

# GUÍA DE ESTUDIO

## UNIDAD DE APRENDIZAJE

### INGENIERÍA DE SOFTWARE

<b>UNIDAD ACADÉMICA:</b> ESCUELA SUPERIOR DE CÓMPUTO, UNIDAD PROFESIONAL INTERDISCIPLINARIA DE INGENIERÍA, CAMPUS ZACATECAS	
<b>PROGRAMA ACADÉMICO:</b> Ingeniería en Sistemas Computacionales	
<b>UNIDAD DE APRENDIZAJE:</b> Ingeniería de Software	<b>SEMESTRE:</b> VI

PROPÓSITO DE LA UNIDAD DE APRENDIZAJE				
Elabora un proyecto de software con base en una metodología de ingeniería de software.				
<b>CONTENIDOS:</b>	I. Fundamentos II. Diseño y construcción III. Pruebas de software IV. Calidad y modelos de madurez			
<b>ORIENTACIÓN DIDÁCTICA:</b>	<b>Métodos de enseñanza</b>		<b>Estrategias de aprendizaje</b>	
	a) Inductivo		a) Estudio de Casos	
	b) Deductivo		b) Aprendizaje Basado en Problemas	
	c) Analógico	X	c) Aprendizaje Orientado a Proyectos	X
	d) Heurístico		d) Aprendizaje Autónomo	
<b>EVALUACIÓN Y ACREDITACIÓN:</b>	Diagnóstica	X	Saberes Previamente Adquiridos	X
	Solución de casos		Organizadores gráficos	X
	Problemas resueltos		Problemarios	
	Reporte de proyectos	X	Exposiciones	
	Reportes de indagación	X	Otras evidencias a evaluar: Las que correspondan	
	Reportes de prácticas			
	Evaluación escrita	X		
<b>BIBLIOGRAFÍA BÁSICA:</b>	<b>Autor(es)</b>	<b>Año</b>	<b>Título del documento</b>	<b>Editorial / ISBN</b>
	IEEE Computer Society	2014	Guide to the Software Engineering Body of Knowledge, Version 3.0	IEEE Computer Society Press/ 9780769551661
	Pressman, R.	2010	Ingeniería del software: Un enfoque práctico	Mc Graw Hill/ 9786071503145
	Project Management Institute	2018	Guía práctica de ágil	Project Management Institute/ 9781628254143
	Sommerville, I.	2011	Ingeniería del software, novena edición	Pearson (Addison-Wesley)/ 9786073206037
	Stellman A.	2014	Learning Agile: Understanding Scrum, Xp, Lean, and Kanban	O'reilly/ 978-1449331924

## Contenido

<b>GUÍA DE ESTUDIO .....</b>	<b>1</b>
<b>UNIDAD DE APRENDIZAJE .....</b>	<b>1</b>
<b>INGENIERÍA DE SOFTWARE .....</b>	<b>1</b>
UNIDAD 1. Fundamentos y gestión del proyecto de software .....	3
1.1 Fundamentos de Ingeniería de Software.....	3
1.2 Plan de proyecto de software.....	3
1.3 Modelos de procesos.....	5
1.4 Metodologías ágiles.....	5
UNIDAD 2. Requerimientos, diseño y modelado .....	7
2.1 Ingeniería de requerimientos .....	7
2.2 Diseño del sistema software.....	7
2.3 Herramientas de modelado .....	7
UNIDAD 3. Pruebas de software .....	8
3.1 Tipos de pruebas .....	8
3.2 Herramientas para pruebas.....	8
3.3 Evaluación de las pruebas.....	8
UNIDAD 4. Calidad del software .....	9
4.1 Calidad del proceso software .....	9
4.2 Calidad del producto software .....	9
4.3 Modelos y normas de calidad .....	9
4.4 Modelos de madurez .....	10
Referencias bibliográficas .....	11

Esta guía integra **todos los temas del curso** y está diseñada para apoyar el **estudio autónomo**, la **preparación para exámenes ETS** y el **análisis crítico** de la Ingeniería de Software.

Se estructura por **unidades**, con conceptos clave, preguntas guía y recomendaciones de estudio.

## UNIDAD 1. Fundamentos y gestión del proyecto de software

**Contexto breve:** La Ingeniería de Software surge para controlar la complejidad, costos y calidad del desarrollo de sistemas, aplicando principios de ingeniería, gestión y mejora continua.

### 1.1 Fundamentos de Ingeniería de Software

**Información breve:** La Ingeniería de Software es una disciplina que integra procesos sistemáticos, métodos técnicos y herramientas para desarrollar software confiable, mantenible y de alta calidad, considerando restricciones de tiempo y costo.

**Conceptos clave:**

- Crisis del software
- Software como producto de ingeniería
- Proceso, método y herramienta
- Calidad y mantenibilidad

**Debes saber:**

- Por qué el desarrollo de software requiere disciplina ingenieril.
- Diferencia entre programar y aplicar ingeniería de software.

**Preguntas guía:**

- ¿Qué problemas resuelve la ingeniería de software?
- ¿Por qué el software necesita procesos formales?

### 1.2 Plan de proyecto de software

### 1.2.1 Ámbito del software

#### Conceptos clave:

- Alcance
- Límites del sistema
- Exclusiones del proyecto

#### Debes saber:

- Distinguir entre alcance y requerimientos.

### 1.2.2 Factibilidad y análisis de riesgo

#### Conceptos clave:

- Factibilidad técnica, económica, operativa, legal y de tiempo
- Identificación y mitigación de riesgos

#### Debes saber:

- Clasificar tipos de factibilidad.
- Identificar riesgos comunes en proyectos de software.

### 1.2.3 Métricas y estimación

#### Conceptos clave:

- Líneas de código (LOC)
- Puntos de función
- Estimación de esfuerzo y costo

#### Debes saber:

- Diferenciar métricas orientadas al tamaño y a la función.

### 1.2.4 Planificación del proyecto

**Conceptos clave:**

- Cronograma
- Diagrama de Gantt
- Herramientas de gestión (Jira, Trello, MS Project)

**Debes saber:**

- Interpretar un plan de trabajo.

### 1.2.5 Supervisión y control del proyecto

**Conceptos clave:**

- Seguimiento del avance
- Métricas de desempeño
- Acciones correctivas

**Debes saber:**

- Comparar lo planeado vs lo real.

### 1.3 Modelos de procesos

**Conceptos clave:**

- Cascada
- Incremental
- Espiral
- Modelo V

**Debes saber:**

- Ventajas y desventajas de cada modelo.

### 1.4 Metodologías ágiles

#### 1.4.1 XP

- Programación en pareja
- Integración continua

#### 1.4.2 Scrum

- Roles
- Eventos
- Artefactos

#### 1.4.3 Kanban

- Flujo de trabajo
- WIP

#### 1.4.4 Lean

- Eliminación de desperdicios
- Mejora continua

#### Debes saber:

- Comparar metodologías ágiles.

## UNIDAD 2. Requerimientos, diseño y modelado

**Contexto breve:** Esta unidad traduce las necesidades del cliente en soluciones técnicas claras, usando requerimientos bien definidos, diseño estructurado y modelos que facilitan la comunicación.

### 2.1 Ingeniería de requerimientos

**Conceptos clave:**

- Requerimientos funcionales
- Requerimientos no funcionales
- Trazabilidad

**Debes saber:**

- Identificar características de un buen requerimiento.

### 2.2 Diseño del sistema software

#### 2.2.1 Arquitectura lógica

- Capas
- Componentes
- Relaciones

#### 2.2.2 Interfaz de usuario (UI)

- Elementos visuales
- Interacción

#### 2.2.3 Experiencia de usuario (UX)

- Usabilidad
- Satisfacción

**Debes saber:**

- Diferenciar UI y UX.

### 2.3 Herramientas de modelado

**Conceptos clave:**

- UML
- Diagramas de casos de uso, clases, secuencia, actividades

**Debes saber:**

- Elegir el diagrama adecuado según el problema.

## **UNIDAD 3. Pruebas de software**

**Contexto breve:** Las pruebas permiten detectar defectos, reducir riesgos y asegurar que el sistema cumple requerimientos funcionales y no funcionales antes de su liberación.

### **3.1 Tipos de pruebas**

#### **3.1.1 Pruebas funcionales**

- Unitarias
- Integración
- Sistema
- Aceptación

#### **3.1.2 Pruebas no funcionales**

- Rendimiento
- Seguridad
- Usabilidad

### **3.2 Herramientas para pruebas**

**Conceptos clave:**

- Selenium
- JMeter
- Automatización

### **3.3 Evaluación de las pruebas**



**Conceptos clave:**

- Cobertura
- Densidad de defectos
- Efectividad de pruebas

**Debes saber:**

- Interpretar resultados de pruebas.

## **UNIDAD 4. Calidad del software**

**Contexto breve:** La calidad del software se asegura mediante normas, métricas y modelos de madurez que permiten evaluar tanto el proceso como el producto y promover la mejora continua.

### **4.1 Calidad del proceso software**

**Conceptos clave:**

- Mejora continua
- Evaluación del proceso

### **4.2 Calidad del producto software**

**Conceptos clave:**

- Funcionalidad
- Confiabilidad
- Usabilidad
- Mantenibilidad

### **4.3 Modelos y normas de calidad**

#### **4.3.1 ISO**

- ISO/IEC 25010

#### **4.3.2 IEEE**

- Estándares de documentación y procesos

#### 4.3.3 IEC

- Normas técnicas

#### 4.3.4 Otras

- ISO/IEC 15504 (SPICE)
- ITIL

### 4.4 Modelos de madurez

#### 4.4.1 PSP

- Enfoque individual

#### 4.4.2 TSP

- Enfoque de equipo

#### 4.4.3 CMMI

- Niveles de madurez

#### 4.4.4 MoProSoft

- Enfoque en PYMES

## Referencias bibliográficas

1. Pressman, R. S., & Maxim, B. R. *Ingeniería de Software: Un enfoque práctico*. McGraw-Hill.
2. Sommerville, I. *Software Engineering*. Pearson.
3. ISO/IEC 25010:2011. *Systems and software Quality Requirements and Evaluation (SQuaRE)*.
4. ISO/IEC 12207:2017. *Software life cycle processes*.
5. IEEE Std 830 / IEEE 29148. *Software Requirements Specification*.
6. CMMI Institute. *CMMI for Development*.
7. SEI. *Personal Software Process (PSP) and Team Software Process (TSP)*.
8. Secretaría de Economía. *MoProSoft: Modelo de Procesos para la Industria de Software*.