INSTITUTO KHIPU

Semestre III

Sesión 03

PROGRAMA DE ESTUDIOS

DESARROLLO DE SISTEMAS DE INFORMACIÓN

ANÁLISIS Y DISEÑO DE SISTEMAS

Tema:

CICLO DE VIDA DE DESARROLLO DE SOFTWARE

CICLO DE VIDA DE DESARROLLO DE SOFTWARE – ANALISIS.

El ciclo de vida del desarrollo de software (SDLC) es el proceso paso a paso de crear un nuevo software. Los ingenieros y desarrolladores de software utilizan modelos de ciclo de vida para mapear las fases del desarrollo de software, asegurando que el proceso sea eficiente y rentable. Este blog discutirá cada paso del SDLC en profundidad y describirá diferentes modelos de uso común.

¿Qué es el Ciclo de Vida del Desarrollo de Software?

El ciclo de vida del desarrollo de software abarca el desarrollo de nuevo software desde la etapa de planificación inicial hasta la implementación y el mantenimiento a largo plazo. Es una herramienta de mapeo que ayuda a los desarrolladores de software a medida que crean un nuevo software.

Diferentes metodologías SDLC se adaptarán mejor a diferentes tipos de proyectos. Cada modelo enfatiza los diferentes beneficios de SDLC.

¿Por qué es importante SDLC?

El software suele ser desarrollado por equipos de ingenieros y desarrolladores. Es fundamental que cada persona que trabaje en el proyecto siga el mismo proceso. Sin esta consistencia, sería casi imposible desarrollar un software exitoso y entregarlo al cliente a tiempo.

SDLC le da a cada proyecto un framework individualizado para trabajar dentro. Mantiene a todo el equipo de desarrollo en la misma página para que el proceso permanezca ordenado y eficiente y resulte en productos de software de alta calidad y bajo costo.

7 fases del Ciclo de Vida del Desarrollo de Software

Hay siete fases del ciclo de vida de desarrollo de software que se siguen en cada uno de los diferentes modelos. Cada fase cumple una función distinta y juntas proporcionan un framework de programación integral para el desarrollo de software eficiente.

Hay requisitos previos que deben cumplirse antes de que cada fase pueda comenzar o finalizar. Estos se conocen como puntos de entrada y puntos de salida. Todos los miembros del equipo deben seguir las fases de SDLC en orden secuencial para garantizar que el software se complete de manera precisa, rentable y antes de la fecha límite del cliente.

1. **Planificación y Análisis de Requerimientos**

La fase de planificación y análisis de requisitos es el primer paso en el desarrollo de software. Aquí es cuando los miembros superiores del equipo recopilan información sobre el software que se desarrollará. Hablan con el cliente para conocer las intenciones del proyecto y luego identificar posibles riesgos, problemas y oportunidades.

Los miembros del equipo a menudo reciben aportes de múltiples partes interesadas y expertos de la industria. En esta fase, el equipo determina qué costos y recursos se requerirán para completar el proyecto.

2. **Definir requisitos**

Una vez que se completa el análisis de requisitos, los requisitos del programa de software se documentan en un documento de especificación de requisitos de software. Deben ser aceptados por las partes interesadas antes de que el equipo de desarrollo pueda comenzar el proceso de diseño.

3. **Diseño y Prototipado**

Durante esta fase, toda la información recopilada en los dos pasos anteriores se reúne cuando el equipo comienza a diseñar la arquitectura del software. Se pueden presentar múltiples diseños para que el equipo pueda decidir cuál es el mejor para seguir adelante.

4. **Desarrollo de software**

Esta fase es la más larga de todas las fases SDLC. Es donde los desarrolladores comienzan a programar y dar vida al proyecto. Puedes hacerlo usando una variedad de herramientas como compiladores, intérpretes y depuradores. El lenguaje de programación que utilices dependerá de los requisitos del software.

5. **Pruebas de software**

Una vez que se completa el desarrollo del software, debe probarse para asegurarse de que cumple con los requisitos que se identificaron en las fases anteriores. Las pruebas generalmente las realizan equipos de control de calidad y pruebas de software. Comprueban que no haya defectos en el código y que el software funcione como se espera.

Si se encuentran defectos durante las pruebas, el código se envía de vuelta al equipo de desarrollo para que lo arreglen. Esta fase continúa hasta que se corrigen todos los errores.

6. **Implementación del software**

Durante la fase de implementación, el software se entrega al cliente y se pone en uso.

7. **Operaciones y Mantenimiento**

Una vez que se implementa el software, el trabajo no ha terminado. Es probable que surjan problemas que no se detectaron durante la fase de prueba. Durante la fase de mantenimiento continuo, los problemas que surgen se solucionan mediante

actualizaciones y parches de software. También se pueden agregar nuevas características a medida que avanza la tecnología.

¿Qué es la Fase de Análisis?

La fase de análisis es una etapa crucial en el desarrollo de software donde se define con precisión qué es lo que el sistema debe hacer. Es como el arquitecto que diseña un edificio antes de construirlo; en este caso, estamos diseñando el software.

Objetivos principales:

Comprender a fondo los requisitos del usuario: ¿Qué necesita el usuario? ¿Cómo utilizará el sistema?

Definir la funcionalidad del sistema: ¿Qué tareas realizará el sistema? ¿Cuáles son sus límites?

Crear una documentación clara y concisa: Esta documentación servirá como base para las siguientes fases del desarrollo.

Un Método Práctico: El Análisis Orientado a Objetos (OOA)

El Análisis Orientado a Objetos (OOA) es un método popular que se alinea bien con la programación orientada a objetos. En este enfoque, el sistema se modela como una colección de objetos que interactúan entre sí.

Pasos clave en el OOA:

Identificación de objetos:

¿Qué son los objetos? Son entidades del mundo real o abstractas que tienen propiedades (atributos) y comportamientos (métodos).

Ejemplos: Un cliente, un producto, una orden, una ventana, un botón.

Definición de atributos:

¿Qué características tiene cada objeto?

Ejemplo: Un cliente puede tener atributos como nombre, dirección, teléfono, historial de compras.

Definición de métodos:

¿Qué acciones puede realizar cada objeto?

Ejemplo: Un cliente puede realizar las acciones de "comprar", "ver historial", "modificar datos".

Identificación de relaciones:

¿Cómo se relacionan los objetos entre sí?

Relaciones comunes: asociación (un cliente puede tener muchas órdenes), agregación (una orden tiene muchos productos), composición (una ventana es parte de una aplicación).

Creación de diagramas UML:

Los diagramas de clases UML son una herramienta visual para representar las clases, sus atributos, métodos y relaciones.

Herramientas y Técnicas Adicionales

Reuniones con usuarios: Realizar entrevistas, encuestas y talleres para recopilar requisitos.

Casos de uso: Describir la interacción entre el usuario y el sistema en diferentes escenarios.

Prototipos: Crear versiones simplificadas del sistema para validar los requisitos con los usuarios.

Herramientas CASE: Software que ayuda a crear y gestionar modelos de sistemas.

Ejemplo Práctico: Una Tienda en Línea

Objetos: Cliente, Producto, Orden, Carrito de compras **Atributos:** Nombre, dirección, precio, cantidad **Métodos:** Comprar, agregar al carrito, calcular total **Relaciones:** Un cliente puede realizar muchas órdenes, una orden tiene muchos productos.

Diagrama de Clases UML (Ejemplo):

Beneficios del OOA

Mayor claridad: Los diagramas UML proporcionan una representación visual y fácil de entender del sistema.

Reutilización: Los objetos y clases pueden reutilizarse en diferentes partes del sistema.

Facilidad de mantenimiento: Los cambios en los requisitos se pueden implementar más fácilmente al modificar los objetos y sus relaciones.

Conclusiones

La fase de análisis es fundamental para el éxito de un proyecto de software. El OOA es un método eficaz para modelar sistemas complejos de manera clara y concisa. Al seguir estos pasos y utilizar las herramientas adecuadas, los ingenieros de software pueden garantizar que el sistema desarrollado cumpla con los requisitos del usuario y sea fácil de mantener.

Actividades para los Estudiantes:

Ejercicio: Crear un diagrama de clases UML para un sistema de gestión de biblioteca.

Estudio de caso: Analizar un sistema existente y crear un modelo de objetos.

Presentación: Explicar los beneficios del OOA y comparar con otros métodos de análisis.

FUENTE:

- https://www.uml.org/
- https://www.codingdojo.la/2023/06/16/guia-del-ciclo-de-vida-del-desarrollo-de-software/
- https://aws.amazon.com/es/what-is/sdlc/
- BURCH, John; GRUDNISKY, Gary. "Diseño de Sistemas de Información", Grupo Noriega editores.
- SENN, James A. "Análisis y diseño de sistemas de información", 2da. ed., McGraw-Hill.


INSTITUTO
KHIPU