UNIVERSIDAD NACIONAL DE SAN AGUSTÍN DE AREQUIPA

VICERRECTORADO ACADÉMICO

FACULTAD DE INGENIERIA DE PRODUCCION Y SERVICIOS

DEPARTAMENTO ACADÉMICO DE INGENIERIA DE SISTEMAS E INFORMATICA

SÍLABO 2023 - B

ASIGNATURA: INGENIERIA DE SOFTWARE II

1. INFORMACIÓN ACADÉMICA

Periodo académico:	2023 - B	
Escuela Profesional:	CIENCIA DE LA COMPUTACIÓN	
Código de la asignatura:	1703237	
Nombre de la asignatura:	INGENIERIA DE SOFTWARE II	
Semestre:	VI (sexto)	
Duración:	17 semanas	
Número de horas (Semestral)	**Teóricas:**	2.00
	Prácticas:	2.00
	Seminarios:	0.00
	Laboratorio:	2.00
	Teórico-prácticas:	0.00
Número de créditos:	4	
Prerrequisitos:	INGENIERIA DE SOFTWARE I (1703132)	

2. INFORMACIÓN DEL DOCENTE, INSTRUCTOR, COORDINADOR

DOCENTE	GRADO ACADÉMICO	DPTO. ACADÉMICO	HORAS	HORARIO
SARMIENTO CALISAYA, EDGAR	Doctor	INGENIERIA DE SISTEMAS E INFORMATICA	0	Lun: 10:40-12:20 Mié: 10:40-12:20

3. INFORMACIÓN ESPECIFICA DEL CURSO (FUNDAMENTACIÓN, JUSTIFICACIÓN)

Los tópicos de este curso extienden las ideas del diseño y construcción de software desde la secuencia de introducción a las herramientas y entornos de ingeniería de software, las técnicas de aseguramiento de calidad y pruebas, y los desafíos encontrados en proyectos de gran escala o sistemas legados. Es una visión más amplia y completa de la Ingeniería de Software apreciada desde un punto de vista de Proyectos

4. COMPETENCIAS/OBJETIVOS DE LA ASIGNATURA

a) Aplicar herramientas y entornos de ingeniería de software que contribuyan en la construcción y documentación del software, con especial énfasis en la comprensión de todo el proceso de desarrollo y los procesos de soporte. Esto debe incluir herramientas de construcción automática, automatización de pruebas e integración continua, incluyendo el control de cambios y versiones.

b) Ejecutar pruebas estáticas y dinámicas de componentes de software mediante el uso de herramientas y técnicas de caja blanca y caja negra en colaboración con los clientes.

c) Planificar y ejecutar el proceso para diseñar casos de prueba para una organización utilizando técnicas de caja negra y caja blanca para medir las métricas de calidad en términos de cobertura y rendimiento.

d) Identificar oportunidades para la evolución de software legado, y aplicar la abstracción de procedimientos, refactorización (refactoring), patrones de diseño y reutilización de software.

5. CONTENIDO TEMATICO

PRIMERA UNIDAD

Capítulo I: Herramientas y Entornos de Ingeniería de Software

- **Tema 01:** Gestión de configuración del software y control de versiones
- **Tema 02:** Gestión de entregas (release).
- **Tema 03:** Herramientas de Ingeniería de Requisitos y Diseño de Software
- **Tema 04:** Herramientas de Pruebas: Depuración, Análisis estático y dinámico
- **Tema 05:** Entornos que automatizan el proceso de construcción de software: Construcción Automática, Integración continua, ...

SEGUNDA UNIDAD

Capítulo II: Verificación y Validación de Software

- **Tema 06:** Verificación y Validación: Control de Calidad, Inspecciones, Revisiones (Complejidad Ciclomática y Cognitiva), Auditorias y Pruebas
- **Tema 07:** Fundamentos de pruebas: Proceso (generación, implementación, ejecución y análisis), Técnicas (Estáticas y Dinámicas), Tipos (usabilidad, confiabilidad, seguridad, desempeño y funcionalidad ? conforme a la especificación), Niveles (unitarias, integración, validación y de sistema) y Estrategias (Caja blanca y caja negra) de prueba
- **Tema 08:** Plan de pruebas: Casos de prueba y Métodos de generación de casos de prueba
- **Tema 09:** Automatización de pruebas: Análisis de Código Fuente, Pruebas de Regresión, Arquitectura de pruebas xUnit, Dobles de Prueba (Test Double: Mocks, Stubs, Spies, Dummies, Fakes, ?)
- **Tema 10:** Seguimiento de defectos
- **Tema 11:** Enfoques de desarrollo basado en pruebas: TDD, BDD, ?

TERCERA UNIDAD

Capítulo III: Evolución de Software

- **Tema 12:** Sistemas Legados: Cambios de software y Preocupaciones (concerns)
- **Tema 13:** Evolución, Mantenimiento y Reingeniería de Software
- **Tema 14:** Refactoring

Tema 15: Migración de Software Legado: De Monolitos a Módulos, Microservicios, ...

Tema 16: Reutilización de Software: Bibliotecas, Frameworks, Componentes y Patrones de Diseño

Tema 17: Líneas de Producto de Software

6. PROGRAMACIÓN DE ACTIVIDADES DE INVESTIG. FORMATIVA Y RESPONSABILIDAD SOCIAL

6.1. Métodos

Método expositivo en las clases teóricas,
Método de elaboración conjunta en las aulas prácticas y en la elaboración del proyecto de investigación,

6.2. Medios

Pizarra, presentaciones, videos, software y guías de práctica.

6.3. Formas de organización

i. Clases teóricas: Conceptos
ii. Practicas: Aplicación de Conceptos
iii. Laboratorio: Uso de Herramientas de Software
iv. Otro: Presentaciones, videos

6.4. Programación de actividades de investigación formativa y responsabilidad social

i. Investigación Formativa: Proyecto Final de Investigación o Implementación
ii. Responsabilidad Social: Producir presentaciones o tutoriales sobre los recursos de software utilizados en laboratorio

7. CRONOGRAMA ACADÉMICO

SEMANA	TEMA	DOCENTE	%	ACUM.
1	Gestión de configuración del software y control de versiones	E. Sarmiento	8	8.00
2	Gestión de entregas (release).	E. Sarmiento	4	12.00
2	Herramientas de Ingeniería de Requisitos y Diseño de Software	E. Sarmiento	4	16.00
3	Herramientas de Pruebas: Depuración, Análisis estático y dinámico	E. Sarmiento	8	24.00
4	Entornos que automatizan el proceso de construcción de software: Construcción Automática, Integración continua, ...	E. Sarmiento	8	32.00
6	Verificación y Validación: Control de Calidad, Inspecciones, Revisiones (Complejidad Ciclomática y Cognitiva), Auditorias y Pruebas	E. Sarmiento	4	36.00
6	Fundamentos de pruebas: Proceso (generación, implementación, ejecución y análisis), Técnicas (Estáticas y Dinámicas), Tipos (usabilidad, confiabilidad, seguridad, desempeño y funcionalidad ? conforme a la especificación), Niveles (unitarias, integración, validación y de sistema) y Estrategias (Caja blanca y caja negra) de prueba	E. Sarmiento	6	42.00
7	Plan de pruebas: Casos de prueba y Métodos de generación de casos de prueba	E. Sarmiento	6	48.00

8	Automatización de pruebas: Análisis de Código Fuente, Pruebas de Regresión, Arquitectura de pruebas xUnit, Dobles de Prueba (Test Double: Mocks, Stubs, Spies, Dummies, Fakes, ?)	E. Sarmiento	8	56.00
9	Seguimiento de defectos	E. Sarmiento	4	60.00
10	Enfoques de desarrollo basado en pruebas: TDD, BDD, ?	E. Sarmiento	6	66.00
12	Sistemas Legados: Cambios de software y Preocupaciones (Conceptos)	E. Sarmiento	3	69.00
12	Evolución, Mantenimiento y Reingeniería de Software	E. Sarmiento	3	72.00
13	Refactoring	E. Sarmiento	6	78.00
14	Migración de Software Legado: De Monolitos a Módulos, Microservicios, ...	E. Sarmiento	6	84.00
15	Reutilización de Software: Bibliotecas, Frameworks, Componentes y Patrones de Diseño	E. Sarmiento	8	92.00
16	Líneas de Producto de Software	E. Sarmiento	8	100.00

8. ESTRATEGIAS DE EVALUACIÓN

8.1. Evaluación del aprendizaje

1.- Evaluación Continua. Se evaluará durante todo el semestre a los estudiantes considerando:

1.1. Su actitud solidaria o egoísta, su interés por aprender, el que sea autodidacta y aplicación de los contenidos, participación en clase, el trabajo de investigación formativa, participación en prácticas de laboratorio, tanto para el primer parcial (EC1), segundo parcial (EC2) y tercer parcial (EC3)

2.- Evaluación Periódica.

2.1 Primer Examen (EP1)

2.2 Segundo Examen (EP2)

2.3 Proyecto Final (PF) de Investigación o Implementación

3.- Examen Subsanación o Recuperación (Sustitutorio):

8.2. Cronograma de evaluación

EVALUACIÓN	FECHA DE EVALUACIÓN	EXAMEN TEORÍA	EVAL. CONTINUA	TOTAL (%)
Primera Evaluación Parcial	04-10-2023	15%	15%	**30%**
Segunda Evaluación Parcial	15-11-2023	15%	15%	**30%**
Tercera Evaluación Parcial	20-12-2023	20%	20%	**40%**
			TOTAL	**100%**

9. REQUISITOS DE APROBACIÓN DE LA ASIGNATURA

Se tomará en cuenta para la aprobación del estudiante, las normas establecidas en el Reglamento General de Evaluación del proceso enseñanza aprendizaje de la UNSA:

a) El alumno tendrá derecho a observar o en su defecto a ratificar las notas consignadas en sus evaluaciones, después de ser entregadas las mismas por parte del profesor, salvo el vencimiento de plazos para culminación del semestre académico, luego del mismo, no se admitirán reclamaciones, alumno que no se haga presente en el día establecido, perderá su derecho a reclamo.

b) Para aprobar el curso el alumno debe obtener una nota igual o superior a 10.5, en el promedio final.

c) El redondeo, sólo se efectuará en el cálculo del promedio final, quedado expresó, que las notas

parciales, no se redondearán individualmente.

d) El alumno que no tenga alguna de sus evaluaciones y no haya solicitado evaluación de rezagados en el plazo oportuno, se le considerará como abandono.

e) Los casos particulares por los cuales el alumno no pudo cumplir con su evaluación en el tiempo establecido, podrá tramitar ante la dirección de escuela, su respectiva justificación, con la cual, el profesor tendrá la obligación de tomarle una nueva evaluación, la misma que sustituirá, la nota en cuestión.

f) El estudiante quedará en situación de ?abandono? si el porcentaje de asistencia es menor al ochenta (80%) por ciento en las actividades que requieran evaluación continua (Prácticas, talleres, seminarios, etc.).

g) A continuación, se muestra la fórmula de Promedio Final (PF):

PF= EC1*0.15 + EX1*0.15 + EC2*0.15 + EX2*0.15 + EC3*0.20 * PF3*0.20

10. BIBLIOGRAFIA: AUTOR, TÍTULO, AÑO, EDITORIAL

10.1. Bibliografía básica obligatoria

a) Pressman, R. S. and Maxim, B. (2014). Ingeniería de Software: Un Enfoque Práctico. McGraw-Hill, 8th edition.

b) Sommerville, I. (2010). Ingeniería de Software. Addison-Wesley, 9th edition.

c) McConnell S. (2011). Code Complete: A practical Handbook of software construction. Segunda edición. Microsoft Press.

d) Winters, T., Manshreck, T., & Wright, H. 2020. Software engineering at google: Lessons learned from programming over time. O'Reilly Media.

10.2. Bibliografía de consulta

a) Sander Rossel. 2017. Continuous Integration, Delivery, and Deployment: Reliable and faster software releases with automating builds, tests, and deployment

b) Viktor Farcic. 2016. The DevOps 2.0 Toolkit: Automating the Continuous Deployment Pipeline with Containerized Microservices.

c) Gene Kim, Patrick Debois, John Willis, Jez Humble. 2016. The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations

d) Abu Sayed Mahfuz. 2016. Software Quality Assurance: Integrating Testing, Security, and Audit (Internal Audit and IT Audit).

e) Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides. 2015. Design patterns: Elements of Reusable Object-Oriented Software

f) Robert C Martin. 2004. Working Effectively with Legacy Code

g) Martin Fowler and Kent Beck. 2018. Refactoring: Improving the Design of Existing Code (2nd Edition).

Arequipa, 26 de Setiembre del 2023

SARMIENTO CALISAYA, EDGAR