



**National University of the Altiplano (UNA)**  
School of Computer Science  
Syllabus 2024-II

**1. COURSE**

CS392. Tópicos en Ingeniería de Software (Elective)

**2. GENERAL INFORMATION**

- 2.1 Course : CS392. Tópicos en Ingeniería de Software
- 2.2 Semester : 9<sup>th</sup> Semester.
- 2.3 Credits : 4
- 2.4 Horas : 2 HT; 4 HP;
  
- 2.5 Duration of the period : 16 weeks
- 2.6 Type of course : Elective
- 2.7 Learning modality : Face to face
- 2.8 Prerequisites : CS391. Software Engineering III. (7<sup>th</sup> Sem)  
CS391. Software Engineering III. (7<sup>th</sup> Sem)

**3. PROFESSORS**

Meetings after coordination with the professor

**4. INTRODUCTION TO THE COURSE**

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 elementos 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.

**5. 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 apoyar la gestión de procesos empresariales
- Entender los diferentes enfoques para modelar y mejorar los procesos de negocio
- Describir y comprender modelos de aseguramiento de la calidad como marco clave para el éxito 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. COMPETENCES**

) ()

) ()

) ()

) ()

6) Apply computer science theory and software development fundamentals to produce computing-based solutions. ()

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. ()

## 7. TOPICS

Unit 1: Diseño de Software (18)	
Competences Expected:	
Topics	Learning Outcomes
<ul style="list-style-type: none"> <li>• Principios de diseño del sistema: niveles de abstracción (diseño arquitectónico y el diseño detallado), separación de intereses, ocultamiento de información, de acoplamiento y de cohesión, de reutilización de estructuras estándar.</li> <li>• Diseño de paradigmas tales como diseño estructurado (descomposición funcional de arriba hacia abajo), el análisis orientado a objetos y diseño, orientado a eventos de diseño, diseño de nivel de componente, centrado datos estructurada, orientada a aspectos, orientado a la función, orientado al servicio.</li> <li>• Modelos estructurales y de comportamiento de los diseños de software.</li> <li>• Diseño de patrones.</li> <li>• Relaciones entre los requisitos y diseños: La transformación de modelos, el diseño de los contratos, invariantes.</li> <li>• Conceptos de arquitectura de software y arquitecturas estándar (por ejemplo, cliente-servidor, n-capas, transforman centrados, tubos y filtros).</li> <li>• El uso de componentes de diseño: selección de componentes, diseño, adaptación y componentes de ensamblaje, componentes y patrones, componentes y objetos (por ejemplo, construir una GUI usando un standard widget set)</li> <li>• Calidad del diseño interno, y modelos para: eficiencia y desempeño, redundancia y tolerancia a fallos, trazabilidad de los requerimientos.</li> <li>• Calidad del diseño interno, y modelos para: eficiencia y desempeño, redundancia y tolerancia a fallos, trazabilidad de los requerimientos.</li> <li>• Medición y análisis de la calidad de un diseño.</li> <li>• Compensaciones entre diferentes aspectos de la calidad.</li> <li>• Aplicaciones en frameworks.</li> <li>• Middleware: El paradigma de la orientación a objetos con middleware, requerimientos para correr y clasificar objetos, monitores de procesamiento de transacciones y el sistema de flujo de trabajo.</li> <li>• Principales diseños de seguridad y codificación (cross-reference IAS/Principles of secure design). <ul style="list-style-type: none"> <li>– Principio de privilegios mínimos</li> <li>– Principio de falla segura por defecto</li> <li>– Principio de aceptabilidad psicológica</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>• Formular los principios de diseño, incluyendo la separación de problemas, ocultación de información, acoplamiento y cohesión, y la encapsulación [Usar]</li> <li>• Usar un paradigma de diseño para diseñar un sistema de software básico y explicar cómo los principios de diseño del sistema se han aplicado en este diseño [Usar]</li> <li>• Construir modelos del diseño de un sistema de software simple los cuales son apropiado para el paradigma utilizado para diseñarlo [Usar]</li> <li>• En el contexto de un paradigma de diseño simple, describir uno o más patrones de diseño que podrían ser aplicables al diseño de un sistema de software simple [Usar]</li> <li>• Para un sistema simple adecuado para una situación dada, discutir y seleccionar un paradigma de diseño apropiado [Usar]</li> <li>• Crear modelos apropiados para la estructura y el comportamiento de los productos de software desde la especificaciones de requisitos [Usar]</li> <li>• Explicar las relaciones entre los requisitos para un producto de software y su diseño, utilizando los modelos apropiados [Usar]</li> <li>• Para el diseño de un sistema de software simple dentro del contexto de un único paradigma de diseño, describir la arquitectura de software de ese sistema [Usar]</li> <li>• Dado un diseño de alto nivel, identificar la arquitectura de software mediante la diferenciación entre las arquitecturas comunes de software, tales como 3 capas (<i>3-tier</i>), <i>pipe-and-filter</i>, y cliente-servidor [Usar]</li> <li>• Investigar el impacto de la selección arquitecturas de software en el diseño de un sistema simple [Usar]</li> <li>• Aplicar ejemplos simples de patrones en un diseño de software [Usar]</li> <li>• Describir una manera de refactorar y discutir cuando esto debe ser aplicado [Usar]</li> <li>• Seleccionar componentes adecuados para el uso en un diseño de un producto de software [Usar]</li> <li>• Explicar cómo los componentes deben ser adaptados para ser usados en el diseño de un producto de software [Usar]</li> <li>• Diseñar un contrato para un típico componente de software pequeño para el uso de un dado sistema [Usar]</li> <li>• Discutir y seleccionar la arquitectura de software adecuada para un sistema de software simple para un dado escenario [Usar]</li> </ul>

**Unit 2: Gestión de Proyectos de Software (14)****Competences Expected:****Topics****Learning Outcomes**

- La participación del equipo:
  - Procesos elemento del equipo, incluyendo responsabilidades de tarea, la estructura de reuniones y horario de trabajo
  - Roles y responsabilidades en un equipo de software
  - Equipo de resolución de conflictos
  - Los riesgos asociados con los equipos virtuales (comunicación, la percepción, la estructura)
- Estimación de esfuerzo (a nivel personal)
- Riesgo.
  - El papel del riesgo en el ciclo de vida
  - Categorías elemento de riesgo, incluyendo la seguridad, la seguridad, mercado, finanzas, tecnología, las personas, la calidad, la estructura y el proceso de
- Gestión de equipos:
  - Organización de equipo y la toma de decisiones
  - Roles de identificación y asignación
  - Individual y el desempeño del equipo de evaluación
- Gestión de proyectos:
  - Programación y seguimiento de elementos
  - Herramientas de gestión de proyectos
  - Análisis de Costo/Beneficio
- Software de medición y técnicas de estimación.
- Aseguramiento de la calidad del software y el rol de las mediciones.
- Riesgo.
  - El papel del riesgo en el ciclo de vida
  - Categorías elemento de riesgo, incluyendo la seguridad, la seguridad, mercado, finanzas, tecnología, las personas, la calidad, la estructura y el proceso de
- En todo el sistema de aproximación al riesgo, incluyendo riesgos asociados con herramientas.

- Discutir los comportamientos comunes que contribuyen al buen funcionamiento de un equipo [Usar]
- Crear y seguir un programa para una reunión del equipo [Usar]
- Identificar y justificar las funciones necesarias en un equipo de desarrollo de software [Usar]
- Entender las fuentes, obstáculos y beneficios potenciales de un conflicto de equipo [Usar]
- Aplicar una estrategia de resolución de conflictos en un ambiente de equipo [Usar]
- Utilizar un método ad hoc para estimar el esfuerzo de desarrollo del software (ejemplo, tiempo) y comparar con el esfuerzo actual requerido [Usar]
- Listar varios ejemplos de los riesgos del software [Usar]
- Describir el impacto del riesgo en el ciclo de vida de desarrollo de software [Usar]
- Describir las diferentes categorías de riesgo en los sistemas de software [Usar]
- Demostrar a través de la colaboración de proyectos de equipo los elementos centrales de la construcción de equipos y gestión de equipos [Usar]
- Describir como la elección de modelos de procesos afectan la estructura organizacional de equipos y procesos de toma de decisiones [Usar]
- Crear un equipo mediante la identificación de los roles apropiados y la asignación de funciones a los miembros del equipo [Usar]
- Evaluar y retroalimentar a los equipos e individuos sobre su desempeño en un ambiente de equipo [Usar]
- Usando un software particular procesar, describir los aspectos de un proyecto que necesita ser planeado y monitoreado, (ejemplo, estimar el tamaño y esfuerzo, un horario, reasignación de recursos, control de configuración, gestión de cambios, identificación de riesgos en un proyecto y gestión) [Usar]
- Realizar el seguimiento del progreso de alguna etapa de un proyecto que utiliza métricas de proyectos apropiados [Usar]
- Comparar las técnicas simples de tamaño de software y estimación de costos [Usar]
- Usar una herramienta de gestión de proyectos para ayudar en la asignación y rastreo de tareas en un proyecto de desarrollo de software [Usar]
- Describir el impacto del riesgo en el ciclo de vida de desarrollo de software [Usar]

<b>Unit 3: (14)</b>	
<b>Competences Expected:</b>	
<b>Topics</b>	<b>Learning Outcomes</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. [Usar]</li> </ul>
<b>Readings :</b> [Sommerville2010], [Pressman2014]	

<b>Unit 4: (14)</b>	
<b>Competences Expected:</b>	
<b>Topics</b>	<b>Learning Outcomes</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. [Usar]</li> </ul>
<b>Readings :</b> [Sommerville2010], [Pressman2014]	

## 8. WORKPLAN

### 8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

### 8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

### 8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

## 9. EVALUATION SYSTEM

\*\*\*\*\* EVALUATION MISSING \*\*\*\*\*

## 10. BASIC BIBLIOGRAPHY