UNIVERSIDAD DE LIMA

SÍLABO 2024-1

FACULTAD DE INGENIERÍA
CARRERA DE INGENIERÍA DE SISTEMAS

ASIGNATURA	**INGENIERÍA DE SOFTWARE II**
ÁREA	INGENIERÍA DE SOFTWARE
CÓDIGO	1327
NIVEL	OCTAVO
CARÁCTER	OBLIGATORIO
REQUISITOS	INGENIERÍA DE SOFTWARE I
CRÉDITOS	CUATRO (4)
HORAS DE TEORÍA SEMANAL	DOS (2)
HORAS DE PRÁCTICA SEMANAL	CUATRO (4)
IDIOMA	ESPAÑOL
PROFESOR(ES)	MUÑOZ CASILDO, NEHIL INDALICIO (Coordinador) NINA HANCO, HERNAN

I. SUMILLA

En la asignatura se estudian las fases del ciclo de vida de desarrollo de software de diseño, implementación y pruebas, con un amplio nivel de detalle y formalidad. Comprende los siguientes temas: metodologías de desarrollo de software y herramientas para implementar los modelos requeridos en el diseño de software, métodos formales, implementación de la solución y ejecución de casos de prueba, técnicas básicas de aseguramiento de calidad aplicadas a las fases de ciclo de vida tratadas, implementación funcional del producto de software.

II. OBJETIVO GENERAL

Aplicar las distintas técnicas y buenas prácticas en cada etapa de la construcción de un producto de software que permitan su correcta implementación.

III. OBJETIVOS ESPECÍFICOS

1. Aplicar una metodología de desarrollo de software.
2. Aplicar principios y buenas prácticas en diseño y codificación de software.
3. Realizar las pruebas de un producto de software.
4. Establecer, seguir y controlar un proceso de desarrollo de software que siga las buenas prácticas utilizadas por la industria.
5. Comprender los principios para el desarrollo de un producto de software que sea duradero y energéticamente eficiente para reducir el impacto negativo en el medio ambiente.

IV. PROGRAMA ANALÍTICO

UNIDAD 1: DISEÑO DE SOFTWARE **24 horas**

PRIMERA SEMANA
DISEÑO ORIENTADO A OBJETOS USANDO UML
Métodos de Ingeniería de Software. Métodos formales. Principios de diseño. Proceso del diseño de software. Modelado y programación orientada a objetos. Diseño detallado. Modelos de diseño. Aspectos de implementación del software. Medidas de calidad del diseño OO.

SEGUNDA SEMANA
PRINCIPIOS DEL DISEÑO ORIENTADO A OBJETOS
Principios DRY (Don't Repeat Yourself). Principio Law of Demeter, Principios SOLID.

TERCERA SEMANA
DISEÑO DE SOFTWARE CON PATRONES
Patrón de Diseño en Ingeniería del Software. Categoría de patrones. Patrones creacionales: Factory Method, Abstract Factory, Singleton.

CUARTA SEMANA
PATRONES ESTRUCTURALES Y DE COMPORTAMIENTO
Patrones Estructurales: Adapter, Facade. Patrones de comportamiento: Strategy, Command. Patrón Data Access Object (DAO).

UNIDAD 2: DISEÑO ARQUITECTÓNICO **12 horas**

QUINTA SEMANA
DISEÑO DEL SISTEMA Y ARQUITECTURA DEL SOFTWARE
Proceso de diseño de la arquitectura: Requerimientos arquitecturales, Atributos de calidad, Especificación de requisitos de atributos de calidad. Estilos de arquitectura. .

SEXTA SEMANA
DOCUMENTACIÓN DE LA ARQUITECTURA
Documentación de la arquitectura modelo 4+1 vistas .
Documentación de la arquitectura modelo C4.

UNIDAD 3: PRÁCTICAS DE DISEÑO ÁGIL **18 horas**

SÉPTIMA SEMANA
PRUEBAS UNITARIAS
Concepto de pruebas unitarias. Pruebas unitarias estáticas y dinámicas. Herramientas para pruebas. Pruebas unitarias en eXtreme Programming.

OCTAVA SEMANA
DISEÑO ÁGIL
Conceptos de diseño ágil. Principios de diseño de paquetes. Arquitectura en proyectos ágiles.

NOVENA SEMANA
REFACTORIZACIÓN
Conceptos de Refactorización. Principios en Refactorización. Bad Smells in Code. Anti-patrones.

UNIDAD 4: PRUEBAS DE SOFTWARE **24 horas**

DÉCIMA SEMANA
ASEGURAMIENTO DE LA CALIDAD DEL SOFTWARE
Validación y verificación (V&V). Conceptos básicos de pruebas. Tipos de pruebas. Pruebas de análisis de cobertura. Técnicas de pruebas de caja negra. Diseño de casos de prueba. Diseño de Plan de pruebas.

UNDÉCIMA SEMANA
PRUEBAS RENDIMIENTO
Introducción a las Pruebas de Rendimiento. Tipos de Pruebas de Rendimiento Performance, Stress y Carga. Concurrencia, Latencia y throughput

DUODÉCIMA SEMANA
FRAMEWORK PARA PRUEBAS UNITARIAS
Elementos del framework para pruebas unitarias. Automatización de pruebas unitarias.

DECIMOTERCERA SEMANA
PRUEBAS DE MUTACIÓN Y PRUEBAS DE INTEGRACIÓN
Fundamentos de pruebas de mutación. Tipos de pruebas de integración. Framework para mocks. Desarrollo basado en pruebas.

UNIDAD 5: INTEGRACIÓN CONTINUA **6 horas**

DECIMOCUARTA SEMANA
FUNDAMENTOS DE INTEGRACIÓN CONTINUA
El problema de la entrega de software. Implementación de integración continua. Implementación de una estrategia de prueba

DECIMOQUINTA SEMANA **6 horas**
PROYECTO
Proyecto integrador del curso

DECIMOSEXTA SEMANA
Retroalimentación del aprendizaje.

DECIMOSÉPTIMA SEMANA
Entrega final de notas

V. METODOLOGÍA

La Universidad de Lima ha diseñado la **Metodología IATC** para orientar el proceso de enseñanza-aprendizaje y asegurar el logro de los objetivos de la siguiente manera:

- **I**mpacto: motivar y generar curiosidad. Presentar objetivos y agenda.
- **A**dquisición del aprendizaje: transmitir el conocimiento con estrategias innovadoras. Promover la interacción.
- **T**ransferencia de lo aprendido: desarrollar actividades significativas. Utilizar estrategias y técnicas didácticas.
- **C**ierre del aprendizaje: concluir sobre el aprendizaje. Reflexionar sobre el logro del objetivo.

La asignatura se desarrollará en un marco teórico - práctico, para lo cual el alumno hará un trabajo de campo, resolverá casuísticas específicas mediante tareas e irá reportando el avance del trabajo de campo durante el ciclo; todo ello con la finalidad de dar una mayor visión al estudiante de la construcción de un producto de software.

Para el trabajo de campo, se simulará un entorno de desarrollo de software real, para que así los alumnos puedan aplicar sus conocimientos adquiridos en cursos anteriores, así como las prácticas y técnicas aprendidas en este curso.

VI. SISTEMA DE EVALUACIÓN

Para el sistema de evaluación, esta asignatura es de naturaleza:

Teórico práctica

La nota final de la asignatura (NF) es el promedio ponderado de las notas obtenidas en el proceso de evaluación continua (EC):

La nota de EC comprende:

Semana	Tipo de evaluación	Peso (%)	Objetivo que se evalúa
5	EE – Examen escrito	20	OE1 y 2
9	PR - Proyecto	25	OE 1, 2 y 4
12	EE – Examen escrito	25	OE 1 y 2
15	PR - Proyecto	30	OE 2.2, 4.1

VII. REFERENCIAS

OBLIGATORIA

1. Bass, L., Clements, P., & Kazman, R. (2003). Software architecture in practice. Addison-Wesley Professional.
2. Kniberg, H. (2007). Scrum y XP desde las trincheras. Estados Unidos: C4Media.
3. Naik, K., & Tripathy, P. (2011). Software testing and quality assurance: theory and practice. John Wiley & Sons.
4. Sommerville, I. (2015). Software engineering 10th Edition. ISBN-10, 137035152, 18.

COMPLEMENTARIA

5. Bennett, S., McRobb, S., & Farmer, R. (2010). Object-oriented systems analysis and design using UML. McGraw-Hill.
6. Brown, A., Johnston, S. K., Larsen, G., & Palistrant, J. (2005). SOA development using the IBM rational software development platform: a practical guide. Rational Software, 14.
7. Fowler, M. (2018). Refactoring: improving the design of existing code. Addison-Wesley Professional.
8. Pressman, R. S., & Troya, J. M. (2010). Ingeniería del software. McGraw-Hill.
9. Rubin, K. S. (2013). Essential Scrum: a practical guide to the most popular agile process. Upper Saddle River, NJ: Addison-Wesley.
10. Späth, P. (2021). Beginning Java MVC 1.0: Model View Controller Development to Build Web, Cloud, and Microservices Applications. Apress.
11. Weitzenfeld, A. (2004). Ingeniería de software orientada a objetos con UML, Java e Internet. Editorial Thomson.