14- Mantenimiento de software (Software Maintenance): es el proceso de adaptar o modificar un sistema de software para corregir fallas o mejorar el rendimiento.

Mantenimiento correctivo (Corrective Maintenance): es la práctica de reparar fallas en sistemas de software.

Insuficiente (Insufficient): Si algo es insuficiente, no es adecuado o lo suficientemente fuerte para un propósito particular.

Mantenimiento adaptativo (Adaptive Maintenance): es la práctica de actualizar el software según cambios en el entorno, como mejoras en hardware o sistemas operativos, sin cambiar la funcionalidad del software.

Ley del cambio continuo (Law of continuing change): es un principio que establece que un sistema en uso debe experimentar un cambio continuo hasta que sea más rentable reestructurar el sistema.

Mejorar (Enhance): Mejorar algo es mejorar su función.

Liberación (Release): es una versión actualizada de un programa de software existente.

Mantenimiento perfectivo (Perfective maintenance): es la práctica de actualizar el software para acomodar nuevos requisitos de usuario.

Reparar (Repair): Reparar algo es corregir partes que no funcionan correctamente.

Ley de la creciente complejidad (Law of increasing complexity): es un principio que establece que una estructura se vuelve más compleja con cada cambio que se le hace.

Código no estructurado (Unstructured code): es el código de un sistema diseñado pobremente o codificado sin una estructura clara u orden.

Mantenimiento preventivo (Preventive maintenance): es la práctica de mejorar la estructura de un sistema para facilitar su mantenimiento.

15- Sistemas heredados (Legacy systems): es un antiguo sistema de software que continúa siendo actualizado y utilizado.

Modernizar (Modernize): Modernizar algo es hacerlo compatible con nueva tecnología o actualizar su apariencia y funcionalidad.

Ingeniería inversa (Reverse engineering): es el proceso de analizar un sistema de software existente y crear una nueva versión del sistema a un nivel más alto de abstracción.

Basado en la web (Web-based): Si algo es basado en la web, se utiliza en Internet.

Renovación (Revamping): es el proceso de actualizar la interfaz de usuario de un programa sin cambiar la estructura del programa.

Redocumentación (Redocumentation): es el proceso de mejorar o simplificar el código de un programa sin cambiar su función o nivel de abstracción.

Recuperación de diseño (Design recovery): es el proceso de crear un programa que sea idéntico a un programa existente en función, pero mejor organizado en abstracción.

Equivalencia funcional (Functional equivalence): es una medida de cuán similares son dos programas en propósito y función, aunque puedan estar codificados de manera diferente.

Reingeniería (Reengineering): La reingeniería, también llamada renovación, es el proceso de realizar cambios funcionales en un sistema.

Renovación (Renovation): La renovación, también llamada reingeniería, es el proceso de realizar cambios funcionales en un sistema.

Reestructuración (Restructuring): es el proceso de actualizar un sistema manteniendo la misma funcionalidad y nivel de abstracción.

1- Modelos conceptuales (Conceptual models): es un modelo técnicamente preciso de un sistema informático que se representa en términos de las reacciones de un sistema a las acciones del usuario.

Interfaz de usuario (User interface): es una colección de atributos que rige la forma en que un usuario interactúa con un sistema.

Modelo mental (Mental model): es la comprensión de un usuario sobre un sistema informático.

Vista cognitiva (Cognitive view): es un medio para entender un sistema que considera lo que un usuario necesita comprender sobre un sistema para poder operarlo.

Vista lingüística (Linguistic view): es un modelo conceptual que describe las interacciones entre un humano y un sistema.

Vista de diseño (Design view): es un modelo conceptual que se centra en el diseño de la interfaz de usuario.

CLG (Comando Lenguaje Gramática) (CLG(Command Language Grammar)): es una estructura gramatical específica que describe los aspectos de la interfaz de usuario de un sistema informático.

Componente conceptual (Conceptual component): es una vista de un sistema que se refiere a las funciones que el sistema realizará para los usuarios.

Componente de comunicación (Communication component): es una vista de un sistema que se refiere al diálogo entre sistemas y usuarios.

Componente material (Material component): es una vista de un sistema que se refiere a los gráficos de la interfaz de usuario y el hardware con el que el usuario interactuará.

Nivel semántico (Semantic level): es una vista de un componente conceptual que describe los objetos del sistema y la delegación general de tareas.

Nivel de tarea (Task level): es una vista del componente conceptual que se refiere a las tareas realizadas tanto por la máquina como por el usuario.

Nivel de diseño espacial (Spatial layout level): es una vista de un componente material que especifica los elementos gráficos que se muestran en la pantalla.

Nivel de aparato (Apparatus level): es una vista del componente material que especifica la forma y sensación de botones, teclas y otro hardware con el que el usuario interactuará.

Nivel de pulsación de tecla (Keystroke level): es una vista del componente de comunicación que describe las acciones físicas de un usuario, como pulsaciones de teclas o clics de mouse.

Nivel de sintaxis (Syntax level): es una vista de un componente de comunicación que describe el estilo de diálogo especificando todas las interacciones entre el usuario y el sistema.

2- Diseño artístico (Artistic Design): es la práctica de utilizar el diseño gráfico para llamar la atención del usuario sobre partes importantes de una interfaz.

Ergonomía (Ergonomics): es el estudio del diseño de hardware destinado a ser operado físicamente por los usuarios.

Modelo Seeheim (Seeheim model): es un modelo de diseño de software que separa la aplicación de la interfaz de usuario.

Paradigma MVC (Modelo-Vista-Controlador) (MVC Paradigm(model-view-controller)): es un patrón de diseño para interfaces de usuario que divide la aplicación en tres áreas: el modelo, la vista y el controlador.

Capas (Layers): es un nivel de operación de un sistema.

HCI (Interacción humano-computadora) (HCI(human-computer interaction)): es el estudio y diseño de interacciones entre usuarios y computadoras.

Análisis de tareas (Task analysis): es el acto de evaluar un sistema complejo en términos de sus usuarios, tareas, hardware, entorno social y entorno físico.

Presentación (Presentation): son los aspectos colectivos de un sistema que son perceptibles para el usuario, como el diseño de la pantalla o el diseño del teclado.

Usuario final (End User): es un consumidor que se convierte en el usuario previsto o principal de un producto.

Diálogo (Dialog): es una comunicación recíproca entre una computadora y un usuario.

Funcionalidad (Functionality): es el rango de operaciones que una computadora o sistema de software puede realizar.

Humanidades (Humanities): es el estudio o enfoque en cómo las personas perciben, aprenden, piensan y sienten.

Diseño centrado en el usuario (User-centered design): es un proceso de diseño que pone gran énfasis en la experiencia de los usuarios finales.

Groupware: es software diseñado para ayudar a un grupo de personas a lograr un objetivo común o completar una tarea colaborativa.

UVM (Máquina Virtual del Usuario) (UVM(user virtual machine)): es el hardware y software de un sistema dado.

3- Crisis del software (Software crisis): es un problema en la industria del software causado por el hecho de que la demanda de nuevas aplicaciones de software es mayor de lo que los desarrolladores de software pueden cumplir.

Reutilización de software (Software reuse): es la práctica de incorporar piezas modificadas o no modificadas de código fuente de software existente en la creación de nuevo software.

Alcance (Scope): es la extensión de algo o el área que incluye.

Ad hoc (Ad hoc): Si algo es ad hoc, es no sistemático y se hace solo para una instancia particular.

Reutilización de caja blanca (White-box reuse): es un método de reutilización de software en el que los elementos de software se modifican antes de incorporarlos en nuevo software.

Técnicas (Techniques): es una habilidad o método específico para hacer o crear algo.

Códigos fuente (Source codes): es una lista de comandos que se ejecutarán en un programa de computadora.

Productos (Products): es algo que está disponible para su compra.

COTS (Comercial, listo para usar) (COTS(commercial, off-the-shelf)): Si un software es COTS, no se modifica desde su estado original y generalmente se desconoce el contenido del software.

Reutilización de caja negra (Black-box reuse): es un método de reutilización de software en el que los elementos de software se reutilizan sin modificación.

Sustancia (Substance): son los componentes, conceptos y procedimientos de algo.

Enfoque (Approach): es una manera o estrategia de hacer o crear algo.

Composicional (Compositional): Si la tecnología es composicional, sus componentes existentes pueden reutilizarse fácilmente en nuevos sistemas.

Generativo (Generative): Si la tecnología es generativa, sus componentes se utilizan para crear programas que generan nuevos programas.

Uso (Usage): es la manera en que algo se utiliza.

4- Rebusca de código (Code scavenging): es el proceso de reutilizar código que ha sido escrito previamente, si resuelve problemas actuales.

Biblioteca de programas (Program library): es una colección de piezas de código listas para usar.

Productos intermedios (Intermediate Products): es un fragmento de código que está listo para ser utilizado en el desarrollo de una aplicación más complicada.

Instanciar (Instantiate): Instanciar algo es completarlo o darle sustancia.

Plantilla (Template): es un componente esquelético que no tiene todos los detalles de un programa completo.

Generador de aplicaciones (Application generator): es una herramienta que ayuda a los ingenieros a escribir programas a gran escala.

Análisis de dominio (Domain analysis): es un proceso que identifica, captura, estructura y reorganiza información para el desarrollo de software.

Sistema de transformación (Transformation system): es una aplicación que ayuda a los ingenieros a transformar sistemas de conjuntos de especificaciones a programas ejecutables.

ADL (Lenguaje de descripción de arquitectura) (ADL(architecture description language)): es un sistema que describe formalmente la configuración arquitectónica de un sistema de software.

MIL (Lenguaje de Interconexión de Módulos) (MIL(Module Interconnection Language)): es una descripción formal de la estructura general de un sistema de software.

Esqueleto (Skeleton): Si un componente es un esqueleto, no se han llenado todos sus detalles.

Middleware: es un software que conecta el sistema operativo de una computadora a aplicaciones individuales y asegura que los programas puedan ejecutarse juntos de manera fluida.

VHLL (Lenguaje de Muy Alto Nivel): es un lenguaje de programación con un alto nivel de abstracción utilizado principalmente por programadores para ayudar en la creación de nuevos programas.

5- Modelo de confiabilidad del software (Software reliability model): es un modelo estadístico que tiene como objetivo predecir y prevenir fallas en el software.

Confiabilidad (Reliability): es la calidad de ser consistente y libre de errores.

Programación robusta (Robust programming): es la práctica de garantizar que los componentes de software funcionen correctamente independientemente de su contexto.

Dominio de excepción (Exception domain): es el conjunto de entradas incorrectas para un componente de software.

Errores de software (Software errors): son fallas o defectos en el software.

Dominios de excepción esperados (Expected exception domains): son las entradas incorrectas que se anticipan y reconocen como válidas por el software.

Umbral (Threshold): es un límite que debe superarse para que ocurra una cierta reacción.

Tolerante a fallas (Fault-tolerant): si un disco es tolerante a fallas, contiene datos de respaldo en caso de falla del software.

Bloques de recuperación (Recovery blocks): es un archivo de datos guardado automáticamente que se utiliza como respaldo en caso de que una operación cause una falla de software.

Programación de N-versiones (N-version programming): es una técnica de tolerancia a fallas en el software en la que se generan múltiples programas funcionalmente equivalentes a partir de las mismas especificaciones iniciales.

BM (Modelo básico de tiempo de ejecución): es un modelo básico de tiempo de ejecución en la confiabilidad del software.

LPM (Modelo de tiempo de ejecución poisson logarítmico): es un modelo de confiabilidad del software en el que la disminución en la intensidad de fallas es constante.

Redundancia: es la inclusión de componentes que no son necesarios o son copias de componentes existentes para garantizar el correcto funcionamiento del software en caso de error o falla.

Probabilidad: es la medida en que algo es probable que ocurra.

6- Ingeniería Asistida por Computadora (CASE, por sus siglas en inglés, Computer Aided Software Engineering): es la aplicación de varios sistemas de apoyo en el proceso de desarrollo de software.

Herramientas (Tools): son instrumentos o aplicaciones utilizados en el desarrollo de software.

Entornos centrados en el lenguaje (Language-centered environments): Un entorno centrado en el lenguaje, también llamado entorno de programación, es un entorno de desarrollo interactivo que contiene herramientas para el desarrollo en un lenguaje de programación específico.

Entornos integrados (Integrated environments): es un entorno de desarrollo que contiene las especificaciones de un producto final.

Entorno centrado en el proceso (Process-centered environment): es un entorno de desarrollo que se enfoca en el proceso de desarrollo de software.

Escala de procesos (Process scale): es una característica de desarrollo de software que especifica si un producto admite el desarrollo de código o actividades humanas generales.

Entornos (Environments): es una aplicación que respalda el desarrollo completo de software.

Juego de herramientas (Toolkit): es un entorno de desarrollo en el que las herramientas son independientes entre sí y no están bien integradas.

Banco de trabajo (Workbench): es un conjunto de herramientas relacionadas que respaldan el proceso de desarrollo de software en un alcance limitado.

Escala de usuarios (User scale): es un sistema que mide la cantidad de usuarios que un producto puede admitir.

Familia (Family): Si un valor en la escala de usuarios es familia, indica que un producto está diseñado para facilitar las interacciones entre los desarrolladores.

Ciudad (City): Si un valor en la escala de usuarios es ciudad, indica que un producto respalda el desarrollo de un sistema más grande que una familia.

Estado (State): Si un valor en la escala de usuarios es estado, indica que un producto se centra en la uniformidad y estandarización en un sistema muy grande.

Individual (Individual): Si un valor en la escala de usuarios es individual, indica que un producto está diseñado para ayudar en la construcción de software por desarrolladores individuales.