

Universidad Católica San Pablo (UCSP)
Escuela Profesional de
Ciencia de la Computación
SILABO

CS392. Tópicos Avanzados en Ingeniería de Software
(Electivo)



2022-II

1. Información general

1.1 Escuela	:	Ciencia de la Computación
1.2 Curso	:	CS392. Tópicos Avanzados en Ingeniería de Software
1.3 Semestre	:	9 ^{no} Semestre.
1.4 Prerrequisitos	:	CS391. Ingeniería de Software III. (7 ^{mo} Sem)
1.5 Condición	:	Electivo
1.6 Modalidad de aprendizaje	:	Virtual
1.7 horas	:	2 HT; 2 HP; 2 HL;
1.8 Créditos	:	4

2. Profesores

3. Fundamentación del curso

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. Resumen

1. Diseño de Software 2. Gestión de Proyectos de Software 3. 4.

5. Objetivos Generales

- 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. Contribución a los resultados (*Outcomes*)

Esta disciplina contribuye al logro de los siguientes resultados de la carrera:

- c) Diseñar, implementar y evaluar un sistema, proceso, componente o programa computacional para alcanzar las necesidades deseadas. (**Usar**)
- d) Trabajar efectivamente en equipos para cumplir con un objetivo común. (**Usar**)
- i) Utilizar técnicas y herramientas actuales necesarias para la práctica de la computación. (**Usar**)
- j) Aplicar la base matemática, principios de algoritmos y la teoría de la CS en el modelamiento y diseño de sistemas. (**Evaluar**)
- m) Transformar sus conocimientos del área de Ciencia de la Computación en emprendimientos tecnológicos. (**Evaluar**)
- o) Comprender que la formación en humanidades de un buen profesional no se desliga ni se opone sino mas bien contribuye al auténtico crecimiento personal. (**Usar**)

7. Contenido

UNIDAD 1: Diseño de Software (18)**Competencias: c,d,i,j,m,o****Contenido**

- 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.
- 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.
- Modelos estructurales y de comportamiento de los diseños de software.
- Diseño de patrones.
- Relaciones entre los requisitos y diseños: La transformación de modelos, el diseño de los contratos, invariantes.
- Conceptos de arquitectura de software y arquitecturas estándar (por ejemplo, cliente-servidor, n-capas, transforman centrados, tubos y filtros).
- 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)
- Diseños de refactorización utilizando patrones de diseño
- Calidad del diseño interno, y modelos para: eficiencia y desempeño, redundancia y tolerancia a fallos, trazabilidad de los requerimientos.
- Medición y análisis de la calidad de un diseño.
- Compensaciones entre diferentes aspectos de la calidad.
- Aplicaciones en frameworks.
- 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.
- Principales diseños de seguridad y codificación (cross-reference IAS/Principles of secure design).
 - Principio de privilegios mínimos
 - Principio de falla segura por defecto
 - Principio de aceptabilidad psicológica

Objetivos Generales

- Formular los principios de diseño, incluyendo la separación de problemas, ocultación de información, acoplamiento y cohesión, y la encapsulación [Usar]
- 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]
- Construir modelos del diseño de un sistema de software simple los cuales son apropiado para el paradigma utilizado para diseñarlo [Usar]
- 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]
- Para un sistema simple adecuado para una situación dada, discutir y seleccionar un paradigma de diseño apropiado [Usar]
- Crear modelos apropiados para la estructura y el comportamiento de los productos de software desde la especificaciones de requisitos [Usar]
- Explicar las relaciones entre los requisitos para un producto de software y su diseño, utilizando los modelos apropiados [Usar]
- 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]
- Dado un diseño de alto nivel, identificar la arquitectura de software mediante la diferenciación entre las arquitecturas comunes de software, tales como 3 capas (*3-tier*), *pipe-and-filter*, y cliente-servidor [Usar]
- Investigar el impacto de la selección arquitecturas de software en el diseño de un sistema simple [Usar]
- Aplicar ejemplos simples de patrones en un diseño de software [Usar]
- Describir una manera de refactorar y discutir cuando esto debe ser aplicado [Usar]
- Seleccionar componentes adecuados para el uso en un diseño de un producto de software [Usar]
- Explicar cómo los componentes deben ser adaptados para ser usados en el diseño de un producto de software [Usar]
- Diseñar un contrato para un típico componente de software pequeño para el uso de un dado sistema [Usar]
- Discutir y seleccionar la arquitectura de software adecuada para un sistema de software simple para un dado escenario [Usar]

UNIDAD 2: Gestión de Proyectos de Software (14)**Competencias: c,d,i,j,m,o****Contenido****Objetivos Generales**

- 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 de la tolerancia de riesgos en el proceso de desarrollo de software [Usar]

UNIDAD 3: (14)	
Competencias: c,d,i,j,m	
Contenido	Objetivos Generales
<ul style="list-style-type: none"> • Administración del servicio como práctica. • Ciclo de vida del servicio. • Definiciones y conceptos genéricos. • Modelos y principios claves. • Procesos. • Tecnología y arquitectura. • Competencia y entrenamiento. 	<ul style="list-style-type: none"> • Utilizar y aplicar correctamente ITIL en el proceso de software. [Usar]
Lecturas: Sommerville (2017), Pressman and Maxim (2015)	

UNIDAD 4: (14)	
Competencias: c,d,i,j,m	
Contenido	Objetivos Generales
<ul style="list-style-type: none"> • Fundamentos e Introducción. • Frameworks de Control y IT Governance. 	<ul style="list-style-type: none"> • Utilizar y aplicar correctamente COBIT en el proceso de software. [Usar]
Lecturas: Sommerville (2017), Pressman and Maxim (2015)	

8. Metodología
<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. Evaluar
<p>Evaluación Continua 1 : 20 %</p> <p>Examen parcial : 30 %</p> <p>Evaluación Continua 2 : 20 %</p> <p>Examen final : 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.