Instituto Tecnológico Superior Zacatecas Occidente

Ingeniería En Sistemas Computacionales

Ingeniería de Software

Tema:

Desarrollo del ciclo de vida

Elaborado por:

Osiel Chávez Flores

Rubén Gómez Barrientos

Osiel Barrientos Ramírez

Fabián Armando Herrera Avalos

Jesús Agustín Juárez Guerrero

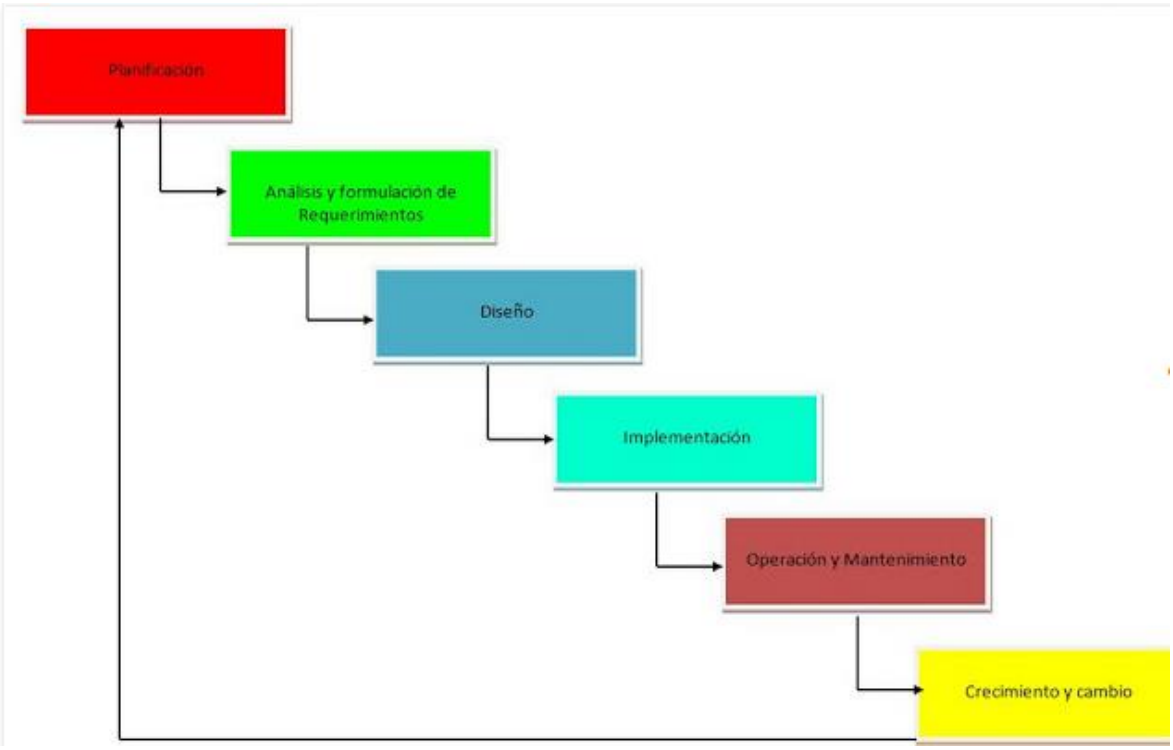
Docente:

I.S.C Ericka Jazmín Robles Gómez

Lugar y fecha de entrega:

Sombrerete Zacatecas a 11 de Marzo del 2015

Diagrama de ciclo de vida



Fases:

Planificación:

Realizar un software que pueda controlar el sistema de tutorías de manera optimizada, para lograr que sea más fácil para las personas encargadas llevar un control apropiado.

Análisis y Diseño de Requerimientos:

Este sistema será administrado por el o la encargado o encargada del área de tutorías de la institución, además que también podrá ser utilizado por los tutores y alumnos los cuales serán la base de dicho sistema.

Diseño:

Se tomara como diseño el mismo proceso que se está llevando en tutorías, con la única diferencia de que será de manera electrónica, con el propósito de tener un control más adecuado y seguro.

Implementación:
Realizar las pruebas pertinentes y verificar que se cumplen con las características identificadas.

Operación y Mantenimiento:

Se instalara dentro del Instituto Tecnológico Superior Zacatecas Occidente, tomando en cuenta que tal vez existan nuevas características que no han sido contempladas y/o características innecesarias, implicando la modificación del software para la adaptación de estas anomalías.

Crecimiento y cambio:
Se evalúa el software de modo que se determina si se puede emplear dentro de la nueva tecnología no afectando la integridad del mismo, de modo que si no es posible que exista una adaptación a lo nuevo, el proceso de diseño del software nuevamente se repite desde el principio.

Lista de entregables importantes por cada fase

Diagrama de procesos

Diagrama de Clases

Diagrama Entidad Relación

Diagrama Meta problema

Diagrama de Dominio

Etc.

Justificación de ciclo de vida

El modelo en cascada proporciona la flexibilidad para poder retroceder, de esta manera lograr buscar mejoras continuas, en caso de que algo salga mal en alguna fase regresarnos las veces que sean necesarias hasta lograr que se cumpla con el objetivo deseado.

Ventajas

El modelo en cascada puede ser apropiado, en general, para proyectos estables (especialmente los proyectos con requisitos no cambiantes) y donde es posible y probable que los diseñadores predigan totalmente áreas de problema del sistema y

produzcan un diseño correcto antes de que empiece la implementación. Funciona bien para proyectos pequeños donde los requisitos están bien entendidos.

Es un modelo en el que todo está bien organizado y no se mezclan las fases. Es simple y fácil de usar.

Debido a la rigidez del modelo es fácil de gestionar ya que cada fase tiene entregables específicos y un proceso de revisión. Las fases son procesadas y completadas de una vez.

Desventajas

En la vida real, un proyecto rara vez sigue una secuencia lineal, esto crea una mala implementación del modelo, lo cual hace que lo lleve al fracaso.

Los resultados y/o mejoras no son visibles progresivamente, el producto se ve cuando ya está finalizado, lo cual provoca una gran inseguridad por parte del cliente que quiere ir viendo los avances en el producto. Esto también implica el tener que tratar con requisitos que no se habían tomado en cuenta desde el principio, y que surgieron al momento de la implementación, lo cual provocará que haya que volver de nuevo a la fase de requisitos.

Hay muchas personas que argumentan que es una mala idea en la práctica, principalmente a causa de su creencia de que es imposible, para un proyecto no trivial, conseguir tener una fase del ciclo de vida del producto software perfecta antes de moverse a las siguientes fases. Por ejemplo, los clientes pueden no ser conscientes exactamente de los requisitos que quieren antes de ver un prototipo del trabajo; pueden cambiar los requisitos constantemente, y los diseñadores e implementadores pueden tener poco control sobre esto. Si los clientes cambian sus requisitos después de que el diseño está terminado, este diseño deberá ser modificado para acomodarse a los nuevos requisitos, invalidando una buena parte del esfuerzo.

Muchas veces se considera un modelo pobre para proyectos complejos, largos, orientados a objetos y por supuesto en aquellos en los que los requisitos tengan un riesgo de moderado a alto de cambiar. Genera altas cantidades de riesgos e incertidumbres.

Riesgos

Difícilmente un cliente va a establecer al principio todos los requisitos necesarios, por lo que provoca un gran atraso trabajando en este modelo, ya que este es muy restrictivo y no permite movilizarse entre fases.

Es inflexible y no motiva al cambio.

Poco apropiado para aplicaciones para la toma de decisiones.

Los usuarios tienen una participación limitada.

Conclusiones

Es el enfoque metodológico que ordena rigurosamente las etapas del ciclo de vida del software, de forma que el inicio de cada etapa debe esperar a la finalización de la inmediatamente anterior.

El modelo en cascada es un proceso de desarrollo secuencial, en el que el desarrollo se ve fluyendo hacia abajo (como una cascada) sobre las fases que componen el ciclo de vida.

Se cree que el modelo en cascada fue el primer modelo de proceso introducido y seguido ampliamente en la ingeniería el software. La innovación estuvo en la primera vez que la ingeniería del software fue dividida en fases separadas.

La primera descripción formal del modelo en cascada se cree que fue en un artículo publicado en 1970 por Winston W. Royce, aunque Royce no usó el término cascada en este artículo. Irónicamente, Royce estaba presentando este modelo como un ejemplo de modelo que no funcionaba, defectuoso.