AUTOEVALUACION DEL CAPITULO 1

PARCIAL No. 1

METODOLOGÍAS DE DESARROLLO EN SOFTWARE

METODOLOGÍAS DE DESARROLLO EN SOFTWARE
ÁREA DE COMPUTACION
DEPARTAMENTO DE ITIN

ACTIVIDAD DE APRENDIZAJE -C1.

Nombres del estudiante:	Ismael Alejandro Silva Flores
Nivel:	Tercer Nivel
NRC:	29022
Asignatura:	Met. de Desarrollo en Software
Nombre del profesor:	Ing. Jenny Alexandra Ruiz Robalino

METODOLOGÍAS DE DESARROLLO EN SOFTWARE

ÁREA DE COMPUTACION

DEPARTAMENTO DE ITIN

1. ¿Qué es ingeniería de software?

La ingeniería de software es "una disciplina de la ingeniería que se preocupa de todos los aspectos de la producción de software" (Sommerville, 2011, p. 7). Su objetivo principal es aplicar un enfoque sistemático, disciplinado y cuantificable al desarrollo, operación y mantenimiento de software (IEEE, 1990). Esto implica la aplicación de teorías, métodos y herramientas para construir software profesional y de alta calidad.

2. ¿Cuáles son las actividades fundamentales de la ingeniería de software?

El proceso de desarrollo de software, o ciclo de vida, se compone de cuatro actividades fundamentales genéricas que se aplican a todos los modelos de proceso (Pressman & Maxim, 2020):

- **Especificación de software:** Definición de la funcionalidad del software y las restricciones operativas.
- **Desarrollo de software:** Diseño y programación del software.
- **Validación de software (Pruebas):** Verificación de que el software cumple con lo especificado por el cliente.
- **Evolución de software (Mantenimiento):** Adaptación y actualización del software para satisfacer las necesidades cambiantes de los clientes y el entorno.

3. ¿Cuál es la diferencia entre ingeniería de software y ciencias de la computación?

Disciplina	Enfoque Principal	Objetivo
Ciencias de la Computación	Es fundamentalmente teórica. Se centra en los fundamentos subyacentes de las computadoras y el software (por ejemplo, teoría de la computación, algoritmos) (Denning, 2004).	Comprender la naturaleza de la computación.
Ingeniería de Software	Es fundamentalmente práctica. Se centra en los problemas de desarrollar y entregar sistemas de software útiles y confiables de manera eficiente y a tiempo (Sommerville, 2011).	Producir software de calidad que cumpla con los requisitos del usuario de forma eficiente y confiable.

La ingeniería de software se apoya en las Ciencias de la Computación para obtener bases teóricas, pero añade aspectos de gestión de proyectos, costos, ética, y la disciplina de ingeniería para la producción industrial (Pressman & Maxim, 2020).

4. ¿Qué es software?

El software no es solo el código del programa. Pressman y Maxim (2020, p. 4) lo definen como:

- **Instrucciones (código):** Programas de computadora que, cuando se ejecutan, proporcionan funcionalidad y características.
- **Estructuras de datos:** Información que permite a los programas manipular la información.
- **Documentación:** Descripción de cómo opera el *software* y cómo usarlo (manuales, modelos de diseño, etc.).

En esencia, el software es el conjunto de programas, datos y documentación asociados necesarios para que un sistema informático funcione.

5. ¿Cuáles son los atributos del buen software?

Un software de calidad debe exhibir ciertas características para ser considerado "bueno" o profesional (Sommerville, 2011):

- **Mantenibilidad:** Debe ser posible que el software evolucione para satisfacer los cambios en los requerimientos.
- **Confiabilidad y Seguridad:** No debe causar fallas operacionales ni daños, y debe protegerse contra ataques externos maliciosos.
- **Eficiencia:** Debe hacer un uso económico de los recursos del sistema (memoria, procesador, etc.).
- **Aceptabilidad (Usabilidad):** Debe ser inteligible, operable y compatible con otros sistemas de software.

6. ¿Cuáles son los principales retos que enfrenta la ingeniería de software?

Los retos modernos se centran en la velocidad de cambio y la complejidad de los sistemas:

- **Reto de la Heterogeneidad:** Desarrollar software que pueda operar y comunicarse de manera fiable en sistemas distribuidos y diversas plataformas (móviles, cloud, IoT) (Sommerville, 2011).
- **Reto de la Entrega Rápida:** La presión para entregar software en plazos cortos, lo que ha impulsado la adopción de metodologías ágiles.
- **Reto de la Confianza:** Garantizar que el software sea seguro contra ciberataques, robusto en caso de fallas y que su comportamiento sea predecible (Pressman & Maxim, 2020).

7. ¿Cuáles son los costos de la ingeniería de software?

El costo del software abarca todo su ciclo de vida y puede dividirse en dos categorías principales (Sommerville, 2011):

- **Costos de Desarrollo:** Son los costos incurridos antes de la entrega del software (análisis, diseño, codificación, pruebas).
- **Costos de Evolución (Mantenimiento):** Son los costos posteriores a la entrega. **Este es el costo más significativo** en el ciclo de vida, a menudo superando el 60% del costo total. Incluye la corrección de errores, la adaptación a nuevos entornos y la adición de nuevas funcionalidades.
- El factor de costo más importante es el **salario de los ingenieros de software** que participan en el proceso.

8. ¿Cuáles son los mejores métodos y técnicas de la ingeniería de software?

No existe un "mejor" método, ya que la elección depende del proyecto. Los métodos se clasifican comúnmente en dos grandes familias:

- **Métodos o Paradigmas Clásicos/Planificados (Ej. Modelo de Cascada):** Se enfocan en una planificación detallada y secuencias rigurosas, ideales para proyectos con requisitos estables y bien definidos (Pressman & Maxim, 2020).
- **Métodos Ágiles (Ej. Scrum, Kanban):** Se enfocan en la adaptabilidad, la entrega continua y la colaboración cercana con el cliente, ideales para proyectos con requisitos cambiantes o con incertidumbre (Beck et al., 2001).
- Las **técnicas** incluyen herramientas de modelado (UML), técnicas de prueba, y métodos de gestión de configuración de software.

9. ¿Qué entiende por paradigmas de la ingeniería de software?

Un paradigma de la ingeniería de software se refiere a un **marco o modelo conceptual** que define la manera en que se estructura y se lleva a cabo el proceso de desarrollo de software. Cada paradigma prescribe un conjunto específico de actividades, artefactos, roles y flujos de trabajo.

Ejemplos comunes de paradigmas son:

- **Paradigma de Cascada** (Secuencial).
- **Paradigma Iterativo/Incremental** (RUP).
- **Paradigma Evolutivo** (Prototipado).
- **Paradigma Ágil** (Scrum).

10. ¿Cuál es la definición del modelo incremental versus otros paradigmas? Características y diferencias con otras ingenierías.

Definición del Modelo Incremental

- El **modelo incremental** (también llamado iterativo e incremental) es un paradigma de desarrollo donde el software se construye en **pequeñas porciones (incrementos o iteraciones)**, que son funcionales y evaluables. Cada incremento añade funcionalidad a la versión anterior.

- **Ventaja clave:** Permite la entrega rápida de un software útil para el cliente (la funcionalidad principal), y reduce el riesgo al abordar el desarrollo en etapas manejables.

Características

- **Iteración:** El desarrollo se repite a lo largo de ciclos de corta duración.
- **Incremento:** Cada ciclo de desarrollo produce una nueva versión ejecutable del sistema.
- **Retroalimentación:** La versión funcional se muestra al cliente para obtener feedback y planificar el siguiente incremento.

Paradigma	Enfoque Principal
Cascada (Clásico)	Lineal y secuencial. Los requisitos se fijan por completo al inicio. La funcionalidad completa se entrega en una sola vez al final.
Incremental	Iterativo y por fases. Se liberan versiones funcionales progresivamente. Los requisitos pueden evolucionar entre incrementos.

Características y Diferencias con Otras Ingenierías

La Ingeniería de Software (IS) se diferencia de la Ingeniería Civil, Mecánica o Eléctrica por la **naturaleza inmaterial** de su producto y por el **costo de los cambios**:

- **Naturaleza Inmaterial (Flexibilidad):** Es más fácil y barato realizar cambios en el software (cambiar código) que en estructuras físicas (cambiar una pared de hormigón). Esto permite que el software sea inherentemente más **flexible** y que los requisitos sean más inestables (Sommerville, 2011).
- **Proceso:** En la IS, la mayor parte del esfuerzo está en el **diseño intelectual** (análisis, diseño) y en la **evolución**. En otras ingenierías, el esfuerzo se traslada a la **manufactura** y a los costos de materiales.

Referencias

Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R. C., Mellor, S., Schwaber, K., Sutherland, J., & Thomas, D. (2001). *Manifiesto por el desarrollo ágil de software* [Manifiesto]. https://agilemanifesto.org/iso/es/manifesto.html

Denning, P. J. (2004). *Computer science: The discipline*. En *Encyclopedia of computer science* (4th ed.). Wiley.

IEEE. (1990). *IEEE standard glossary of software engineering terminology (IEEE Std 610.12-1990)*. IEEE.

Pressman, R. S., & Maxim, B. R. (2020). *Ingeniería de software: Un enfoque práctico* (9.ª ed.). McGraw-Hill Education.

Sommerville, I. (2011). *Ingeniería de software* (9.ª ed.). Pearson Educación.