Nombre del alumno:

Audelí Joachín Velázquez

Nombre del profesor:

Lic. Lepe Arriaga Icel Bernardo

Nombre del trabajo:

Ensayo

Materia:

Ingeniería en software

Licenciatura:

Ingeniería en sistemas computacionales

Grado: octavo cuatrimestre

Grupo: “A”

Frontera Comalapa Chiapas a 06 de marzo de 2022

UNIDAD I

FUNDAMENTOS DE LA INGENIERÍA DEL SOFTWARE

Frontera Comalapa Chiapas a 06 de marzo de 2022

INTRODUCCION

En esta ocasión veremos temas como el desarrollo de un software, todo el proceso que se debe de hacer en la creación de este, por lo que conoceremos las ventajas y desventajas de cada uno de los modelos así como los prototipos y las evoluciones.

Frontera Comalapa Chiapas a 06 de marzo de 2022

FUNDAMENTOS DE LA INGENIERÍA DEL SOFTWARE

DEFINICIÓN Y OBJETIVOS DE LA INGENIERÍA DEL SOFTWARE.

El objetivo principal es un enfoque sistemático, disciplinado y cuantificable aun que en otras palabras es la aplicación de ingeniería al software, surge en el año 1968 por la resolución de problemas y crisis en ese momento del software porque cuando se terminaba el hardware el software no estaba completo. Esto da como resultado el surgimiento de herramientas y procedimientos para apoyar a la ingeniería del software los cuales son las siguientes:

- Mejorar la calidad de los productos de software.
- Aumentar la productividad y el trabajo de los ingenieros de software.
- Utilidad.
- Facilitar el control en el proceso de desarrollo de software.
- Suministrar a los desarrolladores las bases para construir software de alta calidad en una forma eficiente.

CARACTERÍSTICAS Y APLICACIONES DEL SOFTWARE.

Un producto se le puede juzgar por varias razones lo cuales una puede ser la forma de usar o también lo que puede ofrecer así como la satisfacción, los cuales serían los siguientes:

- Operacional
- Transicional
- Mantenimiento

Un software que se ha creado con buena ingeniería, debe tener los siguientes rasgos

En otras palabras podemos mencionar que la ingeniería en software viene de una las ramas de la ciencia de la computación que usa conceptos definidos para que el resultado de estos pueda producir software eficientes, duraderos, escalable, y accesibles a tiempo.

Es así que surge la necesidad de cambios en los requisitos que estos serían:

- Software de gran tamaño
- Escalabilidad
- Costes
- Naturaleza dinámica
- Gestión de calidad

EVOLUCIÓN HISTÓRICA DEL SOFTWARE.

Se le llama evolución a todos aquellos productos que usan principios y métodos de la ingeniería esto incluye el mantenimiento y actualizaciones hasta su terminación, en su realización pasa varios procesos los cuales pasan por los requisitos requeridos esto depende del gusto del cliente por los cambios que ellos sugieren, solo que hay desde cierto punto algunos detalles los cuales serían el cambio constante de la tecnología y un cambio desde cero llevaría tiempo y más inversión por lo que se optaría por el mismo solo que actualizado para no empezar desde cero.

Existen leyes en cuanto a la evolución, los cuales se dividen en tres: S-type' ('static-type', tipo estático), 'P-type' ('practical-type', tipo práctico) y 'E-type' ('embedded-type', tipo embebido o empotrado). De la misma forma existe la evolución del software 'e-type'

existiendo entonces las 8 leyes los cuales son: Cambio continuo, Complejidad creciente, Conservación de la familiaridad, Crecimiento continuado, Decremento de la calidad, Sistemas de retroalimentación, Autorregulación y Estabilidad organizacional.

PERSPECTIVA GENERAL DE LA INGENIERÍA DEL SOFTWARE

En un principio como en todo no se contaba con estándares o métodos sistemáticos pero a partir de la década de los 60 finales de 70 se caracteriza por estableces un producto que se distribución general y es ahí donde el mantenimiento del software cambian los requisitos de los usuarios y que en su mayoría era el mantenimiento era elevado es entonces que aparecen etapas y técnicas como la programación estructurada y las metodologías de diseño.

PROCESOS, MÉTODOS Y HERRAMIENTAS.

Enfocado a la ingeniería, se basa la estructura para el desarrollo y la accesibilidad de la producción de alta calidad de una forma costeable esto es lo que conocemos como método. Existen algunos métodos con variedades que como: descripción del modelo del sistema, guías en el proceso y técnicas.

MODELO CLÁSICO O LINEAL, MODELO EN CASCADA.

Cuando se crea un software tiene su proceso o también le podemos llamas ciclos de vida bien definidas, por lo que los SDLC incluye pasos como los siguientes: comunicación, requisitos del sistema, estudio de factibilidad, análisis del sistema, diseño software, códigos, pruebas, integración, implementación, operaciones y mantenimiento y por último la disposición. Mientas que el modelo cascada se basa en: requisitos del sistema, análisis del sistema, códigos, pruebas, implementación, operaciones y mantenimiento.

CONSTRUCCIÓN DE PROTOTIPOS.

Se basa en crear rápidamente algunas partes del proyecto para poder aclarar dudas e inquietudes para asegurar el desarrollo y que el cliente este de acuerdo porque de lo contrario poder modificarlo en sus momento, los prototipos de este son: escuchar al cliente, construir y revisar la maqueta (prototipo), el cliente prueba la maqueta y lo utiliza para refinar los requisitos del software. Una de las ventajas es que ofrece un mejor enfoque y la forma en la interacción humano-maquina. La desventaja puede ser que el prototipo sea lento, grande o también puede ser que no sea amena o que este en un lenguaje de programación inadecuado.

MODELOS EVOLUTIVOS.

Es interactivo y la accesibilidad de desarrollar versiones siempre más completas, debido a la mercadotecnia a veces no es posible esperar demasiado tiempo un producto completo por lo que se introducen versiones funcional limitada esto es con intención de calmar las tensiones. Existen dos modelos los cuales se describen a continuación.

- Modelos espiral

- Modelo concurrente.
- Modelo incremental

El modelo espiral:

Es un generador de modelos de procesos guiados por el riesgo que se emplea para conducir sistemas intensivos y con muchos usuarios. Se debe de tener en cuenta el riesgo que aparecen a la hora de desarrollar software y son: la planificación, análisis de riesgo e ingeniería que no es más que el siguiente nivel de desarrollo y por último la evaluación del cliente. La característica de este es que está formado por un enfoque cíclico y la utilización de puntos de fijación para asegurar el compromiso y que sean factibles y satisfactorias. Las ventajas son: reduce el riesgo de proyecto, incorpora objetos de calidad e integra el desarrollo con el mantenimiento. Las desventajas: genera mucho tiempo en el desarrollo del sistema, son costosos los modelos, requieren experiencia en la identificación de riesgo.

Modelo concurrente:

Se dirige más a las necesidades del usuario enfocándose a las decisiones y tareas que realice, las características principales son: expresar de manera esquematizada y organizada, cada actividad lleva procesos concurrentes, aplicar a la mayoría de tipos de desarrollo de software, es un módulo aplicable para el cliente soñador, está dirigido básicamente y esencialmente a las necesidades del usuario, es aplicable al cliente servidor.

Ventajas: Es excelente para proyectos en los que se conforman grupos de trabajo, proporciona una imagen exacta del estado actual de un proyecto, no restringe el proyecto a una secuencia de sucesos.

Desventaja: Si no se dan las condiciones específicas no se puede aplicar, Si no existe grupo de trabajo no se puede trabajar en este método, todas las actividades de red existen simultáneamente con otras, los sucesos generados dentro de una actividad, o en algún otro lado de la red de actividad, inician las transiciones entre los estados de otra actividad.

Modelo incremental

Esta combina filosofía y los modelos se basan en cascada. Los modelos se dividen en 4 partes: análisis, código, diseño y prueba. Las características lo podemos representar de la siguiente manera: Se evitan proyectos largos y se entrega “algo de valor” a los usuarios con cierta frecuencia, el usuario se involucra más frecuente, es muy difícil de evaluar el costo, también es difícil de aplicar a los sistemas transaccionales que tienden a ser integrado y trabajar como uno solo, requiere gestores experimentados y por último los errores no se detectan a tiempo.

Conclusión

Es muy importante tomar en cuenta lo anterior para evitar errores, los errores son bueno siempre y cuando no tenga algún costo, es por eso que debemos de tomar en cuenta los consejos para no tener errores que nos cuesten la cancelación del proyecto

Frontera Comalapa Chiapas a 06 de marzo de 2022

Bibliografía

Tomado de libro de consulta de la UDS

Ingeniería en software