PLAN DE ESTUDIOS (PE): Licenciatura en Ingeniería en Tecnologías de la Información.

AREA: Tecnologías de la Información

ASIGNATURA: Métodos formales

CÓDIGO: ITIM-260

CRÉDITOS: 5

FECHA: Junio de 2013

1. DATOS GENERALES

Nivel Educativo:	Licenciatura.
Nombre del Plan de Estudios:	Licenciatura en Ingeniería en Tecnologías de la Información.
Modalidad Académica:	Presencial.
Nombre de la Asignatura:	Métodos formales
Ubicación:	Nivel formativo
Correlación:	
Asignaturas Precedentes:	Matemáticas discretas
Asignaturas Consecuentes:	Web semántica y Ingeniería de conocimiento
Conocimientos, habilidades, actitudes y valores previos:	**C**onocimientos: Ingeniería de software y métodos de matemáticas discretas. Habilidades: Facilidad para modelar aplicaciones software. Actitudes: Colaborativa, positiva y reflexiva Valores: Responsabilidad, puntualidad y solidaridad

2. CARGA HORARIA DEL ESTUDIANTE

Concepto	Horas por periodo		Total de horas por periodo	Número de créditos
	Teoría	Práctica		
Horas teoría y práctica	5	0	5	5
Total	80	0	80	5

3. REVISIONES Y ACTUALIZACIONES

Autores:	Abraham Sánchez López, Claudia Zepeda Cortes
Fecha de diseño:	junio de 2013
Fecha de aprobación por parte de la academia de área	9 de diciembre de 2013
Fecha de aprobación por parte de CDESC-UA	13 de diciembre de 2013
Fecha de revisión del Secretario Académico	*20 de enero de 2014*
Sinopsis de la revisión y/o actualización:	Materia de nueva creación

4. PERFIL DESEABLE DEL PROFESOR (A) PARA IMPARTIR LA ASIGNATURA:

Disciplina profesional:	Ciencias de la computación, tecnologías de la información.
Nivel académico:	Maestría.
Experiencia docente:	Dos años.
Experiencia profesional:	Dos años.

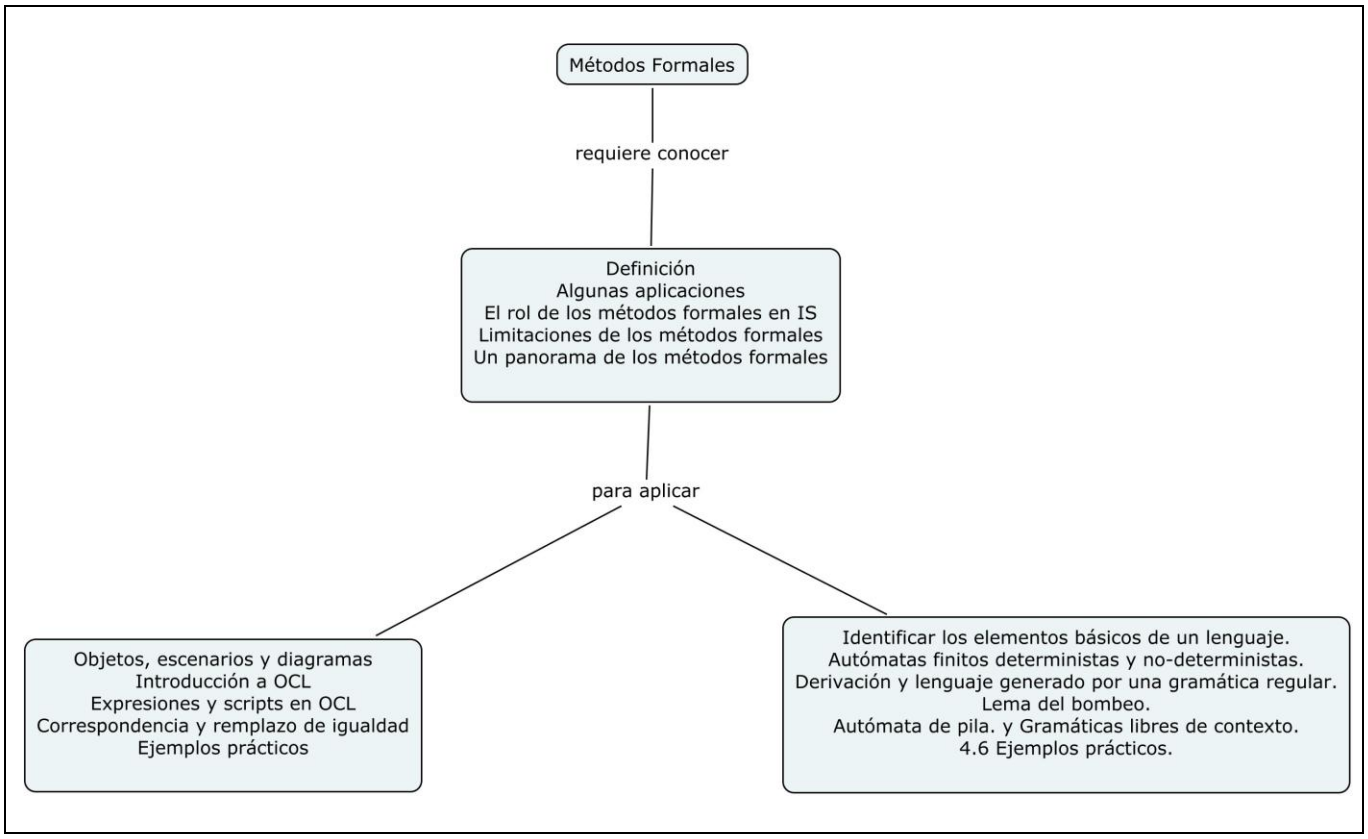
5. OBJETIVOS:

5.1 **General:** Conocer cómo es posible garantizar la rigurosidad, consistencia y completitud del desarrollo de software y con ello evitar los problemas que son el origen de los errores en el software.

5.2 Específicos:

1 Conocer el campo de los métodos formales en la ingeniería de software y su impacto en el desarrollo de aplicaciones
2 Distinguir las ventajas y desventajas del uso de métodos formales en el desarrollo de software de calidad.
3 Conocer y aplicar el lenguaje OCL (lenguaje para la descripción formal de expresiones en los modelos UML).
4 Conocer y aplicar las redes de Petri como uno de los lenguajes formales más ampliamente utilizados en el desarrollo de software formal
5 Conocer y aplicar los autómatas como un lenguaje clásico en el modelado y desarrollo de aplicaciones de software.

6. REPRESENTACIÓN GRÁFICA DE LA ASIGNATURA:



7. CONTENIDO

Unidad	Objetivo Específico	Contenido Temático/Actividades de aprendizaje	Bibliografía	
			Básica	Complementaria
I Introducción a los métodos formales en ingeniería de software	Conocer el campo de los métodos formales en la ingeniería de software y su impacto en el desarrollo de aplicaciones	1.1 Definición 1.2 Algunas aplicaciones 1.3 El rol de los métodos formales en IS 1.4 Limitaciones de los métodos formales 1.5 Un panorama de los métodos formales	Gabbar, Hossam A. (2010) Modern formal methods, verification and applications, Springer-Verlag.	Charatan, Q., Kans, A. (2004) Formal software development: From VDM to Java, Palgrave MacMillan.
II Lenguaje OCL (Object constraint language)	Conocer y aplicar el lenguaje OCL (lenguaje para la descripción formal de expresiones en los modelos UML).	2.1 Objetos, escenarios y diagramas 2.2 Introducción a OCL 2.3 Expresiones y scripts en OCL 2.4 Correspondencia y remplazo de igualdad 2.5 Ejemplos prácticos	Warmer, Jos.,Kleppe, A. (2003) Object constraint language: The getting your models ready for MDA, Addison-Wesley.	Kyas, Marcel. (2006) Verifying OCL specifications of UML models: Tool support and compositionality, Lehmanns.
III Redes de Petri	Conocer y aplicar las redes de Petri como uno de los lenguajes formales más ampliamente utilizados en el desarrollo de software formal	3.1 Introducción 3.2 Modelización con redes de Petri (modelo básico, elementos, evolución) 3.3 Estructuras fundamentales para la modelización de sistemas (paralelismo, sincronización, compartición de recursos, etc.) 3.4 Propiedades de las redes Petri ordinarias 3.5 Redes de Petri coloreadas 3.6 Ejemplos prácticos	Diaz, Michel (editor). (2009) Petri nets: Fundamental models, verification and applications, John Wiley & Sons.	Van der Aalst, Wil Van., Stahl, Christian (2011). Modeling business processes: A petri net-oriented approach, The MIT Press.
IV. Autómatas	Conocer y aplicar los autómatas como un lenguaje clásico en el modelado y desarrollo de aplicaciones de software.	4.1 Identificar los elementos básicos de un lenguaje. 4.2 Autómatas finitos deterministas y no-deterministas. 4.3 Derivación y lenguaje generado por una gramática regular. 4.4 Lema del bombeo. 4.5 Autómata de pila. y Gramáticas libres de contexto. 4.6 Ejemplos prácticos.	Linz, Peter. (2011). An introduction to formal languages and automata, Fifth edition, Jones & Bartlett Learning. Dexter, C. Kozen (2007). Automata and Computability. USA: Springer.	Hopcroft, John E., Motwani, Rajeev., Ullman, Jeffrey D. (2006). Introduction to automata theory, languages and computation, 3rd Edition, Prentice Hall. Kelley, Dean (1995). Automata and Formal Languages: An Introduction. USA: Prentice Hall. (Versión en Ingles). Baral, C. (2003). Knowledge

Unidad	Objetivo Específico	Contenido Temático/Actividades de aprendizaje	Bibliografía	
			Básica	Complementaria
				Representation, Reasoning and Declarative Problem Solving. Cambridge University Press. Baier, Christel and Katoen Joost-Pieter (2008). Principles of Model Checking. England: MIT Press.

8. CONTRIBUCIÓN DEL PROGRAMA DE ASIGNATURA AL PERFIL DE EGRESO

Asignatura	Perfil de egreso (anotar en las siguientes tres columnas, cómo contribuye la asignatura al perfil de egreso)		
	Conocimientos	Habilidades	Actitudes y valores
Métodos formales	Conocer otra perspectiva del modelado de software. Conocer los diferentes lenguajes formales que permiten la especificación y verificación formal de sistemas robustos de software. Conocer y aplicar los lenguajes formales para el desarrollo de sistemas robustos. Conocer la importancia del uso de OCL como elemento adicional al modelado clásico de sistemas de software con UML .	Identificar las diferencias y semejanzas entre el desarrollo de aplicaciones formales y las clásicas. Analizar y clasificar los diferentes lenguajes formales útiles en el desarrollo de software crítico. Distinguir las ventajas y desventajas que ofrece el uso de los diferentes lenguajes formales. Evaluar mediante el desarrollo de una aplicación, el uso del desarrollo formal de sistemas.	Propiciar el interés por el estudio de las tecnologías de vanguardia con actitud propositiva en el desarrollo de aplicaciones críticas. Fomentar mediante otra perspectiva el desarrollo de software. Comprometerse al desarrollo de productos de software de calidad. Centrarse en atender el beneficio que representa el desarrollo formal de sistemas. Visualizar de forma práctica como impacta en la sociedad moderna el uso de las tecnologías de la información.

9. Describa cómo el eje o los ejes transversales contribuyen al desarrollo de la asignatura

Eje (s) transversales	Contribución con la asignatura

Formación Humana y Social	Análisis, reflexión y juicio crítico para utilizar los diferentes lenguajes formales en el modelado de sistemas de software.
Desarrollo de Habilidades en el uso de las Tecnologías de la Información y la Comunicación	Búsqueda de información electrónica relacionada con el desarrollo formal de sistemas de software.
Desarrollo de Habilidades del Pensamiento Complejo	Comprensión del uso de métodos formales en el desarrollo de aplicaciones, como el único medio para el desarrollo de aplicaciones críticas.
Lengua Extranjera	Facilita la comunicación del conocimiento en otros idiomas
Innovación y Talento Universitario	Este curso aporta los elementos diferenciadores en el desarrollo de las aplicaciones críticas, lo que contribuye a una mayor competitividad en el desarrollo de software moderno.
Educación para la Investigación	Habilidad para descubrir y construir nuevos conocimientos aplicables a la solución de problemas planteados en las tecnologías de la información.

10. ORIENTACIÓN DIDÁCTICO-PEDAGÓGICA

Estrategias y Técnicas de aprendizaje-enseñanza	Recursos didácticos
Estrategias de Aprendizaje: El estudiante deberá leer textos, destacará conceptos, elaborará mapas conceptuales, organizará, jerarquizará y aplicará información. **Estrategias de enseñanza:** El profesor Jerarquizará la información y usará preferentemente las técnicas grupales como el aprendizaje colaborativo. **Ambientes de aprendizaje:** Disponibilidad de salones adecuados, bibliotecas y licencias del software requerido. **Actividades y experiencias de aprendizaje:** Se realizarán actividades para el uso del software requerido, también se realizarán actividades que involucren diálogo, redescubrimiento, técnicas grupales, mapas conceptuales, entre otras.	Materiales: - Materiales convencionales: • libros y/o fotocopias - Tableros didácticos: • pizarrón. - Nuevas tecnologías: • PIPE (Platform Indepedent Petri net Editor) • Petri net simulator • Enterprise Architect • OCL-emu - Servicios telemáticos: • Sitios Web • Moodle

11. CRITERIOS DE EVALUACIÓN

Criterios	Porcentaje
▪ Exámenes	40%
▪ Participación en clase	10%
▪ Tareas	20%
▪ Proyecto final	30 %
Total	100%

12. REQUISITOS DE ACREDITACIÓN

Estar inscrito como alumno en la Unidad Académica en la BUAP
Asistir como mínimo al 80% de las sesiones
La calificación mínima para considerar un curso acreditado será de 6
Cumplir con las actividades académicas y cargas de estudio asignadas que señale el PE

13. Anexar (copia del acta de la Academia y de la CDESC- UA con el Vo. Bo. del Secretario Académico)