Universidad Nacional de San Agustín
VICE RECTORADO ACADÉMICO
SILABO

CODIGO DEL CURSO: CS393

1 Datos Generales

FACULTAD : Ingeniería de Producción y Servicios						
DEPARTAMENTO : Ingeniería de Sistemas e Informática			**ESCUELA :** Ciencia de la Computación			
PROFESOR : **TÍTULO :**						
ASIGNATURA : Métodos Formales						
PREREQUISITO: CS260	**CREDITOS:** 4		**Año**: 2010-1 **Sem:** 9^{no} Semestre.		**Total Horas:** 2 HT; 2 HT	2 HP 2 HL
Horario	Lun	Mar	Mie	Jue	Vie	Sáb
Total Semanal						
Aula						

2 Exposición de Motivos

Los desarrollo de software, en gran medida, aún es una actividad artesanal lo que implica que
posible entregar el software correcto, en el tiempo y presupuestos planeados. Los métodos formales
y solidez matemática, a todo el proceso de desarrollo de software, en la búsqueda de la produc
calidad.

2 Objetivo

- Crear especificaciones y diseños matemáticamente precisos utilizando lenguajes de especificación formales. A
lizar las propiedades de las especificaciones y diseños formales.
- Aplicar las técnicas formales de verificación a los segmentos de software con complejidad baja. Discutir y ana
los tipos de modelos existentes para Métodos Formales.
- Discutir el papel de la verificación de las técnicas formales en el contexto de la validación y prueba de softw
Aprender a utilizar los diferentes lenguajes de especificación formal para la especificación y validación de requis
Analizar las propiedades de las especificaciones y diseños formales.
- Utilizar herramientas para transformar especificaciones y diseños. Explicar las ventajas y desventajas potenc
de usar lenguajes de especificación formal. Crear y evaluar aserciones (pre y post condiciones e invariantes),
una variedad de situaciones que se extienden de simples a complejas.
- Con un lenguaje de especificación formal común, formular la especificación de un sistema de software simp
demostrar las ventajas de una perspectiva de calidad.

3 Contenido Temático 3 SE/Métodos Formales.(14 horas)

Objetivos Específicos	Contenidos
▪ Aplicar técnicas de verificación formal a segmentos de software con baja complejidad. ▪ Discutir el rol de las técnicas de verificación formal en el contexto de la validación de software y comparar los beneficios con los de *model checking*. ▪ Explicar los beneficios potenciales y los defectos de usar lenguajes de especificación formal. ▪ Crear y evaluar pre y post-aserciones para una variedad de situaciones desde lo simple hasta lo complejo. ▪ Usar un lenguaje de especificación formal común, formular la especificación de un sistema de software y demostrar los beneficios desde una perspectiva de calidad.	▪ Conceptos ▪ Lenguajes ▪ *Model che* ▪ Especifica ejecutable ▪ Pre-aserci ▪ Verificació ▪ Tools en males. [4]

3 Métodos y Fundamentos Matematicos (12 horas)

Objetivos Específicos	Contenidos
▪ Crear especificaciones y diseños matemáticamente precisos utilizando. lenguajes de especificación formales. ▪ Analizar las propiedades de las especificaciones y diseños formales.	▪ Métodos de constr ▪ Fundamentos mate fos y árboles. 2. Au presiones regulares 4. Precisión numé errores. [4]

3 Modelamiento (12 horas)

Objetivos Específicos	Contenidos	Horas
▪ Aplicar las técnicas formales de verificación a los segmentos de software con complejidad baja. ▪ Discutir y analizar los tipos de modelos existentes para Métodos Formales.	▪ Introducción a los modelos matemáticos y lenguajes de especificación. ▪ Tipos de modelos. ▪ Modelamiento de comportamiento. [4]	

3 Especificacion de Requerimientos (12 horas)

Objetivos Específicos	Contenidos
▪ Discutir el papel de la verificación de las técnicas formales en el contexto de la validación y prueba de software. ▪ Aprender a utilizar los diferentes lenguajes de especificación formal para la especificación y validación de requisitos. ▪ Analizar las propiedades de las especificaciones y diseños formales	▪ Documentación y espec requerimientos. 1. Leng pecificación (OCL, Z, et ▪ Validación de requerimi [2]

3 Diseño (12 horas)

Objetivos Específicos	Contenidos	Horas	Fecha
▪ Utilizar herramientas para transformar especificaciones y diseños. ▪ Explicar las ventajas y desventajas potenciales d eusar lenguajes de especificación formal. ▪ Crear y evaluar aserciones (pre y post condiciones e invariantes), para una variedad de situacioines que se extienden de simples a complejas.	▪ Diseño detallado. ▪ Notaciones de diseño y herramientas de soporte. 1. Análisis de diseño formal. ▪ Evaluación de diseño. 1. Técnicas de evaluación. [1]		

3 Evolución (12 horas)

Objetivos Específicos	Contenidos	Horas	Fe
▪ Con un lenguaje de especificación formal común, formular la especificación de un sistema de software simple y demostrar las ventajas de una perspectiva de calidad.	▪ Actividades de evolución. 1. Refabricación. 2. Transformación de programas. [3]		

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] John V. Guttag and James J. Horning. A tutorial on Larch and LCL, a Larch/C interface language. In S. Prehn and W. J. Toetenel, editors, *VDM91: Formal Software Development Methods*, Delft, October 1991. Springer-Verlag Lecture Notes in Computer Science 551.

[2] Michael Hinchey and C Neville Dean. *Teaching and Learning Formal Methods*. Morgan Kaufmann, September 1996.

[3] Jonathan Jacky. *The Way of Z: Practical Programming with Formal Methods*. Cambridge University Press, November 1996.

[4] John W. Baugh Jr. Formal specification of engineering analysis programs. In E. N. Houstis, J. R Rice, and R. Vichnevetsky, editors, *Expert Systems for Numerical Computing*. North-Holland, 1992.

Docente del curso