UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

Licenciatura en Ciencias de la Computación

Facultad de Ciencias

Programa de la asignatura

Denominación de la asignatura:

Ingeniería de Software

Clave:	Semestre:	Eje temático:	No. Créditos:
0575	6	Ingeniería de Software	10

Carácter: Obligatoria	Horas		Horas por semana	Total de Horas
Tipo: Teórico-Práctica	Teoría:	Práctica:	7	112
	3	4		
Modalidad: Curso	Duración del programa: Semestral			

Asignatura con seriación obligatoria antecedente: Álgebra Superior II; Modelado y Programación

Asignatura con seriación obligatoria subsecuente: Ninguna

Asignatura con seriación indicativa antecedente: Fundamentos de Bases de Datos

Asignatura con seriación indicativa subsecuente: Ninguna

Objetivo general:

Comprender para aplicar los componentes del proceso de desarrollo de software con especial énfasis en los roles que de acuerdo al perfil de egreso es probable que desempeñen.

Índice temático

Unidad	Temas	Horas	
		Teóricas	Prácticas
I	Introducción a la ingeniería de software	6	8
II	Elementos de diseño de sistemas	8	10
III	Procesos de desarrollo de software	9	12
IV	Requerimientos y especificaciones	6	8
V	Arquitectura de software	9	12
VI	Validación y verificación	6	8
VII	Evolución de los sistemas de software	4	6
	Total de horas:	**48**	**64**
	Suma total de horas:	**112**	

Contenido temático	
Unidad	**Tema**
I Introducción a la ingeniería de software	
I.1	Objetivos y campo de acción de la ingeniería de software.
I.2	Ciclo de vida del software.
I.3	Roles comunes en el desarrollo de software.
I.4	Principales modelos de desarrollo de software.
I.5	Disciplinas de la ingeniería de software.
II Elementos de diseño de sistemas	
II.1	Características y beneficios del uso de componentes.
II.2	Cualidades deseables en APIs.
II.3	Diseño de APIs.
II.4	Patrones de diseño.
II.5	Herramientas para el manejo de dependencias.
III Procesos de desarrollo de software	
III.1	Modelo de cascada.
III.2	Modelo iterativo.
III.3	Herramientas de colaboración y control de versiones.
III.4	Métodos guiados por plan.
III.5	Metodologías ágiles.
III.6	Técnicas y herramientas de planeación.
IV Requerimientos y especificaciones	
IV.1	Beneficios de la formalización de requerimientos.
IV.2	Clasificaciones de requerimientos.
IV.3	Proceso de formalización de requerimientos.
IV.4	Casos de uso.
IV.5	Alternativas y complementos para la formalización de requerimientos.
IV.6	Aceptabilidad de incertidumbre en los requerimientos.
IV.7	Rastreo de requerimientos.
IV.8	Herramientas para el desarrollo acelerado de aplicaciones.
V Arquitectura de software	
V.1	Capas y componentes.
V.2	Características sistémicas.
V.3	Enfoque de vistas para documentación de arquitecturas de software.
V.4	Métodos tradicionales para el diseño de arquitecturas de software.
V.5	Enfoques emergentes para el desarrollo de arquitecturas de software.
V.6	Caracterización y uso de patrones arquitectónicos.
VI Validación y verificación	
VI.1	Diferencias entre validación y verificación.
VI.2	Tipos de pruebas. Herramientas para pruebas.
VI.3	Consideraciones de diseño para facilitar las pruebas.
VI.4	Manejo de errores.

VI.5	Herramientas de integración continua.
VI.6	Procesos de validación y aseguramiento de la calidad.
VI.7	Desarrollo guiado por pruebas.
VI.8	Validación de elementos auxiliares (non-code).
VII Evolución de los sistemas de software	
VII.1	Conceptos y principios de mantenibilidad.
VII.2	Análisis de riesgo e impacto.
VII.3	Pruebas de regresión.
VII.4	Reutilización de software.
VII.5	Reingeniería de sistemas.
VII.6	*Refactoring.*

Bibliografía básica:

1. Sommerville, Ian, *Software Engineering (9th edition)*. Addison Wesley, 2010.
2. Shari Lawrence Pfleeger y Joanne M. Atlee, *Software Engineering: Theory and Practice*, Fourth Edition. Prentice Hall, 2010.

Bibliografía complementaria:

1. Cockburn, Alistair, *Crystal Clear: A Human-Powered Methodology for Small Teams*. Addison-Wesley Professional; 1st edition, 2004.
2. Barry W. Boehm, Richard Turner, *Balancing agility and discipline: a guide for the perplexed*. Addison-Wesley, 2003.
3. Shari Lawrence Pfleeger y Joanne M. Atlee, *Software Engineering: Theory and Practice*, Fourth Edition. Prentice Hall, 2010.
4. Eric Evans, *Domain-Driven Design: Tackling Complexity in the Heart of Software*. Addison-Wesley.
5. Alistair Cockburn, *Agile Software Development: The Cooperative Game (2nd edition)*. Addison-Wesley Professional, 2nd edition, 2006.
6. *Collective Wisdom from the Experts. 97 Things Every Software Architect Should Know*. Edited By Richard Monson-Haefel, O'Reilly Media Released: February 2009.

Sugerencias didácticas:		Métodos de evaluación:	
Exposición oral	**(X)**	**Exámenes parciales**	**(X)**
Exposición audiovisual	**(X)**	**Examen final escrito**	**(X)**
Ejercicios dentro de clase	**(X)**	**Trabajos y tareas fuera del aula**	**(X)**
Ejercicios fuera del aula	**(X)**	**Exposición de seminarios por los alumnos**	**( )**
Seminarios	**( )**	**Participación en clase**	**( )**
Lecturas obligatorias	**( )**	**Asistencia**	**( )**
Trabajo de investigación	**( )**	**Seminario**	**( )**
Prácticas de taller o laboratorio	**(X)**		
Prácticas de campo	**( )**	**Otras:** Prácticas de laboratorio. Desarrollo de una aplicación.	
Otras: _____			

Perfil profesiográfico:

Egresado preferentemente de la Licenciatura en Ciencias de la Computación o Matemático con especialidad en Computación con amplia experiencia de programación. Es conveniente que posea un posgrado en la disciplina. Con experiencia docente.