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

Universidad Católica San Pablo 2025-I

CS292. Ingeniería de Software II (Obligatorio)

1. Información general

1.1 Escuela	:	Ciencia de la Computación
1.2 Curso	:	CS292. Ingeniería de Software II
1.3 Semestre	:	6^{to} Semestre
1.4 Prerrequisitos	:	CS291. Ingeniería de Software I. (5^{to} Sem)
1.5 Condición	:	Obligatorio
1.6 Modalidad de aprendizaje	:	Presencial
Horas	:	2 HT; 4 HP;
1.7 Créditos	:	4
1.8 Plan	:	Plan Curricular 2016

2. Profesores

Titular:

- Guillermo Enrique Calderón Ruiz <gcalderon@ucsp.edu.pe>
  - – Doctor en Ciencias de la Ingeniería, Pontificia Universidad Católica de Chile, Chile, 2011.
  - – Maestría en Ingeniería de Sistemas, Universidad Católica Santa María, Perú, 2009..

3. Fundamentación del curso

Los tópicos de este curso extienden las ideas del diseño y desarrollo de software desde la introducción a la programación para abarcar los problemas encontrados en proyectos de gran escala. Es una visión más amplia y completa de la Ingeniería de Software, apreciada desde un punto de vista de Proyectos.

4. Resumen

1. Herramientas y Entornos 2. Verificación y Validación de Software 3. Evolución de Software 4. Gestión de Proyectos de Software

5. Objetivos Generales

- Capacitar a los alumnos para formar parte y definir equipos de desarrollo de software que afronten problemas de envergadura real.
- Familiarizar a los alumnos con el proceso de administración de un proyecto de software de tal manera que sean capaces de crear, mejorar y utilizar herramientas y métricas que les permitan realizar la estimación y seguimiento de un proyecto de software.
- Crear, evaluar e implementar un plan de prueba para segmentos de código de tamaño medio. Distinguir entre los diferentes tipos de pruebas y sentar las bases para crear y mejorar los procedimientos de prueba y las herramientas utilizadas con ese propósito.
- Seleccionar, con justificación, un apropiado conjunto de herramientas para soportar el desarrollo de un rango de productos de software.
- Crear, mejorar y utilizar los patrones existentes para el mantenimiento de software. Dar a conocer las características y patrones de diseño para la reutilización de software.
- Identificar y discutir diferentes sistemas especializados, crear, mejorar y utilizar los patrones especializados para el diseño, implementación, mantenimiento y prueba de sistemas especializados.

6. Contribución a los resultados (Outcomes)

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

1) S.O. Analizar un problema computacional complejo y aplicar los principios computacionales y otras disciplinas relevantes para identificar soluciones. (**Usar**)

2) S.O. Diseñar, implementar y evaluar una solución basada en computación para cumplir con un conjunto determinado de requisitos computacionales en el contexto de las disciplinas del programa. (**Usar**)

3) S.O. Comunicarse efectivamente en diversos contextos profesionales. (**Usar**)

6) S.O. Aplicar la teoría de la computación y los fundamentos del desarrollo de software para producir soluciones basadas en computación. (**Evaluar**)

7. Unidades

UNIDAD 1: Herramientas y Entornos	
Resultados del estudiante: 1,2,3,6	
Contenido	**Objetivos Generales**
• Administración de configuración de software y control de versiones. • Administración de despliegues. • Análisis de requerimientos y herramientas para modelado del diseño. • Herramientas de testing, incluyendo herramientas de análisis estático y dinámico. • Entornos de programación que automatizan el proceso de construcción de partes de programa, por ejemplo, construcciones automatizadas. • Integración continua. • Mecanismos y conceptos de herramientas de integración.	• Administración de configuración de software y control de versiones. [Usar] • Administración de despliegues. [Usar] • Análisis de requerimientos y herramientas para modelado del diseño. [Usar] • Herramientas de testing, incluyendo herramientas de análisis estático y dinámico. [Usar] • Entornos de programación que automatizan el proceso de construcción de partes de programa (por ejemplo, construcciones automatizadas). [Usar] • Integración continua. [Usar] • Mecanismos y conceptos de herramientas de integración. [Usar]
Lecturas: Pressman (2004), Blum (1992), Schach (2004), Wang and King (2000), Keyes (2004), Windle and Abreo (2002), Priest and Sanchez (2001), Schach (2004), Montangero (1996), Ambriola (2001), Conradi (2000), Oquendo (2003)	

UNIDAD 2 : Verificación y Validación de Software	
Resultados del estudiante: 2,3,6	
Contenido	**Objetivos Generales**

• Verificación y validación de conceptos.
• Inspecciones, revisiones, auditorías.
• Tipos de pruebas, incluyendo la interfaz humano-computador, usabilidad, confiabilidad, seguridad, desempeño para la especificación.
• Fundamentos de testing:
  • Pruebas de unit, integración, validación y de sistema.
  • Creación de plan de pruebas y generación de casos de test.
  • Técnicas de test de caja negra y caja blanca.
  • Test de regresión y automatización de pruebas.
• Seguimiento de defectos.
• Limitaciones de testing en dominios particulares, tales como sistemas paralelos o críticos en cuanto a seguridad.
• Enfoques estáticos y enfoques dinámicos para la verificación.
• Desarrollo basado en pruebas.
• Plan de validación, documentación para validación.
• Pruebas orientadas a objetos, sistema de pruebas.
• Verificación y validación de artefactos no codificados (documentación, archivos de ayuda, materiales de entrenamiento).
• Logeo fallido, error crítico y apoyo técnico para dichas actividades.
• Estimación fallida y terminación de las pruebas que incluye el envío por defecto.
	• Distinguir entre la validación y verificación del programa. [Usar]
	• Describir el papel que las herramientas pueden desempeñar en la validación de software. [Usar]
	• Realizar, como parte de una actividad de equipo, una inspección de un segmento de código de tamaño medio. [Usar]
	• Describir y distinguir entre diferentes tipos y niveles de pruebas (unitaria, integración, sistemas y aceptación). [Usar]
	• Describir técnicas para identificar casos de prueba representativos para integración, regresión y pruebas del sistema. [Usar]
	• Crear y documentar un conjunto de pruebas para un segmento de código de mediano tamaño. [Usar]
	• Describir cómo seleccionar buenas pruebas de regresión y automatizarlas. [Usar]
	• Utilizar una herramienta de seguimiento de defectos para manejar defectos de software en un pequeño proyecto de software. [Usar]
	• Discutir las limitaciones de las pruebas en un dominio particular. [Usar]
	• Evaluar un banco de pruebas (a test suite) para un segmento de código de tamaño medio. [Usar]
	• Comparar los enfoques estáticos y dinámicos para la verificación. [Usar]
	• Identificar los principios fundamentales de los métodos de desarrollo basado en pruebas y explicar el papel de las pruebas automatizadas en estos métodos. [Usar]
	• Discutir los temas relacionados con las pruebas de software orientado a objetos. [Usar]
	• Describir las técnicas para la verificación y validación de los artefactos de no código. [Usar]
	• Describir los enfoques para la estimación de fallos. [Usar]

	• Estimar el número de fallos en una pequeña aplicación de software basada en la densidad de defectos y siembra de errores. [Usar] • Realizar una inspección o revisión del código fuente de un software para un proyecto de software de tamaño pequeño o mediano. [Usar]
Lecturas: Pressman (2004), Blum (1992), Schach (2004), Wang and King (2000), Keyes (2004), Windle and Abreo (2002), Priest and Sanchez (2001), Schach (2004), Montangero (1996), Ambriola (2001), Conradi (2000), Oquendo (2003)	

UNIDAD 3: Evolución de Software	
Resultados del estudiante: 1	
Contenido	**Objetivos Generales**
• Desarrollo de Software en el contexto de código grande pre existente: - Cambios de software. - Preocupaciones y ubicación de preocupaciones. - Refactoring. • Evolución de Software. • Características de Software mantenible. • Sistemas de Reingeniería. • Reuso de Software: - Segmentos de código. - Bibliotecas y frameworks. - Componentes. - Líneas de Producto.	• Identificar los problemas principales asociados con la evolución del software y explicar su impacto en el ciclo de vida del software [Usar] • Estimar el impacto del cambio de requerimientos en productos existentes de tamaño medio [Usar] • Usar refactorización en el proceso de modificación de un componente de software [Usar] • Estudiar los desafíos de mejorar sistemas en un entorno cambiante [Usar] • Perfilár los procesos de pruebas de regresión y su rol en el manejo de versiones [Usar] • Estudiar las ventajas y desventajas de diferentes tipos de niveles de confiabilidad [Usar]
Lecturas: Pressman (2004), Blum (1992), Schach (2004), Wang and King (2000), Keyes (2004), Windle and Abreo (2002), Priest and Sanchez (2001), Schach (2004), Montangero (1996), Ambriola (2001), Conradi (2000),Oquendo (2003)	

UNIDAD 4: Gestión de Proyectos de Software	
Resultados del estudiante: 2,3,6	
Contenido	**Objetivos Generales**
• La participación del equipo: - Procesos elementales 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, percepción, estructura). • Estimación de esfuerzo (a nivel personal). • Riesgo: - El papel del riesgo en el ciclo de vida. - Categorías elementales de riesgo, incluyendo seguridad, fiabilidad, mercado, finanzas, tecnología, personas, calidad, estructura y proceso. • Gestión de equipos: - Organización de equipo y toma de decisiones. - Identificación y asignación de roles. - Evaluación del desempeño individual y del equipo. • 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: - Identificación y gestión de riesgos. - Análisis de riesgo y evaluación. - La tolerancia al riesgo (por ejemplo, riesgo adverso, riesgo neutral, búsqueda de riesgo). - Planificación de riesgo.	• 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]

• Enfoque integral al riesgo en todo el sistema, incluyendo riesgos asociados con herramientas.	
Lecturas: Pressman (2004), Blum (1992), Schach (2004), Wang and King (2000), Keyes (2004), Windle and Abreo (2002), Priest and Sanchez (2001), Schach (2004), Montangero (1996), Ambriola (2001), Conradi (2000), Oquendo (2003)	

8. Metodología

1. El profesor del curso presentará clases teóricas de los temas señalados en el programa propiciando la intervención de los alumnos.

2. El profesor del curso presentará demostraciones para fundamentar clases teóricas.

3. El profesor y los alumnos realizarán prácticas.

4. 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.

9. Evaluar

- **Evaluación Permanente 1:** 30%
- **Examen parcial:** 20%
    - **Trabajo Parcial** : 50 %
    - **Examen Parcial** : 50 %
- **Evaluación Permanente 2:** 30%
- **Examen final:** 20%
    - **Trabajo Final** : 50 %
    - **Examen Final** : 50 %

Referencias

- *Ambriola, Vincenzo (July 2001). Software Process Technology. Springer.*
- *Blum, Bruce I. (May 1992). Software Engineering: A Holistic View. 7th. Oxford University Press US.*
  Conradi, R (Mar. 2000). Software Process Technology. Springer.
- *Keyes, Jessica (Feb. 2004). Software Configuration Management. CRC Press.*
  Montangero, Carlo (Sept. 1996). Software Process Technology. Springer.
- *Oquendo, Flavio (Sept. 2003). Software Process Technology. Springer.*
- *Pressman, Roger S. (Mar. 2004). Software Engineering: A Practitioner's Approach. 6th. McGraw-Hill.*
- *Priest, John W. and Jose M. Sanchez (Jan. 2001). Product Development and Design for Manufacturing. Marcel Dekker.*
  Schach, Stephen R (Jan. 2004). Object-Oriented and Classical Software Engineering. McGraw-Hill.
- *Wang, Yingxu and Graham King (Apr. 2000). Software Engineering Processes: Principles and Applications. CRC Press.*
  Windle, Daniel R. and L. Rene Abreo (Aug. 2002). Software Requirements Using the Unified Process. Prentice Hall.