Universidad Nacional de Ingeniería (UNI)
Programa Profesional de
Inteligencia Artificial
Sílabo 2024-I

1. CURSO

CS292. Ingeniería de Software II (Obligatorio)

2. INFORMACIÓN GENERAL

2.1 Curso	:	CS292. Ingeniería de Software II
2.2 Semestre	:	6^{to} Semestre.
2.3 Créditos	:	4
2.4 horas	:	2 HT; 4 HP;
2.5 Duración del periodo	:	16 semanas
2.6 Condición	:	Obligatorio
2.7 Modalidad de aprendizaje	:	Presencial
2.8 Prerrequisitos	:	CS291. Ingeniería de Software I. (5^{to} Sem)

3. PROFESORES

Atención previa coordinación con el profesor

4. INTRODUCCIÓN AL CURSO

Los tópicos de este curso extienden las ideas del diseño y desarrollo de software desde la secuencia de 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.

5. OBJETIVOS

- 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 sea capaz de crear, mejorar y utilizar herramientas y métricas que le 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 , sentar las bases para crear, 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 carácteristicas 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. RESULTADOS DEL ESTUDIANTE

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

2) 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) Comunicarse efectivamente en diversos contextos profesionales. (**Usar**)

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

7. TEMAS

Unidad 1: Herramientas y Entornos (12 horas)	
Resultados esperados: 2,3,6	
Temas	**Objetivos de Aprendizaje**

• 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 (ejem., 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 (ejem., construcciones automatizadas)
	  – Integración continua.
	[Usar]
	• Mecanismos y conceptos de herramientas de integración. [Usar]

Lecturas : [Pre04], [Blu92], [Sch04], [WK00], [Key04], [WA02], [PS01], [Sch04], [Mon96], [Amb01], [Con00], [Oqu03]	

Unidad 2: Verificación y Validación de Software (12 horas)

Resultados esperados: 2,3,6

Temas	**Objetivos de Aprendizaje**
• Verificación y validación de conceptos. • Inspecciones, revisiones, auditorias. • Tipos de pruebas, incluyendo la interfas humano computador, usabildiad, confiabilidad , seguridad,desempeño para la especificación. • Fundamentos de testeo: – 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 testeo 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 la envios 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, integracion, sistemas y aceptacion) [Usar] • Describir tecnicas para identificar casos de prueba representativos para integracion, regresion 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 de código fuente de un software para un proyecto de software de tamaño pequeño o mediano [Usar]

Lecturas : [Pre04], [Blu92], [Sch04], [WK00], [Key04], [WA02], [PS01], [Sch04], [Mon96], [Amb01], [Con00], [Oqu03]

Unidad 3: Evolución de Software (12 horas)	
Resultados esperados: 2,3,6	
Temas	**Objetivos de Aprendizaje**
• 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 sosftware [Usar] • Estudiar los desafios de mejorar sistemas en un entorno cambiante [Usar] • Perfilar 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 : [Pre04], [Blu92], [Sch04], [WK00], [Key04], [WA02], [PS01], [Sch04], [Mon96], [Amb01], [Con00], [Oqu03]	

Unidad 4: Gestión de Proyectos de Software (12 horas)	
Resultados esperados: 2,3,6	
Temas	**Objetivos de Aprendizaje**
• 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. – Identificación de riesgos y gestión. – Análisis riesgo y evaluación. – La tolerancia al riesgo (por ejemplo, riesgo adverso, riesgo neutral, la búsqueda de riesgo) – Planificación de Riesgo • 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 contrucción de equipos y gestión de equipos [Usar]
Lecturas : [Pre04], [Blu92], [Sch04], [WK00], [Key04], [WA02], [PS01], [Sch04], [Mon96], [Amb01], [Con00], [Oqu03]	

8. PLAN DE TRABAJO

8.1 Metodología

Se fomenta la participación individual y en equipo para exponer sus ideas, motivándolos con puntos adicionales en las diferentes etapas de la evaluación del curso.

8.2 Sesiones Teóricas

Las sesiones de teoría se llevan a cabo en clases magistrales donde se realizarán actividades que propicien un aprendizaje activo, con dinámicas que permitan a los estudiantes interiorizar los conceptos.

8.3 Sesiones Prácticas

Las sesiones prácticas se llevan en clase donde se desarrollan una serie de ejercicios y/o conceptos prácticos mediante planteamiento de problemas, la resolución de problemas, ejercicios puntuales y/o en contextos aplicativos.

9. SISTEMA DE EVALUACIÓN

********* EVALUATION MISSING ********

10. BIBLIOGRAFÍA BÁSICA

[Amb01] Vincenzo Ambriola. *Software Process Technology*. Springer, July 2001.

[Blu92] Bruce I. Blum. *Software Engineering: A Holistic View*. 7th. Oxford University Press US, May 1992.

[Con00] R Conradi. *Software Process Technology*. Springer, Mar. 2000.

[Key04] Jessica Keyes. *Software Configuration Management*. CRC Press, Feb. 2004.

[Mon96] Carlo Montangero. *Software Process Technology*. Springer, Sept. 1996.

[Oqu03] Flavio Oquendo. *Software Process Technology*. Springer, Sept. 2003.

[Pre04] Roger S. Pressman. *Software Engineering: A Practitioner's Approach*. 6th. McGraw-Hill, Mar. 2004.

[PS01] John W. Priest and Jose M. Sanchez. *Product Development and Design for Manufacturing*. Marcel Dekker, Jan. 2001.

[Sch04] Stephen R Schach. *Object-Oriented and Classical Software Engineering*. McGraw-Hill, Jan. 2004.

[WA02] Daniel R. Windle and L. Rene Abreo. *Software Requirements Using the Unified Process*. Prentice Hall, Aug. 2002.

[WK00] Yingxu Wang and Graham King. *Software Engineering Processes: Principles and Applications*. CRC Press, Apr. 2000.