

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



CS391. Ingeniería de Software III (Obligatorio)

1. Información general

1.1 Escuela	:	Ciencia de la Computación
1.2 Curso	:	CS391. Ingeniería de Software III
1.3 Semestre	:	7 ^{mo} Semestre.
1.4 Prerrequisitos	:	CS292. Ingeniería de Software II. (6 ^{to} Sem)
1.5 Condición	:	Obligatorio
1.6 Modalidad de aprendizaje	:	Presencial
1.7 horas	:	2 HT; 2 HP;
1.8 Créditos	:	3
1.9 Plan	:	Plan Curricular 2016

2. Profesores

Titular

- Gustavo Delgado Ugarte <ggdelgado@ucsp.edu.pe>
– Master en Ingeniería del Software, Escuela Universitaria de Ingeniería Industrial, Informática y Sistemas - UTA, Chile, 2009.

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, estos elementos son pieza clave y transversal durante todo el proceso productivo.

La construcción de software contempla la implementación y uso de procesos, métodos, modelos y herramientas que permitan lograr la realización de los atributos de calidad de un producto.

4. Resumen

1. Evolución de Software 2. Gestión de Proyectos de Software 3. Gestión de Proyectos de Software 4. Procesos de Software 5. Estándares ISO/IEC

5. Objetivos Generales

- Comprender y poner en práctica los conceptos fundamentales sobre la gestión de proyectos y manejo de equipos de software.
- Comprender los fundamentos de la gestión de proyectos, incluyendo su definición, alcance, y la necesidad de gestión de proyectos en la organización moderna.
- Los alumnos deben comprender los conceptos fundamentales de CMMI, PSP, TSP para que sean adoptados en los proyectos de software.
- Describir y comprender los modelos de aseguramiento de la calidad como marco clave para el éxito de los proyectos de TI.

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. (**Evaluar**)
- 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. (**Evaluar**)
- 3) S.O. Comunicarse efectivamente en diversos contextos profesionales. (**Usar**)
- 5) S.O. Funcionar efectivamente como miembro o líder de un equipo involucrado en actividades apropiadas a la disciplina del programa. (**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) S.O. Desarrollar tecnología computacional buscando el bien común, aportando con formación humana, capacidades científicas, tecnológicas y profesionales para solucionar problemas sociales de nuestro entorno. (**Evaluar**)

7. Contenido

UNIDAD 1: Evolución de Software (12)

Competencias:

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 [Familiarizarse]
- 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 [Familiarizarse]
- Perfilar los procesos de pruebas de regresión y su rol en el manejo de versiones [Familiarizarse]
- Estudiar las ventajas y desventajas de diferentes tipos de niveles de confiabilidad [Familiarizarse]

Lecturas: Pressman and Maxim (2015), Sommerville (2017)

UNIDAD 2: Gestión de Proyectos de Software (10)	
Competencias:	
Contenido	Objetivos Generales
<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 	<ul style="list-style-type: none"> • Discutir los comportamientos comunes que contribuyen al buen funcionamiento de un equipo [Familiarizarse] • 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 [Familiarizarse] • Describir el impacto del riesgo en el ciclo de vida de desarrollo de software [Familiarizarse] • Describir las diferentes categorías de riesgo en los sistemas de software [Familiarizarse] • 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 [Familiarizarse] • 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) [Familiarizarse]
Lecturas: Pressman and Maxim (2015), Sommerville (2017)	

UNIDAD 3: Gestión de Proyectos de Software (8)	
Competencias:	
Contenido	Objetivos Generales
<ul style="list-style-type: none"> • 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. 	<ul style="list-style-type: none"> • 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 [Evaluar] • Identificar riesgos y describir enfoques para manejar riesgos (evitar, aceptar, transferir, mitigar) y caracterizar fortalezas y defectos para cada uno [Familiarizarse] • Explicar cómo el riesgo afecta las decisiones en el proceso de desarrollo de software [Usar] • Identificar los riesgos de seguridad para un sistema de software [Usar] • Demostrar un enfoque sistemático para la tarea de identificar los peligros y riesgos en una situación particular [Usar] • Aplicar los principios básicos del manejo de riesgos en una variedad de escenarios simples incluyendo una situación de seguridad [Usar] • Dirigir un análisis de costo/beneficio para el enfoque de mitigación de riesgos [Usar] • Identificar y analizar alguno de los riesgos para un sistema entero que surgen de aspectos distintos del software [Usar]
Lecturas: Pressman and Maxim (2015), Sommerville (2017)	

UNIDAD 4: Procesos de Software (12)	
Competencias:	
Contenido	Objetivos Generales
<ul style="list-style-type: none"> • Consideraciones a nivel de sistemas, ejem., la interacción del software con su entorno. • Introducción a modelos del proceso de software (e.g., cascada, incremental, ágil): <ul style="list-style-type: none"> – Actividades con ciclos de vida de software. • Programación a gran escala versus programación individual. • Evaluación de modelos de proceso de software. • Conceptos de calidad de software. • Mejoramiento de procesos. • Modelos de madurez de procesos de software. • Mediciones del proceso de software. 	<ul style="list-style-type: none"> • Describa cómo el software puede interactuar y participar en varios sistemas, incluyendo la gestión de información, integración, control de procesos y sistemas de comunicaciones [Usar] • Describir las ventajas y desventajas relativas entre varios modelos importantes de procesos (por ejemplo, la cascada, iterativo y ágil) [Usar] • Describir las diferentes prácticas que son componentes clave de los diversos modelos de procesos [Usar] • Diferenciar entre las fases de desarrollo de software [Usar] • Describir cómo la programación en grandes equipos difiere de esfuerzos individuales con respecto a la comprensión de una gran base de código, lectura de código, comprensión de las construcciones, y comprensión de contexto de cambios [Usar] • Explicar el concepto de ciclo de vida del software y proporcionar un ejemplo que ilustra sus fases incluyendo los entregables que se producen [Usar] • Comparar varios modelos comunes de procesos con respecto a su valor para el desarrollo de las clases particulares de sistemas de software, teniendo en cuenta diferentes aspectos tales como, estabilidad de los requisitos, tamaño y características no funcionales [Usar] • Definir la calidad del software y describir el papel de las actividades de aseguramiento de la calidad en el proceso de software [Usar] • Describir el objetivo y similitudes fundamentales entre los enfoques de mejora de procesos [Usar] • Comparar varios modelos de mejora de procesos, tales como CMM, CMMI, CQI, <i>Plan-Do-Check-Act</i>, o ISO9000 [Usar] • Evaluar un esfuerzo de desarrollo y recomendar cambios potenciales al participar en la mejora de procesos (usando un modelo como PSP) o involucración en una retrospectiva de un proyecto [Usar] • Explicar el papel de los modelos de madurez de procesos en la mejora de procesos [Usar] • Describir varias métricas de procesos para la evaluación y el control de un proyecto [Usar] • Usar las medidas en proyecto para describir el estado actual de un proyecto [Usar]
Lecturas: Pressman and Maxim (2015), Sommerville (2017)	

UNIDAD 5: Estándares ISO/IEC (6)	
Competencias:	
Contenido	Objetivos Generales
<ul style="list-style-type: none"> • ISO 9001:2001. • ISO 9000-3. • ISO/IEC 9126. • ISO/IEC 12207. • ISO/IEC 15939. • ISO/IEC 14598. • ISO/IEC 15504-SPICE. • IT Mark. • SCRUM. • SQuaRE. • CISQ. 	<ul style="list-style-type: none"> • Aprender y aplicar correctamente normas y estándares internacionales. [Usar]
Lecturas: Sommerville (2017), Pressman and Maxim (2015)	

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 Continua 1 : 20 %

Examen parcial : 30 %

Evaluación Continua 2 : 20 %

Examen final : 30 %

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.