Universidad Católica San Pablo
Facultad de Ingeniería y Computación
Escuela Profesional de
Ciencia de la Computación
SILABO

CS290T. Ingeniería de Software I (Obligatorio)

2017-I

1. DATOS GENERALES

1.1 CARRERA PROFESIONAL	:	Ciencia de la Computación
1.2 ASIGNATURA	:	CS290T. Ingeniería de Software I
1.3 SEMESTRE ACADÉMICO	:	5^{to} Semestre.
1.4 PREREQUISITO(S)	:	CS102O. Objetos y Abstracción de Datos. (3^{er} Sem) , CS270T. Bases de Datos I. (4^{to} Sem) , CS130. Introducción a Internet. (3^{er} Sem)
1.5 CARÁCTER	:	Obligatorio
1.6 HORAS	:	2 HT; 2 HP; 2 HL;
1.7 CRÉDITOS	:	4

2. DOCENTE

Dr. Guillermo Enrique Calderón Ruiz

- Dr. Ciencias de la Ingeniería, Pontificia Universidad Católica de Chile, Chile, 2011.
- Mag. Ingeniería de Sistemas, Universidad Católica Santa María, Perú, 2009.
- Prof. Ingeniero de Sistemas, Universidad Católica Santa María, Perú, 1998.

Bach Jorge Homero Neyra Araoz

- Bach Bachiller de Ingeniería de Sistemas, Universidad Nacional San Agustín, Perú, 2006.

3. FUNDAMENTACIÓN DEL CURSO

La taréa de desarrollar software, excepto para aplicaciones sumamente simples, exige la ejecución de un proceso de desarrollo bien definido. Los profesionales de esta área requieren un alto grado de conocimiento de los diferentes modelos e proceso de desarrollo, para que sean capaces de elegir el más idóneo para cada proyecto de desarrollo. Por otro lado, el desarrollo de sistemas de mediana y gran escala requiere del uso de bibliotecas de patrones y componentes y del dominio de técnicas relacionadas al diseño basado en componentes.

4. SUMILLA

1. SE/Diseño de Software.2. SE/Usando APIs.3. SE/Herramientas y Entornos de Software.4. SE/Validación y verificación de software.5. SE/Computación Basada en Componentes.6. SE/Desarrollo de Sistemas Especializados.7. SE/Mejorando la programación: robustez y seguridad.

5. OBJETIVO GENERAL

- Brindar al alumno un marco teórico y práctico para el desarrollo de software bajo estándares de calidad.
- Familiarizar al alumno con los procesos de modelamiento y construcción de software a través del uso de herramientas CASE.
- Los alumnos debe ser capaces de seleccionar Arquitecturas y Plataformas tecnológicas ad-hoc a los escenarios de implementación.
- Aplicar el modelamiento basado en componentes y fin de asegurar variables como calidad, costo y *time-to-market* en los procesos de desarrollo.
- Brindar a los alumnos mejores prácticas para la verificación y validación del software.

6. CONTRIBUCIÓN A LA FORMACIÓN PROFESIONAL Y FORMACIÓN GENERAL

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

b) Analizar problemas e identificar y definir los requerimientos computacionales apropiados para su solución. [**Nivel Bloom: 4**]

c) Diseñar, implementar y evaluar un sistema, proceso, componente o programa computacional para alcanzar las necesidades deseadas. [**Nivel Bloom: 4**]

d) Trabajar efectivamente en equipos para cumplir con un objetivo común. [**Nivel Bloom: 3**]

f) Comunicarse efectivamente con audiencias diversas. [**Nivel Bloom: 3**]

i) Utilizar técnicas y herramientas actuales necesarias para la práctica de la computación. [**Nivel Bloom: 3**]

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. [**Nivel Bloom: 3**]

k) Aplicar los principios de desarrollo y diseño en la construcción de sistemas de software de complejidad variable. [**Nivel Bloom: 3**]

7. CONTENIDOS

UNIDAD 1: SE/Diseño de Software.(12 horas)

Nivel Bloom: 4

OBJETIVO GENERAL	**CONTENIDO**
▪ Discutir las propiedades del buen diseño de softwarem incluyendo la naturaleza y el rol de la documentación asociada. ▪ Evaluar la calidad de múltiples diseños de software basados en principios y conceptos de diseño claves. ▪ Seleccionar y aplicar patrones de diseño apropiados en la construcción de una aplicación de software. ▪ Crear y especificar el diseño de software para un producto de software de tamaño medio usando una especificación de requerimientos de software, una metodología de diseño de programas aceptado (ejemplo orientado a objetos o estructurado) y una notación de diseño apropiada. ▪ Conducir una revisión de diseño de software con material de código abierto utilizando lineamientos apropiados. ▪ Evaluar un diseño de software a nivel componente. ▪ Evaluar un diseño de software a nivel componente desde la perspectiva de reuso.	▪ Conceptos y principios fundamentales de diseño. ▪ El rol y uso de contratos. ▪ Patrones de diseño. ▪ Arquitectura de software. ▪ Diseño estructurado. ▪ Análisis y diseño orientado a objetos. ▪ Diseño a nivel componente. ▪ Cualidades de diseño. ▪ Aspectos internos tales como bajo acoplamiento. ▪ Aspectos externos como confiabilidad,mantenimiento, usabilidad, desempeño. ▪ Otros abordajes: centrado en datos, orientado a aspectos, orientado a funciones, orientado a servicios, métodos ágiles. ▪ Diseño reusable. ▪ Uso de material de código abierto.

Lecturas: [Pressman, 2005], [Sommerville, 2008], [Larman, 2008]

UNIDAD 2: SE/Usando APIs.(6 horas)	
Nivel Bloom: 3	
OBJETIVO GENERAL	**CONTENIDO**
▪ Explicar el valor de las interfaces para programación de aplicaciones (APIs) en el desarrollo de software. ▪ Usar navegadores de clases y herramientas relacionadas durante el desarrollo de aplicaciones usando APIs. ▪ Diseñar, implementar, probar y depurar programas que usan paquetes API de larga escala.	▪ Programación usando API. ▪ Diseño de API. ▪ Navegadores de clases (*Class browsers*) y herramientas relacionadas. ▪ Depuración en el entorno API. ▪ Introducción a la computación basada en componentes.
Lecturas: [Pressman, 2005], [Sommerville, 2008]	

UNIDAD 3: SE/Herramientas y Entornos de Software.(8 horas)	
Nivel Bloom: 3	
OBJETIVO GENERAL	**CONTENIDO**
▪ Seleccionar con justificación un apropiado conjunto de herramientas para soportar el desarrollo de un rango de productos de software. ▪ Analizar y evaluar un conjunto de herramientas en una área dada del desarrollo de software (ej: administración, modelamiento o pruebas). ▪ Demostrar la capacidad para usar un rango de herramientas de software en soporte del desarrollo de un producto de software de tamaño medio.	▪ Entornos de programación. ▪ Análisis de requerimientos y herramientas de modelamiento de diseño. ▪ Herramientas de pruebas incluyendo herramientas de análisis estático y dinámico. ▪ Herramientas de administración de configuración. ▪ Manejo de la configuración y herramientas de control de versión. ▪ Mecanismos de integración de herramientas.
Lecturas: [Pressman, 2005], [Sommerville, 2008], [Long, 2007]	

UNIDAD 4: SE/Validación y verificación de software.(8 horas)	
Nivel Bloom: 3	
OBJETIVO GENERAL	**CONTENIDO**
▪ Distinguir entre validación de programas y verificación. ▪ Describir el rol que las herramientas pueden jugar en la validación de software. ▪ Distinguir entre los diferentes tipos y niveles de pruebas (unidad, integración, sistemas y aceptación) para productos de software de tamaño medio y el material relacionado. ▪ Crear, evaluar e implementar un plan de prueba para segmentos de código de tamaño medio. ▪ Encargarse, como parte de una actividad de equipo, de una inspección de un segmento de código de tamaño medio. ▪ Discutir los temas concernientes a la prueba de software orientado a objetos..	▪ Distinción entre verificación y validación. ▪ Abordajes estáticos y dinámicos. ▪ Planeamiento de la validación y documentación para la validación. ▪ Diferentes tipos de tests, interfase humano-computador, usabilidad, confiabilidad, seguridad, conformidad con la especificación. ▪ Fundamentos del *Testing* incluyendo la creación de planes de prueba y la generación de casos de prueba. ▪ Técnicas de prueba de caja blanca y caja negra. ▪ Semilla por defecto. ▪ Unidad, integración, validación y sistemas de prueba. ▪ Prueba orientado a objetos, pruebas de sistema. ▪ Medidas de procesos, diseño, programa. ▪ Verificación y validación de partes que no son componentes (documentación, archivos de ayuda, material de entrenamiento). ▪ Defecto de historial (*fault logging*), defecto de rastreo y soporte técnico para esas actividades. ▪ Test de regresión. ▪ Inspecciones, revisiones, auditorías.
Lecturas: [Pressman, 2005], [Sommerville, 2008], [Larman, 2008]	

UNIDAD 5: SE/Computación Basada en Componentes.(14 horas)	
Nivel Bloom: 3	
OBJETIVO GENERAL	**CONTENIDO**
▪ Explicar y aplicar principios reconocidos para la construcción de componentes de software de alta calidad. ▪ Discutir y seleccionar una arquitectura, para un sistema basado en componentes, disponible para un escenario dado. ▪ Identificar el tipo de manejo de eventos implementado en una o mas APIs dadas. ▪ Explicar el rol de los objetos en sistemas *middleware* y la relación con componentes. ▪ Aplicar métodos orientados a componentes para el diseño de un rango de software incluyendo aquellos requeridos para transacciones concurrentes, servicios de comunicación confiables, servicios incluyendo interacción de bases de datos para consulta remota y administración de bases de datos, comunicación segura y acceso.	▪ Fundamentos. a) La definición y naturaleza de los componentes. b) Componentes e interfaces. c) Interfaces como contratos. d) Los beneficios de los componentes. e) Técnicas básicas f) Diseño de componentes y ensamblaje. g) Relaciones con el modelo cliente-servidor y con patrones. h) Uso de objetos y servicios del ciclo de vida del objeto. i) Uso de objetos *brokers*. j) *Marshalling*. ▪ Aplicaciones (incluyendo el uso de componentes para móviles). ▪ Patrones como son utilizados en análisis y diseño. Contexto de uso incluyendo arquitecturas empresariales. ▪ Arquitectura de sistemas basados en componentes. ▪ Diseño orientado a componentes. ▪ Entornos de aplicación. ▪ Manejo de eventos: detección, notificación y respuesta. ▪ *Middleware*. a) El paradigma orientado a objetos dentro del *middleware*. b) Agente de petición de objeto (*Object request brokers*). c) Monitores del procesamiento de transacciones. d) Sistemas de flujo de información (*workflow*). e) Estado del arte de las herramientas.
Lecturas: [Pressman, 2005], [Sommerville, 2008], [Larman, 2008]	

UNIDAD 6: SE/Desarrollo de Sistemas Especializados.(4 horas)	
Nivel Bloom: 3	
OBJETIVO GENERAL	**CONTENIDO**
▪ Identificar y discutir diferentes sistemas especializados. ▪ Discutir el ciclo de vida y tópicos sobre el proceso de software en el ámbito de sistemas diseñados para un contexto especializado incluyendo sistemas que podrían tener que operar en un modo de operación degradado. ▪ Seleccionar, con la justificación apropiada, métodos que darán como resultado el desarrollo eficiente y efectivo y el mantenimiento de sistemas de software especializado. ▪ Dado un contexto específico y un conjunto de tópicos profesionales relacionados, discutir como, un ingeniero de software envuelto en el desarrollo de sistemas especializados, debe de responder a estos tópicos. ▪ Sintetizar los temas técnicos centrales asociados con la implementación del crecimiento de sistemas especializados..	▪ Sistemas en tiempo real. ▪ Sistemas cliente-servidor. ▪ Sistemas distribuidos. ▪ Sistemas paralelos. ▪ Sistemas basados en web. ▪ Sistemas de alta integridad.
Lecturas: [Pressman, 2005], [Sommerville, 2008], [Larman, 2008]	

UNIDAD 7: SE/Mejorando la programación: robustez y seguridad.(8 horas)	
Nivel Bloom: 3	
OBJETIVO GENERAL	**CONTENIDO**
▪ Reescribir un programa simple para remover vulnerabilidades comunes tales como desborde de *buffers*, desborde de enteros y condiciones de corrida. ▪ Presentar y aplicar los principios de la menor parte de privilegio y escenarios seguros por defecto. ▪ Escribir una librería simple que desarrolle algunas tareas no triviales y no finalice la ejecución de un programa sin observar como este fue ejecutado.	▪ Programación a la defensiva: a) Principios de diseño y codificación seguros. b) Principio de la menor parte de privilegio. c) Principios escenarios seguros por defecto. ▪ Principio de aceptación psicológica: a) Como detectar problemas potenciales en seguridad de programas. b) Desborde de *buffers* de otros tipos. c) Condiciones de corrida (*race conditions*). d) Inicialización inapropiada incluyendo privilegios escogidos. e) Chequeo de la entrada. f) Asumir éxito y correctitud. g) Validación de presupuestos. ▪ ¿Cómo documentar consideraciones de seguridad en el uso de un programa?.
Lecturas: [Pressman, 2005], [Sommerville, 2008], [Larman, 2008]	

8. METODOLOGÍA

Evaluación Permanente 1 : 20 %

Evaluación Parcial : 30 %

Trabajo Parcial : 40 %

Examen Parcial : 60 %

Evaluación Permanente 2 : 20 %

Evaluación Final : 30 %

Trabajo Final : 50 %

Examen Final : 50 %

9. EVALUACIONES

Evaluación Permanente 1 : 20 %

Examen Parcial : 30 %

Evaluación Permanente 2 : 20 %

Examen Final : 30 %

Referencias

[Larman, 2008] Larman, C. (2008). *Applying UML and Patterns*. Prentice Hall.

[Long, 2007] Long, F. (2007). *Software Engineering Environments*. Peter Norton Foundation Series. Springer.

[Pressman, 2005] Pressman, R. S. (2005). *Software Engineering: A Practitioner's Approach*. McGraw-Hill, 6th edition.

[Sommerville, 2008] Sommerville, I. (2008). *Software Engineering*. Addison Wesley, 7th edition. ISBN: 0321210263.