CICLO FORMATIVO DE GRADO SUPERIOR - TÉCNICO EN ADMINISTRACIÓN DE SISTEMAS INFORMÁTICOS EN REDES

FUNDAMENTOS DE HARDWARE

Ciclo de vida de un Software

Nombre y apellidos: Wuke Zhang

- **1-ASIR**

Realizar un breve trabajo de investigación, respondiendo a las siguientes preguntas:

1. Investiga sobre las distintas etapas que componen el ciclo de vida de un software. Explica la importancia de cada una de estas etapas.

2. Analiza cómo se realiza la planificación a la hora de desarrollar un software incluyendo la definición de objetivos, requisitos y plazos.

3. Investiga cómo se realiza la verificación y validación de un software, incluyendo explicaciones sobre las pruebas de unidad, integración y aceptación.

4. Analiza la importancia del ciclo de vida del software en la calidad del producto final y en la satisfacción del cliente.

El trabajo deberá tener:

- Extensión mínima de 6 páginas, excluyendo portada, índice y bibliografía.
- Letra Arial 12, interlineado 1,15.
- Se penalizará notablemente no incluir fuentes bibliográficas variadas o que estas no concuerden con los contenidos del trabajo.

Indice:

Portada..... 1

Indice..... 2

Introduccion
..... 3

Contenido
..... 4

Bibliografia
..... 5

1. Investiga sobre las distintas etapas que componen el ciclo de vida de un software. Explica la importancia de cada una de estas etapas.

Definición de necesidades:
Las necesidades del software son los requisitos y objetivos que deben ser cumplidos a lo largo de todo el ciclo de vida del desarrollo, asegurando que el producto final sea funcional, eficiente y satisfaga las necesidades de los usuarios y las partes interesadas.
Las necesidades pueden ser tanto funcionales como no funcionales. Las necesidades funcionales se refieren a las acciones específicas que el software debe realizar, como procesar datos, generar informes o permitir interacciones con el usuario. Por otro lado, las necesidades no funcionales abarcan aspectos como el rendimiento, la seguridad, la usabilidad y la escalabilidad del software.

Análisis:
Por supuesto, hay que averiguar qué es exactamente lo que tiene que hacer el software. Por eso, la etapa de análisis en el ciclo de vida del software corresponde al proceso a través del cual se intenta descubrir qué es lo que realmente se necesita y se llega a una comprensión adecuada de los requerimientos del sistema (las características que el sistema debe poseer).

Diseño
En esta fase se estudian posibles opciones de implementación para el software que hay que construir, así como decidir la estructura general del mismo. El diseño es una etapa compleja y su proceso debe realizarse de manera iterativa.
Es posible que la solución inicial no sea la más adecuada, por lo que en tal caso hay que refinarla. No obstante, hay catálogos de patrones de diseño muy útiles que recogen errores que otros han cometido para no caer en la misma trampa.

Codificación
En esta fase hay que elegir las herramientas adecuadas, un entorno de desarrollo que facilite el trabajo y un lenguaje de programación apropiado para el tipo de software a construir. Esta elección dependerá tanto de las decisiones de diseño tomadas como del entorno en el que el software deba funcionar.
Al programar, hay que intentar que el código no sea indescifrable siguiendo distintas pautas como las siguientes:

- Evitar bloques de control no estructurados.
- Identificar correctamente las variables y su alcance.
- Elegir algoritmos y estructuras de datos adecuadas para el problema.
- Mantener la lógica de la aplicación lo más sencilla posible.
- Documentar y comentar adecuadamente el código de los programas.
- Facilitar la interpretación visual del código utilizando reglas de formato de código previamente consensuadas en el equipo de desarrollo.

También hay que tener en cuenta la adquisición de recursos necesarios para que el software funcione, además de desarrollar casos de prueba para comprobar el funcionamiento del mismo según se vaya programando.

Pruebas
Como errar es humano, la fase de pruebas del ciclo de vida del software busca detectar los fallos cometidos en las etapas anteriores para corregirlos. Por supuesto, lo ideal es hacerlo antes de que el usuario final se los encuentre. Se dice que una prueba es un éxito si se detecta algún error.

Instalación o despliegue y validación
La siguiente fase es poner el software en funcionamiento, por lo que hay que planificar el entorno teniendo en cuenta las dependencias existentes entre los diferentes componentes del mismo.
Es posible que haya componentes que funcionen correctamente por separado, pero que al combinarlos provoquen problemas. Por ello, hay que usar combinaciones conocidas que no causen problemas de compatibilidad.

Evolución y mantenimiento
Esta es una de las fases más importantes del ciclo de vida de desarrollo del software. Puesto que el software ni se rompe ni se desgasta con el uso, su mantenimiento incluye tres puntos diferenciados:

- Eliminar los defectos detectados durante su vida útil (mantenimiento correctivo).
- Adaptarlo a nuevas necesidades (mantenimiento adaptativo).
- Añadirle nuevas funcionalidades (mantenimiento perfectivo).

Aunque suene contradictorio, cuanto mejor es el software más tiempo hay que invertir en su mantenimiento. La principal razón es que se usará más (incluso de formas que no se habían previsto) y, por ende, habrá más propuestas de mejoras.

2. Analiza cómo se realiza la planificación a la hora de desarrollar un software incluyendo la definición de objetivos, requisitos y plazos.

La planificación en el desarrollo de software es un proceso crítico que abarca varios aspectos clave para asegurar el éxito del proyecto. A continuación, ampliaré los pasos involucrados en la planificación y proporcionaré más detalles:

1. **Definición de Objetivos**:
    - Antes de comenzar cualquier proyecto, es fundamental establecer objetivos claros. Estos pueden incluir metas específicas, como funcionalidades a implementar, mejoras en la eficiencia o la satisfacción del cliente.
    - Los objetivos deben ser realistas, medibles y alineados con las

necesidades del negocio o del cliente.

2. **Requisitos**:
    - La fase de definición de requisitos implica comprender las necesidades del cliente o usuario final. Esto incluye entrevistas, encuestas y análisis de documentos.
    - Los requisitos deben ser detallados y específicos. Deben cubrir funcionalidades, restricciones, interfaces y expectativas de rendimiento.
3. **Estimación de Recursos y Tiempo**:
    - La estimación precisa de recursos (como personal, hardware y software) y tiempo es crucial. Se deben considerar factores como la complejidad del proyecto, la experiencia del equipo y la disponibilidad de recursos.
    - Las técnicas de estimación, como la estimación paramétrica o la descomposición de tareas, ayudan a calcular los esfuerzos necesarios.
4. **Planificación Temporal**:
    - Crear un cronograma realista es esencial. Se deben asignar tareas, establecer hitos y considerar dependencias entre actividades.
    - Herramientas como el diagrama de Gantt o el método PERT pueden ayudar a visualizar y gestionar el tiempo.
5. **Gestión de Riesgos**:
    - Identificar y evaluar riesgos potenciales es parte integral de la planificación. Esto incluye riesgos técnicos, de recursos, de comunicación y otros.
    - Se deben desarrollar estrategias para mitigar o responder a estos riesgos.
6. **Comunicación y Colaboración**:
    - Mantener una comunicación efectiva con el cliente, el equipo y otras partes interesadas es crucial. Esto ayuda a comprender cambios en los requisitos y a adaptarse según sea necesario.
    - La colaboración temprana y continua garantiza que todos estén alineados con los objetivos del proyecto.
7. **Actualización Continua**:
    - La planificación no es estática. A medida que avanza el proyecto, se deben revisar y ajustar los planes según los cambios en los requisitos o las circunstancias.
    - La adaptabilidad es clave para mantener el proyecto en el camino correcto.

En resumen, la planificación de proyectos de desarrollo de software es un proceso iterativo y colaborativo. Requiere una combinación de habilidades técnicas, comunicación efectiva y enfoque sistemático para lograr resultados exitosos. Al seguir estos pasos, los equipos pueden anticipar desafíos, optimizar recursos y cumplir con los plazos establecidos.

3. Investiga cómo se realiza la verificación y validación de un software, incluyendo explicaciones sobre las pruebas de unidad, integración y aceptación.

La **verificación y validación** son dos procesos fundamentales en el desarrollo de software que aseguran que el producto final cumpla con los requisitos y expectativas del usuario. Aquí tienes una explicación detallada de estos procesos y las pruebas involucradas:

Verificación de Software

La verificación se centra en el proceso de desarrollo, asegurándose de que el software se construye correctamente. Incluye actividades como:

- **Revisión de Código**: Evaluación del código fuente para verificar que sigue las pautas de programación y las mejores prácticas.
- **Pruebas de Unidad**: Pruebas a nivel de módulos o unidades de código para identificar errores en funciones específicas.
- **Pruebas de Integración**: Evaluación de cómo los distintos módulos o componentes interactúan entre sí.
- **Análisis Estático**: Uso de herramientas para buscar problemas en el código sin ejecutar el programa.
- **Verificación de Diseño**: Revisión de la arquitectura y diseño del software para asegurar que cumple con los requisitos.

Validación de Software

La validación se enfoca en confirmar que el software cumple con las necesidades del usuario final. Incluye actividades como:

- **Pruebas Funcionales**: Verificación de que el software realiza sus funciones según los requisitos especificados.
- **Validación de Requisitos**: Confirmación de que los requisitos del software son claros, completos y alineados con las necesidades del usuario.

- **Validación de Cumplimiento Normativo**: Asegurarse de que el software cumple con estándares y regulaciones aplicables.
- **Validación de Documentación**: Verificación de que la documentación del software es precisa y está actualizada.

Pruebas de Unidad

Son pruebas de bajo nivel que se realizan cerca de la fuente de la aplicación, probando métodos y funciones individuales de las clases, componentes o módulos.

Pruebas de Integración

Verifican que los distintos módulos o servicios utilizados por la aplicación funcionan bien en conjunto, como la interacción con la base de datos o entre microservicios.

Pruebas de Aceptación

Estas pruebas verifican si todo el sistema funciona según lo previsto y si cumple con las especificaciones y requisitos del usuario final.

La combinación de estas pruebas durante el ciclo de desarrollo del software es crucial para entregar un producto robusto, funcional y seguro.

4. Analiza la importancia del ciclo de vida del software en la calidad del producto final y en la satisfacción del cliente.

El ciclo de vida del desarrollo de software (CVDS) es fundamental para la calidad del producto final y la satisfacción del cliente por varias razones:

Estructura y Planificación: Proporciona una estructura que guía a los equipos de desarrollo a través de las fases necesarias para crear software de alta calidad, desde el análisis de requisitos hasta el mantenimiento1.

Calidad y Estándares: Asegura que el software cumpla con los estándares de calidad y eficiencia requeridos, lo que resulta en un producto final confiable y eficaz1.

Gestión de Cambios e Innovación: Facilita la adaptación a nuevas tecnologías,

metodologías y herramientas emergentes, lo que permite a las empresas mantenerse competitivas y responder a las necesidades cambiantes del mercado1.

Satisfacción del Cliente: Al seguir un proceso detallado, las empresas pueden cumplir mejor con los requisitos técnicos y las expectativas del cliente, lo que lleva a una mayor satisfacción del usuario final2.

Mejora Continua: El CVDS incluye fases de prueba y mantenimiento que permiten iterar y mejorar el software basándose en el feedback de los usuarios, asegurando así que el producto evolucione y se mantenga relevante3.

Bibliografía:

Ciclo de vida del Desarrollo de Software: Fases y ejemplos - Vidasoft

https://informatecdigital.com/desarrollo/las-etapas-del-desarrollo-de-software-una-guia-completa/

https://vidasoft.es/blog/desarrollo-producto/ciclo-de-vida-del-desarrollo-de-software-fases-ejemplos-modelos-y-mejores-practicas/

https://intelequia.com/es/blog/post/ciclo-de-vida-del-software-todo-lo-que-necesitas-saber

https://www.ctr.unican.es/asignaturas/Ingenieria_Software_4_F/Doc/M7_09_VerificacionValidacion-2011.pdf

https://es.linkedin.com/pulse/la-planificaci%C3%B3n-de-un-proyecto-software-

https://intelequia.com/es/blog/post/ciclo-de-vida-del-software-todo-lo-que-necesitas-saber

https://html.rincondelvago.com/el-ciclo-de-vida-del-software.html

https://chat.openai.com/share/155d082d-b505-41ff-bb8b-10278a02af26