Ingeniería de Software

Syllabus

Equipo Docente

Yadran Eterovic

Profesor Asociado. Departamento de Ciencia de la Computación. Coordinador Docente Postgrado, Departamento de Ciencia de la Computación, Representante de Pregrado, Departamento de Ciencia de la Computación.

Descripción del curso

En este curso los estudiantes aprenderán conceptos, técnicas y metodologías que se utilizan para el desarrollo de software confiable y robusto. Los estudiantes serán capaces de entender las distintas etapas del proceso de desarrollo de software, incluyendo la especificación de requisitos, el diseño, el desarrollo, la gestión, técnicas de verificación y validación de software.

Resultados de Aprendizaje

- Aplicar el desarrollo de un sistema de una manera metódica, considerando requisitos, diseño modular, para su implementación que identifique y minimice los riesgos, codificando para su integración de manera colaborativa, usando métodos para identificar y prevenir fallas.
- Desarrollar requisitos claros, concisos y precisos para el desarrollo de un nuevo producto de software (sistema), basados en las necesidades de los usuarios y otros interesados.

- Aplicar principios y patrones al diseñar un sistema y al evaluar el diseño de un sistema pensando en su escalabilidad y mantenibilidad (abstracción, descomposición, ocultación de información, acoplamiento, cohesión, etc.)
- Crear diagramas de clases en UML que modelen el dominio de un problema y la arquitectura de software de un sistema.
- Crear diagramas de secuencia, de estados, y de actividades en UML que modelen los casos de uso y, más en general, el comportamiento de un sistema.
- Aplicar técnicas de testing simples a distintos niveles de un producto de software para verificar y validar la correcta funcionalidad del producto.

Estructura del curso

El curso está estructurado de la siguiente forma:

Unidad 1: Proceso

- Proceso/modelo en cascada
- Procesos iterativos: prototipos y RUP
- Procesos incrementales métodos ágiles
- Scrum y Kanban

Unidad 2: Gestión del Proyecto

- Requisitos Funcionales y no funcionales
- Casos de uso y relatos de usuarios
- Actividades de gestión estimaciones
- Planeación de producto, release y sprint

Unidad 3: Diseño y Arquitectura

- Conceptos fundamentales
- Atributos de un buen diseño acoplamiento y cohesión
- Diagramas UML de clases, secuencia y estados patrones de diseño

- Arquitecturas cliente servidor y multicapas
- Arquitectura orientada a servicios y microservicios

Unidad 4: Aseguramiento de Calidad (QA)

- Definiciones de calidad
- Prevención de defectos
- Detección y eliminación de defectos (testing)

El requisito académico se cumple realizando todos los test (evaluaciones) del curso. El alumno solamente podrá aprobar el curso si aprueba todos los test. Los test se aprueban si se obtiene el 50% de las respuestas correctas. El promedio final del curso será el promedio de la nota final de cada módulo.

Actividad	Evaluación
Cuestionarios	25% nota final
Tareas	25% nota final
Participación en foro	10% nota final
Trabajo final	50% nota final

Plataforma e Información General

Duración: 90 horas totales. 24 directas y 66 de trabajo autónomo.
CRÉDITOS : 5 créditos UC, equivalentes a 3 créditos SCT
REQUISITOS: Sin requisitos
RESTRICCIONES: Programa MDS
CONECTOR: No aplica
CARÁCTER: Mínimo
TIPO: Cátedra
CALIFICACIÓN: Estándar
NIVEL FORMATIVO: Magíster

Política de entregas de evaluaciones calificadas fuera de plazo

En caso de entregar una evaluación calificada, sea esta Tarea o Cuestionario, fuera del plazo informado (fecha límite), se aplicará un descuento progresivo a la nota máxima por entrega tardía. El plazo para entregar evaluaciones o tareas fuera de plazo será de 7 días desde la fecha límite. Luego de los 7 días de plazo adicional, el alumno obtendrá una nota de 0% en dicha evaluación.

Si por razones de fuerza mayor, el alumno/a no pudiera rendir la prueba dentro del plazo regular o excepcional, deberá enviar una solicitud al correo de Soporte de su programa, adjuntando respaldos para que su requerimiento sea evaluado por la Unidad Académica (UA). La resolución de esta solicitud quedará a criterio de la UA.

Bibliografía

Mínima:

- Fundamentals of Software Engineering (2nd Edition), Carlo Ghezzi, Mehdi Jazayeri, Dino Mandrioli, Prentice Hall; September 29, 2002.
- Software Engineering: (Update) (8th Edition), Ian Sommerville; Addison Wesley, June 4, 2006.
- Software Engineering: A Practitioner's Approach /(7 edition), Roger Pressman; McGraw-Hill Science/Engineering/Math, January 20, 2009

Complementaria

- Clean Code: A Handbook of Agile Software Craftsmanship is a book written by Robert. C. Martin.
- User Stories Applied: For Agile Software Development is a book written by Mike Cohn.