UNIVERSIDAD DE LAS FUERZAS ARMADAS

Carrera: Ingeniería en Tecnologías de la información.

Asignatura: Metodología del desarrollo de software

Tema del taller: Autoevaluación 1

Docente: Ing. Jenny Ruiz

Integrantes: Kevin Cañola, Cristhian Marcalla, Mateo Lugmaña, Eduardo Tasiguano

Fecha: 13-10-2025 **Paralelo:** 29022

Autoevaluación 1

1. La persona que acuñó por primera vez el término "ingeniería del software" fue:

Margaret Hamilton.

Justificación:

Margaret Hamilton utilizó por primera vez el término *software engineering* durante su trabajo en el *MIT Instrumentation Laboratory* en los años 60, cuando lideró el desarrollo del software de navegación del programa Apollo de la NASA. Lo empleó para resaltar la importancia de aplicar principios de ingeniería al desarrollo de software, en un momento en que esta práctica aún no era reconocida formalmente como una rama de la ingeniería.

Bibliografía:

- Piattini, M. (2016). *Ingeniería del Software: Un enfoque práctico*. Ra-Ma Editorial.
- ACM. (2020). *Computing Curricula 2020: Paradigms in Software Engineering and Computer Science*.

2. Los elementos que componen el software son:

Programas, procedimientos, documentación y datos relacionados.

Justificación:

El software no se limita únicamente al código ejecutable; también incluye la documentación técnica y los procedimientos operativos que permiten su correcto funcionamiento, mantenimiento y uso. Estos componentes conforman un sistema integral que facilita la interacción entre el usuario y la máquina.

Bibliografía:

- Pressman, R. S., & Maxim, B. R. (2015). *Software Engineering: A Practitioner's Approach* (8.ª ed.). McGraw-Hill.
- Sommerville, I. (2016). *Software Engineering* (10.ª ed.). Pearson Education.

3. Oficialmente, el término ingeniería del software se acuñó en:

La Conferencia de la OTAN de 1968.

Justificación:

El término "ingeniería del software" fue formalizado en la *Conferencia de la OTAN sobre Ingeniería de Software* celebrada en Garmisch, Alemania, en 1968. En este evento, expertos discutieron la llamada "crisis del software", que reflejaba los problemas de fiabilidad, mantenimiento y sobrecostos en el desarrollo de sistemas informáticos complejos.

Bibliografía:

- Bauer, F. (1968). *Software Engineering: NATO Science Committee Report*.
- Hinojosa, C. (2019). *Fundamentos de Ingeniería de Software*. Universidad Autónoma de México.

4. La definición de tipo de software correcto es:

Software que hace uso de algoritmos no numéricos para resolver problemas complejos para los que no son adecuados el cálculo o el análisis directo (software de inteligencia artificial).

Justificación:

El software de inteligencia artificial (IA) se caracteriza por emplear técnicas heurísticas y algoritmos no numéricos, como redes neuronales, sistemas expertos o aprendizaje automático, para resolver problemas que requieren razonamiento, aprendizaje o reconocimiento de patrones, en lugar de simples cálculos matemáticos.

Bibliografía:

- García, J. (2018). *Introducción a la Ingeniería del Software*. Alfaomega.
- Piattini, M. (2016). *Ingeniería del Software: Un enfoque práctico*. Ra-Ma Editorial.

5. ¿Cuáles son los atributos de un buen software?

Mantenible, confiable y fácil de utilizar.

Justificación:

Según la norma ISO/IEC 25010, un software de calidad debe ser mantenible (fácil de modificar y actualizar), confiable (funcionar correctamente bajo condiciones establecidas) y usable (intuitivo y accesible para el usuario). Estos atributos garantizan la sostenibilidad, efectividad y satisfacción del cliente.

Bibliografía:

- Sommerville, I. (2016). *Software Engineering* (10.ª ed.). Pearson Education.
- ISO/IEC 25010:2011. *Systems and Software Quality Requirements and Evaluation (SQuaRE)*.

6. Las características del software son:

El software se desarrolla o modifica con intelecto, no se fabrica en el sentido clásico.

Justificación:

El software no se produce en una línea de ensamblaje como los productos físicos; se desarrolla a través del pensamiento, la lógica y la creatividad humana. Su construcción implica análisis, diseño, codificación y verificación, actividades basadas en conocimiento, no en manufactura repetitiva.

Bibliografía:

- Pressman, R. S., & Maxim, B. R. (2015). *Ingeniería del Software: Un enfoque práctico*. McGraw-Hill.
- Sommerville, I. (2016). *Software Engineering* (10.ª ed.). Pearson Education.

7. La crisis del software se refiere a los problemas que desde sus inicios ha ido experimentado este. Muchas veces los problemas de gran magnitud se generan debido a la mínima eficacia que presenta una gran cantidad de empresas al momento de realizar un software.

Verdadero.

Justificación:

La llamada *crisis del software* surgió por la dificultad de entregar programas confiables, a tiempo y dentro del presupuesto. La falta de métodos formales y planificación provocó fallas graves, sobrecostos y sistemas no funcionales, lo que impulsó el surgimiento de la ingeniería del software como disciplina formal.

Bibliografía:

- Bauer, F. (1968). *NATO Software Engineering Report*.
- Piattini, M. (2016). *Ingeniería del Software: Un enfoque práctico*. Ra-Ma Editorial.

8. A partir del siguiente gráfico, los nombres de las fases del modelo en Cascada (Waterfall) son:

Comunicación, planificación, diseño-modelado, construcción-implementación, entrega-implantación.

Justificación:

El modelo en cascada propone un flujo secuencial en el que cada fase depende de la anterior: primero se comunica con el cliente (requisitos), se planifica el proyecto, se diseña el sistema, se construye y prueba el código, y finalmente se entrega e implanta el producto. Este enfoque fue uno de los primeros en formalizar el proceso de desarrollo.

Bibliografía:

- Royce, W. W. (1970). *Managing the Development of Large Software Systems*. Proceedings of IEEE WESCON.
- Sommerville, I. (2016). *Software Engineering* (10.ª ed.). Pearson Education.

9. El modelo de proceso de software en espiral propuesto por Boehm conjuga la naturaleza iterativa de la construcción de prototipos con los aspectos controlados y sistemáticos del modelo lineal secuencial. La etapa que no pertenece al modelo es:

Definición de un paradigma de desarrollo.

Justificación:

El modelo espiral de Boehm consta de cuatro actividades principales: planificación, análisis de riesgos, desarrollo/validación y evaluación del cliente. No incluye una etapa llamada "definición de un paradigma de desarrollo", ya que el modelo mismo es un paradigma que guía el proceso iterativo y controlado de desarrollo.

Bibliografía:

- Boehm, B. W. (1988). *A Spiral Model of Software Development and Enhancement*. IEEE Computer, 21(5).
- Pressman, R. S., & Maxim, B. R. (2015). *Software Engineering: A Practitioner's Approach*. McGraw-Hill.

10. Se construye un buen sistema de información considerando que el punto de partida es:

Utilizar un proceso definido con fases claras, donde cada una de estas genera un producto final.

Justificación:

Un desarrollo exitoso depende de un proceso estructurado con etapas bien definidas (análisis, diseño, implementación, prueba y mantenimiento). Cada fase produce entregables verificables que aseguran calidad, trazabilidad y control del proyecto, minimizando errores y costos.

Bibliografía:

- Piattini, M. (2016). *Ingeniería del Software: Un enfoque práctico*. Ra-Ma Editorial.
- Sommerville, I. (2016). *Software Engineering* (10.ª ed.). Pearson Education.