EX-2026-00509716- -UNC-ME#FAMAF

PROGRAMA DE ASIGNATURA	
ASIGNATURA: Ingeniería del Software I	**AÑO:** 2026
CARÁCTER: Obligatoria	**UBICACIÓN EN LA CARRERA:** 3° año 2° cuatrimestre
CARRERA: Licenciatura en Ciencias de la Computación	
RÉGIMEN: Cuatrimestral	**CARGA HORARIA:** 120 horas

FUNDAMENTOS Y OBJETIVOS

La materia se organiza en clases teóricas, clases prácticas y actividades de laboratorio. En las clases teóricas se brindan los contenidos fundamentales de la asignatura. En las clases prácticas se ejercita sobre los temas cubiertos en la teoría, con especial énfasis en actividades de análisis y diseño orientado a objetos, especificación de requisitos y testing. Las clases de laboratorios se utilizan para llevar adelante un proyecto de desarrollo, de tamaño mediano, que es resuelto en grupos y en el cual los alumnos experimentan los problemas que surgen en el desarrollo de un sistema real, siguiendo todas las etapas que involucra el desarrollo de un proyecto real. Las clases teóricas son complementadas con charlas de temáticas variadas vinculadas a la ingeniería de software, brindadas por docentes de la asignatura e invitados de la industria local.

Lograr que el alumno/a sea capaz de:
Entender y aplicar actividades de análisis y especificación de requerimientos, diseño, codificación y testing de software.
Manejar elementos de planificación, especificación y documentación de proyectos usando los paradigmas funcionales y orientado a objetos para el análisis y el diseño de sistemas.

CONTENIDO

1. **Introducción**

El dominio del problema. La "Crisis del Software" y su legado.
El desafío de la Ingeniería del Software, como disciplina para la Calidad.
El enfoque de la Ingeniería del Software, y su importancia frente a la actualidad donde los sistemas computacionales son ubicuos.

2. **El proceso del software Procesos.**

Modelo de procesos. Componentes. Enfoque ETVX.
Características deseadas del proceso del software: Predecible y repetible, Tolerante a cambios, Testeable y Mantenible.
Proceso de desarrollo del software, etapas fundamentales.
Modelos de procesos de desarrollo: Cascada, Prototipado, Iterativo, Time-boxing.
Otros procesos del software: Administración del proyecto, Proceso de inspección, Administración de configuración, Administración de cambios, Administración del proceso (CMM).

3. **Análisis y especificación de los requerimientos del software**

Requerimientos del software, Necesidad de la especificación de requerimientos, Proceso de requerimientos.
Análisis del problema: Enfoque informal, Modelo de flujo de datos (DFD), Modelo orientado a objetos (UML), Prototipado.
Especificación de los requerimientos del software: Características, Componentes, Lenguajes

de especificación, Estructura de un documento.
Especificación funcional con Casos de Uso: Conceptos, Estructura, Abstracción. Validación.
Métricas: Tamaño, Calidad.

4. Arquitectura del software

Rol de la arquitectura del software.
Vistas: Módulos, Componentes y conectores, Asignación de recursos.
Vista de Componentes y Conectores (C&C;). Estilos arquitectónicos para C&C;: Tubos y Filtros, Datos compartidos, Cliente-servidor, Publish-subscribe, Peer-to-peer, Procesos que se comunican. Patrones arquitectónicos: Monolito, Multi-tier, Web architecture - REST API, Microservices, Event driven.
Documentación del diseño arquitectónico.
Arquitectura en comparación con el diseño. Preservación de la integridad de una arquitectura.
Evaluación de las arquitecturas (método de análisis ATAM).

5. Planeamiento del proyecto de software

Planeamiento del proceso.
Estimación del esfuerzo: Incertidumbres, Construcción de los modelos (estimaciones top-down y bottom-up). El modelo COCOMO. Planificación y recursos humanos: Planificación global y detallada, Estructura del equipo de trabajo.
Plan del Control de Calidad: Introducción y eliminación de errores, Enfoques, Plan. Administración del Riesgo: Conceptos, Evaluación, Control.
Planeamiento del seguimiento del proyecto: Mediciones, Seguimiento observacional, Registro del seguimiento.

6. Diseño orientado a funciones

Niveles en el proceso de diseño.
Principios del diseño: Particionado y jerarquía, Abstracción, Modularidad. Estrategias top-down y bottom-up.
Acoplamiento y Cohesión.
Notación y especificación del diseño.
Metodología de diseño estructurado: Cuatro pasos elementales, Heurísticas de diseño, Análisis de transacción (diagrama de estructura).
Verificación.
Métricas: de red, de estabilidad y de flujo de información.

7. Diseño orientado a objetos

Conceptos de la orientación a objetos: Clases, Objetos, Relación entre objetos, Herencia, Polimorfismo.
Conceptos de diseño: Acoplamiento, Cohesión, Principio abierto-cerrado. UML.
Una metodología de diseño: Modelado dinámico, Modelado funcional, Definición de clases y operaciones, Optimización. ORM.
Métricas.

8. Diseño detallado

Lenguaje de diseño de procesos (PDL). Diseño lógico (del algoritmo). Modelo de estado de clases (Autómatas de estado finito). Refinamiento en abstracciones de datos e invariantes de representación.
Verificación: Recorrido del diseño, Revisión crítica (bajo proceso de inspección), Verificación de consistencia y Uso de técnicas formales.
Métricas: Complejidad Ciclomática, Vínculos de datos, Métricas de cohesión.

9. Codificación

Principios y pautas para la programación: Errores comunes, Programación estructurada,

Ocultamiento de la información, Prácticas de programación, Estándares de Codificación. Proceso de Codificación: Incremental, Dirigido por test, Programación de a pares, Control del código fuente y construcción (build).
Refactorización: Conceptos básicos, Malos olores, Refactorizaciones comunes. Verificación: Inspección del código, Test de unidad, Análisis Estático, Métodos formales.
Métricas: Tamaño y Complejidad.

10. Testing

Conceptos fundamentales: Defecto y desperfecto (fault & failure), Oráculos, Casos de test y criterios de selección, Psicología del test.
Testing de caja negra: Particionado por clases de equivalencia, Análisis de valores l mites, Grafo de causa-efecto, Testing de a pares, Casos especiales, Testing basado en estados (Máquinas de estado finitas).
Testing de caja blanca: Criterios basados en flujo de control, Criterios basados en flujo de datos, Testing por mutación, Generación de casos de tests y herramientas de soporte.
El proceso de testing: Niveles, Plan, Especificación de los casos de test, ejecución de los casos de test, Análisis, Registro de defectos y seguimiento. Análisis y Prevención de los defectos.
Métricas. Estimación de la con habilidad.

11. Aspectos profesionales y sociales

Computación y sociedad.
Propiedad intelectual, licencia de software y contratos informáticos.
Aspectos legales.
Responsabilidad y ética profesional.

BIBLIOGRAFÍA

BIBLIOGRAFÍA BÁSICA

- El curso sigue fundamentalmente el libro: Pankaj Jalote. An Integrated Approach to Software Engineering, Third Edition. Springer. 2005. ISBN: 0-387-20881-X.

BIBLIOGRAFÍA COMPLEMENTARIA

- F. Brooks. The Mythical Man Month. Addison-Wesley. 1995.
- R. Glass. The Relationship Between Theory and Practice in Software Engineering. Communications of the ACM, 39(11):1113. Nov. 1996.
- IEEE. Estándares de la IEEE sobre la Ingeniería del Software.
- B. Meyer. Object-Oriented Software Construction (2nd Edition). Prentice Hall. 2000. - D. Parnas. Software Engineering: An Unconsummated Marriage. Communications of the ACM, 40(9):128. Sep. 1997.
- Sun Microsystems. Java Code Convention. 1997.
- R. Stallman et al. GNU coding standards. 2007.

METODOLOGÍA DE ENSEÑANZA

El curso se desarrolla mediante un enfoque teórico-práctico, que combina la fundamentación teórica con la aplicación práctica, simulando entornos reales de desarrollo de software. La metodología es la siguiente:

1. Clases Teóricas:
    - Impartidas siguiendo el libro de texto base (Jalote) complementado con material extra (artículos, casos de estudio, etc.) para actualizar y profundizar los temas.

- Se enfocan en la exposición de conceptos, principios y técnicas de la Ingeniería del Software, fomentando la participación y el debate.

2. Sesiones Prácticas o de laboratorio:
   - Los estudiantes resuelven ejercicios y problemas prácticos.
   - Durante estas sesiones, los docentes responden dudas y proporcionan retroalimentación inmediata sobre los ejercicios y problemas, asegurando la comprensión de los temas vistos en la teoría.
3. Proyecto Integrador:
   - Se asignan tareas para realizar fuera del aula, con un plazo aproximado de dos semanas para su entrega.
   - Los estudiantes tienen la oportunidad de consultar y discutir con sus docentes durante las clases prácticas, para aclarar dudas y recibir orientación.
   - Cada tarea va acompañada de clases específicas que preparan a los estudiantes para resolver los problemas planteados.
   - Los estudiantes, organizados en grupos, desarrollan un proyecto de software que abarca todo el ciclo de vida.
   - Cada grupo es asignado a docentes tutores que realizan seguimiento y brindan asesoría personalizada.
   - Se utilizan herramientas típicas de la industria (como sistemas de control de versiones, entornos de desarrollo integrado, gestores de proyectos) y se sigue un proceso de desarrollo iterativo (por ejemplo, Scrum o similar) para gestionar el trabajo.
   - El proyecto permite aplicar los conocimientos teóricos en un contexto real, desarrollando habilidades técnicas y de trabajo en equipo.
4. Evaluación Continua:
   - La evaluación se basa en exámenes parciales de las clases, y los proyectos, donde se valora no solo el resultado, sino también el proceso, la calidad del trabajo y la capacidad de aplicar metodologías de ingeniería del software.

EVALUACIÓN

FORMAS DE EVALUACIÓN

- Dos evaluaciones parciales y un recuperatorio.
- Un proyecto práctico.
- Un coloquio para estudiantes que aspiren a la promoción.
- Un examen final.

REGULARIDAD

• Cumplir un mínimo de 70% de asistencia a clases prácticas, o de laboratorio.
• Aprobar con una nota no menor a 4 (cuatro), correspondiente al 60%, al menos dos evaluaciones parciales o un parcial y el recuperatorio del otro.
• Aprobar el 60% de los trabajos prácticos y laboratorios.

PROMOCIÓN

• Cumplir un mínimo de 80% de asistencia a clases teóricas, prácticas, o de laboratorio.
• Aprobar todas las evaluaciones parciales con una nota no menor a 6 (seis) correspondiente al 73%, y obteniendo un promedio no menor a 7 (siete) correspondiente al 80%.
• Aprobar el proyecto.
• Aprobar un coloquio.