected defects (testing, code review, etc.) **IV:** Software reliability, system reliability and failure behavior **V:** Defect injection and removal cycle, and different approaches for defect removal **VI:** Compare the "error budget" approach to reliability with the "error-free" approach and identify domains where each is relevant.

Learning Outcomes **I:** Describe how to determine the level of reliability required by a software system [Describe] **II:** Explain the problems that exist in achieving very high levels of reliability [Explain] **III:** Understand approaches to minimizing faults that can be applied at each stage of the software lifecycle [Explain]

Bibliography: [Pressman and Maxim, 2014]

5.11 Formal Methods, 5 hr(s)

Competences: AG-C03,AG-C09

Topics: **I:** Formal specification of interfaces (a) Specification of pre- and post- conditions (b) Formal languages for writing and analyzing pre- and post-conditions. **II:** Problem areas well served by formal methods (a) Lock-free programming, data races (b) Asynchronous and distributed systems, deadlock, livelock, etc. **III:** Comparison to other tools and techniques for defect detection (a) Testing (b) Fuzzing **IV:** Formal approaches to software modeling and analysis (a) Model checkers (b) Model finders

Learning Outcomes **I:** Describe the role formal specification and analysis techniques can play in the development of complex software and compare their use as validation and verification techniques with testing [Describe] **II:** Apply formal specification and analysis techniques to software designs and programs with low complexity [Apply] **III:** Explain the potential benefits and drawbacks of using formal specification languages [Explain]

Bibliography: [Sommerville, 2015]

6. Bibliography

[Bondi, 2015] Bondi, A. B. (2015). *Foundations of Software and System Performance Engineering: Process, Performance Modeling, Requirements, Testing, Scalability, and Practice*. Addison-Wesley, Upper Saddle River, NJ.

[Fowler, 2018] Fowler, M. (2018). *Refactoring: Improving the Design of Existing Code*. Addison-Wesley Professional, 2nd edition.

[Gregg, 2020] Gregg, B. (2020). *Systems Performance: Enterprise and the Cloud*. Addison-Wesley Professional, Boston, MA, 2nd edition.

[Pressman and Maxim, 2014] Pressman, R. S. and Maxim, B. (2014). *Software Engineering: A Practitioner's Approach*. McGraw-Hill, 8th edition.

[Sommerville, 2015] Sommerville, I. (2015). *Software Engineering*. Pearson, 10th edition.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

CS3P3 Internet of Things (Elective)

1. General information

- **Semester**: 9th Sem. **Credits**: 3
- **Hour of this course**: **Theory**: 1 hours; **Laboratory**: 4 hours;
- **Prerrequisites**:
  - CS3P1 Parallel and Distributed Computing (8th Sem-Pg. 430)

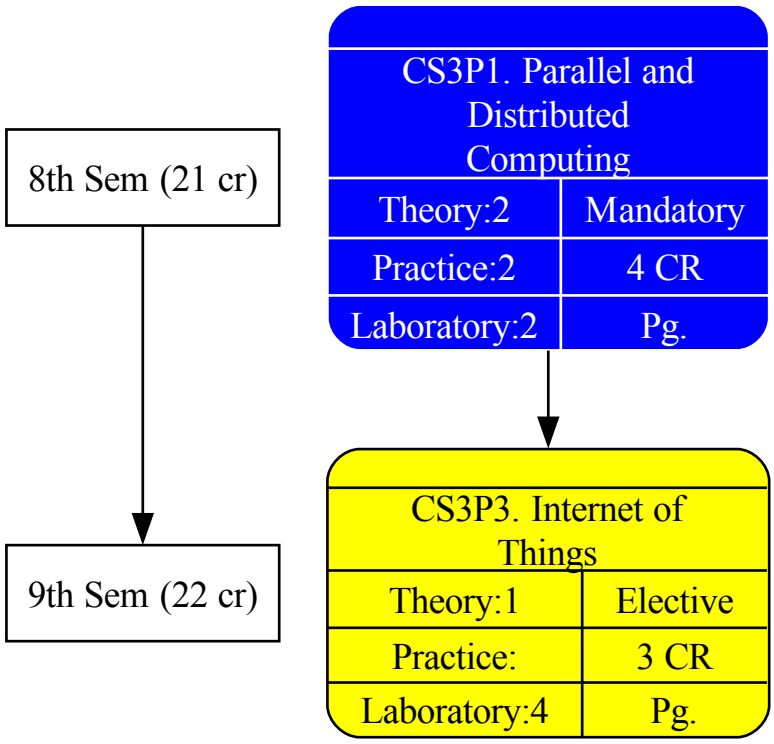

Figure 5.65: Connection Map. CS3P3 Internet of Things

2. Course Rationale

The Internet of Things (IoT) has transformed how we interact with the physical world by connecting billions of devices. This course explores the architectural foundations of IoT, emphasizing device mobility, wireless communication, and the societal implications of pervasive sensing. Students will analyze the technical challenges of constrained systems and the ethical responsibilities associated with large-scale data collection in interconnected environments.

3. Goals of the Course

- Understand the architecture of IoT systems and the role of mobility in network design.
- Evaluate the societal and ethical impacts of IoT technologies.
- Implement communication protocols for constrained and mobile devices.
- Analyze security risks in the lifecycle of an IoT ecosystem.

4. General Competences (*Outcomes*)

AG-C09) Design and Development of Solutions: Designs, implements, and evaluates solutions for complex computing problems. (Usage)

AG-C11) Use of Tools: Applies modern computing tools in problem solving. (Usage)

5. Content

5.1 Mobility Support, 12 hr(s)

Competences: AG-C09,AG-C11

Topics: **I:** Principles of cellular communication (e.g., 4G, 5G) **II:** Principles of Wireless LANs (mainly 802.11) **III:** Device to device communication (e.g., IoT communication) **IV:** Multi-hop wireless networks (e.g., ad hoc networks, opportunistic, delay tolerant)

Learning Outcomes **I:** Describe some aspects of cellular communication such as registration [Describe] **II:** Describe how 802.11 supports mobile users [Describe] **III:** Describe practical uses of device-to-device communication, as well as multihop [Describe] **IV:** Describe one type of mobile network such as ad hoc [Describe]

Bibliography: [Hanes et al., 2017, Buyya and Dastjerdi, 2016]

5.2 Security Ethics, Law and Society, 12 hr(s)

Competences: AG-C09,AG-C11

Topics: **I:** Societal impacts of Internet of Things (IoT) devices and other emerging technologies on security and privacy Civil Liberties and Expression, Data, Surveillance and Privacy Technology, Privacy Philosophical Foundations and Legislation, Computer Crimes and Attacks, Security Laws, Policies and Responsibility. **II:** Privacy by design and privacy engineering Civil Liberties and Expression, Data, Surveillance and Privacy Technology, Privacy Philosophical Foundations and Legislation, Computer Crimes and Attacks, Security Laws, Policies and Responsibility.

Learning Outcomes **I:** Analyze the societal impacts of pervasive IoT surveillance [Assessment]. **II:** Apply privacy-by-design principles to IoT data architectures [Usage].

Bibliography: [Hanes et al., 2017]

5.3 Networked Applications, 24 hr(s)

Competences: AG-C09,AG-C11

Topics: **I:** Distributed application paradigms (e.g., client/server, peer-to-peer, cloud, edge, and fog). See also: Communication, Coordination **II:** Protocols for constrained networks (CoAP, MQTT, LoRaWAN).

Learning Outcomes **I:** Define the principles of naming, addressing, resource location [Define] **II:** Analyze the needs of specific networked application demands [Analyze] **III:** Describe the details of one application layer protocol [Describe] **IV:** Implement a simple client-server socket-based application [Implement]

Bibliography: [Hanes et al., 2017]

6. Bibliography

[Buyya and Dastjerdi, 2016] Buyya, R. and Dastjerdi, A. V. (2016). *Internet of Things: Principles and Paradigms*. Morgan Kaufmann.

[Hanes et al., 2017] Hanes, D., Salgueiro, G., and Grossetete, P. (2017). *IoT Fundamentals: Networking Technologies, Protocols, and Use Cases for the Internet of Things*. Cisco Press.

Tenth Semester

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

CS3P2 Cloud Computing (Mandatory)

1. General information

- **Semester**: 10th Sem. **Credits**: 3
- **Hour of this course**: **Theory**: 1 hours; **Practice**: 2 hours; **Laboratory**: 2 hours;
- **Prerrequisites**:
  - CS370 Big Data (9th Sem-Pg. 475)

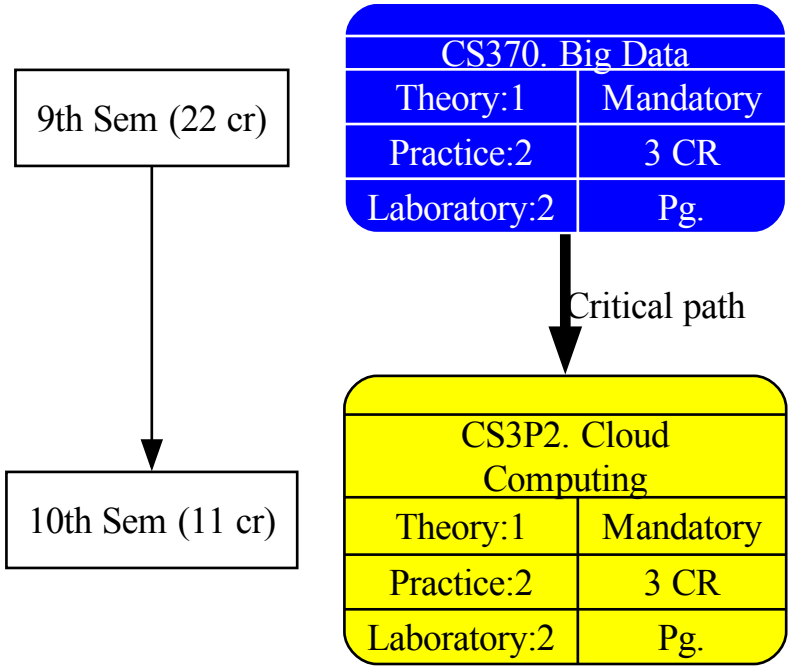

Figure 5.66: Connection Map. CS3P2 Cloud Computing

2. Course Rationale

Modern computing relies heavily on distributed architectures and cloud-native platforms. This course provides an in-depth exploration of cloud computing services, distributed database concepts, and web platforms. Students will learn to design scalable architectures, understand the implications of distributed data consistency, and evaluate the privacy and security challenges inherent in cloud-based environments.

3. Goals of the Course

- Understand the architecture and deployment models of cloud computing.
- Design distributed database solutions considering consistency and availability.
- Develop applications using modern cloud-based web platforms and frameworks.
- Analyze security and privacy implications in distributed and cloud services.

4. General Competences (*Outcomes*)

AG-C09) Design and Development of Solutions: Designs, implements, and evaluates solutions for complex computing problems. (Usage)

AG-C11) Use of Tools: Applies modern computing tools in problem solving. (Usage)

5. Content

5.1 Cloud Architecture and Data Management, 12 hr(s)

Competences: AG-C09,AG-C11

Topics: **I:** Web platforms, frameworks, or meta-frameworks (a) Cloud services (b) API, Web Components **II:** Computing services NoSQL Systems (a) Cloud Hosting (b) Scalability (e.g., Autoscaling, Clusters) (c) Cost estimation for services **III:** Architecture (a) Monoliths vs Microservices (b) Micro-frontends (c) Event-Driven vs RESTful architectures: advantages and disadvantages (d) Serverless, cloud computing on demand

Learning Outcomes **I:** Design and implement a web-based application using a microservice architecture design [Design] **II:** Describe the constraints, such as hosting, services, and scalability, related to web platforms [Describe]

Bibliography: [Marinescu, 2022, Buyya et al., 2013]

5.2 Distributed Databases/Cloud Computing, 10 hr(s)

Competences: AG-C09,AG-C11

Topics: **I:** Distributed databases/cloud-based systems

Learning Outcomes **I:** Describe the key components of a distributed DBMS, including distributed data storage, query processing, and transaction management [Describe] **II:** Analyze the trade-offs between parallel DBMS architectures: shared memory, shared disk, and shared nothing [Analyze] **III:** Describe data replication strategies and weak consistency models in distributed database systems [Describe]

Bibliography: [Buyya et al., 2013]

5.3 Parallel and Distributed Computing, 8 hr(s)

Competences: AG-C09,AG-C11

Topics: **I:** Safety and liveness: (a) Race conditions (b) Dependencies/preconditions (c) Fault models (d) Termination **II:** Programming models: (a) Actor models (b) Procedural and reactive models (c) Synchronous/asynchronous programming models (d) Data parallelism **III:** Properties: (a) Order-based properties: i. Commutativity ii. Independence (b) Consistency-based properties: i. Atomicity ii. Consensus **IV:** Execution control: (a) Async await (b) Promises (c) Threads **V:** Communication and coordination: (a) Mutexes (b) Message-passing (c) Shared memory (d) Cobegin-coend (e) Monitors (f) Channels (g) Threads (h) Guards **VI:** Futures **VII:** Language support for data parallelism such as forall, loop unrolling, map/reduce **VIII:** Effect of memory-consistency models on language semantics and correct code generation **IX:** Representational State Transfer Application Programming Interfaces (REST APIs) **X:** Technologies and approaches: cloud computing, high performance computing, quantum computing, ubiquitous computing **XI:** Overheads of message-passing **XII:** Granularity of program for efficient exploitation of concurrency **XIII:** Concurrency and other programming paradigms (e.g., functional)

Learning Outcomes I: Explain why programming languages do not guarantee sequential consistency in the presence of data races and what programmers must do as a result [Explain] **II:** Implement correct concurrent programs using multiple programming models, such as shared memory, actors, futures, synchronization constructs, and data-parallelism primitives [Implement] **III:** Use a message-passing model to analyze a communication protocol [Use] **IV:** Use synchronization constructions such as monitor/synchronized methods in a simple program [Use] **V:** Modeling data dependency using simple programming constructs involving variables, read and write [Implement] **VI:** Modeling control dependency using simple constructs such as selection and iteration [Implement] **VII:** Explain how REST API's integrate applications and automate processes [Explain] **VIII:** Explain benefits, constraints and challenges related to distributed and parallel computing [Explain]

Bibliography: [Marinescu, 2022]

5.4 Networked Applications, 6 hr(s)

Competences: AG-C09,AG-C11

Topics: I: Distributed application paradigms (e.g., client/server, peer-to-peer, cloud, edge, and fog). See also: Communication, Coordination

Learning Outcomes I: Define the principles of naming, addressing, resource location [Define] **II:** Analyze the needs of specific networked application demands [Analyze] **III:** Describe the details of one application layer protocol [Describe] **IV:** Implement a simple client-server socket-based application [Implement]

Bibliography: [Marinescu, 2022]

5.5 Algorithms: Application Domains Survey, 4 hr(s)

Competences: AG-C09,AG-C11

Topics: I: Survey of common application domains across multicore, reactive, data parallel, cluster, cloud, open distributed systems, and frameworks (with reference to the following table). (a) Multicore: Typical Execution agents: Threads. Typical Communication mechanisms: Shared memory, Atomics, locks. Typical Algorithmic domains: Resource management, data processing. Typical Engineering goals: Throughput, latency, energy. (b) Reactive: Typical Execution agents: Handlers, threads. Typical Communication mechanisms: I/O Channels. Typical Algorithmic domains: Services, real-time. Typical Engineering goals: Latency. (c) Data parallel: Typical Execution agents: GPU, SIMD, accelerators, hybrid. Typical Communication mechanisms: Heterogeneous memory. Typical Algorithmic domains: Linear algebra, graphics, data analysis. Typical Engineering goals: Throughput, energy. (d) Cluster: Typical Execution agents: Managed hosts. Typical Communication mechanisms: Sockets, channels. Typical Algorithmic domains: Simulation, data analysis. Typical Engineering goals: Throughput. (e) Cloud: Typical Execution agents: Provisioned hosts. Typical Communication mechanisms: Service APIs. Typical Algorithmic domains: Web applications. Typical Engineering goals: Scalability. (f) Open distributed: Typical Execution agents: Autonomous hosts. Typical Communication mechanisms: Sockets, Data stores. Typical Algorithmic domains: Fault tolerant data stores and services. Typical Engineering goals: Reliability.

Learning Outcomes I: Extend an event-driven sequential program by establishing a new activity in an event handler (for example a new thread in a GUI action handler) [Design] **II:** Improve the performance of a sequential component by introducing parallelism and/or distribution [Create] **III:** Choose among different parallel/distributed designs for components of a given system [Assess]

Bibliography: [Marinescu, 2022]

5.6 Security Mindset and Principles, 8 hr(s)

Competences: AG-C09,AG-C11

Topics: I: Applications of a security mindset: web, cloud, and mobile devices System Design, Common Aspects: Platforms, APIs and Constraints, Common Aspects: Platform Languages and Web Programming Patterns. **II:** Privacy implications of widespread data collection including but not limited to transactional databases, data warehouses, surveillance systems, cloud computing, and artificial intelligence.

Learning Outcomes I: Analyze the security risks associated with distributed cloud applications [Usage]. **II:** Evaluate the privacy implications of data storage in multi-tenant cloud environments [Assessment].

Bibliography: [Marinescu, 2022]

6. Bibliography

[Buyya et al., 2013] Buyya, R., Vecchiola, C., and Selvi, S. T. (2013). *Mastering Cloud Computing: Foundations and Applications Programming.* Morgan Kaufmann, 1st edition.

[Marinescu, 2022] Marinescu, D. C. (2022). *Cloud Computing: Theory and Practice.* Morgan Kaufmann, 3rd edition.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

CS404 Research Workshop (Mandatory)

1. General information

- **Semester**: 10th Sem. **Credits**: 4
- **Hour of this course**: **Theory**: 2 hours; **Laboratory**: 4 hours;
- **Prerrequisites**:
  - CS403 Capstone Project II (9th Sem-Pg. 480)

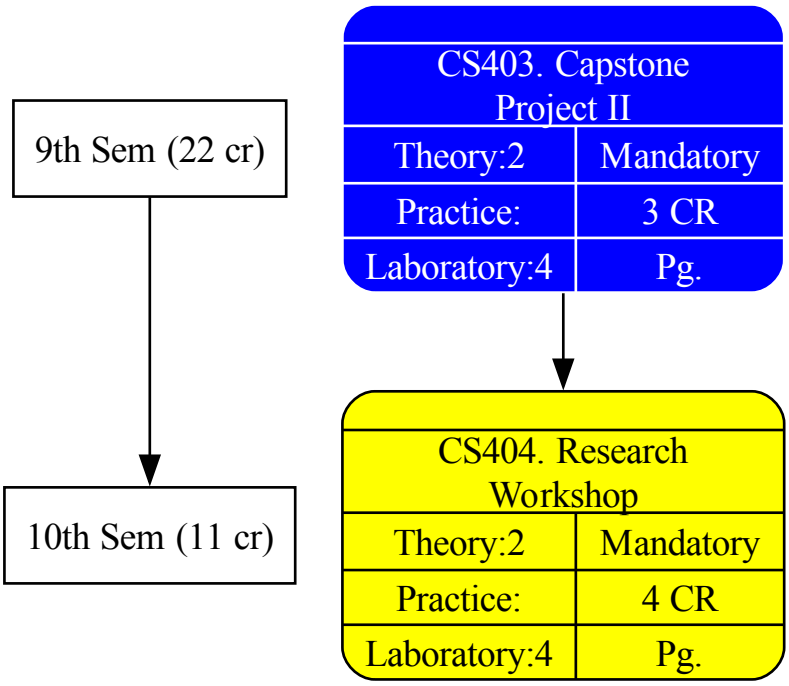

Figure 5.67: Connection Map. CS404 Research Workshop

2. Course Rationale

This course aims to enable students to complete their thesis with sufficient quality for public defense, finishing experiments, final chapters, advisor revisions, and the formal institutional submission.

3. Goals of the Course

- Complete the experimental part of the thesis with statistical rigor and full analysis of results.
- Write the final thesis chapters: conclusions, contributions, and future work.
- Ensure the document complies with the institutional format and has complete narrative coherence.
- Incorporate advisor revisions until obtaining an approved document ready for defense.
- Prepare and deliver the formal thesis presentation before the faculty committee.
- The deliverables for this course are:

Midterm: Advanced draft with completed experiments, introduction, theoretical framework, proposal, and results chapters in LaTeX.

Final: Complete thesis document approved by the advisor and ready for public defense.

4. General Competences (*Outcomes*)

AG-C02) Ethics: Applies ethical principles and commits to professional ethics and the standards of professional computing practice. (Assessment)

AG-C04) Communication: Communicates effectively in complex computing activities. (Assessment)

AG-C05) Project Management: Applies management principles in computing to manage projects. (Assessment)

AG-C08) Problem Analysis: Identifies, formulates, and analyzes complex computing problems. (Assessment)

AG-C09) Design and Development of Solutions: Designs, implements, and evaluates solutions for complex computing problems. (Assessment)

5. Content

5.1 Completion of Experiments and Results Analysis, 12 hr(s)

Competences: AG-C05,AG-C08,AG-C09

Topics: **I:** Full execution of the experimental protocol defined in the thesis plan. **II:** Final statistical analysis: significance tests, confidence intervals, and effect size. **III:** Objective comparison with baselines and state-of-the-art using established metrics. **IV:** Interpretation of results: findings, limitations, and threats to validity. **V:** Reproducible documentation: code, datasets, configurations, and results.

Learning Outcomes **I:** Complete the execution of the experimental protocol covering all defined cases. [Assessment] **II:** Analyze results with statistical rigor and interpret findings with respect to the hypothesis. [Assessment] **III:** Document experiments, configurations, and results in a reproducible manner. [Usage]

Bibliography: [IEEE-Computer Society, 2008, ACM, 2008]

5.2 Writing the Final Thesis Chapters, 12 hr(s)

Competences: AG-C02,AG-C04,AG-C09

Topics: **I:** Writing the results chapter: high-quality tables, figures, and comparative analysis. **II:** Writing the conclusions chapter: findings, original contributions, and limitations. **III:** Writing the future work chapter: extensions and open research problems. **IV:** Writing the final introduction: motivation, problem statement, and contributions summary. **V:** Writing the executive abstract in both Spanish and English.

Learning Outcomes **I:** Write the results chapter with clear visualizations and comparative analysis. [Assessment] **II:** Formulate the conclusions and original contributions of the thesis with precision. [Assessment] **III:** Write the abstract and final introduction articulating the problem and the contributions. [Assessment]

Bibliography: [IEEE-Computer Society, 2008, ACM, 2008]

5.3 Document Formatting, Integration, and Quality, 10 hr(s)

Competences: AG-C02,AG-C04,AG-C09

Topics: **I:** Verification of compliance with the institutional thesis format in LaTeX. **II:** Quality review of figures, tables, and equations in the final document. **III:** Narrative and terminological consistency across all thesis chapters. **IV:** Review and update of bibliography: coverage, BibTeX format, and correct citations. **V:** Generation of the final document: PDF, cover page, indices, and appendices.

Learning Outcomes **I:** Verify that the document fully complies with the institutional thesis format. [Assessment] **II:** Correct the narrative and terminological consistency across all chapters. [Assessment] **III:** Produce the final PDF document with cover page, indices, and complete bibliography. [Usage]

Bibliography: [IEEE-Computer Society, 2008, ACM, 2008]

5.4 Advisor Review and Correction Cycles, 10 hr(s)

Competences: AG-C02,AG-C08,AG-C09

Topics: **I:** Structured revision dynamic: submission, review, and response to the advisor. **II:** Incorporation of advisor feedback: prioritization and argued response. **III:** Review of proposal soundness and experimental validity according to the advisor. **IV:** Handling academic disagreements with the advisor constructively and ethically. **V:** Advisor approval criteria for authorizing the thesis defense.

Learning Outcomes **I:** Systematically incorporate advisor observations with argued responses. [Assessment] **II:** Improve the scientific soundness of the document through revision cycles. [Assessment] **III:** Obtain formal advisor approval for the thesis defense authorization. [Assessment]

Bibliography: [IEEE-Computer Society, 2008, ACM, 2008]

5.5 Defense Preparation and Rehearsal, 8 hr(s)

Competences: AG-C02,AG-C04,AG-C05

Topics: **I:** Design of the oral presentation: structure, narrative, and time management. **II:** Preparation of technical slides: figures, results, and key comparisons. **III:** Defense techniques for jury questions: argumentation and objection handling. **IV:** Defense rehearsal with the advisor and/or academic peers. **V:** Stress management and effective communication in academic evaluation contexts.

Learning Outcomes **I:** Design and deliver a clear and structured oral presentation of the thesis work. [Assessment] **II:** Defend methodology and results before critical questions from the jury. [Assessment] **III:** Demonstrate full command of the topic through prior defense rehearsals. [Usage]

Bibliography: [IEEE-Computer Society, 2008, ACM, 2008]

5.6 Formal Submission and Pre-Defense, 8 hr(s)

Competences: AG-C02,AG-C04,AG-C05

Topics: **I:** Institutional thesis submission process: documentation and regulatory deadlines. **II:** Predefense before faculty authorities: evaluation of the final draft. **III:** Incorporation of pre-defense committee corrections into the document. **IV:** Administrative procedures: jury designation and defense scheduling. **V:** Ethics in dissemination: open access, institutional repository, and authorship rights.

Learning Outcomes I: Submit the complete thesis document meeting all institutional requirements. [Assessment] **II:** Present the final draft before the pre-defense committee and incorporate their corrections. [Assessment] **III:** Complete the administrative procedures for scheduling the public defense. [Usage]

Bibliography: [IEEE-Computer Society, 2008, ACM, 2008]

6. Bibliography

[ACM, 2008] ACM (2008). *Digital Library*. Association for Computing Machinery. http://portal.acm.org/dl.cfm.

[IEEE-Computer Society, 2008] IEEE-Computer Society (2008). *Digital Library*. IEEE-Computer Society. http://www.computer.org/publications/dlib.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

AI367 Topics in Artificial Intelligence (Mandatory)

1. General information

- **Semester**: 10th Sem. **Credits**: 4
- **Hour of this course**: **Theory**: 2 hours; **Practice**: 2 hours; **Laboratory**: 2 hours;
- **Prerrequisites**:
  - AI264 Deep Learning (7th Sem-Pg. 411)

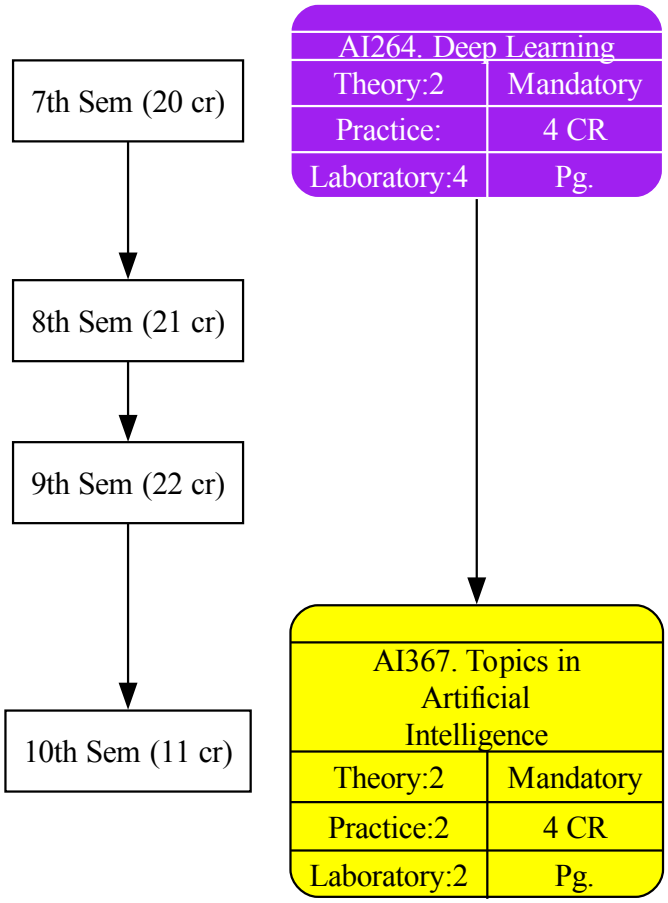

Figure 5.68: Connection Map. AI367 Topics in Artificial Intelligence

2. Course Rationale

Artificial Intelligence is a rapidly evolving field. This course explores advanced and contemporary topics that are currently shaping the industry and research. Focus is placed on Generative AI, Large Language Models (LLMs), and Advanced Reinforcement Learning. Students will analyze the architectures behind modern AI systems, such as Transformers, and discuss the technical and societal challenges of deploying autonomous intelligent agents at scale.

3. Goals of the Course

- Understand the architecture and training process of Large Language Models.
- Master the principles of Generative AI for text and image synthesis.
- Implement advanced Reinforcement Learning algorithms for decision making.
- Apply prompt engineering and fine-tuning techniques to existing models.
- Critically evaluate the latest research papers and trends in AI.

4. General Competences (*Outcomes*)

AG-C06) Life-long Learning: Recognizes the importance of continuous learning and adaptation to new technologies. (Usage)

5. Content

5.1 Machine Learning Fundamentals, 6 hr(s)

Competences: AG-C06

Topics: **I:** Definition and examples of a broad variety of machine learning tasks: (a) Supervised learning: i. Classification ii. Regression (b) Reinforcement learning (c) Unsupervised learning: i. Clustering **II:** Fundamental ideas: (a) No free lunch theorem: no one learner can solve all problems; representational design decisions have consequences. (b) Sources of error and undecidability in machine learning **III:** A simple statistical-based supervised learning such as linear regression or decision trees: (a) Focus on how they work without going into mathematical or optimization details; enough to understand and use existing implementations correctly **IV:** The overfitting problem/controlling solution complexity (regularization, pruning - intuition only): (a) The bias (underfitting) - variance (overfitting) tradeoff

Learning Outcomes **I:** Describe the differences among the three main styles of learning (supervised, reinforcement, and unsupervised) and determine which is appropriate to a particular problem domain [Explain] **II:** Differentiate the terms of AI, machine learning, and deep learning [Evaluate] **III:** Frame an application as a classification problem, including the available input features and output to be predicted (e.g., identifying alphabetic characters from pixel grid input) [Apply] **IV:** Identify overfitting in the context of a problem and learning curves and describe solutions to overfitting [Evaluate] **V:** Explain how machine learning works as an optimization/search process [Explain]

Bibliography: [Goodfellow et al., 2016, Foster, 2022]

5.2 Machine Learning Data, Representations and Evaluation, 5 hr(s)

Competences: AG-C06

Topics: **I:** Working with Data: (a) Data preprocessing: i. Importance and pitfalls of preprocessing choices (b) Handling missing values (imputing, flag-as-missing): i. Implications of imputing vs flag-as-missing (c) Encoding categorical variables, encoding real-valued data (d) Normalization/standardization (e) Emphasis on real data, not textbook examples **II:** Representations: (a) Hypothesis spaces and complexity (b) Simple basis feature expansion, such as squaring univariate features (c) Learned feature representations **III:** Machine learning evaluation: (a) Separation of train, validation, and test sets (b) Performance metrics for classifiers (c) Estimation of test performance on held-out data (d) Tuning the parameters of a machine learning model with a validation set (e) Importance of understanding what a model is doing, where its pitfalls/shortcomings are, and the implications of its decisions

Learning Outcomes **I:** Apply two or more simple statistical learning algorithms to a classification task and measure the classifiers' accuracy [Apply] **II:** Implement a statistical learning algorithm and the corresponding optimization process to train the classifier and obtain a prediction on new data [Apply] **III:** Explain proper ML evaluation procedures, including the differences

between training and testing performance, and what can go wrong with the evaluation process leading to inaccurate reporting of ML performance [Explain] **IV:** Compare two machine learning algorithms on a dataset, implementing the data preprocessing and evaluation methodology (e.g., metrics and handling of train/test splits) from scratch [Apply]

Bibliography: [Goodfellow et al., 2016, Foster, 2022]

5.3 Neural Networks and Ethics, 5 hr(s)

Competences: AG-C06

Topics: I: Basic neural networks: (a) Fundamentals of understanding how neural networks work and their training process, without details of the calculations (b) Basic introduction to generative neural networks (e.g., large language models) **II:** Ethics for Machine Learning: (a) Focus on real data, real scenarios, and case studies (b) Dataset/algorithmic/evaluation bias and unintended consequences

Learning Outcomes I: Describe the neural network training process and resulting learned representations [Explain]

Bibliography: [Goodfellow et al., 2016, Foster, 2022]

5.4 Machine Learning Formulation and Evaluation, 3 hr(s)

Competences:

Topics: I: Formulation of simple machine learning as an optimization problem, such as least squares linear regression or logistic regression: (a) Objective function (b) Gradient descent (c) Regularization to avoid overfitting (mathematical formulation) **II:** Ensembles of models: (a) Simple weighted majority combination **III:** Performance evaluation: (a) Other metrics for classification (e.g., error, precision, recall) (b) Performance metrics for regressors (c) Confusion matrix (d) Cross-validation: i. Parameter tuning (grid/random search, via cross-validation)

Learning Outcomes I: Compare and contrast several learning techniques (e.g., decision trees, logistic regression, naive Bayes, neural networks, and belief networks), providing examples of when each strategy is superior [Compare] **II:** Evaluate the performance of a simple learning system on a real-world dataset [Evaluate] **III:** Characterize the state of the art in learning theory, including its achievements and shortcomings [Explain] **IV:** Explain the problem of overfitting, along with techniques for detecting and managing the problem [Explain] **V:** Explain the triple tradeoff among the size of a hypothesis space, the size of the training set, and performance accuracy [Explain]

Bibliography: [Goodfellow et al., 2016]

5.5 Deep Learning, RL and Ethics, 3 hr(s)

Competences:

Topics: I: Deep learning: (a) Deep feed-forward networks (intuition only, no mathematics) (b) Convolutional neural networks (intuition only, no mathematics) (c) Visualization of learned feature representations from deep nets (d) Other architectures (generative NN, recurrent NN, transformers, etc.) **II:** Overview of reinforcement learning methods **III:** Two or more applications of machine learning algorithms: (a) E.g., medicine and health, economics, vision, natural language, robotics, game play **IV:** Ethics for Machine Learning: (a) Continued focus on real data, real scenarios, and case studies (b) Privacy (c) Fairness (d) Intellectual property (e) Explainability

Learning Outcomes I: Visualize the training progress of a neural network through learning curves in a well-established toolkit (e.g., TensorBoard) and visualize the learned features of the network [Apply] **II:** Given a real-world application of machine learning, describe ethical issues regarding the choices of data, preprocessing steps, algorithm selection, and visualization/presentation of results [Apply]

Bibliography: [Goodfellow et al., 2016]

5.6 Natural Language Processing, 20 hr(s)

Competences: AG-C06

Topics: **I:** Deterministic and stochastic grammars **II:** Parsing algorithms: (a) CFGs and chart parsers (e.g., CYK) (b) Probabilistic CFGs and weighted CYK **III:** Representing meaning/Semantics: (a) Logic-based knowledge representations (b) Semantic roles (c) Temporal representations (d) Beliefs, desires, and intentions **IV:** Corpus-based methods **V:** N-grams and HMMs **VI:** Smoothing and backoff **VII:** Examples of use: POS tagging and morphology **VIII:** Information retrieval: (a) Vector space model: i. TF & IDF (b) Precision and recall **IX:** Information extraction **X:** Language translation **XI:** Text classification, categorization: (a) Bag of words model **XII:** Deep learning for NLP: (a) RNNs (b) Transformers (c) Multimodal embeddings (e.g., images + text) (d) Generative language models

Learning Outcomes **I:** Define and contrast deterministic and stochastic grammars, providing examples to show the adequacy of each [Define] **II:** Simulate, apply, or implement classic and stochastic algorithms for parsing natural language [Apply] **III:** Identify the challenges of representing meaning [Analyze] **IV:** List the advantages of using standard corpora. Identify examples of current corpora for a variety of NLP tasks [List] **V:** Identify techniques for information retrieval, language translation, and text classification [Analyze] **VI:** Implement a TF/IDF transform, use it to extract features from a corpus, and train an off-the-shelf machine learning algorithm using those features to do text classification [Apply]

Bibliography: [Tunstall et al., 2022, Russell and Norvig, 2020]

5.7 Machine Learning Fundamentals, 6 hr(s)

Competences: AG-C06

Topics: **I:** Definition and examples of a broad variety of machine learning tasks: (a) Supervised learning: i. Classification ii. Regression (b) Reinforcement learning (c) Unsupervised learning: i. Clustering **II:** Fundamental ideas: (a) No free lunch theorem: no one learner can solve all problems; representational design decisions have consequences. (b) Sources of error and undecidability in machine learning **III:** A simple statistical-based supervised learning such as linear regression or decision trees: (a) Focus on how they work without going into mathematical or optimization details; enough to understand and use existing implementations correctly **IV:** The overfitting problem/controlling solution complexity (regularization, pruning - intuition only): (a) The bias (underfitting) - variance (overfitting) tradeoff

Learning Outcomes **I:** Describe the differences among the three main styles of learning (supervised, reinforcement, and unsupervised) and determine which is appropriate to a particular problem domain [Explain] **II:** Differentiate the terms of AI, machine learning, and deep learning [Evaluate] **III:** Frame an application as a classification problem, including the available input features and output to be predicted (e.g., identifying alphabetic characters from pixel grid input) [Apply] **IV:** Identify overfitting in the context of a problem and learning curves and describe solutions to overfitting [Evaluate] **V:** Explain how machine learning works as an optimization/search process [Explain]

Bibliography: [Sutton and Barto, 2018]

5.8 Machine Learning Data, Representations and Evaluation, 5 hr(s)

Competences: AG-C06

Topics: **I:** Working with Data: (a) Data preprocessing: i. Importance and pitfalls of preprocessing choices (b) Handling missing values (imputing, flag-as-missing): i. Implications of imputing

vs flag-as-missing (c) Encoding categorical variables, encoding real-valued data (d) Normalization/standardization (e) Emphasis on real data, not textbook examples **II:** Representations: (a) Hypothesis spaces and complexity (b) Simple basis feature expansion, such as squaring univariate features (c) Learned feature representations **III:** Machine learning evaluation: (a) Separation of train, validation, and test sets (b) Performance metrics for classifiers (c) Estimation of test performance on held-out data (d) Tuning the parameters of a machine learning model with a validation set (e) Importance of understanding what a model is doing, where its pitfalls/shortcomings are, and the implications of its decisions

Learning Outcomes **I:** Apply two or more simple statistical learning algorithms to a classification task and measure the classifiers' accuracy [Apply] **II:** Implement a statistical learning algorithm and the corresponding optimization process to train the classifier and obtain a prediction on new data [Apply] **III:** Explain proper ML evaluation procedures, including the differences between training and testing performance, and what can go wrong with the evaluation process leading to inaccurate reporting of ML performance [Explain] **IV:** Compare two machine learning algorithms on a dataset, implementing the data preprocessing and evaluation methodology (e.g., metrics and handling of train/test splits) from scratch [Apply]

Bibliography: [Sutton and Barto, 2018]

5.9 Neural Networks and Ethics, 5 hr(s)

Competences: AG-C06

Topics: **I:** Basic neural networks: (a) Fundamentals of understanding how neural networks work and their training process, without details of the calculations (b) Basic introduction to generative neural networks (e.g., large language models) **II:** Ethics for Machine Learning: (a) Focus on real data, real scenarios, and case studies (b) Dataset/algorithmic/evaluation bias and unintended consequences

Learning Outcomes **I:** Describe the neural network training process and resulting learned representations [Explain]

Bibliography: [Sutton and Barto, 2018]

5.10 AI Applications and Societal Impact, 6 hr(s)

Competences: AG-C06

Topics: **I:** At least one application of AI to a specific problem and field, such as medicine, health, sustainability, social media, economics, education, robotics, etc. (choose at least one for the CS Core): (a) Formulating and evaluating a specific application as an AI problem: i. How to deal with underspecified or ill-posed problems (b) Data availability/scarcity and cleanliness: i. Basic data cleaning and preprocessing ii. Data set bias (c) Algorithmic bias (d) Evaluation bias (e) Assessment of societal implications of the application **II:** Deployed deep generative models: (a) High-level overview of deep image generative models (e.g., as of 2023, DALL-E, Midjourney, Stable Diffusion, etc.), their uses, and their shortcomings/pitfalls. (b) High-level overview of large language models (e.g., as of 2023, ChatGPT, Bard, etc.), their uses, and their shortcomings/pitfalls. **III:** Overview of societal impact of AI: (a) Ethics (b) Fairness (c) Trust/explainability (d) Privacy and usage of training data (e) Human autonomy and oversight/regulations/legal requirements (f) Sustainability

Learning Outcomes **I:** Given a real-world application domain and problem, formulate an AI solution to it, identifying proper data/input, preprocessing, representations, AI techniques, and evaluation metrics/methodology [Apply]

Bibliography: [Russell and Norvig, 2020, Bostrom, 2014]

5.11 Additional AI Applications and Societal Impact, 6 hr(s)

Competences: AG-C06

Topics: **I:** One or more additional applications of AI to a broad set of problems and diverse fields, such as medicine, health, sustainability, social media, economics, education, robotics, etc. (choose a different area from that chosen for the CS Core): (a) Formulating and evaluating a specific application as an AI problem: i. How to deal with underspecified or ill-posed problems (b) Data availability/scarcity and cleanliness: i. Basic data cleaning and preprocessing ii. Data set bias (c) Algorithmic bias (d) Evaluation bias (e) Assessment of societal implications of the application **II:** Additional depth on deployed deep generative models: (a) Introduction to how deep image generative models work, (e.g., as of 2023, DALL-E, Midjourney, Stable Diffusion) including discussion of attention (b) Introduction to how large language models work, (e.g., as of 2023, ChatGPT, Bard) including discussion of attention (c) Idea of foundational models, how to use them, and the benefits/issues with training them from big data **III:** Analysis and discussion of the societal impact of AI: (a) Ethics (b) Fairness (c) Trust/explainability (d) Privacy and usage of training data (e) Human autonomy and oversight/regulations/legal requirements (f) Sustainability

Learning Outcomes **I:** Analyze the societal impact of one or more specific real-world AI applications, identifying issues regarding ethics, fairness, bias, trust, and explainability [Evaluate] **II:** Describe some of the failure modes of current deep generative models for language or images, and how this could affect their use in an application [Describe]

Bibliography: [Russell and Norvig, 2020, Bostrom, 2014]

6. Bibliography

[Bostrom, 2014] Bostrom, N. (2014). *Superintelligence: Paths, Dangers, Strategies*. Oxford University Press.

[Foster, 2022] Foster, D. (2022). *Generative Deep Learning*. O'Reilly Media, 2nd edition.

[Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press.

[Russell and Norvig, 2020] Russell, S. and Norvig, P. (2020). *Artificial Intelligence: A Modern Approach*. Pearson, 4th edition.

[Sutton and Barto, 2018] Sutton, R. S. and Barto, A. G. (2018). *Reinforcement Learning: An Introduction*. MIT Press, 2nd edition.

[Tunstall et al., 2022] Tunstall, L., von Werra, L., and Wolf, T. (2022). *Natural Language Processing with Transformers*. O'Reilly Media.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

AI368 Evolutionary Computing (Mandatory)

1. General information

- **Semester**: 10th Sem. **Credits**: 4
- **Hour of this course**: **Theory**: 2 hours; **Practice**: 2 hours; **Laboratory**: 2 hours;
- **Prerrequisites**:
  - AI264 Deep Learning (7th Sem-Pg. 411)

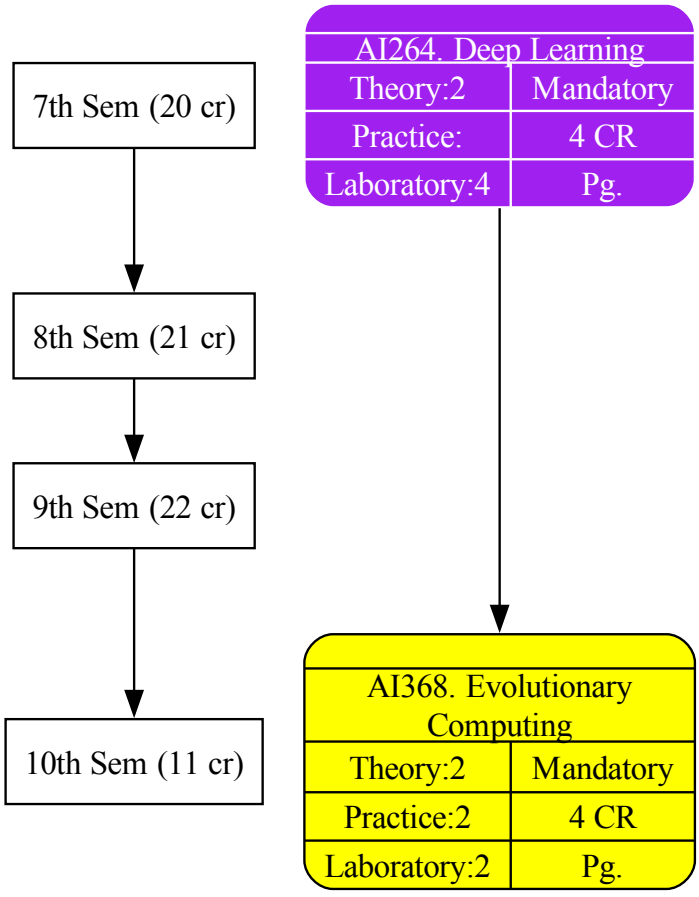

Figure 5.69: Connection Map. AI368 Evolutionary Computing

2. Course Rationale

Evolutionary Computing is a subfield of Artificial Intelligence inspired by natural selection and biological evolution. This course explores metaheuristic optimization algorithms such as Genetic Algorithms, Evolution Strategies, and Genetic Programming. These techniques are essential for solving complex optimization and search problems where traditional analytical methods are insufficient. Students will learn to design, implement, and analyze evolutionary systems applied to engineering, data science, and autonomous agents.

3. Goals of the Course

- Understand the principles of natural selection applied to computation.
- Master the design of representations, operators, and fitness functions.
- Implement and tune Genetic Algorithms and Evolution Strategies.
- Analyze the convergence and performance of bio-inspired algorithms.
- Apply evolutionary techniques to multi-objective optimization problems.

4. General Competences (*Outcomes*)

AG-C08) Problem Analysis: Identifies, formulates, and analyzes complex computing problems. (Usage)

AG-C12) Applies computer science theory and software development fundamentals to produce computer-based solutions. (Usage)

5. Content

5.1 Foundations of Evolutionary Computation, 12 hr(s)

Competences: AG-C08,AG-C12

Topics: **I:** Introduction to Evolutionary Algorithms (EAs) and biological metaphors. **II:** The General Evolutionary Algorithm cycle: Initialization, Selection, Variation, Replacement. **III:** Components: Populations, chromosomes, genes, alleles. **IV:** Fitness landscapes, genotype-phenotype mapping, and search spaces.

Learning Outcomes **I:** Explain the relationship between biological evolution and artificial search processes [Familiarity] **II:** Identify the components of a generic evolutionary algorithm and their roles [Usage] **III:** Describe the concept of fitness landscapes and their impact on EA performance [Familiarity]

Bibliography: [Eiben and Smith, 2015, Back, 1996]

5.2 Genetic Algorithms and Variants, 16 hr(s)

Competences: AG-C08,AG-C12

Topics: **I:** Canonical Genetic Algorithms (CGA): binary representation, selection schemes (roulette, tournament). **II:** Genetic Operators: Crossover (single-point, uniform, arithmetic), Mutation (bit-flip, Gaussian). **III:** Schema Theorem and Building Block Hypothesis. **IV:** Advanced GA variants: Real-coded GAs, Genetic Programming (GP) with tree representations.

Learning Outcomes **I:** Design appropriate representations and operators for specific optimization problems [Assessment] **II:** Implement a canonical Genetic Algorithm for function optimization [Usage] **III:** Apply Genetic Programming to symbolic regression or automatic programming tasks [Assessment]

Bibliography: [Eiben and Smith, 2015, Mitchell, 1998]

5.3 Evolution Strategies and Evolutionary Programming, 18 hr(s)

Competences: AG-C08,AG-C12

Topics: **I:** Evolution Strategies (ES): (μ, λ) and $(\mu + \lambda)$ selection mechanisms. **II:** Self-adaptation of strategy parameters (mutation step sizes). **III:** Covariance Matrix Adaptation (CMA-ES). **IV:** Evolutionary Programming (EP) and its focus on behavioral evolution.

Learning Outcomes **I:** Implement Evolution Strategies with self-adaptive mutation for continuous optimization [Assessment] **II:** Compare the performance of GAs vs ES on benchmark problems [Usage] **III:** Explain the role of strategy parameter adaptation in convergence speed [Familiarity]

Bibliography: [Eiben and Smith, 2015, Deb, 2001]

5.4 Multi-objective and Hybrid Evolutionary Algorithms, 18 hr(s)

Competences: AG-C08,AG-C12

Topics: **I:** Multi-objective Optimization: Pareto optimality, dominance relations. **II:** Multi-objective EAs: NSGA-II, SPEA2, MOEA/D. **III:** Diversity preservation mechanisms and archiving strategies. **IV:** Memetic Algorithms: Hybridization with local search. **V:** Constraint handling techniques in evolutionary search.

Learning Outcomes **I:** Solve multi-objective problems using Pareto-based selection methods [Assessment] **II:** Design a memetic algorithm combining global and local search [Assessment] **III:** Evaluate the trade-offs between convergence and diversity in MOEAs [Usage]

Bibliography: [Eiben and Smith, 2015, Deb, 2001]

6. Bibliography

[Back, 1996] Back, T. (1996). *Evolutionary Algorithms in Theory and Practice*. Oxford University Press.

[Deb, 2001] Deb, K. (2001). *Multi-Objective Optimization using Evolutionary Algorithms*. Wiley.

[Eiben and Smith, 2015] Eiben, A. E. and Smith, J. E. (2015). *Introduction to Evolutionary Computing*. Springer, 2nd edition.

[Mitchell, 1998] Mitchell, M. (1998). *An Introduction to Genetic Algorithms*. MIT Press.

Chapter 6

Professors & Courses

Taught courses by professor

- **Dr. Ernesto Cuadros-Vargas**
  - CS312 Advanced Data Structures
  - CS395 Software Quality

Professor assigned per course

- First Semester
  - BIC101. Introduction to Computing (Pág 239)
    - There is no professor for this course
  - BMA101. Linear Algebra (Pág 244)
    - There is no professor for this course
  - BMA102. Differential Calculus (Pág 246)
    - There is no professor for this course
  - BCH101. Chemistry I (Pág 249)
    - There is no professor for this course
  - BEI101. Technical and professional English I (Pág 254)
    - There is no professor for this course
- Second Semester
  - CS100. Introduction to Computer Science (Pág 257)
    - There is no professor for this course
  - CS112. Objects-oriented programming I (Pág 268)
    - There is no professor for this course
  - CS1D1. Discrete Structures (Pág 273)
    - There is no professor for this course
  - BMA103. Integral Calculus (Pág 276)
    - There is no professor for this course
  - BFI101. Physics I (Pág 278)
    - There is no professor for this course
  - BEI102. Technical and professional English II (Pág 283)
    - There is no professor for this course
- Third Semester
  - CS113. Objects-oriented programming II (Pág 287)
    - There is no professor for this course
  - CS2B1. Platform Based Development (Pág 293)
    - There is no professor for this course
  - AI161. Applied AI (Pág 297)
    - There is no professor for this course
  - BMA104. Advanced Differential and Integral Calculus (Pág 301)

    - There is no professor for this course
  - STA251. Probability Calculation (Pág 304)
    - There is no professor for this course
  - BEI201. Technical and professional English III (Pág 309)
    - There is no professor for this course
- Fourth Semester
  - CS210. Algorithms and Data Structures (Pág 313)
    - There is no professor for this course
  - CS211. Theory of Computation (Pág 318)
    - There is no professor for this course
  - CS221. Computer Systems Architecture (Pág 323)
    - There is no professor for this course
  - CS271. Data Management (Pág 327)
    - There is no professor for this course
  - CS401. Research Methodology (Pág 331)
    - There is no professor for this course
  - MA202. Numerical Methods (Pág 335)
    - There is no professor for this course
  - BEI202. Technical and professional English IV (Pág 340)
    - There is no professor for this course
- Fifth Semester
  - CS212. Analysis and Design of Algorithms (Pág 344)
    - There is no professor for this course
  - CS261. Artificial Intelligence (Pág 349)
    - There is no professor for this course
  - CS272. Databases II (Pág 356)
    - There is no professor for this course
  - CS291. Software Engineering I (Pág 360)
    - There is no professor for this course
  - CS2S1. Operating systems (Pág 365)
    - There is no professor for this course
  - FG106. Theater (Pág 370)
    - There is no professor for this course
  - BEI203. Technical and professional English V (Pág 374)
    - There is no professor for this course
- Sixth Semester
  - CS231. Networking and Communication (Pág 378)
    - There is no professor for this course
  - CS311. Algorithms for Complex Problems (Pág 383)
    - There is no professor for this course
  - CS312. Advanced Data Structures (Pág 386)
    - Ernesto Cuadros-Vargas (ecuadrosv@uni.edu.pe)
  - CS342. Compilers (Pág 389)
    - There is no professor for this course
  - AI263. Introduction to Machine Learning (Pág 392)
    - There is no professor for this course
  - FI201. Computational Physics (Pág 395)
    - There is no professor for this course
- Seventh Semester
  - CS251. Computer Graphics (Pág 398)
    - There is no professor for this course
  - CS292. Software Engineering II (Pág 401)
    - There is no professor for this course
  - CS2H1. User Experience (UX) (Pág 406)
    - There is no professor for this course
  - AI264. Deep Learning (Pág 411)

        - There is no professor for this course
    - FG211. Professional Ethics (Pág 414)
        - There is no professor for this course
    - FG350. Leadership and Performance (Pág 416)
        - There is no professor for this course
- Eighth Semester
    - CS281. Computing in Society (Pág 419)
        - There is no professor for this course
    - CS3I1. Computer Security (Pág 425)
        - There is no professor for this course
    - CS3P1. Parallel and Distributed Computing (Pág 430)
        - There is no professor for this course
    - CS402. Capstone Project I (Pág 439)
        - There is no professor for this course
    - EX301. Extracurricular Activities (Pág 443)
        - There is no professor for this course
    - ET201. Entrerpreneurship I (Pág 445)
        - There is no professor for this course
    - AI268. Computational Vision (Pág 449)
        - There is no professor for this course
    - CS391. Software Engineering III (Pág 455)
        - There is no professor for this course
    - CS393. Information systems (Pág 459)
        - There is no professor for this course
    - CS394. Software Architecture and DevOps for Cloud Software (Pág 462)
        - There is no professor for this course
    - CS395. Software Quality (Pág 468)
        - Ernesto Cuadros-Vargas (ecuadrosv@uni.edu.pe)
- Ninth Semester
    - CS370. Big Data (Pág 475)
        - There is no professor for this course
    - CS400. Pre-professional internships (Pág 478)
        - There is no professor for this course
    - CS403. Capstone Project II (Pág 480)
        - There is no professor for this course
    - AI365. Advanced Generative AI Models (Pág 484)
        - There is no professor for this course
    - CB309. Bioinformatics (Pág 486)
        - There is no professor for this course
    - ET301. Entrerpreneurship II (Pág 489)
        - There is no professor for this course
    - AI369. Robotics (Pág 492)
        - There is no professor for this course
    - CS351. Topics in Computer Graphics (Pág 496)
        - There is no professor for this course
    - CS353. Quantum Computing (Pág 500)
        - There is no professor for this course
    - CS392. Tópicos en Ingeniería de Software (Pág 504)
        - There is no professor for this course
    - CS3P3. Internet of Things (Pág 511)
        - There is no professor for this course
- Tenth Semester
    - CS3P2. Cloud Computing (Pág 513)
        - There is no professor for this course
    - CS404. Research Workshop (Pág 517)
        - There is no professor for this course

- AI367. Topics in Artificial Intelligence (Pág 521)
  - There is no professor for this course
- AI368. Evolutionary Computing (Pág 527)
  - There is no professor for this course

Chapter 7

Equivalences with Other Curriculum Plans

Below are the equivalences of this curriculum with the previous curriculum plan(s). For more details on the proposed curriculum, see section 4.2 (p. 186).

Equivalence from 2027 to 2018

Primer Semestre – Plan2027			Plan 2018			
Código	**Curso**	**Cr**	**Código**	**Curso**	**Sem**	**Cr**
BIC101	Introducción a la Computación	4	BIC01	Introducción a la Computación	1st Sem	2
BMA101	Algebra Lineal	4	BMA03	Álgebra Lineal	1st Sem	4
BMA102	Cálculo Diferencial	5	BMA01	Cálculo Diferencial	1st Sem	5
BCH101	Química I	5	BQU01	Química I	1st Sem	5
BEI101	Inglés I	2	CL002	Inglés 1	2nd Sem	2

Segundo Semestre – Plan2027			Plan 2018			
Código	**Curso**	**Cr**	**Código**	**Curso**	**Sem**	**Cr**
CS100	Introducción a la Ciencia de la Computación	3			1st Sem	
CS112	Programación Orientada a Objetos I	4	CC112	Fundamentos de programación	2nd Sem	4
CS1D1	Estructuras Discretas	4	CM2H1	Matemática Discreta	3rd Sem	4
BMA103	Cálculo Integral	5	BMA02	Cálculo Integral	2nd Sem	5
BFI101	Física I	5	BFI01	Física I	1st Sem	5
BEI102	Inglés II	2	BIE01	Idioma Extranjero Avanzado	4th Sem	2

Tercer Semestre – Plan2027			Plan 2018			
Código	**Curso**	**Cr**	**Código**	**Curso**	**Sem**	**Cr**
CS113	Programación Orientada a Objetos II	4	CC211	Programación Orientada a Objetos	3rd Sem	4
CS2B1	Desarrollo Basado en Plataformas	4			3rd Sem	
AI161	IA Aplicada	4			3rd Sem	
STA251	Estadística y Probabilidades	3	CM1H2	Cálculo de probabilidades	2nd Sem	3

Cuarto Semestre – Plan2027			Plan 2018			
Código	**Curso**	**Cr**	**Código**	**Curso**	**Sem**	**Cr**
CS210	Algoritmos y Estructuras de Datos	4	CC232	Algoritmo y Estructura de Datos	4th Sem	4
CS211	Teoría de la Computación	4	CC321	Teoría de Autómatas Lenguaje y Computación	5th Sem	4
CS221	Arquitectura de Computadores	4	CC221	Arquitectura de Computadores	3rd Sem	4
CS271	Bases de Datos I	4	CC202	Base de Datos	4th Sem	4
CS401	Metodología de la Investigación	2	CC4H1	Metodología de la Investigación	7th Sem	2
MA202	Métodos Numéricos	3	CM4F1	Análisis y Modelamiento Numérico I	5th Sem	5

Quinto Semestre – Plan2027			Plan 2018			
Código	**Curso**	**Cr**	**Código**	**Curso**	**Sem**	**Cr**
CS212	Análisis y Diseño de Algoritmos	4	CC371	Análisis y Diseño de Algoritmos	5th Sem	4
CS261	Inteligencia Artificial	4	CC421	Inteligencia Artificial	7th Sem	4
CS272	Bases de Datos II	3			5th Sem	
CS291	Ingeniería de Software I	4	CC341	Ingeniería de Software	5th Sem	4
CS2S1	Sistemas Operativos	4	CC222	Sistemas Operativos	4th Sem	4

Sexto Semestre – Plan2027			Plan 2018			
Código	**Curso**	**Cr**	**Código**	**Curso**	**Sem**	**Cr**
CS231	Redes y Comunicación	3	CC311	Computación Centrada en Redes	5^{th} Sem	4
CS311	Algoritmos para Problemas Complejos	3			6^{th} Sem	
CS312	Estructuras de Datos Avanzadas	4			6^{th} Sem	
CS342	Compiladores	4			5^{th} Sem	
AI263	Introducción al Aprendizaje de Máquina	4			6^{th} Sem	4
FI201	Física Computacional	3			2^{nd} Sem	

Séptimo Semestre – Plan2027			Plan 2018			
Código	**Curso**	**Cr**	**Código**	**Curso**	**Sem**	**Cr**
CS251	Computación Gráfica	4	CC431	Computación Gráfica	7^{th} Sem	4
CS292	Ingeniería de Software II	4	CC3S2	Desarrollo de Software	6^{th} Sem	4
CS2H1	Experiencia de Usuario (UX)	4	CC451	Interacción Humano-Computador	7^{th} Sem	4
AI264	Aprendizaje Profundo	4			7^{th} Sem	
FG211	Ética Profesional	2	BEF01	Ética y Filosofía Política	6^{th} Sem	2

Octavo Semestre – Plan2027			Plan 2018			
Código	**Curso**	**Cr**	**Código**	**Curso**	**Sem**	**Cr**
CS281	Computación en la Sociedad	2			8^{th} Sem	
CS3I1	Seguridad en Computación	3	CC411	Seguridad en Sistemas Informáticos	7^{th} Sem	4
CS3P1	Computación Paralela y Distribuída	4	CC4P1	Programación Concurrente y Distribuida	7^{th} Sem	4
CS402	Proyecto de tesis 1	4	CC412	Proyecto de Tesis I	8^{th} Sem	4
EX301	Actividades Extracurriculares	2	BAE01	Actividades Extracurriculares	10^{th} Sem	1
CS391	Ingeniería de Software III	3			8^{th} Sem	
CS393	Sistemas de Información	3			7^{th} Sem	

Noveno Semestre – Plan2027			Plan 2018			
Código	**Curso**	**Cr**	**Código**	**Curso**	**Sem**	**Cr**
CS370	Big Data	3	CC442	Minería de Datos	8^{th} Sem	4
CS400	Prácticas Pre-profesionales	2	XP200	Prácticas Pre-Profesionales II	10^{th} Sem	2
CS403	Proyecto de tesis 2	3	CC511	Proyecto de Tesis II	9^{th} Sem	4
AI365	Modelos Generativos Avanzados en IA	4			9^{th} Sem	
CB309	Bioinformática	4			9^{th} Sem	
CS351	Tópicos en Computación Gráfica	3			9^{th} Sem	
CS353	Computación Cuántica	3			10^{th} Sem	
CS392	Tópicos en Ingeniería de Software	3			9^{th} Sem	
CS3P3	Internet de las Cosas	3			10^{th} Sem	

Décimo Semestre – Plan2027			Plan 2018			
Código	**Curso**	**Cr**	**Código**	**Curso**	**Sem**	**Cr**
CS3P2	Cloud Computing	3	CC422	Infraestructuras de Computación	8^{th} Sem	4
CS404	Taller de Investigación	4	CC512	Proyecto de Tesis III	10^{th} Sem	4

Equivalence from 2018 to 2027

Primer Semestre – Plan 2018			Plan2027			
Código	**Curso**	**Cr**	**Código**	**Curso**	**Sem**	**Cr**
BIC01	Introducción a la Computación	2	BIC101	Introducción a la Computación	1st Sem	4
BMA03	Álgebra Lineal	4	BMA101	Algebra Lineal	1st Sem	4
BMA01	Cálculo Diferencial	5	BMA102	Cálculo Diferencial	1st Sem	5
BQU01	Química I	5	BCH101	Química I	1st Sem	5
			CS100	Introducción a la Ciencia de la Computación	2nd Sem	3
BFI01	Física I	5	BFI101	Física I	2nd Sem	5

Segundo Semestre – Plan 2018			Plan2027			
Código	**Curso**	**Cr**	**Código**	**Curso**	**Sem**	**Cr**
CL002	Inglés 1	2	BEI101	Inglés I	1st Sem	2
CC112	Fundamentos de programación	4	CS112	Programación Orientada a Objetos I	2nd Sem	4
BMA02	Cálculo Integral	5	BMA103	Cálculo Integral	2nd Sem	5
CM1H2	Cálculo de probabilidades	3	STA251	Estadística y Probabilidades	3rd Sem	3
			FI201	Física Computacional	6th Sem	3

Tercer Semestre – Plan 2018			Plan2027			
Código	**Curso**	**Cr**	**Código**	**Curso**	**Sem**	**Cr**
CM2H1	Matemática Discreta	4	CS1D1	Estructuras Discretas	2nd Sem	4
CC211	Programación Orientada a Objetos	4	CS113	Programación Orientada a Objetos II	3rd Sem	4
CC221	Arquitectura de Computadores	4	CS221	Arquitectura de Computadores	4th Sem	4

Cuarto Semestre – Plan 2018			Plan2027			
Código	**Curso**	**Cr**	**Código**	**Curso**	**Sem**	**Cr**
BIE01	Idioma Extranjero Avanzado	2	BEI102	Inglés II	2nd Sem	2
CC232	Algoritmo y Estructura de Datos	4	CS210	Algoritmos y Estructuras de Datos	4th Sem	4
CC202	Base de Datos	4	CS271	Bases de Datos I	4th Sem	4
CC222	Sistemas Operativos	4	CS2S1	Sistemas Operativos	5th Sem	4
BEG01	Economía General					

Quinto Semestre – Plan 2018			Plan2027			
Código	**Curso**	**Cr**	**Código**	**Curso**	**Sem**	**Cr**
CC321	Teoría de Autómatas Lenguaje y Computación	4	CS211	Teoría de la Computación	4th Sem	4
CM4F1	Análisis y Modelamiento Numérico I	5	MA202	Métodos Numéricos	4th Sem	3
CC371	Análisis y Diseño de Algoritmos	4	CS212	Análisis y Diseño de Algoritmos	5th Sem	4
CC341	Ingeniería de Software	4	CS291	Ingeniería de Software I	5th Sem	4
CC311	Computación Centrada en Redes	4	CS231	Redes y Comunicación	6th Sem	3

Sexto Semestre – Plan 2018			Plan2027			
Código	**Curso**	**Cr**	**Código**	**Curso**	**Sem**	**Cr**
		4	AI263	Introducción al Aprendizaje de Máquina	6th Sem	4
CC3S2	Desarrollo de Software	4	CS292	Ingeniería de Software II	7th Sem	4
BEF01	Ética y Filosofía Política	2	FG211	Ética Profesional	7th Sem	2
BRN01	Realidad Nacional, Constitución y Derechos Humanos					
CC3M2	Matemática Computacional					

Séptimo Semestre – Plan 2018			Plan2027			
Código	**Curso**	**Cr**	**Código**	**Curso**	**Sem**	**Cr**
CC4H1	Metodología de la Investigación	2	CS401	Metodología de la Investigación	4th Sem	2
CC421	Inteligencia Artificial	4	CS261	Inteligencia Artificial	5th Sem	4
CC431	Computación Gráfica	4	CS251	Computación Gráfica	7th Sem	4
CC451	Interacción Humano-Computador	4	CS2H1	Experiencia de Usuario (UX)	7th Sem	4
CC411	Seguridad en Sistemas Informáticos	4	CS3I1	Seguridad en Computación	8th Sem	3
CC4P1	Programación Concurrente y Distribuida	4	CS3P1	Computación Paralela y Distribuída	8th Sem	4
			CS393	Sistemas de Información	8th Sem	3

Octavo Semestre – Plan 2018			Plan2027			
Código	**Curso**	**Cr**	**Código**	**Curso**	**Sem**	**Cr**
CC412	Proyecto de Tesis I	4	CS402	Proyecto de tesis 1	8^{th} Sem	4
			CS391	Ingeniería de Software III	8^{th} Sem	3
CC442	Minería de Datos	4	CS370	Big Data	9^{th} Sem	3
CC422	Infraestructuras de Computación	4	CS3P2	Cloud Computing	10^{th} Sem	3

Noveno Semestre – Plan 2018			Plan2027			
Código	**Curso**	**Cr**	**Código**	**Curso**	**Sem**	**Cr**
CC511	Proyecto de Tesis II	4	CS403	Proyecto de tesis 2	9^{th} Sem	3
CC521	Programación Evolutiva					

Décimo Semestre – Plan 2018			Plan2027			
Código	**Curso**	**Cr**	**Código**	**Curso**	**Sem**	**Cr**
BAE01	Actividades Extracurriculares	1	EX301	Actividades Extracurriculares	8^{th} Sem	2
XP200	Prácticas Pre-Profesionales II	2	CS400	Prácticas Pre-profesionales	9^{th} Sem	2
			CS3P3	Internet de las Cosas	9^{th} Sem	3
CC512	Proyecto de Tesis III	4	CS404	Taller de Investigación	10^{th} Sem	4

Chapter 8

Laboratories

BIC101. Introduction to Computing (Mandatory) 1st Sem, Lab: 4 hr(s)

Basic Computing Lab:

- **Hardware**: 30 workstations (Intel Core i5, 8GB RAM, 256GB SSD, integrated graphics/sound, Gigabit Ethernet).
- **OS**: Dual-boot Linux (Ubuntu LTS) / Windows 11 Pro.
- **Programming**: Python, Java, C++ (C++11/14/17), C#, Prolog, Lex/Yacc.
- **Tools**: Git, VS Code, Eclipse, Jupyter Notebook.

BCH101. Chemistry I (Mandatory) 1st Sem, Lab: 2 hr(s)

Chemistry Lab:

- **Equipment**: Analytical balances, microscopes, spectrophotometers, glassware kits.
- **Safety**: Fume hoods, emergency showers, chemical spill kits.
- **Reagents**: Acids/bases, buffers, organic solvents, pH indicators.

CS112. Objects-oriented programming I (Mandatory) 2nd Sem, Lab: 4 hr(s)

Basic Computing Lab:

- **Hardware**: 30 workstations (Intel Core i5, 8GB RAM, 256GB SSD, integrated graphics/sound, Gigabit Ethernet).
- **OS**: Dual-boot Linux (Ubuntu LTS) / Windows 11 Pro.
- **Programming**: Python, Java, C++ (C++11/14/17), C#, Prolog, Lex/Yacc.
- **Tools**: Git, VS Code, Eclipse, Jupyter Notebook.

BFI101. Physics I (Mandatory) 2nd Sem, Lab: 4 hr(s)

Physics Laboratory:

- **Equipment and Tools**: Optical benches, force tables, pendulums, dynamic carts, spectrometers, PASCO sensors, and measurement modules (multimeters, power supplies, calipers).
- **Hardware and Technology**: Academic-use laptops, Arduino sensors, and modules for data acquisition.
- **Software**: Arduino IDE, MATLAB for analysis, Microsoft Office, and specialized experimental data acquisition software.
- **Services and Experiments**: Physics I and II lab sessions (Mechanics, Thermodynamics, Electromagnetism, Waves), instrumentation training, and support for research projects.

CS113. Objects-oriented programming II (Mandatory) 3rd Sem, Lab: 2 hr(s)

Basic Computing Lab:

- **Hardware**: 30 workstations (Intel Core i5, 8GB RAM, 256GB SSD, integrated graphics/sound, Gigabit Ethernet).
- **OS**: Dual-boot Linux (Ubuntu LTS) / Windows 11 Pro.
- **Programming**: Python, Java, C++ (C++11/14/17), C#, Prolog, Lex/Yacc.
- **Tools**: Git, VS Code, Eclipse, Jupyter Notebook.

CS2B1. Platform Based Development (Mandatory) 3rd Sem, Lab: 4 hr(s)

Advanced Computing Lab:

- **Hardware**: 30 high-end PCs (Intel Core i7, 16GB RAM, 512GB NVMe SSD, NVIDIA GTX 1660, Gigabit Ethernet).
- **OS**: Linux (Fedora) / Windows 11 Pro.
- **Programming**: C++20, Python (SciPy stack), Perl, R.
- **Software**:
  - *Development*: J2EE, .NET 6, MATLAB, GCC/Clang.
  - *DB/Cloud*: PostgreSQL, MongoDB, Docker.
  - *Specialized*: Rational Rose, VTK, LEDA, 8086 emulators.

AI161. Applied AI (Mandatory) 3rd Sem, Lab: 4 hr(s)

Basic Computing Lab:

- **Hardware**: 30 workstations (Intel Core i5, 8GB RAM, 256GB SSD, integrated graphics/sound, Gigabit Ethernet).
- **OS**: Dual-boot Linux (Ubuntu LTS) / Windows 11 Pro.
- **Programming**: Python, Java, C++ (C++11/14/17), C#, Prolog, Lex/Yacc.
- **Tools**: Git, VS Code, Eclipse, Jupyter Notebook.

CS210. Algorithms and Data Structures (Mandatory) 4th Sem, Lab: 2 hr(s)

Advanced Computing Lab:

- **Hardware**: 30 high-end PCs (Intel Core i7, 16GB RAM, 512GB NVMe SSD, NVIDIA GTX 1660, Gigabit Ethernet).
- **OS**: Linux (Fedora) / Windows 11 Pro.
- **Programming**: C++20, Python (SciPy stack), Perl, R.
- **Software**:
  - *Development*: J2EE, .NET 6, MATLAB, GCC/Clang.
  - *DB/Cloud*: PostgreSQL, MongoDB, Docker.
  - *Specialized*: Rational Rose, VTK, LEDA, 8086 emulators.

CS211. Theory of Computation (Mandatory) 4th Sem, Lab: 2 hr(s)

Basic Computing Lab:

- **Hardware**: 30 workstations (Intel Core i5, 8GB RAM, 256GB SSD, integrated graphics/sound, Gigabit Ethernet).
- **OS**: Dual-boot Linux (Ubuntu LTS) / Windows 11 Pro.
- **Programming**: Python, Java, C++ (C++11/14/17), C#, Prolog, Lex/Yacc.

- **Tools**: Git, VS Code, Eclipse, Jupyter Notebook.

CS221. Computer Systems Architecture (Mandatory) 4th Sem, Lab: 4 hr(s)

Networks & Cybersecurity Lab:
- **Hardware**: 30 PCs + dedicated network racks (Cisco switches/routers, Raspberry Pi clusters).
- **OS**: Linux (Kali, CentOS) / Windows Server.
- **Software**: Wireshark, GNS3, VMware ESXi, 8086 emulators.
- **Focus**: Network simulation, penetration testing, IoT protocols.

CS271. Data Management (Mandatory) 4th Sem, Lab: 4 hr(s)

Advanced Computing Lab:
- **Hardware**: 30 high-end PCs (Intel Core i7, 16GB RAM, 512GB NVMe SSD, NVIDIA GTX 1660, Gigabit Ethernet).
- **OS**: Linux (Fedora) / Windows 11 Pro.
- **Programming**: C++20, Python (SciPy stack), Perl, R.
- **Software**:
  - *Development*: J2EE, .NET 6, MATLAB, GCC/Clang.
  - *DB/Cloud*: PostgreSQL, MongoDB, Docker.
  - *Specialized*: Rational Rose, VTK, LEDA, 8086 emulators.

CS212. Analysis and Design of Algorithms (Mandatory) 5th Sem, Lab: 2 hr(s)

Advanced Computing Lab:
- **Hardware**: 30 high-end PCs (Intel Core i7, 16GB RAM, 512GB NVMe SSD, NVIDIA GTX 1660, Gigabit Ethernet).
- **OS**: Linux (Fedora) / Windows 11 Pro.
- **Programming**: C++20, Python (SciPy stack), Perl, R.
- **Software**:
  - *Development*: J2EE, .NET 6, MATLAB, GCC/Clang.
  - *DB/Cloud*: PostgreSQL, MongoDB, Docker.
  - *Specialized*: Rational Rose, VTK, LEDA, 8086 emulators.

CS261. Artificial Intelligence (Mandatory) 5th Sem, Lab: 2 hr(s)

Basic Computing Lab:
- **Hardware**: 30 workstations (Intel Core i5, 8GB RAM, 256GB SSD, integrated graphics/sound, Gigabit Ethernet).
- **OS**: Dual-boot Linux (Ubuntu LTS) / Windows 11 Pro.
- **Programming**: Python, Java, C++ (C++11/14/17), C#, Prolog, Lex/Yacc.
- **Tools**: Git, VS Code, Eclipse, Jupyter Notebook.

CS272. Databases II (Mandatory) 5th Sem, Lab: 2 hr(s)

Advanced Computing Lab:
- **Hardware**: 30 high-end PCs (Intel Core i7, 16GB RAM, 512GB NVMe SSD, NVIDIA GTX 1660, Gigabit Ethernet).
- **OS**: Linux (Fedora) / Windows 11 Pro.
- **Programming**: C++20, Python (SciPy stack), Perl, R.

- **Software**:
  - *Development*: J2EE, .NET 6, MATLAB, GCC/Clang.
  - *DB/Cloud*: PostgreSQL, MongoDB, Docker.
  - *Specialized*: Rational Rose, VTK, LEDA, 8086 emulators.

CS291. Software Engineering I (Mandatory) 5th Sem, Lab: 2 hr(s)

Advanced Computing Lab:

- **Hardware**: 30 high-end PCs (Intel Core i7, 16GB RAM, 512GB NVMe SSD, NVIDIA GTX 1660, Gigabit Ethernet).
- **OS**: Linux (Fedora) / Windows 11 Pro.
- **Programming**: C++20, Python (SciPy stack), Perl, R.
- **Software**:
  - *Development*: J2EE, .NET 6, MATLAB, GCC/Clang.
  - *DB/Cloud*: PostgreSQL, MongoDB, Docker.
  - *Specialized*: Rational Rose, VTK, LEDA, 8086 emulators.

CS2S1. Operating systems (Mandatory) 5th Sem, Lab: 2 hr(s)

Advanced Computing Lab:

- **Hardware**: 30 high-end PCs (Intel Core i7, 16GB RAM, 512GB NVMe SSD, NVIDIA GTX 1660, Gigabit Ethernet).
- **OS**: Linux (Fedora) / Windows 11 Pro.
- **Programming**: C++20, Python (SciPy stack), Perl, R.
- **Software**:
  - *Development*: J2EE, .NET 6, MATLAB, GCC/Clang.
  - *DB/Cloud*: PostgreSQL, MongoDB, Docker.
  - *Specialized*: Rational Rose, VTK, LEDA, 8086 emulators.

CS231. Networking and Communication (Mandatory) 6th Sem, Lab: 2 hr(s)

Advanced Computing Lab:

- **Hardware**: 30 high-end PCs (Intel Core i7, 16GB RAM, 512GB NVMe SSD, NVIDIA GTX 1660, Gigabit Ethernet).
- **OS**: Linux (Fedora) / Windows 11 Pro.
- **Programming**: C++20, Python (SciPy stack), Perl, R.
- **Software**:
  - *Development*: J2EE, .NET 6, MATLAB, GCC/Clang.
  - *DB/Cloud*: PostgreSQL, MongoDB, Docker.
  - *Specialized*: Rational Rose, VTK, LEDA, 8086 emulators.

CS311. Algorithms for Complex Problems (Mandatory) 6th Sem, Lab: 2 hr(s)

Basic Computing Lab:

- **Hardware**: 30 workstations (Intel Core i5, 8GB RAM, 256GB SSD, integrated graphics/sound, Gigabit Ethernet).
- **OS**: Dual-boot Linux (Ubuntu LTS) / Windows 11 Pro.
- **Programming**: Python, Java, C++ (C++11/14/17), C#, Prolog, Lex/Yacc.
- **Tools**: Git, VS Code, Eclipse, Jupyter Notebook.

CS312. Advanced Data Structures (Mandatory) 6th Sem, Lab: 2 hr(s)

Advanced Computing Lab:

- **Hardware**: 30 high-end PCs (Intel Core i7, 16GB RAM, 512GB NVMe SSD, NVIDIA GTX 1660, Gigabit Ethernet).
- **OS**: Linux (Fedora) / Windows 11 Pro.
- **Programming**: C++20, Python (SciPy stack), Perl, R.
- **Software**:
  - *Development*: J2EE, .NET 6, MATLAB, GCC/Clang.
  - *DB/Cloud*: PostgreSQL, MongoDB, Docker.
  - *Specialized*: Rational Rose, VTK, LEDA, 8086 emulators.

CS342. Compilers (Mandatory) 6th Sem, Lab: 2 hr(s)

Advanced Computing Lab:

- **Hardware**: 30 high-end PCs (Intel Core i7, 16GB RAM, 512GB NVMe SSD, NVIDIA GTX 1660, Gigabit Ethernet).
- **OS**: Linux (Fedora) / Windows 11 Pro.
- **Programming**: C++20, Python (SciPy stack), Perl, R.
- **Software**:
  - *Development*: J2EE, .NET 6, MATLAB, GCC/Clang.
  - *DB/Cloud*: PostgreSQL, MongoDB, Docker.
  - *Specialized*: Rational Rose, VTK, LEDA, 8086 emulators.

AI263. Introduction to Machine Learning (Mandatory) 6th Sem, Lab: 4 hr(s)

Basic Computing Lab:

- **Hardware**: 30 workstations (Intel Core i5, 8GB RAM, 256GB SSD, integrated graphics/sound, Gigabit Ethernet).
- **OS**: Dual-boot Linux (Ubuntu LTS) / Windows 11 Pro.
- **Programming**: Python, Java, C++ (C++11/14/17), C#, Prolog, Lex/Yacc.
- **Tools**: Git, VS Code, Eclipse, Jupyter Notebook.

FI201. Computational Physics (Mandatory) 6th Sem, Lab: 2 hr(s)

Physics Laboratory:

- **Equipment and Tools**: Optical benches, force tables, pendulums, dynamic carts, spectrometers, PASCO sensors, and measurement modules (multimeters, power supplies, calipers).
- **Hardware and Technology**: Academic-use laptops, Arduino sensors, and modules for data acquisition.
- **Software**: Arduino IDE, MATLAB for analysis, Microsoft Office, and specialized experimental data acquisition software.
- **Services and Experiments**: Physics I and II lab sessions (Mechanics, Thermodynamics, Electromagnetism, Waves), instrumentation training, and support for research projects.

CS251. Computer Graphics (Mandatory) 7th Sem, Lab: 4 hr(s)

Advanced Computing Lab:

- **Hardware**: 30 high-end PCs (Intel Core i7, 16GB RAM, 512GB NVMe SSD, NVIDIA GTX 1660, Gigabit Ethernet).
- **OS**: Linux (Fedora) / Windows 11 Pro.
- **Programming**: C++20, Python (SciPy stack), Perl, R.

- **Software**:
  - *Development*: J2EE, .NET 6, MATLAB, GCC/Clang.
  - *DB/Cloud*: PostgreSQL, MongoDB, Docker.
  - *Specialized*: Rational Rose, VTK, LEDA, 8086 emulators.

CS292. Software Engineering II (Mandatory) 7th Sem, Lab: 2 hr(s)

Advanced Computing Lab:
- **Hardware**: 30 high-end PCs (Intel Core i7, 16GB RAM, 512GB NVMe SSD, NVIDIA GTX 1660, Gigabit Ethernet).
- **OS**: Linux (Fedora) / Windows 11 Pro.
- **Programming**: C++20, Python (SciPy stack), Perl, R.
- **Software**:
  - *Development*: J2EE, .NET 6, MATLAB, GCC/Clang.
  - *DB/Cloud*: PostgreSQL, MongoDB, Docker.
  - *Specialized*: Rational Rose, VTK, LEDA, 8086 emulators.

CS2H1. User Experience (UX) (Mandatory) 7th Sem, Lab: 4 hr(s)

Basic Computing Lab:
- **Hardware**: 30 workstations (Intel Core i5, 8GB RAM, 256GB SSD, integrated graphics/sound, Gigabit Ethernet).
- **OS**: Dual-boot Linux (Ubuntu LTS) / Windows 11 Pro.
- **Programming**: Python, Java, C++ (C++11/14/17), C#, Prolog, Lex/Yacc.
- **Tools**: Git, VS Code, Eclipse, Jupyter Notebook.

AI264. Deep Learning (Mandatory) 7th Sem, Lab: 4 hr(s)

Basic Computing Lab:
- **Hardware**: 30 workstations (Intel Core i5, 8GB RAM, 256GB SSD, integrated graphics/sound, Gigabit Ethernet).
- **OS**: Dual-boot Linux (Ubuntu LTS) / Windows 11 Pro.
- **Programming**: Python, Java, C++ (C++11/14/17), C#, Prolog, Lex/Yacc.
- **Tools**: Git, VS Code, Eclipse, Jupyter Notebook.

CS3I1. Computer Security (Mandatory) 8th Sem, Lab: 2 hr(s)

Advanced Computing Lab:
- **Hardware**: 30 high-end PCs (Intel Core i7, 16GB RAM, 512GB NVMe SSD, NVIDIA GTX 1660, Gigabit Ethernet).
- **OS**: Linux (Fedora) / Windows 11 Pro.
- **Programming**: C++20, Python (SciPy stack), Perl, R.
- **Software**:
  - *Development*: J2EE, .NET 6, MATLAB, GCC/Clang.
  - *DB/Cloud*: PostgreSQL, MongoDB, Docker.
  - *Specialized*: Rational Rose, VTK, LEDA, 8086 emulators.

CS3P1. Parallel and Distributed Computing (Mandatory) 8th Sem, Lab: 2 hr(s)

Advanced Computing Lab:
- **Hardware**: 30 high-end PCs (Intel Core i7, 16GB RAM, 512GB NVMe SSD, NVIDIA GTX 1660, Gigabit Ethernet).

- **OS**: Linux (Fedora) / Windows 11 Pro.
- **Programming**: C++20, Python (SciPy stack), Perl, R.
- **Software**:
  - *Development*: J2EE, .NET 6, MATLAB, GCC/Clang.
  - *DB/Cloud*: PostgreSQL, MongoDB, Docker.
  - *Specialized*: Rational Rose, VTK, LEDA, 8086 emulators.

CS402. Capstone Project I (Mandatory) 8th Sem, Lab: 4 hr(s)

Advanced Computing Lab:

- **Hardware**: 30 high-end PCs (Intel Core i7, 16GB RAM, 512GB NVMe SSD, NVIDIA GTX 1660, Gigabit Ethernet).
- **OS**: Linux (Fedora) / Windows 11 Pro.
- **Programming**: C++20, Python (SciPy stack), Perl, R.
- **Software**:
  - *Development*: J2EE, .NET 6, MATLAB, GCC/Clang.
  - *DB/Cloud*: PostgreSQL, MongoDB, Docker.
  - *Specialized*: Rational Rose, VTK, LEDA, 8086 emulators.

AI268. Computational Vision (Elective) 8th Sem, Lab: 2 hr(s)

Advanced Computing Lab:

- **Hardware**: 30 high-end PCs (Intel Core i7, 16GB RAM, 512GB NVMe SSD, NVIDIA GTX 1660, Gigabit Ethernet).
- **OS**: Linux (Fedora) / Windows 11 Pro.
- **Programming**: C++20, Python (SciPy stack), Perl, R.
- **Software**:
  - *Development*: J2EE, .NET 6, MATLAB, GCC/Clang.
  - *DB/Cloud*: PostgreSQL, MongoDB, Docker.
  - *Specialized*: Rational Rose, VTK, LEDA, 8086 emulators.

CS393. Information systems (Elective) 8th Sem, Lab: 2 hr(s)

Advanced Computing Lab:

- **Hardware**: 30 high-end PCs (Intel Core i7, 16GB RAM, 512GB NVMe SSD, NVIDIA GTX 1660, Gigabit Ethernet).
- **OS**: Linux (Fedora) / Windows 11 Pro.
- **Programming**: C++20, Python (SciPy stack), Perl, R.
- **Software**:
  - *Development*: J2EE, .NET 6, MATLAB, GCC/Clang.
  - *DB/Cloud*: PostgreSQL, MongoDB, Docker.
  - *Specialized*: Rational Rose, VTK, LEDA, 8086 emulators.

CS370. Big Data (Mandatory) 9th Sem, Lab: 2 hr(s)

Advanced Computing Lab:

- **Hardware**: 30 high-end PCs (Intel Core i7, 16GB RAM, 512GB NVMe SSD, NVIDIA GTX 1660, Gigabit Ethernet).
- **OS**: Linux (Fedora) / Windows 11 Pro.
- **Programming**: C++20, Python (SciPy stack), Perl, R.
- **Software**:
  - *Development*: J2EE, .NET 6, MATLAB, GCC/Clang.
  - *DB/Cloud*: PostgreSQL, MongoDB, Docker.
  - *Specialized*: Rational Rose, VTK, LEDA, 8086 emulators.

CS403. Capstone Project II (Mandatory) 9th Sem, Lab: 4 hr(s)

Advanced Computing Lab:

- **Hardware**: 30 high-end PCs (Intel Core i7, 16GB RAM, 512GB NVMe SSD, NVIDIA GTX 1660, Gigabit Ethernet).
- **OS**: Linux (Fedora) / Windows 11 Pro.
- **Programming**: C++20, Python (SciPy stack), Perl, R.
- **Software**:
  - *Development*: J2EE, .NET 6, MATLAB, GCC/Clang.
  - *DB/Cloud*: PostgreSQL, MongoDB, Docker.
  - *Specialized*: Rational Rose, VTK, LEDA, 8086 emulators.

AI365. Advanced Generative AI Models (Mandatory) 9th Sem, Lab: 4 hr(s)

Basic Computing Lab:

- **Hardware**: 30 workstations (Intel Core i5, 8GB RAM, 256GB SSD, integrated graphics/sound, Gigabit Ethernet).
- **OS**: Dual-boot Linux (Ubuntu LTS) / Windows 11 Pro.
- **Programming**: Python, Java, C++ (C++11/14/17), C#, Prolog, Lex/Yacc.
- **Tools**: Git, VS Code, Eclipse, Jupyter Notebook.

AI369. Robotics (Elective) 9th Sem, Lab: 2 hr(s)

Advanced Computing Lab:

- **Hardware**: 30 high-end PCs (Intel Core i7, 16GB RAM, 512GB NVMe SSD, NVIDIA GTX 1660, Gigabit Ethernet).
- **OS**: Linux (Fedora) / Windows 11 Pro.
- **Programming**: C++20, Python (SciPy stack), Perl, R.
- **Software**:
  - *Development*: J2EE, .NET 6, MATLAB, GCC/Clang.
  - *DB/Cloud*: PostgreSQL, MongoDB, Docker.
  - *Specialized*: Rational Rose, VTK, LEDA, 8086 emulators.

CS351. Topics in Computer Graphics (Elective) 9th Sem, Lab: 2 hr(s)

Advanced Computing Lab:

- **Hardware**: 30 high-end PCs (Intel Core i7, 16GB RAM, 512GB NVMe SSD, NVIDIA GTX 1660, Gigabit Ethernet).
- **OS**: Linux (Fedora) / Windows 11 Pro.
- **Programming**: C++20, Python (SciPy stack), Perl, R.
- **Software**:
  - *Development*: J2EE, .NET 6, MATLAB, GCC/Clang.
  - *DB/Cloud*: PostgreSQL, MongoDB, Docker.
  - *Specialized*: Rational Rose, VTK, LEDA, 8086 emulators.

CS353. Quantum Computing (Elective) 9th Sem, Lab: 2 hr(s)

Advanced Computing Lab:

- **Hardware**: 30 high-end PCs (Intel Core i7, 16GB RAM, 512GB NVMe SSD, NVIDIA GTX 1660, Gigabit Ethernet).
- **OS**: Linux (Fedora) / Windows 11 Pro.
- **Programming**: C++20, Python (SciPy stack), Perl, R.

- **Software**:
  - *Development*: J2EE, .NET 6, MATLAB, GCC/Clang.
  - *DB/Cloud*: PostgreSQL, MongoDB, Docker.
  - *Specialized*: Rational Rose, VTK, LEDA, 8086 emulators.

CS392. Tópicos en Ingeniería de Software (Elective) 9th Sem, Lab: 2 hr(s)

Advanced Computing Lab:

- **Hardware**: 30 high-end PCs (Intel Core i7, 16GB RAM, 512GB NVMe SSD, NVIDIA GTX 1660, Gigabit Ethernet).
- **OS**: Linux (Fedora) / Windows 11 Pro.
- **Programming**: C++20, Python (SciPy stack), Perl, R.
- **Software**:
  - *Development*: J2EE, .NET 6, MATLAB, GCC/Clang.
  - *DB/Cloud*: PostgreSQL, MongoDB, Docker.
  - *Specialized*: Rational Rose, VTK, LEDA, 8086 emulators.

CS3P3. Internet of Things (Elective) 9th Sem, Lab: 4 hr(s)

Advanced Computing Lab:

- **Hardware**: 30 high-end PCs (Intel Core i7, 16GB RAM, 512GB NVMe SSD, NVIDIA GTX 1660, Gigabit Ethernet).
- **OS**: Linux (Fedora) / Windows 11 Pro.
- **Programming**: C++20, Python (SciPy stack), Perl, R.
- **Software**:
  - *Development*: J2EE, .NET 6, MATLAB, GCC/Clang.
  - *DB/Cloud*: PostgreSQL, MongoDB, Docker.
  - *Specialized*: Rational Rose, VTK, LEDA, 8086 emulators.

CS3P2. Cloud Computing (Mandatory) 10th Sem, Lab: 2 hr(s)

Advanced Computing Lab:

- **Hardware**: 30 high-end PCs (Intel Core i7, 16GB RAM, 512GB NVMe SSD, NVIDIA GTX 1660, Gigabit Ethernet).
- **OS**: Linux (Fedora) / Windows 11 Pro.
- **Programming**: C++20, Python (SciPy stack), Perl, R.
- **Software**:
  - *Development*: J2EE, .NET 6, MATLAB, GCC/Clang.
  - *DB/Cloud*: PostgreSQL, MongoDB, Docker.
  - *Specialized*: Rational Rose, VTK, LEDA, 8086 emulators.

CS404. Research Workshop (Mandatory) 10th Sem, Lab: 4 hr(s)

Advanced Computing Lab:

- **Hardware**: 30 high-end PCs (Intel Core i7, 16GB RAM, 512GB NVMe SSD, NVIDIA GTX 1660, Gigabit Ethernet).
- **OS**: Linux (Fedora) / Windows 11 Pro.
- **Programming**: C++20, Python (SciPy stack), Perl, R.
- **Software**:
  - *Development*: J2EE, .NET 6, MATLAB, GCC/Clang.
  - *DB/Cloud*: PostgreSQL, MongoDB, Docker.
  - *Specialized*: Rational Rose, VTK, LEDA, 8086 emulators.

AI367. Topics in Artificial Intelligence (Mandatory) 10th Sem, Lab: 2 hr(s)

Advanced Computing Lab:

- **Hardware**: 30 high-end PCs (Intel Core i7, 16GB RAM, 512GB NVMe SSD, NVIDIA GTX 1660, Gigabit Ethernet).
- **OS**: Linux (Fedora) / Windows 11 Pro.
- **Programming**: C++20, Python (SciPy stack), Perl, R.
- **Software**:
  - *Development*: J2EE, .NET 6, MATLAB, GCC/Clang.
  - *DB/Cloud*: PostgreSQL, MongoDB, Docker.
  - *Specialized*: Rational Rose, VTK, LEDA, 8086 emulators.

AI368. Evolutionary Computing (Mandatory) 10th Sem, Lab: 2 hr(s)

Advanced Computing Lab:

- **Hardware**: 30 high-end PCs (Intel Core i7, 16GB RAM, 512GB NVMe SSD, NVIDIA GTX 1660, Gigabit Ethernet).
- **OS**: Linux (Fedora) / Windows 11 Pro.
- **Programming**: C++20, Python (SciPy stack), Perl, R.
- **Software**:
  - *Development*: J2EE, .NET 6, MATLAB, GCC/Clang.
  - *DB/Cloud*: PostgreSQL, MongoDB, Docker.
  - *Specialized*: Rational Rose, VTK, LEDA, 8086 emulators.