PROGRAMA de Elementos de Ingeniería de Software

Carreras: Tecnicatura Universitaria en Programación Informática - Licenciatura en Informática

Asignatura: Elementos de Ingeniería de Software

Núcleo al que pertenece: Básico

Profesores: Pablo Suárez, Diego Sanchez y Warmi Guercio

Asignaturas Correlativas: Programación con Objetos 2

Objetivos:

Que le estudiante:

- Entienda que para llevar a cabo un proyecto de desarrollo de software hace falta llevar a cabo varias actividades además de programar, y tenga una noción de cuáles son estas actividades y las técnicas asociadas a cada una.
- Conozca el concepto de metodología como definición de las actividades que involucra el desarrollo de software, su articulación y los roles que ocupan las personas que participan.
- Conozca los conceptos principales asociados a metodologías ágiles y estructuradas, las actividades y roles que involucran, y algunas similitudes y diferencias entre ambos enfoques.
- Pueda interpretar requerimientos funcionales y no funcionales, y tenga noción de las actividades asociadas a tareas de programación necesarias para lograr la concreción y verificación de los mismos.
- Comprenda la relevancia de los distintos tipos de testing existentes y el alcance de cada uno de ellos e identifique cuales son los más relacionados con las actividades de un programador.
- Tenga una pequeña experiencia práctica aplicando las actividades y metodologías que se describen en la materia.

Contenidos mínimos:

- Teoría General del Sistema. Sistemas de Información.
- Metodologías ágiles: actividades, productos, formas de articulación, roles. Ejemplos: Scrum.

- Metodologías estructuradas: actividades, productos, formas de articulación, roles. Ejemplos: UP.
- Debate sobre similitudes y diferencias entre metodologías ágiles y estructuradas.
- Concepto de ciclo de vida, relación con distintas metodologías.
- Métricas: que son, que miden, para qué sirven, cuando sirven. Ejemplos de métricas asociadas a desarrollo de software en general y actividades de programación en particular.
- Estimación de esfuerzos: relevancia de la experiencia previa para estimar, heurísticas utilizadas. Pertinencia de estimaciones relativas. Técnicas de estimación asociadas a metodologías ágiles.
- Conceptos de requerimiento funcional y no funcional, pertinencia de definiciones comprensibles y adecuadas.
- Comprensión de requerimientos funcionales, detección de inconsistencias. Implementación en código de requerimientos funcionales, verificación de que el código construido cumple los requerimientos.
- Problemas asociados a requerimientos no funcionales: pertinencia de definiciones medibles, nociones sobre técnicas de verificación, posibilidad de garantizarse por construcción.
- Nociones sobre distintos tipos de testing: de unidad, funcional, de sistema, de stress, de carga. Cualidades deseadas y técnicas para lograrlas: regresión, automatización, independencia. Ejemplos concretos de test de unidad y de test funcional. Noción de coverage.
- Prácticas asociadas a extreme programming: peer programming, relevancia de tests automáticos, integración continua, interacción de las actividades de coding y refactor.
- Noción de TDD.
- Nociones de riesgo y plan de contingencia.
- Ingeniería de Software de Sistema de Tiempo Real

Carga horaria semanal: 6 hs

Programa analítico:

Unidad 1: Surgimiento y problemáticas del desarrollo de software

Evolución de la computación y los sistemas de software. La naturaleza del software. Definición de proyecto. Proyectos e ingeniería de software. Formas de contratación. Modelo de ciclo de vida.

Unidad 2: Metodologías de desarrollo de software

Metodologías orientadas al plan y metodologías ágiles. El Proceso Unificado y Scrum. El manifiesto ágil

Unidad 3: Gestión de proyectos

Definición de proyecto. Variables de proyecto. Criterios de éxito. Estimación y planificación. Formas de contratación

Unidad 4: Herramientas y artefactos para el inicio de proyecto

Visión de proyecto. Definición de Alcance. Visual Story Mapping. User Stories. Criterios INVEST. Objetivos SMART. Definición de hecho. Modelado de Dominio, Diagramas UML: clases, actividades, de estado.

Unidad 5: Herramientas y prácticas técnicas para la construcción

Behaviour-Driven Development, Test-Driven Development, Integración continua. Control de la configuración, versionado. Arquitectura de software, patrones de diseño y atributos de calidad.

Unidad 6: Herramientas de gestión

Introducción a la gestión de riesgos. Radiadores de Información. Métricas. Cobertura. Burndown Chart. Comunicación y Manejo de expectativas.

Unidad 7: Proceso unificado.

Ideas centrales. Fases, roles y artefactos. Especificación con casos de uso. Vista 4+1 de arquitectura.

Unidad 8: Puesta en marcha

Gestión de ambientes. Operación de aplicaciones. El movimiento Devops. Entrega continua.

Bibliografía obligatoria:

- Shore & Warden, The Art of Agile Development, O'Reilly Media. 2008.
- Paez et. al., Construcción de Software: una mirada ágil, EDUNTREF. 2014.
- Frederick P. Brooks, Jr., No Silver Bullet Essence and Accidents of Software Engineering. Computer 20, 4, 10-19. 1987.
- David Parnas. Software Engineering - Missing in Action: A Personal Perspective. Computer 44, 10, 54-58. 2011.
- Alistair Cockburn, Agile Software Development, Addison-Wesley Professional , 2001
- Carlos Fontela. Estado del arte y tendencias en Test-Driven Development. Universidad Nacional de La Plata Facultad de Informática Trabajo Final Integrador de la Especialización en Ingeniería de Software. 2011.
- Roger Pressman, Software Engineering, a practitioner's approach. McGraw-Hill Education. 8th Edition, 2014.

Bibliografía de consulta:

- Kent Beck. *Extreme Programming Explained: Embrace Change*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA. 1999
- Philippe Kruchten. *The Rational Unified Process: An Introduction, Second Edition* (2nd ed.). Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA. 2000.
- Pierre Bourque (Redactor), Richard E. Fairley (Redactor), Guide to the Software Engineering Body of Knowledge" - IEEE Computer Society. 2014.

Organización de las clases:

Las clases serán en modalidades teórica, práctica y teórico-práctico dependiendo del tema a desarrollar.
En las clases teóricas se reforzará con un material de lectura y en algunos casos con cuestionarios para realizar a través del campus.
Se dará a les alumnes ejercicios para realizar con el objetivo de asentar los conceptos trabajados en clase. Algunos de estos ejercicios formarán parte de su evaluación.
Durante las últimas 6 semanas de clases, se trabajará con les alumnes en un trabajo práctico grupal, con entregas semanales incrementales, donde tendrán oportunidad de poner en práctica los temas desarrollados durante el curso de la materia. El trabajo finalizará con una exposición de cada grupo frente al resto de sus compañeres.

Prácticas:

Los temas principales del trabajo grupal son: planificación, estimación, comunicación del estado del proyecto, control de versiones, integración contínua, mejora contínua.

El objetivo de la práctica es trabajar y evaluar integralmente la totalidad de las ceremonias de un proyecto ágil, desde la incepción y planificación, pasando por el desarrollo iterativo y concluyendo con la muestra final de los resultados del proyecto.

El trabajo grupal consta de las siguientes fases:

- Inicio / planificación general
- Iteraciones donde se trabaja en la evolución del proyecto grupal (producto y gestión)
- Cierre y muestra final.

Parte de la carga horaria de esta asignatura forma parte del grupo de espacios curriculares de la carrera que implementan horas de Instancias de Formación de Prácticas Profesionales Supervisadas (IFPPS), que se regulan a través del Reglamento aprobado por Res. (CDCyT) N.º 0340/21, o cualquier otra resolución que la modifique o la reemplace. En estas instancias se evalúa a les estudiantes por la aplicación de los conceptos teóricos/prácticos aprendidos en relación con los objetivos de aprendizaje. Así mismo se formulan otros objetivos vinculados a la solución de un problema del mercado laboral, en el cual se insertarán como profesionales. De esta manera, en lo que respecta a las horas totales de la materia, se dedican 10 horas totales para las IFPPS y las 98 horas restantes para clases teóricas y prácticas (trabajos prácticos usuales) necesarias para el aprendizaje de los conceptos.

Modalidad de evaluación:

Los mecanismos de evaluación en modalidades libre y presencial de esta asignatura están reglamentados según los siguientes artículos del Régimen de estudios de la UNQ (Res. CS 201/18).

En la modalidad de libre, se evaluarán los contenidos de la asignatura con un examen escrito, un examen oral y se replicará, en forma resumida, el trabajo del proyecto de trabajo grupal, realizando al menos 3 (tres) iteraciones del mismo.

CRONOGRAMA TENTATIVO

Semana	Tema/unidad	Actividad*				Evaluación
		Teórico	Práctico			
			Res Prob.	Lab.	Otros Especificar	
1	Introdución a Ingeniería de Software	x				
2	Introducción a la gestión de proyectos	x				
3	Metodologías Agiles	x			Dinámica	
4	Modelado de dominio	x	x			
5	TDD		x			
6	Práctica TDD - BDD		x			
7	Estimación - Retrospectivas	x	x			
8	Proceso Unificado – Casos de uso	x				
9	Actividad de repaso				Dinámica	
10	Inicio Trabajo Grupal		x			
11	Seguimiento trabajo grupal		x			
12	Seguimiento trabajo grupal		x			
13	Seguimiento trabajo grupal		x			
14	Seguimiento trabajo grupal		x			
15	Exposición de trabajos grupales				Exposición	
16	Evaluación					x
17	Actividad de retrospectiva				Dinámica	

***INDIQUE CON UNA CRUZ LA MODALIDAD**

Firmado digitalmente por: SUAREZ Mariana Alejandra
Directora del Departamento de Ciencia y Tecnologla
UNIVERSIDAD NACIONAL DE QUILMES