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

CS393. Métodos Formales (Electivo)

2017-I

1. DATOS GENERALES		
1.1 CARRERA PROFESIONAL	:	Ciencia de la Computación
1.2 ASIGNATURA	:	CS393. Métodos Formales
1.3 SEMESTRE ACADÉMICO	:	9^{no} Semestre.
1.4 PREREQUISITO(S)	:	CS260. Lógica Computacional. (6^{to} Sem)
1.5 CARÁCTER	:	Electivo
1.6 HORAS	:	2 HT; 2 HP; 2 HL;
1.7 CRÉDITOS	:	4

2. DOCENTE

Prof. Jhon Humberto Cano Chávez

- Prof. Ingeniería de Sistemas, Universidad Católica Santa María, Perú, 2000.

3. FUNDAMENTACIÓN DEL CURSO

Los desarrollo de software, en gran medida, aún es una actividad artesanal lo que implica que muchas veces no es posible entregar el software correcto, en el tiempo y presupuestos planeados. Los métodos formales intentan dar rigidez y solidez matemática, a todo el proceso de desarrollo de software, en la búsqueda de la producción de software de calidad.

4. SUMILLA

1. SE/Métodos Formales.2. Métodos y Fundamentos Matematicos 3. Modelamiento 4. Especificacion de Requerimientos 5. Diseño 6. Evolución

5. OBJETIVO GENERAL

- Crear especificaciones y diseños matemáticamente precisos utilizando lenguajes de especificación formales. Analizar las propiedades de las especificaciones y diseños formales.
- 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.
- 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.
- Utilizar herramientas para transformar especificaciones y diseños. Explicar las ventajas y desventajas potenciales de usar lenguajes de especificación formal. Crear y evaluar aserciones (pre y post condiciones e invariantes), para 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 simple y demostrar las ventajas de una perspectiva de calidad.

6. CONTRIBUCIÓN A LA FORMACIÓN PROFESIONAL Y FORMACIÓN GENERAL

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

a) Aplicar conocimientos de computación y de matemáticas apropiadas para la disciplina. [**Nivel Bloom: 4**]

b) Analizar problemas e identificar y definir los requerimientos computacionales apropiados para su solución. [**Nivel Bloom: 3**]

c) Diseñar, implementar y evaluar un sistema, proceso, componente o programa computacional para alcanzar las necesidades deseadas. [**Nivel Bloom: 3**]

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

l) Desarrollar principios investigación en el área de computación con niveles de competividad internacional. [**Nivel Bloom: 3**]

7. CONTENIDOS

UNIDAD 1: SE/Métodos Formales.(14 horas)

Nivel Bloom: 3

OBJETIVO GENERAL	CONTENIDO
▪ 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 de métodos formales. ▪ Lenguajes de especificación formal. ▪ *Model checking*. ▪ Especificaciones ejecutables y no ejecutables. ▪ Pre-aserciones y post-aserciones. ▪ Verificación formal. ▪ Tools en el soporte a métodos formales.

Lecturas: [Jr., 1992]

UNIDAD 2: Métodos y Fundamentos Matematicos (12 horas)

Nivel Bloom: 3

OBJETIVO GENERAL	CONTENIDO
▪ Crear especificaciones y diseños matemáticamente precisos utilizando. lenguajes de especificación formales. ▪ Analizar las propiedades de las especificaciones y diseños formales.	▪ Métodos de construcción formal. ▪ Fundamentos matemáticos. 1. Grafos y árboles. 2. Autómata finito, expresiones regulares. 3. Gramáticas. 4. Precisión numérica, exactitud, y errores.

Lecturas: [Jr., 1992]

UNIDAD 3: Modelamiento (12 horas)	
Nivel Bloom: 3	
OBJETIVO GENERAL	**CONTENIDO**
▪ 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.
Lecturas: [Jr., 1992]	

UNIDAD 4: Especificacion de Requerimientos (12 horas)	
Nivel Bloom: 4	
OBJETIVO GENERAL	**CONTENIDO**
▪ 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 especificación de requerimientos. 1. Lenguajes de especificación (OCL, Z, etc.). ▪ Validación de requerimientos.
Lecturas: [Hinchey and Dean, 1996]	

UNIDAD 5: Diseño (12 horas)	
Nivel Bloom: 3	
OBJETIVO GENERAL	**CONTENIDO**
▪ 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.
Lecturas: [Guttag and Horning, 1991]	

UNIDAD 6: Evolución (12 horas)	
Nivel Bloom: 4	
OBJETIVO GENERAL	**CONTENIDO**
▪ 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.
Lecturas: [Jacky, 1996]	

8. METODOLOGÍA

El profesor del curso presentará clases teóricas de los temas señalados en el programa propiciando la intervención de los alumnos.

El profesor del curso presentará demostraciones para fundamentar clases teóricas.

El profesor y los alumnos realizarán prácticas

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. EVALUACIONES

Evaluación Permanente 1 : 20 %

Examen Parcial : 30 %

Evaluación Permanente 2 : 20 %

Examen Final : 30 %

Referencias

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

[Hinchey and Dean, 1996] Hinchey, M. and Dean, C. N. (1996). *Teaching and Learning Formal Methods*. Morgan Kaufmann.

[Jacky, 1996] Jacky, J. (1996). *The Way of Z: Practical Programming with Formal Methods*. Cambridge University Press.

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