cilred.com

CICLO DE VIDA DEL SOFTWARE & METODOLOGIAS DE DESARROLLO DE SOFTWARE

ING. EDUARDO CRUZ ROMERO

eduar14_cr@hotmail.com

cilred.com

CICLO DE VIDA DEL SOFTWARE

Para apreciar un poco más el problema del software, es necesario considerar su ciclo de vida, correspondiente a las diversas etapas por las cuales debe pasar un sistema, comenzando con la formulación de un problema, seguido por la especificación de requisitos, análisis, diseño, implementación o codificación, integración y pruebas del software, después viene una fase operacional durante la cual se mantiene y extiende el sistema.

Todo desarrollo de software incluye aspectos esenciales, como la planeación, correspondiente a las etapas de requisitos, análisis y diseño, junto con aspectos secundarios o accidentales, como codificación y pruebas. Según Brooks, existe una regla empírica (tbumb rule) que dice que para el desarrollo de un proyecto de software se debe asignar 1/3 del tiempo a la planeación, 1/6 a codificación, 1/4 a pruebas de componentes y 1/4 a pruebas del sistema, como se muestra en la figura. En otras palabras en total 2/3 a lo accidental, mientras que solo 1/3 a lo esencial, esto es preocupante ya que dedicamos menos tiempo a lo esencial que a lo accidental.

De manera adicional, la mayor parte de los avances en la productividad del software se han dado históricamente gracias a herramientas, ambientes y lenguajes de programación que reducen el esfuerzo en el desarrollo de las tareas secundarias o accidentales. La premisa de Brooks de mejorar por un orden de magnitud los aspectos esenciales del desarrollo de software significaría perfeccionar por 10 el esfuerzo dedicado a lo esencial, ósea de 1/10 a 10/10. Esto se lograría dedicando 10 veces más de tiempo a lo esencial que a lo accidental. ¿Entonces como hacer para mejorar tan radicalmente la productividad del software? Aunque no todos coinciden en la gravedad de la situación, la gran mayoría de los expertos en el área de ingeniería de software están de acuerdo en que se requiere de un proceso de desarrollo de software eficiente y sistemático. Por ejemplo, Ed Yourdon, autor de los libros Decline and Fall or the American Programmer y Rise and Resurrection of the American Programmer discute que aunque no hay un solo desarrollo que sea “bala de plata” se puede considerar varios aspectos que juntos puede dar ese incremento en el orden de magnitud. Estos aspectos incluyen seguir buenos procesos y metodologías, utilizar tecnología de objetos y herramientas avanzadas, incluir rehusó y métricas de software, y considerar de manera especial el aspecto humano (peopleware).

COMENTARIOS:

La vida de un proyecto de un software es prácticamente difícil de determinar ya que desde la concepción del proyecto se plantean objetivos, se programan tiempos y se definen las características necesarias que contendrá el software, todo lo anterior se basa en las diferentes etapas, que son: el análisis, diseño, codificación, pruebas, implementación y mantenimiento, de lo anterior se puede concluir que un proyecto de software debe ser planeado muy detalladamente ya que cualquier error o característica que se omita en la planeación afectara las siguientes etapas trayendo consigo un mayor gasto del ya planeado, mayor tiempo del establecido y un producto con características no deseadas, así mismo se debe considerarse que un proyecto de software aun después de ser liberado no tiene la madures deseada, por lo tanto puede que sea poco fiable, por eso deben existir actualizaciones o revisiones posteriores al proyecto, esto es necesario para corregir errores cometidos en las primera versión del producto que hayan pasado desapercibidos para el grupo de personas encargadas del mismo, como se menciona en el texto anterior lo mejor y lo más viable que se puede hacer, es realizar una buena planeación, es decir dedicarle más de 1/3 del tiempo asignado al proyecto a la planeación.

METODOLOGÍA DE DESARROLLO DE SOFTWARE

Una metodología de desarrollo de software se refiere a un framework que es usado para estructurar, planear y controlar el proceso de desarrollo en sistemas de información.

A lo largo del tiempo, una gran cantidad de métodos han sido desarrollados diferenciándose por su fortaleza y debilidad.

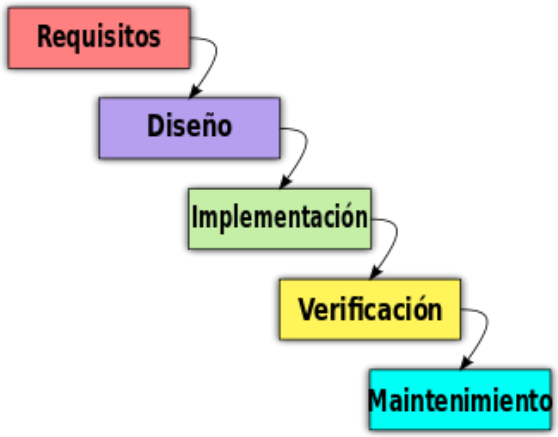
El framework para metodología de desarrollo de software consiste en:

- Una filosofía de desarrollo de programas de computación con el enfoque del proceso de desarrollo de software.
- Herramientas, modelos y métodos para asistir al proceso de desarrollo de software.

Estos frameworks son a menudo vinculados a algún tipo de organización, que además desarrolla, apoya el uso y promueve la metodología. La metodología es a menudo documentada en algún tipo de documentación formal.

CASCADA

Es un proceso secuencial de desarrollo en el que los pasos de desarrollo son vistos hacia abajo (como en una cascada de agua) a través de las fases de análisis de las necesidades, el diseño, implementación, pruebas (validación), la integración, y mantenimiento. La primera descripción formal del modelo de cascada se cita a menudo a un artículo publicado por Winston Royce W. en 1970, aunque Royce no utiliza el término "cascada".

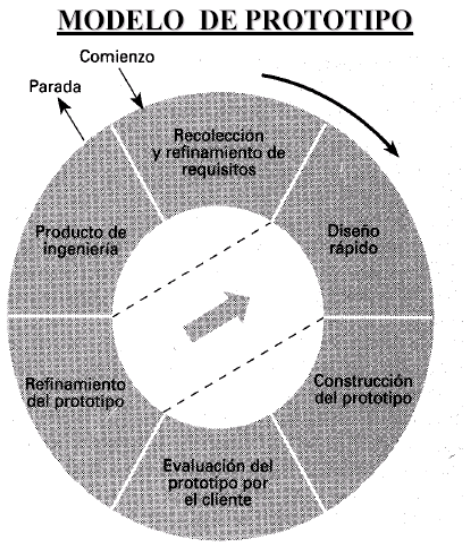


Los principios básicos del modelo de cascada son los siguientes:

- El proyecto está dividido en fases secuenciales, con cierta superposición y splashback aceptable entre fases.
- Se hace hincapié en la planificación, los horarios, fechas, presupuestos y ejecución de todo un sistema de una sola vez.
- Un estricto control se mantiene durante la vida del proyecto a través de la utilización de una amplia documentación escrita, así como a través de comentarios y aprobación / signoff por el usuario y la tecnología de la información de gestión al final de la mayoría de las fases antes de comenzar la próxima fase.

PROTOTIPADO

El Modelo de prototipos, en Ingeniería de software, pertenece a los modelos de desarrollo evolutivo. El prototipo debe ser construido en poco tiempo, usando los programas adecuados y no se debe utilizar muchos recursos.



El diseño rápido se centra en una representación de aquellos aspectos del software que serán visibles para el cliente o el usuario final. Este diseño conduce a la construcción de un prototipo, el cual es evaluado por el cliente para una retroalimentación; gracias a ésta se refinan los requisitos del software que se desarrollará. La interacción ocurre cuando el prototipo se ajusta para satisfacer las necesidades del cliente. Esto permite que al mismo tiempo el desarrollador entienda mejor lo que se debe hacer y el cliente vea resultados a corto plazo.

Etapas:

- Plan rápido
- Modelado, diseño rápido
- Construcción del Prototipo
- Desarrollo, entrega y retroalimentación
- Comunicación

Ventajas:

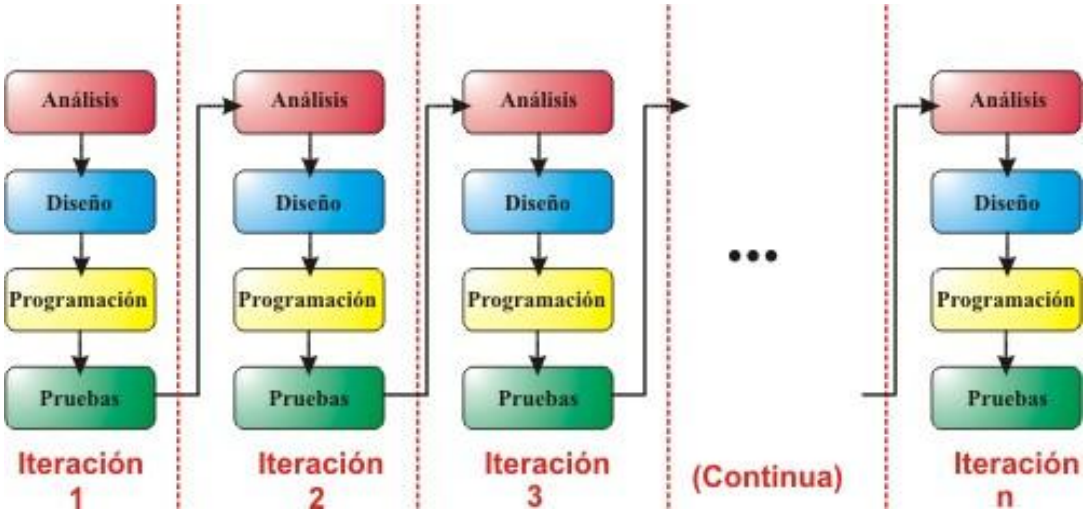
- Este modelo es útil cuando el cliente conoce los objetivos generales para el software, pero no identifica los requisitos detallados de entrada, procesamiento o salida.
- También ofrece un mejor enfoque cuando el responsable del desarrollo del software está inseguro de la eficacia de un algoritmo, de la adaptabilidad de un sistema operativo o de la forma que debería tomar la interacción humano-máquina.

INCREMENTAL

Provee una estrategia para controlar la complejidad y los riesgos, desarrollando una parte del producto software reservando el resto de aspectos para el futuro.

Los principios básicos son:

- Una serie de mini-Cascadas se llevan a cabo, donde todas las fases de la cascada modelo de desarrollo se han completado para una pequeña parte de los sistemas, antes de proceder a la próxima incremental
- Se definen los requisitos antes de proceder con lo evolutivo, se realiza un mini-Cascada de desarrollo de cada uno de los incrementos del sistema
- El concepto inicial de software, análisis de las necesidades, y el diseño de la arquitectura y colectiva básicas se definen utilizando el enfoque de cascada, seguida por iterativo de prototipos, que culmina en la instalación del prototipo final.



ESPIRAL

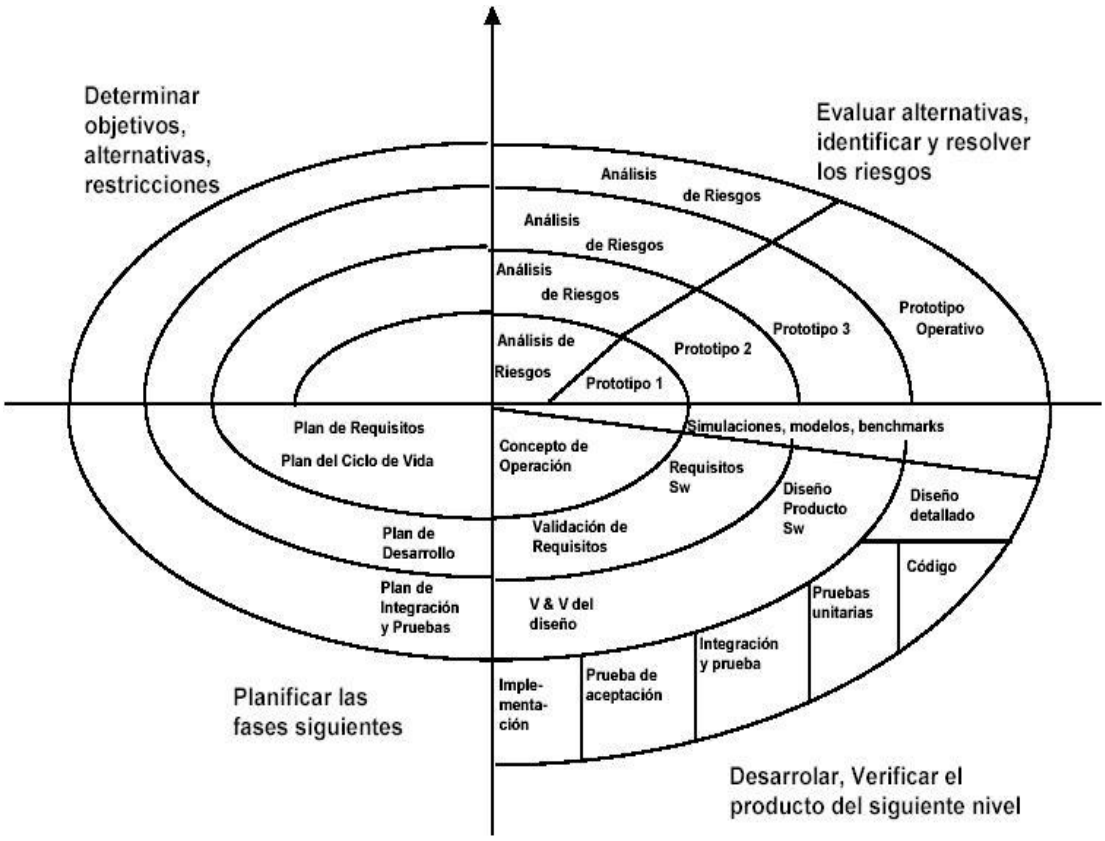
El desarrollo en espiral es un modelo de ciclo de vida del software definido por primera vez por Barry Boehm en 1986, utilizado generalmente en la Ingeniería de software. Las actividades de este modelo se conforman en una espiral, en la que cada bucle o iteración representa un conjunto de actividades. Las actividades no están fijadas a ninguna prioridad, sino que las siguientes se eligen en función del análisis de riesgo, comenzando por el bucle interior.

Los principios básicos son:

- La atención se centra en la evaluación y reducción del riesgo del proyecto dividiendo el proyecto en segmentos más pequeños y proporcionar más facilidad de cambio durante el proceso de desarrollo, así como ofrecer la oportunidad de evaluar los riesgos y con un peso de

la consideración de la continuación del proyecto durante todo el ciclo de vida.

- Cada viaje alrededor de la espiral atraviesa cuatro cuadrantes básicos:
  1. Determinar objetivos, alternativas, y desencadenantes de la iteración.
  2. Evaluar alternativas; Identificar y resolver los riesgos.
  3. Desarrollar y verificar los resultados de la iteración.
  4. Plan de la próxima iteración.
- Cada ciclo comienza con la identificación de los interesados y sus condiciones de ganancia, y termina con la revisión y examinación.



RAPID APPLICATION DEVELOPMENT (RAD)

El desarrollo rápido de aplicaciones (RAD) es una metodología de desarrollo de software, que implica el desarrollo iterativo y la construcción de prototipos. El desarrollo rápido de aplicaciones es un término originalmente utilizado para describir un proceso de desarrollo de software introducido por James Martin en 1991.

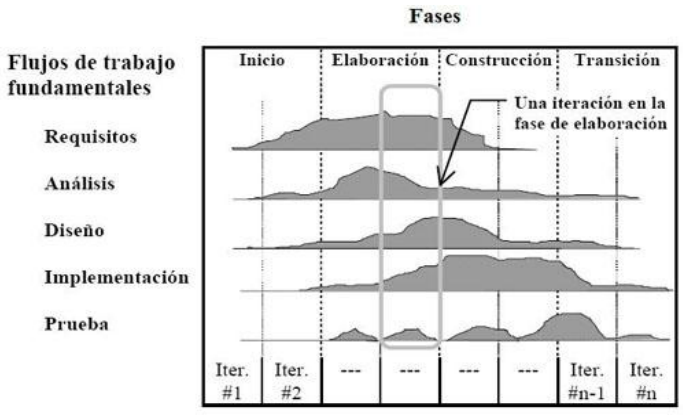
Principios básicos:

- Objetivo clave es para un rápido desarrollo y entrega de una alta calidad en un sistema de relativamente bajo coste de inversión.

- Intenta reducir los riesgos inherentes del proyecto partiéndolo en segmentos más pequeños y proporcionar más facilidad de cambio durante el proceso de desarrollo.
- Orientación dedicada a producir sistemas de alta calidad con rapidez, principalmente mediante el uso de iteración por prototipos (en cualquier etapa de desarrollo), promueve la participación de los usuarios y el uso de herramientas de desarrollo computarizadas. Estas herramientas pueden incluir constructores de Interfaz gráfica de usuario (GUI), Computer Aided Software Engineering (CASE) las herramientas, los sistemas de gestión de bases de datos (DBMS), lenguajes de programación de cuarta generación, generadores de código, y técnicas orientada a objetos.
- Hace especial hincapié en el cumplimiento de la necesidad comercial, mientras que la ingeniería tecnológica o la excelencia es de menor importancia.
- Control de proyecto implica el desarrollo de prioridades y la definición de los plazos de entrega. Si el proyecto empieza a aplazarse, se hace hincapié en la reducción de requisitos para el ajuste, no en el aumento de la fecha límite.
- En general incluye Joint application development (JAD), donde los usuarios están intensamente participando en el diseño del sistema, ya sea a través de la creación de consenso estructurado en talleres, o por vía electrónica.
- La participación activa de los usuarios es imprescindible.
- Iterativamente realiza la producción de software, en lugar de enfocarse en un prototipo.
- Produce la documentación necesaria para facilitar el futuro desarrollo y mantenimiento.

PROCESO DE DESARROLLO UNIFICADO (RUP)

Durante varios años se ha utilizado el modelo tradicional en cascada, demostrando en la práctica que no refleja en la realidad la complejidad inherente al proceso de desarrollo de software. Este problema es derivado de la naturaleza implícita de la estructura de este modelo, definido por una secuencia de grandes etapas que requieren alcanzar hitos que deben ser concluidos antes de continuar con la siguiente fase.



Como una alternativa de solución a este problema, se definieron posteriormente los modelos iterativos e incrementales que trabajan adecuadamente con niveles altos de riesgo, y permiten entregar liberaciones de software en etapas tempranas; tal es el caso del Proceso Unificado propuesto por IBM, que incluye prácticas claves y aspectos relacionados a la planeación estratégica y administración de riesgos; y actualmente guían de forma natural el proceso de desarrollo de software complejo por lo que ha sido considerado como un estándar el desarrollo de software en las empresas.

El proceso unificado conocido como RUP, es un modelo de software que permite el desarrollo de software a gran escala, mediante un proceso continuo de pruebas y retroalimentación, garantizando el cumplimiento de ciertos estándares de calidad. Aunque con el inconveniente de generar mayor complejidad en los controles de administración del mismo. Sin embargo, los beneficios obtenidos recompensan el esfuerzo invertido en este aspecto.

El proceso de desarrollo constituye un marco metodológico que define en términos de metas estratégicas, objetivos, actividades y artefactos (documentación) requerido en cada fase de desarrollo. Esto permite enfocar esfuerzo de los recursos humanos en términos de habilidades, competencias y capacidades a asumir roles específicos con responsabilidades bien definidas.

Fase de concepción: Esta fase tiene como propósito definir y acordar el alcance del proyecto con los patrocinadores, identificar los riesgos potenciales asociados al proyecto, proponer una visión muy general de la arquitectura de software y producir el plan de las fases y el de iteraciones.

Fase de elaboración: En la fase de elaboración se seleccionan los casos de uso que permiten definir la arquitectura base del sistema y se desarrollaran en esta fase, se realiza la especificación de los casos de uso seleccionados y el primer análisis del dominio del problema, se diseña la solución preliminar.

Fase de construcción: El propósito de esta fase es completar la funcionalidad del sistema, para ello se deben clarificar los requerimientos pendientes, administrar los cambios de acuerdo a las evaluaciones realizados por los usuarios y se realizan las mejoras para el proyecto.

Fase de transición: El propósito de esta fase es asegurar que el software esté disponible para los usuarios finales, ajustar los errores y defectos encontrados en las pruebas de aceptación, capacitar a los usuarios y proveer el soporte técnico necesario. Se debe verificar que el producto cumpla con las especificaciones entregadas por las personas involucradas en el proyecto.

COMENTARIOS:

Para desarrollar un proyecto un software es necesario contar con una metodología de desarrollo ya que estas metodologías son las que nos guiaran durante el proceso de desarrollo del proyecto, las metodologías que actualmente existen fueron inventadas en diferentes tiempos y todas ellas son de suma importancia ya que cada una tiene sus características y ventajas distintas, en el modelo en cascada por ejemplo: hace referencia a que el proceso de desarrollo de software tiene que ser visto en forma de cascada es decir de arriba hacia abajo empezando con los requisitos, diseño, implementación, verificación y mantenimiento, todas estas etapas están unidas unas con otras para llevar una secuencia, además de que en este modelo se considera a la planeación y los presupuestos de sistema de una sola vez, y lo mismo se realiza para la siguiente versión. En el modelo de prototipos o prototipado se refiere a que el prototipo debe ser construido en poco tiempo considerando los aspectos y características principales que serán visibles para el usuario final el cual debe evaluar y aprobar, para su posterior continuación o realizar las correcciones necesarias, en cambio el modelo incremental provee de estrategias para controlar los riesgos durante el desarrollo del proyecto, ya que trabaja con una serie de mini-cascadas en donde todas las fases de la cascada del modelo de desarrollo se han contemplado para una pequeña parte de los sistemas, antes de proceder a la próxima etapa. Para el modelo en espiral las actividades se representan en forma de espiral en las que cada ciclo representa un conjunto de actividades que no están fijadas a ninguna prioridad, en este modelo se centra la atención en la evaluación y reducción del riesgo del proyecto esto se consigue dividiendo el proyecto en segmentos mas pequeños y así proporcionar más facilidad de cambio durante el proceso de desarrollo, en el caso del desarrollo rápido de aplicaciones (RAD) es una metodología de desarrollo de software, que implica el desarrollo iterativo y la construcción de prototipos.

En lo personal recomiendo la metodología RUP ya que cuenta con solo cuatro etapas, concepción, elaboración, construcción y transición, de manera general podríamos decir que la fase de concepción es donde se plantea el problema y las características generales del sistema, en la fase de elaboración es donde se diseñan las ventanas y se elaboran los diagramas correspondientes además de que se puede iniciar la programación y se hace el calendario del proyecto, así también se define la arquitectura de sistema, para la fase de construcción se realiza la total programación del sistema, se revisa el código y se prueba el software, se integran los módulos y se puede proporcionar una primera versión para pruebas del cliente o usuarios de sistema, y en la fase de transición no es otra cosa más que el despliegue o distribución de producto ya sea una versión beta, también se realiza la documentación necesaria y se crea un equipo de mantenimiento o corrección de errores, una vez liberada la primera versión se puede empezar a preparar para una segunda versión con los resultados obtenidos de la primera además de que esta segunda versión prácticamente reduce los costos y tiempos a la mitad, la metodología RUP puede ser usada para proyectos de software o proyectos de sitios web completos.