UNIVERSIDAD NACIONAL DE SAN AGUSTÍN DE AREQUIPA

VICERRECTORADO ACADÉMICO

FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS

DEPARTAMENTO ACADÉMICO DE INGENIERIA DE SISTEMAS E INFORMATICA

SÍLABO 2025 - B

ASIGNATURA: INGENIERIA DE SOFTWARE II

1. INFORMACIÓN ACADÉMICA

Periodo académico:	2025 - 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	DSc.	INGENIERIA DE SISTEMAS E INFORMATICA	0	Mié: 12:20-14:00 Jue: 12:20-14:00
SARMIENTO CALISAYA, EDGAR	DSc.	INGENIERIA DE SISTEMAS E INFORMATICA	0	Mié: 14:00-15:40 Jue: 14:00-15:40

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) Describe los aspectos que son importantes para gestionar la evolución del software a lo largo de su ciclo de vida, y así asegurar consistencia, integridad y trazabilidad.

b) Gestiona repositorios de proyectos en una plataforma única y utiliza herramientas de control de versiones y cambios.

c) Realiza análisis estáticos automáticos de código fuente para detectar errores, smells y vulnerabilidades.

d) Construye un proyecto mediante sistemas de construcción automática.

e) Realiza empaquetado, publicación, despliegue y ejecución automática de un sistema de software en un repositorio central y un entorno en línea (host).

f) Construye un pipeline de CI/CD sencillo para integrar e desplegar automáticamente un software.

g) Mide la complejidad (legibilidad, testabilidad y mantenibilidad) de un programa mediante el desarrollo de un diagrama de flujo de control del código.

h) Diseña, implementa y ejecuta casos de prueba de un componente de software utilizando técnicas de caja blanca para medir la cobertura del código.

i) Diseña, implementa y ejecuta casos de prueba para un sistema bajo prueba utilizando técnicas de caja negra para medir métricas de calidad en términos de funcionalidad.

j) Realiza pruebas automáticas de rendimiento (carga, volumen y estrés).

k) Realiza pruebas automáticas de seguridad (penetración) para detectar vulnerabilidades en el software.

l) Realiza seguimiento al ciclo de vida de los defectos utilizando herramientas de seguimiento y notificación de issues, y realizando pruebas de regresión.

m) Refactoriza un software legado (heredado) mediante análisis estático para eliminar errores, smells y vulnerabilidades, o añadir nuevos requisitos/características.

n) Evoluciona un software legado (heredado) mediante la adición de nuevos requisitos/características.

o) Analiza la arquitectura de un software legado para identificar potenciales problemas de seguridad, rendimiento y confiabilidad, y migra arquitecturas monolíticas a microservicios o guiadas por eventos.

p) Analiza la arquitectura de un software legado para identificar potenciales problemas de reusabilidad, y mejora su capacidad de reutilización mediante patrones de diseño o líneas de producto de software.

5. CONTENIDO TEMATICO

PRIMERA UNIDAD

Capítulo I: Gestión de Configuración de Software

Tema 01: Herramientas y Entornos de Ingeniería de Software

Tema 02: Gestión de Configuración de Software

Tema 03: Gestión de Versiones y Cambios

Tema 04: Construcción Automática y Análisis Estático de Código Fuente

Tema 05: Gestión de Entregas (release)

Tema 06: DevOps y CI/CD

SEGUNDA UNIDAD

Capítulo II: Verificación y Validación de Software

Tema 07: Verificación y Validación: Control de Calidad, Inspecciones, Revisiones (Complejidad

Ciclomática y Cognitiva), Auditorias y Pruebas

Tema 08: Fundamentos de Pruebas: Proceso (generación, implementación, ejecución y análisis), Técnicas (Estáticas y Dinámicas), Tipos (usabilidad, confiabilidad, seguridad, desempeño y funcionalidad), Niveles (unitarias, integración, validación y de sistema) y Estrategias (Caja blanca y caja negra) de prueba

Tema 09: Métodos de Generación de Casos de Prueba: Caja Blanca

Tema 10: Métodos de Generación de Casos de Prueba: Caja Negra

Tema 11: Automatización de Pruebas: Pruebas de Regresión, Arquitectura de Pruebas xUnit, Dobles de Prueba (Mocks, Stubs, Spies, Dummies y Fakes), y Enfoques de desarrollo guiado por pruebas: TDD, ATDD y BDD

Tema 12: Plan de Pruebas y Seguimiento de Defectos

TERCERA UNIDAD

Capítulo III: Evolución de Software

Tema 13: Sistemas Legados: Cambios de software y Preocupaciones (concerns)

Tema 14: Evolución, Mantenimiento y Reingeniería de Software

Tema 15: Refactoring

Tema 16: Rediseño: Migración de Arquitecturas Monolíticas a Arquitecturas Modulares, Basadas en Microservicios, o Microservicios Guiadas por Eventos.

Tema 17: Reutilización de Software: Bibliotecas, Frameworks, Componentes, Patrones de Diseño y Líneas de Producto de Software

6. ESTRATEGIAS DE ENSEÑANZA APRENDIZAJE

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	Herramientas y Entornos de Ingeniería de Software	E. Sarmiento	4	4.00
2	Gestión de Configuración de Software	E. Sarmiento	4	8.00

3	Gestión de Versiones y Cambios	E. Sarmiento	6	14.00
3	Construcción Automática y Análisis Estático de Código Fuente	E. Sarmiento	8	22.00
4	Gestión de Entregas (release)	E. Sarmiento	6	28.00
5	DevOps y CI/CD	E. Sarmiento	6	34.00
7	Verificación y Validación: Control de Calidad, Inspecciones, Revisiones (Complejidad Ciclomática y Cognitiva), Auditorias y Pruebas	E. Sarmiento	6	40.00
7	Fundamentos de Pruebas: Proceso (generación, implementación, ejecución y análisis), Técnicas (Estáticas y Dinámicas), Tipos (usabilidad, confiabilidad, seguridad, desempeño y funcionalidad), Niveles (unitarias, integración, validación y de sistema) y Estrategias (Caja blanca y caja negra) de prueba	E. Sarmiento	6	46.00
8	Métodos de Generación de Casos de Prueba: Caja Blanca	E. Sarmiento	6	52.00
9	Métodos de Generación de Casos de Prueba: Caja Negra	E. Sarmiento	6	58.00
10	Automatización de Pruebas: Pruebas de Regresión, Arquitectura de Pruebas xUnit, Dobles de Prueba (Mocks, Stubs, Spies, Dummies y Fakes), y Enfoques de desarrollo guiado por pruebas: TDD, ATDD y BDD	E. Sarmiento	6	64.00
11	Plan de Pruebas y Seguimiento de Defectos	E. Sarmiento	4	68.00
13	Sistemas Legados: Cambios de software y Preocupaciones (conceptos)	E. Sarmiento	4	72.00
13	Evolución, Mantenimiento y Reingeniería de Software	E. Sarmiento	4	76.00
14	Refactoring	E. Sarmiento	8	84.00
15	Rediseño: Migración de Arquitecturas Monolíticas a Arquitecturas Modulares, Basadas en Microservicios, o Microservicios Guiadas por Eventos.	E. Sarmiento	8	92.00
16	Reutilización de Software: Bibliotecas, Frameworks, Componentes, Patrones de Diseño y 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 Tercer Examen (EP3)
2.4 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	01-10-2025	15%	15%	**30%**

Segunda Evaluación Parcial	19-11-2025	15%	15%	**30%**
Tercera Evaluación Parcial	18-12-2025	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 Nota Final (NF):

NF= EC1*0.15 + EP1*0.15 + EC2*0.15 + EP2*0.15 + (0.5*EC3 + 0.5*PF3)*0.20 + EP3*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.

e) Valente, M. T. (2024). Software Engineering: A Modern Approach. Disponible en: https://leanpub.com/softengbook

10.2. Bibliografía de consulta

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

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

c) Kim, G., Debois, P., Willis, J., and Humble, J. (2016). The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations

d) Mahfuz, A.S. (2016). Software Quality Assurance: Integrating Testing, Security, and Audit (Internal Audit and IT Audit).
e) Gamma, E., Helm, R., Johnson, R., and Vlissides, J. (2015). Design patterns: Elements of Reusable Object-Oriented Software
f) Martin, R.C. (2004). Working Effectively with Legacy Code
g) Fowler, M., and Beck, K. (2018). Refactoring: Improving the Design of Existing Code (2nd Edition).
h) Sarmiento-Calisaya, E. et al. (2024). Introducing Computer Science Undergraduate Students to DevOps Technologies from Software Engineering Fundamentals. IEEE/ACM International Conference on Software Engineering. Preprint disponible en: http://dx.doi.org/10.13140/RG.2.2.21730.95685
i) Henderson, F. (2019). Software Engineering at Google. Disponible en: https://arxiv.org/pdf/1702.01715

Arequipa, 12 de Setiembre del 2025

SARMIENTO CALISAYA, EDGAR