Universidad Nacional de San Agustín
VICE RECTORADO ACADÉMICO
SILABO

CODIGO DEL CURSO: CS290T

1 Datos Generales

FACULTAD : Ingeniería de Producción y Servicios						
DEPARTAMENTO : Ingeniería de Sistemas e Informática			**ESCUELA :** Ciencia de la Computación			
PROFESOR : **TÍTULO :**						
ASIGNATURA : Ingeniería de Software I						
PREREQUISITO:	**CREDITOS:**		**Año**: 2010-1		**Total Horas:** 2 HT;	
CS102O,CS270T,CS130	4		**Sem:** 5^{to} Semestre.		2 HT	2 HP 2 HL
Horario	Lun	Mar	Mie	Jue	Vie	Sáb
Total Semanal						
Aula						

2 Exposición de Motivos

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

2 Objetivo

- 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 h
mientas CASE.
- Los alumnos debe ser capaces de seleccionar Arquitecturas y Plataformas tecnológicas ad-hoc a los escenari
implementación.
- Aplicar el modelamiento basado en componentes y fin de asegurar variables como calidad, costo y *time-to-m*
en los procesos de desarrollo.
- Brindar a los alumnos mejores prácticas para la verificación y validación del software.

3 Contenido Temático 3 SE/Diseño de Software.(12 horas)

Objetivos Específicos	Contenidos
▪ 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.	▪ Concepto les de dis ▪ El rol y u ▪ Patrones ▪ Arquitect ▪ Diseño es ▪ Análisis y tos. ▪ Diseño a ▪ Cualidad ▪ Aspectos acoplamie ▪ Aspectos bilidad,m desempeñ ▪ Otros abo orientado funciones todos ágil ▪ Diseño re ▪ Uso de m [3], [4], [1]

3 SE/Usando APIs.(6 horas)

Objetivos Específicos	Contenidos	Hora
▪ 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. [3], [4]	

	Objetivos Específicos	Contenidos
3 SE/Herramientas y Entornos de Software.(8 horas)	▪ 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 pro ▪ Análisis de requ mientas de mode ▪ Herramientas de do herramientas y dinámico. ▪ Herramientas de configuración. ▪ Manejo de la co mientas de contr ▪ Mecanismos de rramientas. [3], [4], [2]

	Objetivos Específicos	Contenidos
3 SE/Validación y verificación de software.(8 horas)	▪ 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 v dación. ▪ Abordajes estátic ▪ Planeamiento de l cumentación para ▪ Diferentes tipos se humano-compu confiabilidad, seg dad con la especif ▪ Fundamentos del do la creación de y la generación de ▪ Técnicas de prue y caja negra. ▪ Semilla por defect ▪ Unidad, integrac sistemas de prueb ▪ Prueba orientado de sistema. ▪ Medidas de proc grama. ▪ Verificación y val que no son compo tación, archivos d de entrenamiento ▪ Defecto de histor defecto de rastreo para esas activida ▪ Test de regresión. ▪ Inspecciones, revi [3], [4], [1]

	Objetivos Específicos	Contenidos
3 SE/Computación Basada en Componentes.(14 horas)	▪ 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 y naturaleza b) Componen terfaces como neficios de los nicas básicas nentes y ensa con el modelo patrones. h) vicios del cic i) Uso de obj *halling*. ▪ Aplicaciones componentes ▪ Patrones con análisis y dis incluyendo a riales. ▪ Arquitectura componentes. ▪ Diseño orient ▪ Entornos de a ▪ Manejo de ev ficación y resp ▪ *Middleware*. orientado a ob *leware*. b) de objeto (*O* c) Monitores transacciones de informació tado del arte [3], [4], [1]

3 SE/Desarrollo de Sistemas Especializados.(4 horas)

Objetivos Específicos	Contenidos
▪ 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 tier ▪ Sistemas cliente ▪ Sistemas distrib ▪ Sistemas paralel ▪ Sistemas basado ▪ Sistemas de alta [3], [4], [1]

3 SE/Mejorando la programación: robustez y seguridad.(8 horas)

Objetivos Específicos	Conten
▪ 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.	▪ Pr a) fic de c) de ▪ Pr ca po ma otr rri za vil en tu ▪ ¿C de ma [3], [4], [

4 Actividades

- Asignaciones

- Controles de Lectura
- Exposiciones

5 Recursos Materiales

- Apuntes del curso
- Libro(s) de la bibliografía

6 Metodología

- Clase Magistral.
- Taller didáctico.
- Social Constructivismo.
- Prácticas personales y en grupo.

7 Evaluación

La nota final (NF) se obtiene de la siguiente manera:

NE Nota de Exámenes 60 %, esta nota se divide en
- Exámen Parcial 40 %
- Examen Final 60 %

NT Nota de Trabajos e Intervención en clase 40 %

$$NF = 0{,}6 * NE + 0{,}4 * NT$$

Referencias

[1] Craig Larman. *Applying UML and Patterns.* Prentice Hall, 2008.

[2] F.W. Long. *Software Engineering Environments.* Peter Norton Foundation Series. Springer, October 2007.

[3] Roger S. Pressman. *Software Engineering: A Practitioner's Approach.* McGraw-Hill, 6th edition, March 2005.

[4] Ian Sommerville. *Software Engineering.* Addison Wesley, 7th edition, May 2008. ISBN: 0321210263.

Docente del curso