

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 2021 - A

ASIGNATURA: INGENIERIA DE SOFTWARE I

1. INFORMACIÓN ACADÉMICA

Periodo académico:	2021 - A	
Escuela Profesional:	CIENCIA DE LA COMPUTACIÓN	
Código de la asignatura:	1703132	
Nombre de la asignatura:	INGENIERIA DE SOFTWARE I	
Semestre:	V (quinto)	
Duración:	17 semanas	
Número de horas (Semestral)	Teóricas:	2.0
	Prácticas:	0.0
	Seminarios:	0.0
	Laboratorio:	2.0
	Teórico-prácticas:	0.0
Número de créditos:	3	
Prerrequisitos:	CIENCIAS DE LA COMPUTACION II (1702118) BASE DE DATOS I (1702226)	

2. INFORMACIÓN DEL DOCENTE, INSTRUCTOR, COORDINADOR

DOCENTE	GRADO ACADÉMICO	DPTO. ACADÉMICO	HORAS	HORARIO
SARMIENTO CALISAYA, EDGAR	Doctor Informática	INGENIERIA DE SISTEMAS E INFORMATICA	0	Mié: 08:50-10:30
SARMIENTO CALISAYA, EDGAR	Doctor Informática	INGENIERIA DE SISTEMAS E INFORMATICA	0	Mar: 15:50-17:30

3. INFORMACIÓN ESPECIFICA DEL CURSO (FUNDAMENTACIÓN, JUSTIFICACIÓN)

La tarea de desarrollar software, excepto para aplicaciones sumamente simples, exige la ejecución de un proceso de desarrollo bien definido. Los profesionales de esta área requieren un alto grado de conocimiento de los diferentes modelos de proceso de desarrollo, para que sean capaces de elegir el más idóneo para cada proyecto de desarrollo. Este curso introduce las principales actividades, tareas, prácticas

4. COMPETENCIAS/OBJETIVOS DE LA ASIGNATURA

- a) Ser capaz de aplicar los principios y tecnologías de ingeniería de software para asegurar que las implementaciones de software son robustos, fiables y apropiados para su público objetivo.
- b) Entendimiento de lo que las tecnologías actuales pueden y no pueden lograr.
- c) Entendimiento del concepto del ciclo de vida, incluyendo la importancia de sus fases (planificación, desarrollo, implementación y evolución).
- d) Entender las implicaciones de ciclo de vida para el desarrollo de todos los aspectos de los sistemas informáticos (incluyendo software, hardware, y la interfaz de la computadora humana)
- e) Comprender la relación entre la calidad y la gestión del ciclo de vida.
- f) Capacidad para participar de forma activa y coordinada en un equipo.
- g) Modelar y diseñar sistemas de computadora de una manera que se demuestre comprensión del balance entre las opciones de diseño.
- h) Identificar y analizar los criterios y especificaciones apropiadas a los problemas específicos, y planificar estrategias para su solución.
- i) Implementar la teoría apropiada, prácticas y herramientas para la especificación, diseño, implementación y mantenimiento, así como la evaluación de los sistemas basados en computadoras.
- j) Especificar, diseñar e implementar sistemas basados en computadoras.
- k) Implementar efectivamente las herramientas que se utilizan para la construcción y la documentación de software, con especial énfasis en la comprensión de todo el proceso involucrado en el uso de computadoras para resolver problemas prácticos. Esto debe incluir herramientas para el control de software, incluyendo el control de versiones y gestión de la configuración.

5. CONTENIDO TEMATICO

PRIMERA UNIDAD

Capítulo I: Ingeniería de Requisitos

Tema 01: Fundamentos: Requisito

Tema 02: Los Límites del Sistema y el Contexto.

Tema 03: Elicitación de Requisitos.

Tema 04: Especificación de Requisitos: Lenguaje Natural y Modelos

Tema 05: Análisis de Requisitos.

Tema 06: Gestión de Requisitos, Documento de Especificación de Requisitos y Herramientas.

SEGUNDA UNIDAD

Capítulo II: Diseño de Software

Tema 07: Principios de diseño de software

Tema 08: Paradigmas de diseño: estructurado, orientado a objetos, aspectos, funciones, servicios

Tema 09: Modelos estructurales y de comportamiento del diseño orientado a objetos

Tema 10: Arquitectura de software: Fundamentos y Documentación

Tema 11: Estilos y patrones de arquitectura: filtros y tuberías, capas, eventos, micro kernel, plugins, micro servicios

TERCERA UNIDAD

Capítulo III: Construcción de Software

Tema 13: Estilos de programación.

Tema 14: Normas/Estándares/Estilos/Guías/Convenciones de codificación: Codificación Legible (clean code).

Tema 15: Estrategias de Integración: top-down, bottom-up, sandwich

Tema 16: Prácticas de desarrollo de software orientado a objetos: Principios SOLID

Tema 17: Enfoques de desarrollo de software: Domain-driven Design - DDD

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 acrílica, plumones, cañón multimedia, videos, software y guías de práctica.

6.3. Formas de organización

- i. Clases teóricas: Conceptos
- ii. Practicas: Aplicación de Conceptos
- iii. Laboratorio: Uso de Recursos 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	Fundamentos: Requisito	E. Sarmiento	3	3.00
2	Los Límites del Sistema y el Contexto.	E. Sarmiento	3	6.00
3	Elicitación de Requisitos.	E. Sarmiento	6	12.00
4	Especificación de Requisitos: Lenguaje Natural y Modelos	E. Sarmiento	6	18.00
5	Análisis de Requisitos.	E. Sarmiento	6	24.00
5	Gestión de Requisitos, Documento de Especificación de Requisitos y Herramientas.	E. Sarmiento	6	30.00
7	Principios de diseño de software	E. Sarmiento	6	36.00
7	Paradigmas de diseño: estructurado, orientado a objetos, aspectos, funciones, servicios	E. Sarmiento	6	42.00
8	Modelos estructurales y de comportamiento del diseño orientado a objetos	E. Sarmiento	6	48.00
9	Arquitectura de software: Fundamentos y Documentación	E. Sarmiento	6	54.00

10	Estilos y patrones de arquitectura: filtros y tuberías, capas, eventos, micro kernel, plugins, micro servicios	E. Sarmiento	8	62.00
11	Componentes de Software, Bibliotecas, Frameworks y Middlewares.	E. Sarmiento	6	68.00
13	Estilos de programación.	E. Sarmiento	6	74.00
13	Normas/Estándares/Estilos/Guías/Convenciones de codificación: Codificación Legible (clean code).	E. Sarmiento	6	80.00
14	Estrategias de Integración: top-down, bottom-up, sandwich	E. Sarmiento	6	86.00
14	Prácticas de desarrollo de software orientado a objetos: Principios SOLID	E. Sarmiento	6	92.00
15	Enfoques de desarrollo de software: Domain-driven Design - DDD	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	13-05-2021	20%	9%	29%
Segunda Evaluación Parcial	16-06-2021	20%	9%	29%
Tercera Evaluación Parcial	14-07-2021	27%	15%	42%
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 expreso, 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):

$$PF = EC1 \cdot 0.09 + EP1 \cdot 0.20 + EC2 \cdot 0.09 + EP2 \cdot 0.20 + EC3 \cdot 0.15 + PF \cdot 0.27$$

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 2: A practical Handbook of software construction. Segunda edición. Microsoft Press.

10.2. Bibliografía de consulta

a) Karl E Wieggers, Joy Beatty. 2003. Software Requirements.

b) Klaus Pohl, Chris Rupp. 2015. Requirements Engineering Fundamentals, Rocky Nook Inc.

c) M Richards. 2015. Software architecture patterns.

<https://www.oreilly.com/programming/free/software-architecture-patterns.csp>

d) Martin Fowler, Dave Rice, Matthew Foemmel, Edward Hieatt, Robert Mee, and Randy Stafford. 2002. Patterns of Enterprise Application Architecture

e) Robert C. Martin. 2017. Clean Architecture: A Craftsman?s Guide to Software Structure and Design

f) Robert C. Martin. 2008. Clean Code: A Handbook of Agile Software Craftsmanship

g) Lopes, Cristina Videira. Exercises in programming style. CRC Press, 2014. Disponible en:

http://ecoop14.it.uu.se/programme/Lopes_ECOOP_2014.pdf

<https://github.com/crista/exercises-in-programming-style>

h) Eric Evans. Domain-Driven Design: Tackling Complexity in the Heart of Software. 2003

Arequipa, 15 de Abril del 2021

SARMIENTO CALISAYA, EDGAR