Índice del resumen

Deuda técnica

costo del re-trabajo adicional causado por la elección de la solución más rápida en lugar de la más efectiva.

DEUDA TÉCNICA = DEUDA TECNOLÓGICA = DEUDA DE CÓDIGO

1. Prudente y deliberada: Se toma una decisión consciente de asumir una deuda para obtener una entrega rápida.
2. Imprudente y deliberada: Se prioriza la rapidez en la entrega a pesar de saber que se podría producir un mejor código.
3. Prudente e inadvertida: Se descubre una mejor solución después de haber implementado el código.
4. Imprudente e inadvertida: El equipo no tiene el conocimiento necesario para producir el mejor código y comete errores sin darse cuenta.

Ciclo de vida del software

Es un proceso que describe las diferentes fases por las que pasa un proyecto de software, desde la concepción de la idea hasta la finalización del desarrollo, el mantenimiento y la eventual retirada del software. Este proceso se divide en etapas que están diseñadas para garantizar que el software se desarrolle de manera eficiente, efectiva y de alta calidad.

requisitos -> que

La ingeniería de requerimientos es como hacer una lista de compras para una receta o un plan para construir algo. Es importante entender lo que las personas quieren para poder hacer un buen trabajo, y también es importante hacer cambios y seguir todo en orden para que todo funcione bien.

Requisito: es una característica del software que alguien quiere, necesita u ordena. describen lo que hace el software y las limitaciones que debería tener. son importantes porque nos comunican las expectativas de los consumidores de productos software. los requisitos solo tratan de descubrir los objetivos del sistema

requisito = definición = lo que **debería hacer** el software **!= como**

¿Qué queremos que haga el sistema?
¿Quiénes son los usuarios del sistema?
¿Cuáles son las necesidades de los usuarios?
¿Qué debe hacer el sistema para lograr esas necesidades?

se dividen en 2:

Requisitos Funcionales

Los requerimientos funcionales son como una lista de deseos que las personas hacen para un software nuevo. Estos requerimientos describen todas las cosas que el software debe poder hacer, como por ejemplo mostrar una lista de tareas o enviar mensajes. También describen cómo todas estas funciones del software deben trabajar juntas. Los requerimientos funcionales son importantes porque son la base desde la cual se diseña el software. Los ingenieros de software comienzan con estos requerimientos para saber qué características y funcionalidades deben incluir en el software.

Requisitos no Funcionales

Los requerimientos no funcionales describen cómo debería ser el software en cuanto a su desempeño, seguridad, mantenimiento y apariencia. Son características que no se relacionan con las acciones del software sino con su entorno y restricciones, como la cantidad de transacciones que debe procesar, cumplir con las leyes de seguridad y acceso, y tener una interfaz fácil de usar. Estos requerimientos se pueden dividir en tres categorías: de producto, de organización y externos, dependiendo si se refieren al código, políticas de la empresa o leyes y regulaciones. Analizar y categorizar los requerimientos no funcionales es importante para la ingeniería de requisitos, que es la parte del proceso de desarrollo de software que se encarga de definir lo que el software debe hacer y cómo debe hacerlo.

Especificaciones

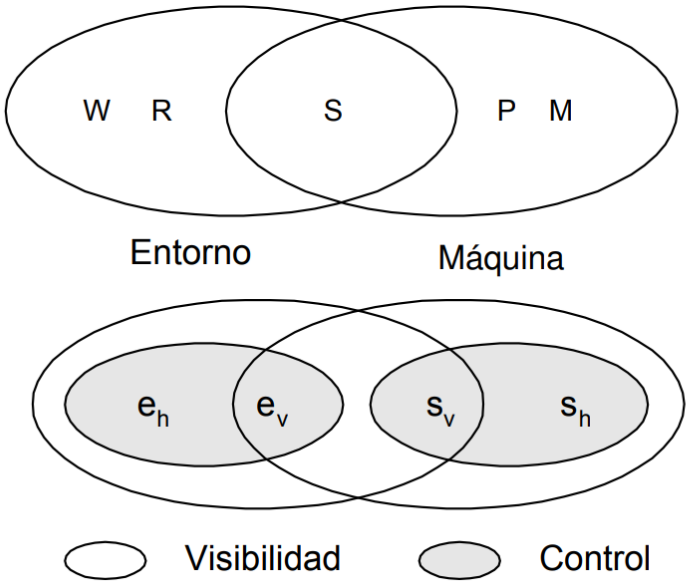
Las especificaciones son documentos técnicos que describen las características y el comportamiento del sistema. Ayudan a tener un entendimiento claro del producto a desarrollar y minimizan las fallas en el software. En ellas se plasman todo lo acordado entre el equipo de desarrollo y el cliente sobre lo que debe hacer el sistema. Las especificaciones detallan las funciones del software, los datos que debe capturar y almacenar y la información que debe generar. Una especificación es un requerimiento ya analizado.

Especificaciones vs Requisitos

Los requisitos son lo que el usuario necesita del sistema de forma sencilla y sin jerga técnica, mientras que las especificaciones son notas técnicas sobre lo que se debe hacer para cumplir con el requisito. Los requisitos funcionales definen las funcionalidades que el sistema debe tener, mientras que los no funcionales definen las características esperadas del sistema. Las especificaciones son importantes para tener un entendimiento claro del producto que se requiere desarrollar y minimizar las fallas en el software.

Requisitos: lo que el usuario quiere.
Especificaciones: lo que el sistema hace.

Modelo WRSPM



W: dominio de aplicación

R: requerimientos

S: especificación

P: programa que implementa S

M: plataforma de cómputo

e y *s*: fenómenos del entorno y del sistema

El modelo WRSPM ayuda a identificar los elementos involucrados en la solución de los problemas y consta de los siguientes elementos:

1. Dominio (W): son las suposiciones del mundo que son verdaderas y tienen un impacto en nuestro sistema y en nuestro dominio de problemas.
    a. Conocimiento de dominio: es el conocimiento sobre el mundo real relevante al proyecto de desarrollo y provee el contexto donde las acciones de la máquina (sistema) tienen significado o son útiles.
2. Requerimientos (R): son los requisitos y representan la comprensión del idioma del usuario de lo que el usuario quiere de la solución.
3. Especificación (S): es la interfaz entre cómo el sistema cumplirá con esos requisitos. Está escrito en el lenguaje del sistema que dice en español sencillo lo que hará el sistema.
4. Programa (P): es lo que escribirán los desarrollos de software. El programa cumplirá con las especificaciones para proporcionar el objetivo del usuario para los requisitos.
5. Máquina (M): es la especificación de hardware.
6. EH (Elementos del Entorno Ocultos al Sistema) se refiere a los elementos del entorno que son importantes para el usuario, pero que no son directamente visibles para el sistema.
7. EV (Elementos del Entorno Visibles para el Sistema) son los elementos del entorno que el sistema puede detectar y utilizar para llevar a cabo su tarea.
8. SV (Elementos del Sistema Visibles en el Entorno) son los elementos del sistema que el usuario puede ver y utilizar para interactuar con el sistema.
9. SH (Elementos del Sistema Ocultos al Entorno) son los elementos del sistema que son importantes para la realización de la tarea, pero que no son visibles para el usuario.

Para que la solución sea correcta, se deben cumplir las siguientes ecuaciones:

Si [W, S] implican R y [M, P] implican S, entonces la solución es correcta.

Dado [W, S], las suposiciones del Mundo y las Especificaciones, implican que se cumplen los requisitos. Dado [M, P], la Máquina y el Programa, implican que la Especificación se han cumplido.

Es importante tener cuidado con las suposiciones del mundo y asegurarse de investigar todas las suposiciones para garantizar que se cumplan estas ecuaciones. Además, tener en cuenta que el modelo WRSPM no garantiza que la solución sea infalible y que pueden surgir problemas en la programación y la prueba de la solución.

diseño -> como

El diseño es como la planificación que haces antes de crear algo, ya sea un dibujo o un software. Los patrones de diseño son como plantillas que te ayudan a hacer diseños más bonitos y efectivos. Los patrones de diseño de software y de arquitectura son herramientas que ayudan a los programadores a escribir programas de computadora de una manera más efectiva.

El modelado de software es importante porque permite comunicar ideas, evaluar alternativas, aproximarse gradualmente al producto y visualizarlo antes de construirlo.

Los modelos son representaciones abstractas del software que capturan una vista del sistema del mundo real y describen los aspectos relevantes del sistema a un nivel de detalle adecuado. Es importante tener trazabilidad entre los modelos para asegurarse de que cada requisito esté respaldado por una pieza de código y viceversa.

Los diagramas son representaciones gráficas de un modelo y se utilizan para expresar el producto desde diferentes perspectivas.

El nivel de aprovechamiento de los modelos varía desde simplemente documentar después de construir el producto hasta la generación automática de código a partir de modelos.

La especificación de análisis es independiente de la plataforma de implementación y se enfoca en establecer el "qué" en lugar del "cómo".

Enfoques más recientes como MDA establecen modelos independientes de la plataforma (CIM), modelos independientes de la plataforma (PIM) y modelos específicos de la plataforma (PSM).

ENFOQUE ESTRUCTURADO

Está el modelado de procesos y el modelado de datos.

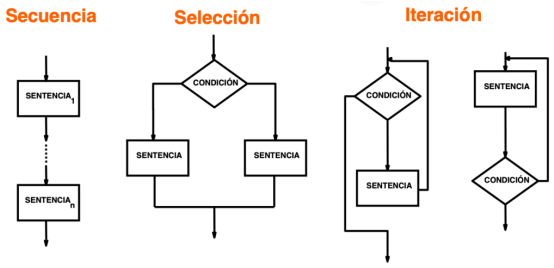
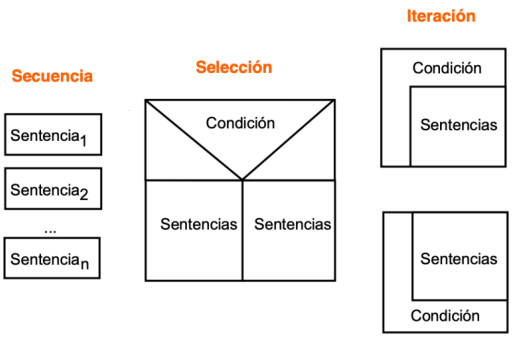
Modelado de procesos

El modelado de procesos se refiere a la representación visual de cómo se llevan a cabo las actividades en un sistema. En el análisis estructurado, se utiliza el Diagrama de Flujo de Datos (DFD), que muestra los procesos que transforman y manipulan flujos de datos. El DFD es simple y fácil de entender.

La explosión de un proceso implica desglosar un proceso padre en un nuevo DFD con más detalles. Las normas a seguir incluyen numerar los procesos hijos y mantener un DFD equilibrado.

La especificación de procesos es la descripción de la lógica interna de los procesos de último nivel, y se utiliza para definir cómo se transforman las entradas en salidas. Las herramientas que se utilizan para especificar los procesos incluyen la descripción narrativa, el lenguaje estructurado o el pseudocódigo, los árboles de decisión, las tablas de decisión, los diagramas de flujo y los diagramas de Nassi-Schneiderman.

El pseudocódigo se utiliza para especificar los detalles de un procedimiento y es muy preciso. Los árboles de decisión son útiles cuando el número de acciones es reducido, mientras que las tablas de decisión son recomendadas cuando hay muchas combinaciones y se quiere revisar en detalle. Los diagramas de flujo son una de las primeras notaciones de modelado para algoritmos y código, y se utilizan para representar la secuencia, selección e iteración de las acciones. Los diagramas de Nassi-Schneiderman tienen un nivel de detalle similar al pseudocódigo.

Especificación de Proceso Diagrama de Flujo	Especificación de Proceso Diagramas de Nassi-Schneiderman
	

implementación -> construir

pruebas -> pruebas

mantenimiento -> arreglar errores

LA NATURALEZA DEL SOFTWARE

¿Qué es el Software?

- Instrucciones, estructuras de datos y documentación que brindan características, función y rendimiento deseado.
- Desarrollado o modificado con el intelecto.
- No se desgasta y la mayoría del software sigue siendo personalizado.

Diseño y mantenimiento del software

- Mejorar el diseño para reducir el deterioro del software.
- Los métodos de la ingeniería de software reducen la magnitud de los picos y pendientes de la curva real.

Dominios de aplicación del software

- Software de sistema, aplicación, ingeniería/científico, integrado, línea de productos, WebApps, IA.
- Nuevas categorías: computación de mundo abierto, ubicua, netsourcing, código abierto, procesamiento de datos.

Software heredado

- Debe cambiar para adaptarse a nuevas necesidades informáticas, tecnológicas y de negocio.
- Puede necesitar ampliarse, rediseñarse o hacerse interoperable con otros sistemas o bases de datos más modernas.

LA NATURALEZA ÚNICA DE LAS APLICACIONES WEB

Características de las Aplicaciones Web

- **Uso intensivo de redes** para atender a una comunidad diversa de clientes.
- **Concurrencia** de un gran número de usuarios con patrones de uso variados.
- **Carga impredecible** con cambios en el número de usuarios de forma drástica de un día a otro.
- **Rendimiento** rápido para evitar que los usuarios se vayan a otra aplicación.
- **Disponibilidad** las 24 horas del día, los 365 días del año para usuarios de todo el mundo.
- **Orientadas a los datos**, principalmente para acceder a información de bases de datos externas.
- Contenido sensible, con calidad y naturaleza estética importante para la calidad de la aplicación.
- **Evolución continua**, con actualizaciones frecuentes en el contenido.

- **Inmediatez** en llegar al mercado con plazos de algunos días o semanas.
- **Seguridad** estricta para proteger contenido sensible y garantizar la transmisión segura de datos.
- **Estética** atractiva para el éxito de la aplicación, especialmente en aplicaciones comerciales o de ventas.

INGENIERÍA DE SOFTWARE

- **Realidades de la Ingeniería de Software**: se debe comprender el problema antes de desarrollar una solución, el diseño es fundamental, el software debe ser de alta calidad y mantenible.
- **Definición de la Ingeniería de Software**: establecimiento y uso de principios sólidos de ingeniería para obtener económicamente software confiable y eficiente en máquinas reales.
- **Definición del IEEE**: aplicación sistemática, disciplinada y cuantificable al desarrollo, operación y mantenimiento de software, y estudio de enfoques según la definición anterior.
- **Compromiso** con la calidad: la cultura de mejora continua es esencial en la ingeniería de software.
- **Proceso**: el proceso de ingeniería de software une las capas de la tecnología y permite el desarrollo racional y oportuno del software.
- **Métodos**: incluyen un conjunto amplio de tareas como comunicación, análisis de requerimientos, modelación de diseño, construcción del programa, pruebas y apoyo, y se basan en principios fundamentales.
- **Herramientas**: proporcionan un apoyo automatizado o semiautomatizado para el proceso y los métodos, y cuando se integran forman un sistema de ingeniería de software asistido por computadora.

El proceso del software

- Un proceso es un conjunto de actividades, acciones y tareas que se ejecutan cuando se va a crear un producto del trabajo.
- Una actividad busca lograr un objetivo amplio y se desarrolla sin importar el dominio de la aplicación, tamaño del proyecto, complejidad del esfuerzo o grado de rigor con el que se usará la ingeniería de software.
- Una acción es un conjunto de tareas que producen un producto importante del trabajo.
- Una tarea se centra en un objetivo pequeño pero bien definido y produce un resultado tangible.
- El proceso impone consistencia y estructura al conjunto de actividades.
- No es un procedimiento, sino un conjunto organizado de ellos.
- Permite capturar y transmitir experiencia de un proyecto a proyectos futuros.
- El proceso permite/sugiere técnicas y herramientas.
- La estructura del proceso guía la acción, permitiendo examinar, entender, controlar y mejorar actividades.
- El proceso se puede ver como una caja negra o en una visión transparente.

Framework de actividades

- Comunicación: antes de comenzar cualquier trabajo técnico, es importante comunicarse y colaborar con el cliente y otros participantes.
- Planificación: crea un "mapa" que guía al equipo en el proyecto.
- Modelado: se crean modelos para entender mejor los requisitos del software y el diseño que los satisfará.
- Implementación: combina la generación de código y las pruebas necesarias.
- Despliegue: se entrega el software al consumidor que lo evalúa y da retroalimentación.
- Actividades sombrilla
    - Son complementarias a las actividades estructurales del proceso.
    - Se aplican a lo largo del proyecto y ayudan al equipo a administrar y controlar el avance, la calidad, el cambio y el riesgo.
    - Se centran en la administración, el seguimiento y el control del proyecto.
- Seguimiento y Control del Proyecto de Software: permite evaluar el progreso del equipo de software comparándolo con el plan del proyecto y tomar acciones necesarias para cumplir con la programación de actividades.
- Administración del riesgo: evalúa los riesgos que pueden afectar el resultado del proyecto o la calidad del producto.
- Aseguramiento de la calidad del software: define y ejecuta las actividades necesarias para garantizar la calidad del software.
- Revisiones técnicas: evalúa los productos del trabajo de ingeniería de software para detectar y eliminar errores antes de que se propaguen a la siguiente actividad.
- Medición: define y recopila mediciones del proceso, proyecto y producto para ayudar al equipo a entregar software que satisfaga las necesidades de los participantes.
- Administración de la configuración del software: administra los efectos del cambio a lo largo del proceso del software.
- Administración de la reutilización: define criterios para volver a usar el producto del trabajo, incluidos los componentes del software, y establece mecanismos para obtener componentes reutilizables.
- Preparación y producción del producto del trabajo: agrupa las actividades necesarias para crear productos del trabajo, como modelos, documentos, registros, formatos y listas.

Adaptación de un modelo de proceso

- El proceso de ingeniería de software debe ser ágil y adaptable.
- La adaptación del proceso es esencial para el éxito del proyecto.
- Factores que se consideran en la adaptación del proceso: flujo de actividades, grado de definición de las tareas, identificación y requerimientos de los productos, aseguramiento de la calidad, seguimiento y control del proyecto, detalle y rigor en la descripción del proceso, involucramiento del cliente y autonomía del equipo, y prescripción de la organización y roles del equipo.

La esencia de la práctica

- George Polya describe la esencia de la solución de problemas y la práctica de ingeniería de software.
- Pasos para solucionar problemas: comprender el problema, planear la solución, ejecutar el plan y examinar la precisión del resultado.
- En el primer paso, es importante entender el problema haciendo preguntas sobre los participantes, incógnitas, posibilidad de fraccionarlo y representarlo gráficamente.
- En el segundo paso, se debe hacer un pequeño diseño y considerar si hay patrones o soluciones reutilizables.
- En el tercer paso, se ejecuta el plan y se revisa que la solución se ajuste al diseño y se hagan pruebas para validar la corrección del algoritmo.
- En el cuarto paso, se examina el resultado y se prueban todas las partes de la solución, y se validan contra los requerimientos de los participantes.

Principios generales de la ingeniería de software propuestos por David Hooker

- La razón por la que todo existe
- KISS (¡Mantenlo simple, estúpido!) o MSE (Mantéalo sencillo, estúpido)
- Mantener la visión
- Lo que tú produces, otros lo consumirán
- Estar abierto al futuro
- Planifique con anticipación para la reutilización
- ¡Piensa!

Explicación de los principios:

- Primer principio: Un sistema de software existe para dar valor a los usuarios, por lo que todas las decisiones deben tomarse teniendo esto en mente.
- Segundo principio: El diseño de software debe ser tan simple como sea posible, pero no más, para conseguir un sistema fácil de mantener y menos propenso a errores.
- Tercer principio: Una visión clara es esencial para el éxito de un proyecto de software, ya que sin ella, el sistema corre el riesgo de convertirse en una urdimbre de diseños incompatibles.
- Cuarto principio: Otros consumirán lo que usted produce, por lo que debe establecer especificaciones, diseñar e implementar con la seguridad de que alguien más tendrá que entenderlo.
- Quinto principio: Un sistema con larga vida útil tiene más valor, por lo que los sistemas deben ser fáciles de adaptar a cambios futuros.
- Sexto principio: La reutilización ahorra tiempo y esfuerzo, por lo que lograr un alto nivel de reutilización es una meta difícil pero importante al desarrollar un sistema de software.
- Séptimo principio: Siempre piense en el proceso de diseño del software y tome decisiones reflexivas.