



National University of Engineering (UNI)

School of Computer Science
Syllabus 2024-II

1. COURSE

CS292. Software Engineering II (Mandatory)

2. GENERAL INFORMATION

2.1 Course	: CS292. Software Engineering II
2.2 Semester	: 6 th 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	: Mandatory
2.7 Learning modality	: Face to face
2.8 Prerequisites	: CS291. Software Engineering I. (5 th Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

The topics of this course extend the ideas of software design and development from the introduction sequence to programming to encompass the problems encountered in large-scale projects. It is a broader and more complete view of Software Engineering appreciated from a Project point of view.

5. GOALS

- Enable students to be part of and define software development teams facing real-world problems.
- familiarize the students with the process of administering a software project in such a way as to be able to create, improve and use tools and metrics that allow them to carry out the estimation and monitoring of a software project
- Create, evaluate and execute a test plan for medium-sized code segments, Distinguish between different types of tests, lay the foundation for creating, improve test procedures and tools for these purposes
- Select with justification an appropriate set of tools to support the development of a range of software products.
- Create, improve and use existing patterns for software maintenance. Disclose features and design patterns for software reuse.
- Identify and discuss different specialized systems, create, improve and use specialized standards for the design, implementation, maintenance and testing of specialized systems.

6. COMPETENCES

- 1) Analyze a complex computing problem and apply principles of computing and other relevant disciplines to identify solutions. (Usage)
- 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. (Usage)
- 3) Communicate effectively in a variety of professional contexts.. (Usage)
- 6) Apply computer science theory and software development fundamentals to produce computing-based solutions. (Assessment)

7. TOPICS

Unit 1: Herramientas y Entornos (12 hours)	
Competences Expected:	
Topics	Learning Outcomes
• Administración de configuración de software y control de versiones. • Administración de despliegues. • Análisis de requerimientos y herramientas para modelado del diseño. • Herramientas de <i>testing</i> 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 <i>testing</i> 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]
Readings : [Pre04], [Blu92], [Sch04], [WK00], [Key04], [WA02], [PS01], [Sch04], [Mon96], [Amb01], [Con00], [Oqu03]	

Unit 2: Verificación y Validación de Software (12 hours)	
Competences Expected:	
Topics	Learning Outcomes
• Verificación y validación de conceptos. • Inspecciones, revisiones, auditorias. • Tipos de pruebas, incluyendo la interfaz humano computador, usabilidad, 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 (<i>a test suite</i>) 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 las limitaciones de las pruebas en un dominio particular [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]
Readings : [Pre04], [Blu92], [Sch04], [WK00], [Key04], [WA02], [PS01], [Sch04], [Mon96], [Amb01], [Con00], [Oqu03]	

Unit 3: Evolución de Software (12 hours)	
Competences Expected:	
Topics	Learning Outcomes
• Desarrollo de Software en el contexto de código grande pre existente – Cambios de software – Preocupaciones y ubicación de preocupaciones – <i>Refactoring</i> • Evolución de Software. • Características de Software mantenible. • Sistemas de Reingeniería. • Reuso de Software. – Segmentos de código – Bibliotecas y <i>frameworks</i> – 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] • 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]
Readings : [Pre04], [Blu92], [Sch04], [WK00], [Key04], [WA02], [PS01], [Sch04], [Mon96], [Amb01], [Con00], [Oqu03]	

Unit 4: Gestión de Proyectos de Software (12 hours)	
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. – 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 construcción de equipos y gestión de equipos [Usar]
Readings : [Pre04], [Blu92], [Sch04], [WK00], [Key04], [WA02], [PS01], [Sch04], [Mon96], [Amb01], [Con00], [Oqu03]	

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

- [Blu92] Bruce I. Blum. *Software Engineering: A Holistic View*. 7th. Oxford University Press US, May 1992.
- [Mon96] Carlo Montangero. *Software Process Technology*. Springer, Sept. 1996.
- [Con00] R Conradi. *Software Process Technology*. Springer, Mar. 2000.
- [WK00] Yingxu Wang and Graham King. *Software Engineering Processes: Principles and Applications*. CRC Press, Apr. 2000.
- [Amb01] Vincenzo Ambriola. *Software Process Technology*. Springer, July 2001.
- [PS01] John W. Priest and Jose M. Sanchez. *Product Development and Design for Manufacturing*. Marcel Dekker, Jan. 2001.
- [WA02] Daniel R. Windle and L. Rene Abreo. *Software Requirements Using the Unified Process*. Prentice Hall, Aug. 2002.
- [Oqu03] Flavio Oquendo. *Software Process Technology*. Springer, Sept. 2003.
- [Key04] Jessica Keyes. *Software Configuration Management*. CRC Press, Feb. 2004.
- [Pre04] Roger S. Pressman. *Software Engineering: A Practitioner's Approach*. 6th. McGraw-Hill, Mar. 2004.
- [Sch04] Stephen R Schach. *Object-Oriented and Classical Software Engineering*. McGraw-Hill, Jan. 2004.