



Universidad de Ingeniería y Tecnología
Escuela Profesional de
Ciencia de la Computación
Silabo del curso – Periodo Académico 2017-II

1. **Código del curso y nombre:** CS292. Ingeniería de Software II
2. **Créditos:** 4
3. **Horas de Teoría y Laboratorio:** 2 HT; 4 HP;
4. **Docente(s)**

Atención previa coordinación con el profesor

5. Bibliografía

- [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.

6. Información del curso

- (a) **Breve descripción del 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.
- (b) **Prerrequisitos:** CS291. Ingeniería de Software I. (5^{to} Sem)
- (c) **Tipo de Curso:** Electivo

7. Competencias

- 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 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

8. Contribución a los resultados (*Outcomes*)

- b) Analizar problemas e identificar y definir los requerimientos computacionales apropiados para su solución. (**Usar**)
- c) Diseñar, implementar y evaluar un sistema, proceso, componente o programa computacional para alcanzar las necesidades deseadas. (**Usar**)
- f) Comunicarse efectivamente con audiencias diversas. (**Usar**)
- i) Utilizar técnicas y herramientas actuales necesarias para la práctica de la computación. (**Evaluar**)
- k) Aplicar los principios de desarrollo y diseño en la construcción de sistemas de software de complejidad variable. (**Usar**)

9. Competencias (IEEE)

- C7.** Ser capaz de aplicar los principios y tecnologías de ingeniería de software para asegurar que las implementaciones de software son robustos, fiables y apropiados para su público objetivo.⇒ **Outcome b,k**
- C8.** Entendimiento de lo que las tecnologías actuales pueden y no pueden lograr.⇒ **Outcome b,c,k**
- C11.** Entendimiento del concepto del ciclo de vida, incluyendo la importancia de sus fases (planificación, desarrollo, implementación y evolución).⇒ **Outcome c**
- C12.** Entender las implicaciones de ciclo de vida para el desarrollo de todos los aspectos de los sistemas informáticos (incluyendo software, hardware, y la interfaz de la computadora humana).⇒ **Outcome c,i**
- C18.** Capacidad para participar de forma activa y coordinada en un equipo.⇒ **Outcome k**
- CS1.** Modelar y diseñar sistemas de computadora de una manera que se demuestre comprensión del balance entre las opciones de diseño.⇒ **Outcome c**
- CS2.** Identificar y analizar los criterios y especificaciones apropiadas a los problemas específicos, y planificar estrategias para su solución.⇒ **Outcome b,c**
- CS4.** Implementar la teoría apropiada, prácticas y herramientas para la especificación, diseño, implementación y mantenimiento, así como la evaluación de los sistemas basados en computadoras.⇒ **Outcome b,c,i**
- CS5.** Especificar, diseñar e implementar sistemas basados en computadoras.⇒ **Outcome b,c,i**
- CS10.** Implementar efectivamente las herramientas que se utilizan para la construcción y la documentación de software, con especial énfasis en la comprensión de todo el proceso involucrado en el uso de computadoras para resolver problemas prácticos. Esto debe incluir herramientas para el control de software, incluyendo el control de versiones y gestión de la configuración.⇒ **Outcome i,k**

10. Lista de temas a estudiar en el curso

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

11. Metodología y Evaluación

Metodología:

Sesiones Teóricas:

El desarrollo de las sesiones teóricas está focalizado en el estudiante, a través de su participación activa, resolviendo problemas relacionados al curso con los aportes individuales y discutiendo casos reales de la industria. Los alumnos desarrollarán a lo largo del curso un proyecto de aplicación de las herramientas recibidas en una empresa.

Sesiones de Laboratorio:

Las sesiones prácticas se desarrollan en laboratorio. Las prácticas de laboratorio se realizan en equipos para fortalecer su comunicación. Al inicio de cada laboratorio se explica el desarrollo de la práctica y al término se destaca las principales conclusiones de la actividad en forma grupal.

Exposiciones individuales o grupales:

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.

Lecturas:

A lo largo del curso se proporcionan diferentes lecturas, las cuales son evaluadas. El promedio de las notas de las lecturas es considerado como la nota de una práctica calificada. El uso del campus virtual UTEC Online permite a cada estudiante acceder a la información del curso, e interactuar fuera de aula con el profesor y con los otros estudiantes.

Sistema de Evaluación:

12. Contenido

Unidad 1: Herramientas y Entornos (12)	
Competencias esperadas: C1	
Objetivos de Aprendizaje	Tópicos
<ul style="list-style-type: none">• 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)<ul style="list-style-type: none">– Integración continua.[Usar]• Mecanismos y conceptos de herramientas de integración. [Usar]	<ul style="list-style-type: none">• 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)<ul style="list-style-type: none">– Integración continua.• Mecanismos y conceptos de herramientas de integración.
Lecturas : [Pre04], [Blu92], [Sch04], [WK00], [Key04], [WA02], [PS01], [Sch04], [Mon96], [Amb01], [Con00], [Oqu03]	

Unidad 2: Verificación y Validación de Software (12)	
Competences esperadas: C20	
Objetivos de Aprendizaje	Tópicos
<ul style="list-style-type: none"> • 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 (<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 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] 	<ul style="list-style-type: none"> • 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 testeo: <ul style="list-style-type: none"> – 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 envíos por defecto.
Lecturas : [Pre04], [Blu92], [Sch04], [WK00], [Key04], [WA02], [PS01], [Sch04], [Mon96], [Amb01], [Con00], [Oqu03]	

Unidad 3: Evolución de Software (12)	
Competences esperadas: C20	
Objetivos de Aprendizaje	Tópicos
<ul style="list-style-type: none"> • 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] 	<ul style="list-style-type: none"> • Desarrollo de Software en el contexto de código grande pre existente <ul style="list-style-type: none"> – Cambios de software – Preocupaciones y ubicación de preocupaciones – <i>Refactoring</i> • Evolución de Software. • Características de Software mantenible. • Sistemas de Reingeniería. • Reuso de Software. <ul style="list-style-type: none"> – Segmentos de código – Bibliotecas y <i>frameworks</i> – Componentes – Líneas de Producto
Lecturas : [Pre04], [Blu92], [Sch04], [WK00], [Key04], [WA02], [PS01], [Sch04], [Mon96], [Amb01], [Con00], [Oqu03]	

Unidad 4: Gestión de Proyectos de Software (12)	
Competences esperadas: C24	
Objetivos de Aprendizaje	Tópicos
<ul style="list-style-type: none"> • 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] 	<ul style="list-style-type: none"> • La participación del equipo: <ul style="list-style-type: none"> – 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. <ul style="list-style-type: none"> – 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: <ul style="list-style-type: none"> – Organización de equipo y la toma de decisiones – Roles de identificación y asignación – Individual y el desempeño del equipo de evaluación • Gestión de proyectos: <ul style="list-style-type: none"> – 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. <ul style="list-style-type: none"> – 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.
Lecturas : [Pre04], [Blu92], [Sch04], [WK00], [Key04], [WA02], [PS01], [Sch04], [Mon96], [Amb01], [Con00], [Oqu03]	



Universidad de Ingeniería y Tecnología
Escuela Profesional de
Ciencia de la Computación
Silabo del curso – Periodo Académico 2017-II

1. **Código del curso y nombre:** CS251. Computación Gráfica
2. **Créditos:** 4
3. **Horas de Teoría y Laboratorio:** 2 HT; 4 HP;
4. **Docente(s)**

Atención previa coordinación con el profesor

5. Bibliografía

- [HB90] Donald Hearn and Pauline Baker. *Computer Graphics in C*. Prentice Hall, 1990.
- [Hug+13] John F. Hughes et al. *Computer Graphics - Principles and Practice 3rd Edition*. Addison-Wesley, 2013.
- [Shr+13] Dave Shreiner et al. *OpenGL, Programming Guide, Eighth Edition*. Addison-Wesley, 2013.
- [Wol11] David Wolff. *OpenGL 4.0 Shading Language Cookbook*. Packt Publishing, 2011.

6. Información del curso

- (a) **Breve descripción del curso** Ofrece una introducción para el área de Computación Gráfica, la cual es una parte importante dentro de Ciencias de la Computación. El proposito de este curso es investigar los principios, técnicas y herramientas fundamentales para esta área.
- (b) **Prerrequisitos:** CS312. Estructuras de Datos Avanzadas. (6^{to} Sem)
- (c) **Tipo de Curso:** Electivo

7. Competencias

- Acercar al alumno a conceptos y técnicas usados en aplicaciones gráficas 3-D complejas.
- Dar al alumno las herramientas necesarias para determinar que software gráfico y que plataforma son los más adecuados para desarrollar una aplicación específica.

8. Contribución a los resultados (Outcomes)

- a) Aplicar conocimientos de computación y de matemáticas apropiadas para la disciplina. (**Usar**)
- b) Analizar problemas e identificar y definir los requerimientos computacionales apropiados para su solución. (**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 Ciencia de la Computación en el modelamiento y diseño de sistemas computacionales de tal manera que demuestre comprensión de los puntos de equilibrio involucrados en la opción escogida. (**Usar**)

9. Competencias (IEEE)

- C1.** La comprensión intelectual y la capacidad de aplicar las bases matemáticas y la teoría de la informática (*Computer Science*).⇒ **Outcome a**
- C2.** Capacidad para tener una perspectiva crítica y creativa para identificar y resolver problemas utilizando el pensamiento computacional.⇒ **Outcome b**
- C5.** Capacidad para implementar algoritmos y estructuras de datos en el software.⇒ **Outcome b**