

# CIENCIA DE LA COMPUTACIÓN

---

**INGENIERÍA DE SOFTWARE**

**4 CRÉDITOS**



# ÍNDICE

<b>ÍNDICE</b>	2
<b>UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA</b>	3
<b>SILABO 2021-1</b>	3
ASIGNATURA	3
DATOS GENERALES	3
Ciclo: 5º	3
Créditos: Cuatro (4) créditos	3
Horas de teoría: Dos (2) semanales	3
Horas de práctica: Cuatro (4) quincenales	3
Duración del período: Dieciséis (16) semanas	3
Condición: Obligatorio	3
Modalidad: Virtual	3
Requisitos:	3
PROFESORES	3
Profesor coordinador del curso	3
Profesor(es) instructor(es) del curso	3
INTRODUCCIÓN AL CURSO	3
OBJETIVOS	4
COMPETENCIAS Y CRITERIOS DE DESEMPEÑO	4
RESULTADOS DE APRENDIZAJE	5
TEMAS	5
PLAN DE TRABAJO	6
Metodología	6
Sesiones de teoría	6
Sesiones de práctica (laboratorio o taller)	6
SISTEMA DE EVALUACIÓN	7
SESIONES DE APOYO O TUTORÍAS	8
REFERENCIAS BIBLIOGRÁFICAS	8

# UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

## SILABO 2021-1

### 1. ASIGNATURA

CS2901 - Ingeniería de Software I

### 2. DATOS GENERALES

**2.1 Ciclo:** 5º

**2.2 Créditos:** Cuatro (4) créditos

**2.3 Horas de teoría:** Dos (2) semanales

**2.4 Horas de práctica:** Cuatro (4) semanales

**2.5 Duración del período:** Dieciséis (16) semanas

**2.6 Condición:** Obligatorio

**2.7 Modalidad:** Virtual

**2.8 Requisitos:**

- CS1103 Programación Orientada a Objetos II
- CS2701 Base de Datos I

### 3. PROFESORES

#### 3.1 Profesor coordinador del curso

Jesus Bellido Angulo ([jbellido@utec.edu.pe](mailto:jbellido@utec.edu.pe))

Horario de atención: Martes de 4 a 5PM

#### 3.2 Profesor(es) instructor(es) del curso

Jesus Bellido Angulo ( [jbellido@utec.edu.pe](mailto:jbellido@utec.edu.pe) )

Horario de atención: : Martes de 4 a 5PM

Teofilo Chambilla Aquino ( [tchambilla@utec.edu.pe](mailto:tchambilla@utec.edu.pe) )

Horario de atención: Previa coordinación con el profesor

### 4. INTRODUCCIÓN AL CURSO

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. Por otro lado, el desarrollo de sistemas de mediana y gran escala requiere del uso de biblioteca de patrones y componentes del dominio de técnicas relacionadas al diseño basado en componentes.

## 5. OBJETIVOS

**Sesión 1:** Describir cómo el proceso de ingeniería de requisitos apoya la obtención y validación de los requisitos de comportamiento.

**Sesión 2:** Interpretar un modelo de requisitos dada por un sistema de software simple.

**Sesión 3:** Describir los retos fundamentales y técnicas comunes que se utilizan para la obtención de requisitos.

**Sesión 4:** Identificar los requisitos funcionales y no funcionales en una especificación de requisitos dada por un sistema de software.

**Sesión 5:** Realizar una revisión de un conjunto de requisitos de software para determinar la calidad de los requisitos con respecto a las características de los buenos requisitos

**Sesión 6:** Aplicar elementos clave y métodos comunes para la obtención y el análisis para producir un conjunto de requisitos de software para un sistema de software de tamaño medio.

**Sesión 7:** Crear un prototipo de un sistema de software para reducir el riesgo en los requisitos.

**Sesión 8:** Formular los principios de diseño, incluyendo la separación de responsabilidades, encapsulamiento, acoplamiento y cohesión, y la encapsulación.

**Sesión 9:** Usar un paradigma de diseño para diseñar un sistema de software básico y explicar cómo los principios de diseño del sistema se han aplicado en este diseño.

**Sesión 10:** Construir modelos del diseño de un sistema de software simple los cuales son apropiado para el paradigma utilizado para diseñarlo.

**Sesión 11:** Seleccionar componentes adecuados para el uso en un diseño de un producto de software

**Sesión 13:** Describir uno o más patrones de diseño que podrían ser aplicables al diseño de un sistema de software simple.

**Sesión 14:** Construir modelos de verificación y validación de software por medio de pruebas de unidad, funcionales, de integración y de stress.

**Sesión 15:** Utilizar herramientas y metodologías de Integración Continua.

## 6. COMPETENCIAS Y CRITERIOS DE DESEMPEÑO

Los criterios de desempeño que se van a trabajar en este curso son:

Fecha de actualización: 09/04/2021

Revisado y aprobado por el Centro de Excelencia en Enseñanza y Aprendizaje y la Dirección de Ciencia de la Computación

- 3.2. Diseña, implementa y evalúa soluciones a problemas complejos de computación (*nivel 2*)
- 5.1. Trabaja eficazmente en equipo (*nivel 2*)
- 5.2. Lidera eficazmente equipos diversos (*nivel 2*)
- 6.1. Se comunica eficazmente de forma oral en una variedad de contextos profesionales. (*nivel 2*)
- 6.2. Se comunica eficazmente de forma escrita en una variedad de contextos profesionales(*nivel 2*)

## 7. RESULTADOS DE APRENDIZAJE

Al finalizar el curso de Ingeniería de software se espera que el estudiante sea capaz de:

- RA1.** Construir software con estándares de calidad que satisface las necesidad de un proyecto de desarrollo avanzado usando herramientas y metodologías adecuadas.
- RA2.** Desempeñar el rol de un miembro del equipo de desarrollo de software en las distintas fases de la metodología.
- RA3.** Planificar el trabajo del equipo de desarrollo de software coordinando con los miembros del equipo y stakeholders.
- RA4.** Elaborar demostraciones que reflejen el avance del proyecto de desarrollo de software y negociar los entregables con los stakeholders.
- RA5.** Documentar los artefactos de software de forma precisa y clara.

## 8. TEMAS

### 1. Introducción

- 1. Ingeniería de Software
- 2. Metodologías Tradicionales
- 3. Metodologías Ágiles
- 4. Planificación
- 5. Estimación

### 2. Análisis

- 1. Requerimientos
  - 1. Obtención
  - 2. Requerimientos Funcionales
  - 3. Requerimientos No Funcionales
- 2. User stories
- 3. Use Cases
  - 1. Diagrama de Casos de Uso

### **3. Diseño**

1. Principios de diseño
  1. Cohesión
  2. Acoplamiento
2. Patrones de Diseño de Diseño de Software
3. Arquitectura de Software
  1. Diagrama de Arquitectura de Software
  2. Estilo de arquitectura de Software
4. UML
  1. Diagrama de Clases
  2. Diagrama de Secuencia

### **4. Implementación**

1. Code Review
2. Code Quality

### **5. Pruebas**

1. Unit Testing
2. Functional Testing
3. Automated Function Testing
4. Integration Testing
5. Stress Testing

### **6. Deployment**

1. Development Environment
2. Production Environment
3. Integración Continua

## **9. PLAN DE TRABAJO**

### **9.1 Metodología**

Este curso presenta por metodología activa el aprendizaje clásico y el aprendizaje basado en problemas; ambos son fundamentales para introducir al estudiante a los conceptos básicos y afianzar la base necesaria para los siguientes cursos de carrera. Ambos aumentan el interés del estudiante y promueven su compromiso en el aprendizaje.

### **9.2 Sesiones de teoría**

Las sesiones de teoría se llevan a cabo en clases magistrales donde se realizarán actividades que propicien un aprendizaje activo, con dinámicas que permitan a los estudiantes interiorizar los conceptos. Se fomenta la participación individual y en equipo para exponer sus ideas, motivándolos con puntos adicionales en las diferentes etapas de la evaluación del curso.

### **9.3 Sesiones de práctica (laboratorio o taller)**

En las sesiones de laboratorio se propondrán problemas para verificar que los alumnos hayan alcanzado el logro planteado para cada una de las unidades de

aprendizaje, estas actividades les permitirá aplicar los conocimientos adquiridos durante las sesiones de teoría y además estos retos permitirá la evaluación del desempeño de los alumnos.

## 10. SISTEMA DE EVALUACIÓN

Parte de la evaluación continua serán presentaciones grupales, y prácticas individuales en clase y laboratorio. Las prácticas en clase serán presentadas el mismo día salvo algunas excepciones. Mientras que las prácticas de laboratorio tendrán una fecha de entrega.

EVALUACIÓN	TEORÍA (T)	LABORATORIO (L)
	*La ponderación de la evaluación se hará si ambas partes están aprobadas	Evaluación Continua (C1) (5%) Evaluación Continua (C2) (5%) Examen Parcial (E1) (20%) Examen Final (E2) (20%)
	50%	50%
	<b>100%</b>	

Donde:

- **C1** (Semanas 1 - 7) : Tareas + Lecturas + Participación activa en clase.
- **C2** (Semanas 8 - 15) : Tareas + Lecturas + Participación activa en clase.
- **E1** : (Semana 8) Examen Parcial.
- **E2** : Examen Final tomado al finalizar el semestre.
- **P1** : Etapa de Análisis (Semana 5). Esta nota puede ser obtenida en dos intentos:
  - Primer intento: Será evaluado sobre la nota 20.
  - Segundo intento: Reemplazará la nota obtenida durante el primer intento (inclusive si es menor). La nota máxima en este intento es 15.
- **P2** : Etapa de Diseño (Semana 7). Esta nota puede ser obtenida en dos intentos:
  - Primer intento: Será evaluado sobre la nota 20.
  - Segundo intento: Reemplazará la nota obtenida durante el primer intento (inclusive si es menor). La nota máxima en este intento es 15.
- **P3**: Etapa de Implementación y Testeo (Semanas 8 - 13). La nota Dev es el promedio de tres entregas Dev 1.1, Dev 1.2, Dev 1.3. La nota de cada entrega es otorgada por los profesores del curso y el cliente.
- **P4**: Etapa de Pruebas. Esta nota corresponde a la fase de pruebas de unidad, pruebas funcionales y de stress.
- **P4**: Demo Final (Semana 15). Esta nota es otorgada por los profesores del curso y el cliente.

Las rúbricas que permitirán medir las actividades más significativas del curso y que, además se relacionan con la evaluación de las competencias del estudiante son: [enlace](#)

## 11. SESIONES DE APOYO O TUTORÍAS

Este apartado permite formalizar los espacios de apoyo a los estudiantes y que éstos tengan la atención NECESARIA y el tiempo disponible para presentar sus dudas y consultas acerca del curso:

Semana	Fecha/ Hora	Tema a tratar	Objetivos de la sesión
2	27/04/2021 17:00 PM – 18:00 PM	Práctica	Metodologías de Desarrollo
4	11/05/2021 17:00 PM – 18:00 PM	Proyecto	Análisis: Requerimientos Funcionales y Diagrama de Clases
6	25/05/2021 17:00 PM – 18:00 PM	Proyecto	Diseño: Diagrama de Clases, Secuencia y de Arquitectura de Software
8	08/06/2020 17:00 PM – 18:00 PM	Examen 1	Responder a las dudas sobre las preguntas de examen 1, temas relacionados, contenidos, tipos de ejercicios.
10	22/06/2020 17:00 PM – 18:00 PM	Proyecto	Implementación: Revisar la demo funcional 1.0
12	06/07/2020 17:00 PM – 18:00 PM	Proyecto	Implementación: Revisar la demo funcional 1.2
14	20/07/2020 17:00 PM – 18:00 PM	Proyecto	Implementación: Revisar la demo funcional 2.0
16	03/08/2020 17:00 PM – 18:00 PM	Examen 2	Responder a las dudas sobre las preguntas de examen 2, temas relacionados, contenidos, tipos de ejercicios.

## 12. REFERENCIAS BIBLIOGRÁFICAS

- Sommerville, I. (2018). Software engineering 11th Edition.
- Martin, R. C. (2018). Clean architecture: a craftsman's guide to software structure and design. Prentice Hall.
- Martin, R. C. (2013). Clean Code-Refactoring, Patterns, Testen und Techniken für sauberen Code: Deutsche Ausgabe. MITP-Verlags GmbH & Co. KG.
- Freeman, E., Robson, E., Bates, B., & Sierra, K. (2008). Head first design patterns. " O'Reilly Media, Inc."
- Eriksson, H. E., Penker, M., Lyons, B., & Fado, D. (2003). UML 2 Toolkit, CafeScribe (Vol. 26). John Wiley & Sons.

Fecha de actualización: 09/04/2021

Revisado y aprobado por el Centro de Excelencia en Enseñanza y Aprendizaje y la Dirección de Ciencia de la Computación