

**San Pablo Catholic University (UCSP)**  
**Undergraduate Program in**  
**Computer Science**  
**SILABO**

**CS392. Advanced Topics in Software Engineering**  
**(Elective)**



2020-I

**1. General information**

1.1 School	:	Ciencia de la Computación
1.2 Course	:	CS392. Advanced Topics in Software Engineering
1.3 Semester	:	9 <sup>no</sup> Semestre.
1.4 Prerequisites	:	CS391. Software Engineering III. (7 <sup>th</sup> Sem)
1.5 Type of course	:	Elective
1.6 Learning modality	:	Virtual
1.7 Horas	:	2 HT; 2 HP; 2 HL;
1.8 Credits	:	4

**2. Professors**

**3. Course foundation**

El desarrollo de software requiere del uso de mejores prácticas de desarrollo, gestión de proyectos de TI, manejo de equipos y uso eficiente y racional de frameworks de aseguramiento de la calidad y de Gobierno de Portfolios, estos elemento son pieza clave y transversal para el éxito del proceso productivo.

Este curso explora el diseño, selección, implementación y gestión de soluciones TI en las Organizaciones. El foco está en las aplicaciones y la infraestructura y su aplicación en el negocio.

**4. Summary**

1. Software Design 2. Software Project Management 3. 4.

**5. Generales Goals**

- Entender una variedad de frameworks para el análisis de arquitectura empresarial y la toma de decisiones
- Utilizar técnicas para la evaluación y gestión del riesgo en el portfolio de la empresa
- Evaluar y planificar la integración de tecnologías emergentes
- Entender el papel y el potencial de las TI para a apoyar la gestión de procesos empresariales
- Entender los difentes enfoques para modelar y mejorar los procesos de negocio
- Describir y comprender modelos de aseguramiento de la calidad como marco clave para el éxitos de los proyectos de TI.
- Comprender y aplicar el framework de IT Governance como elemento clave para la gestión del portfolio de aplicaciones Empresariales

## 6. Contribution to Outcomes

This discipline contributes to the achievement of the following outcomes:

- 1) Analyze a complex computing problem and to apply principles of computing and other relevant disciplines to identify solutions. (**Assessment**)
- 2) Design, implement and evaluate a computing-based solution to meet a given set of computing requirements in the context of the program's discipline. (**Assessment**)
- 3) Communicate effectively in a variety of professional contexts. (**Usage**)
- 5) Function effectively as a member or leader of a team engaged in activities appropriate to the program's discipline. (**Usage**)
- 6) Apply computer science theory and software development fundamentals to produce computing-based solutions. (**Assessment**)
- 7) Develop computational technology for the well-being of all, contributing with human formation, scientific, technological and professional skills to solve social problems of our community. (**Assessment**)

## 7. Content

**UNIT 1: Software Design (18)****Competences:****Content****Generales Goals**

- System design principles: levels of abstraction (architectural design and detailed design), separation of concerns, information hiding, coupling and cohesion , re-use of standard structures
- Design Paradigms such as structured design (top-down functional decomposition), object-oriented analysis and design, event driven design, component-level design, data-structured centered, aspect oriented, function oriented, service oriented
- Structural and behavioral models of software designs
- Design patterns
- Relationships between requirements and designs: transformation of models, design of contracts, invariants
- Software architecture concepts and standard architectures (e.g. client-server, n-layer, transform centered, pipes-and-filters)
- The use of component desing: component selection, design, adaptation and assembly of components, component and patterns, components and objects (for example, building a GUI using a standar widget set)
- Refactoring designs using design patterns
- Internal design qualities, and models for them: efficiency and performance, redundancy and fault tolerance, traceability of requeriments
- Measurement and analysis of design quality
- Tradeoffs between different aspects of quality
- Application frameworks
- Middleware: the object-oriented paradigm within middleware, object request brokers and marshalling, transaction processing monitors, workflow systems
- Principles of secure design and coding
  - Principle of least privilege
  - Principle of fail-safe defaults
  - Principle of psychological acceptability

- Articulate design principles including separation of concerns, information hiding, coupling and cohesion, and encapsulation [Usage]
- Use a design paradigm to design a simple software system, and explain how system design principles have been applied in this design [Usage]
- Construct models of the design of a simple software system that are appropriate for the paradigm used to design it [Usage]
- Within the context of a single design paradigm, describe one or more design patterns that could be applicable to the design of a simple software system [Usage]
- For a simple system suitable for a given scenario, discuss and select an appropriate design paradigm [Usage]
- Create appropriate models for the structure and behavior of software products from their requirements specifications [Usage]
- Explain the relationships between the requirements for a software product and its design, using appropriate models [Usage]
- For the design of a simple software system within the context of a single design paradigm, describe the software architecture of that system [Usage]
- Given a high-level design, identify the software architecture by differentiating among common software architectures such as 3-tier, pipe-and-filter, and client-server [Usage]
- Investigate the impact of software architectures selection on the design of a simple system [Usage]
- Apply simple examples of patterns in a software design [Usage]
- Describe a form of refactoring and discuss when it may be applicable [Usage]
- Select suitable components for use in the design of a software product [Usage]
- Explain how suitable components might need to be adapted for use in the design of a software product [Usage]
- Design a contract for a typical small software component for use in a given system [Usage]
- Discuss and select appropriate software architecture for a simple system suitable for a given scenario [Usage]
- Apply models for internal and external qualities in designing software components to achieve an acceptable tradeoff between conflicting quality aspects [Usage]

## UNIT 2: Software Project Management (14)

### Competences:

#### Content

- Team participation
  - Team processes including responsibilities for task, meeting structure, and work schedule
  - Roles and responsibilities in a software team
  - Team conflict resolution
  - Risks associated with virtual teams (communication, perception, structure)
- Effort estimation (at the personal level)
- Risk
  - The role of risk in the lifecycle
  - Risk categories including security, safety, market, financial, technology, people, quality, structure and process
- Team management
  - Team organization and decision-making
  - Role identification and assignment
  - Individual and team performance assessment
- Project management
  - Scheduling and tracking
  - Project management tools
  - Cost/benefit analysis
- Software measurement and estimation techniques
- Software quality assurance and the role of measurements
- Risk
  - The role of risk in the lifecycle
  - Risk categories including security, safety, market, financial, technology, people, quality, structure and process
- System-wide approach to risk including hazards associated with tools

#### Generales Goals

- Discuss common behaviors that contribute to the effective functioning of a team [Usage]
- Create and follow an agenda for a team meeting [Usage]
- Identify and justify necessary roles in a software development team [Usage]
- Understand the sources, hazards, and potential benefits of team conflict [Usage]
- Apply a conflict resolution strategy in a team setting [Usage]
- Use an ad hoc method to estimate software development effort (eg, time) and compare to actual effort required [Usage]
- List several examples of software risks [Usage]
- Describe the impact of risk in a software development lifecycle [Usage]
- Describe different categories of risk in software systems [Usage]
- Demonstrate through involvement in a team project the central elements of team building and team management [Usage]
- Describe how the choice of process model affects team organizational structures and decision-making processes [Usage]
- Create a team by identifying appropriate roles and assigning roles to team members [Usage]
- Assess and provide feedback to teams and individuals on their performance in a team setting [Usage]
- Using a particular software process, describe the aspects of a project that need to be planned and monitored, (eg, estimates of size and effort, a schedule, resource allocation, configuration control, change management, and project risk identification and management) [Usage]
- Track the progress of some stage in a project using appropriate project metrics [Usage]
- Compare simple software size and cost estimation techniques [Usage]
- Use a project management tool to assist in the assignment and tracking of tasks in a software development project [Usage]
- Describe the impact of risk tolerance on the software development process [Usage]
- Identify risks and describe approaches to managing risk (avoidance, acceptance, transference, mitigation), and characterize the strengths and short-

<b>UNIT 3: (14)</b>	
<b>Competences:</b>	
<b>Content</b>	<b>Generales Goals</b>
<ul style="list-style-type: none"> <li>• Administración del servicio como práctica.</li> <li>• Ciclo de vida del servicio.</li> <li>• Definiciones y conceptos genéricos.</li> <li>• Modelos y principios claves.</li> <li>• Procesos.</li> <li>• Tecnología y arquitectura.</li> <li>• Competencia y entrenamiento.</li> </ul>	<ul style="list-style-type: none"> <li>• Utilizar y aplicar correctamente ITIL en el proceso de software. [Usage]</li> </ul>
<b>Readings:</b> Sommerville (2017), Pressman and Maxim (2015)	

<b>UNIT 4: (14)</b>	
<b>Competences:</b>	
<b>Content</b>	<b>Generales Goals</b>
<ul style="list-style-type: none"> <li>• Fundamentos e Introducción.</li> <li>• Frameworks de Control y IT Governance.</li> </ul>	<ul style="list-style-type: none"> <li>• Utilizar y aplicar correctamente COBIT en el proceso de software. [Usage]</li> </ul>
<b>Readings:</b> Sommerville (2017), Pressman and Maxim (2015)	

8. Methodology
<p>El profesor del curso presentará clases teóricas de los temas señalados en el programa propiciando la intervención de los alumnos.</p> <p>El profesor del curso presentará demostraciones para fundamentar clases teóricas.</p> <p>El profesor y los alumnos realizarán prácticas</p> <p>Los alumnos deberán asistir a clase habiendo leído lo que el profesor va a presentar. De esta manera se facilitará la comprensión y los estudiantes estarán en mejores condiciones de hacer consultas en clase.</p>

9. Assessment
<p><b>Continuous Assessment 1</b> : 20 %</p> <p><b>Partial Exam</b> : 30 %</p> <p><b>Continuous Assessment 2</b> : 20 %</p> <p><b>Final exam</b> : 30 %</p>

## References

Pressman, Roger S. and Bruce Maxim (Jan. 2015). *Software Engineering: A Practitioner's Approach*. 8th. McGraw-Hill.  
Sommerville, Ian (Mar. 2017). *Software Engineering*. 10th. Pearson.