DISEÑO DE APLICACIONES WEB/MÓVILES

CICLO DE VIDA DE DESARROLLO DE SOFTWARE

Contenido

Introducción

El ciclo de vida del desarrollo de software es un modelo aplicado al desarrollo de un producto de software. Hay varios modelos a seguir, cada uno describe un enfoque diferente para distintas actividades que tienen lugar durante todo el proceso.

La utilización de un ciclo de vida de software tiene como objetivo encontrar procesos sistemáticos, reproducibles y predecibles que mejoren la productividad y la calidad.

Sin una gestión del proyecto, éstos corren el riesgo de demorarse o consumir un presupuesto mayor que el planeado.

Modelo en Cascada

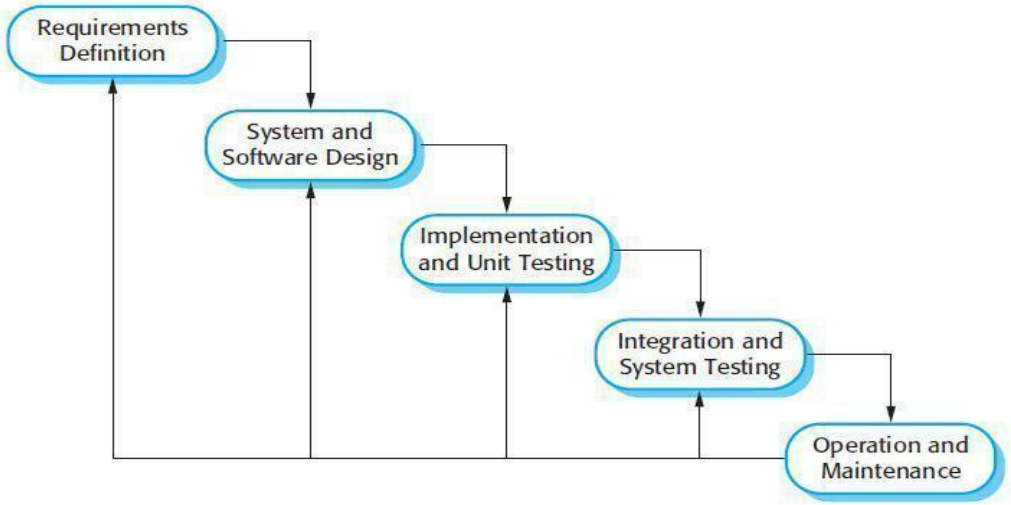
La versión original de este modelo fue propuesta por Winston W. Royce en 1970 y posteriormente revisada por Barry Boehm en 1980 e Ian Sommerville en 1985

Fue el primer modelo propuesto, en el cual las etapas que lo componen se representan cayendo en cascada, desde una etapa, hasta la siguiente.

Una etapa de desarrollo debe completarse antes de dar comienzo a la siguiente. De esta forma, cuando todos los requerimientos del cliente han sido identificados, analizados para comprobar su integridad y consistencia, y documentados en un documento de requerimientos, recién entonces el equipo de desarrollo puede seguir con las actividades de diseño del sistema.

El modelo en cascada presenta una visión muy clara de cómo se suceden las etapas durante el desarrollo, y sugiere a los desarrolladores cuál es la secuencia de eventos que podrán encontrar.

Las etapas son:



Primera etapa: Análisis y Definición de Requerimientos

Los servicios restricciones y metas del sistema se definen a partir de las consultas que son realizadas a los usuarios.
El resultado de estas consultas se traduce y sirve en una especificación del sistema.
Se definen Requisitos Funcionales y No Funcionales.

Segunda etapa: Diseño del Sistema y del Software

El proceso de diseño del sistema divide los requerimientos del sistema en hardware, software, firmware según corresponda.
Se establece una arquitectura completa del sistema.
En la etapa de diseño del software se identifican y describen las abstracciones del sistema de software y sus relaciones.

Tercera etapa: Implementación

Durante esta etapa el desarrollo del software se lleva a cabo como un conjunto de unidades de programas. Pueden realizarse pruebas unitarias para verificar que cada una cumpla con su función.

Cuarta etapa: Integración y prueba del sistema

Se buscan sistemáticamente y se corrigen todos los errores antes de ser entregado al usuario final.

Quinta etapa: Mantenimiento del sistema

El sistema ya se encuentra en funcionamiento práctico.
Implica un sostenimiento adecuado para que el software funcione adecuadamente.

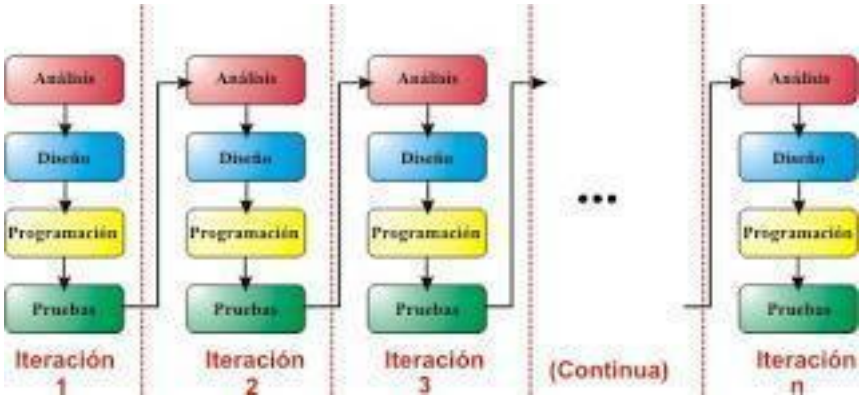
Ventajas del modelo

- Es un modelo fácil de implementar y entender.
- Está orientado a documentos.
- Promueve una metodología de trabajo sistemática: Definir antes que diseñar, diseñar antes que codificar

Desventajas del modelo

- Es rígido.
- Poco flexible.
- Muchas restricciones.
- Es costoso: cualquier error eleva el costo.
- Difícil de detectar errores.
- Se necesita tener en claro los requerimientos al comienzo del proyecto.

Modelo ITERATIVO O Incremental



Se basa en la idea de desarrollar una implementación inicial, construyendo secciones reducidas del software, luego exponerla a los comentarios del usuario y refinando a medida que se van liberando las diferentes versiones, en las que se irán agregando funcionalidades, hasta que se llega a un sistema adecuado.

La idea es que se planifique un proyecto en distintos bloques temporales denominados iteración. En una iteración se repite un determinado proceso de trabajo que brinda un resultado más completo para un producto final, de forma que quien lo utilice reciba beneficios de este proyecto de manera creciente.

Ventajas del Modelo Incremental:

- Se reduce el tiempo de desarrollo inicial ya que se implementa una funcionalidad parcial.
- Si se detecta un error, se desecha la última iteración.
- No es necesario tener en claro TODOS los requerimientos al comienzo del proyecto.

Desventajas del Modelo Incremental:

- La entrega temprana de los proyectos produce la creación de sistemas demasiados simples a primera impresión del cliente.
- Requiere de un cliente involucrado durante todo el curso del proyecto. Hay clientes que simplemente no estarán dispuestos a invertir el tiempo necesario.
- La entrega de un programa que es parcial pero funcional puede hacer vulnerable al programa debido a la falta de robustez en su sistema.

MODELO EN ESPIRAL

El Modelo en Espiral, propuesto originalmente por BOEHM en 1986, es un modelo de proceso de software evolutivo donde se conjuga la naturaleza de construcción de prototipos con los aspectos controlados y sistemáticos del Modelo en Cascada.

Proporciona el potencial para el desarrollo rápido de versiones incrementales del software que no se basa en fases claramente definidas y separadas para crear un sistema.

En el modelo espiral, el software se desarrolla en una serie de versiones incrementales. Durante las primeras iteraciones la versión incremental podría ser un modelo en papel o un prototipo, durante las últimas iteraciones se producen versiones cada vez más completas del sistema diseñado.

El modelo en espiral se divide en un número de actividades de marco de trabajo, también llamadas REGIONES DE TAREAS.

Cada una de las regiones están compuestas por un conjunto de tareas del trabajo llamado CONJUNTO DE TAREAS que se adaptan a las características

del proyecto que va a emprenderse en todos los casos se aplican actividades de protección.

Se puede observar que son 4 áreas bien definidas:

Definición de objetivos

Se establece la especificación del alcance del producto que se obtendrá al final de la iteración (catálogo de objetivos y requisitos), se identificarán riesgos y se determinarán posibles alternativas de solución.

Análisis del riesgo

Se analizan los **riesgos** involucrados en el desarrollo del proyecto.

Desarrollar y probar

Comprende las actividades de análisis, diseño, construcción e implantación de la versión del producto.

Planificación

Se procede a realizar un estudio de la situación actual del sistema y del proyecto donde se determina la necesidad o no de una nueva evolución y en el caso de que sea necesaria se realiza la planificación de esta.

El movimiento de la espiral, ampliando con cada iteración su amplitud radial, indica que cada vez se van construyendo versiones sucesivas del software, cada vez más completas.

Uno de los puntos más interesantes del modelo, es la introducción al proceso de desarrollo a las actividades de análisis de los riesgos asociados al desarrollo y a la evaluación por parte del cliente de los resultados del software.

A diferencia del modelo de proceso clásico que termina cuando se entrega el software, el modelo en espiral puede adaptarse y aplicarse a lo largo de la vida del software. Una visión alternativa del modelo en espiral puede ser considerada examinando el eje de punto de entrada en el proyecto.

Ventajas del Modelo en Espiral

- Es un buen modelo para el desarrollo de grandes proyectos.
- Se puede combinar con otros modelos de desarrollo.
- Puede comenzar con un Proyecto con alto grado de incertidumbre.
- Bajo riesgo de retraso en caso de detección de errores
- El error se soluciona en la rama de la espiral correspondiente.

Desventajas del Modelo en Espiral

- El costo temporal que se suma en cada vuelta.
- Requiere comunicación con el usuario
- En muchos casos se considera complicado la realización del análisis de riesgos, ya que en función del sistema de información resulta muy complejo definir alternativas de solución y determinar con claridad cuál de ellas es la más apropiada.
- Es complicado realizar una planificación global del proyecto, ya que eso dependerá de cómo evolucione el proyecto, producto y usuarios en las distintas iteraciones. Se puede establecer una planificación inicial, con una serie de evoluciones, pero existe una cierta incertidumbre en cuanto hasta dónde se podrá llegar con el presupuesto inicial.

Modelo Ágil

El desarrollo ágil de software utiliza un desarrollo iterativo como base para abogar por un punto de vista más ligero y centrado en las personas que en el caso de las soluciones tradicionales.

Los procesos ágiles utilizan retroalimentación en lugar de planificación, como principal mecanismo de control. La retroalimentación se canaliza por medio de pruebas periódicas y frecuentes versiones del software.

Se basan en:

- Valores.
- Principios.
- Cambios de cultura.

Se enfocan en:

- La creación constante de valor agregado en un entorno incierto a partir del aprendizaje colaborativo entre las personas.

Cumplen con el manifiesto ágil:

1. Individuos e interacciones sobre procesos y herramientas.
2. Producto funcionando sobre documentación detallada.
3. Colaboración con el cliente sobre negociación contractual.
4. Respuesta ante el cambio sobre seguir un plan.

Principios del manifiesto ágil

1. Satisfacción del cliente: Nuestra mayor prioridad es satisfacer al cliente mediante la entrega temprana y continua de producto con valor.
2. Adaptación al Cambio: Aceptamos que los requisitos cambien, incluso en etapas tardías del desarrollo. Los procesos Ágiles aprovechan el cambio para proporcionar ventaja competitiva al cliente.
3. Entrega frecuente y en poco tiempo: Entregamos producto funcionando frecuentemente, entre dos semanas y un mes, con preferencia al período de tiempo más corto posible.
4. Colaboración continua: Los responsables de negocio y los desarrolladores trabajamos juntos de forma cotidiana durante todo el proyecto.
5. Motivación del equipo: Los proyectos se desarrollan en torno a individuos motivados. Hay que darles el entorno y el apoyo que necesitan, y contarles la ejecución del trabajo.
6. Diálogo: El método más eficiente y efectivo de comunicar información al equipo de desarrollo y entre sus miembros es la conversación cara a cara.
7. Producto funcionando: El producto funcionando es la medida principal de progreso.
8. Sostenible y Constante: Los procesos ágiles promueven el desarrollo sostenible. Los promotores, desarrolladores y usuarios debemos ser capaces de mantener un ritmo constante de forma indefinida.

9. Atención a los Detalles: La atención continua a la excelencia técnica y al buen diseño mejora la agilidad.
10. Simplicidad: La simplicidad (el arte de maximizar la cantidad de trabajo no realizado), es esencial.
11. Autoorganización: Las mejores arquitecturas, requisitos y diseños emergen de equipos auto organizados.
12. Mejora continua: A intervalos regulares el equipo reflexiona sobre cómo ser más efectivo para a continuación ajustar y perfeccionar su comportamiento en consecuencia.

Ventajas de la agilidad

- Mayor satisfacción del cliente.
- Más motivación e involucramiento del equipo de desarrollo.
- Valor agregado inmediato y sostenible al negocio.
- Mejora la calidad del producto (se eliminan características innecesarias).
- Detección temprana de errores o problemas.
- Adaptabilidad al cambio (competitividad en el mercado).
- Fomentan la comunicación e interacción cara a cara.

Metodología ágil SCRUM

SCRUM es una de las metodologías ágiles más populares y utilizadas actualmente en el desarrollo de software.

Permite encontrar prácticas emergentes en dominios complejos, para trabajar colaborativamente, en equipo y obtener el mejor resultado posible en un proyecto. Estas prácticas se apoyan unas a otras y su selección tiene origen en un estudio de la manera de trabajar de equipos altamente productivos.

Está pensado para construir productos de software de manera incremental, realizando entregas parciales y regulares del producto en tiempos fijos llamados sprints. Los sprints suelen durar entre 1 y 3 semanas.

En cada sprint se entrega un incremento de Producto (versión mejorada de la entrega anterior).

Posee:

Roles

Product Owner:

- Representa la voz del cliente.
- Se asegura de que el equipo trabaje de forma adecuada desde la perspectiva del negocio.
- Define las prioridades.
- Dueño del backlog.

Dev Team:

- Desarrolla el producto.
- Organiza el trabajo.
- Reporta el progreso.
- Es un equipo multidisciplinario.

Scrum Master:

- Cuida el proceso.
- Protege al equipo.
- Refuerza las reglas.
- Elimina obstáculos que impiden que el equipo alcance objetivos.

Stakeholder:

- Observa y aconseja.

Artefactos

Product Backlog:

- Lista de requerimientos.
- Priorizadas por valor Negocio.
- Administrada por el Product Owner.
- Evoluciona a lo largo del proyecto.
- Los cambios no afectan al Sprint activo.
- Cualquiera puede agregar requerimientos.

Sprint backlog:

- Lista de requerimientos del Sprint.
- Administrada por Dev. Team.
- Solo el team puede modificarlo.
- Siempre visible.

Incremento del producto:

- Versión del producto.
- Potencialmente entregable.
- Funcionalidad trabajando.
- Cumple definición de hecho (Definition Of Done).

Otros artefactos:

Definition of Done (DoD):

La DoD es un documento que define qué se considera hecho en un equipo Scrum. La idea es establecer una serie de criterios comunes para especificar cuándo un ítem está completamente terminado y que aplique a todos los ítems que forman parte del incremento.

Definition of Ready (DoR):

El DoR es un documento que define cuándo un requerimiento (historia de usuario o similar) se considera listo para que el equipo de desarrollo pueda entenderlo, valorarlo e incluirlo en un Sprint Planning con idea de incluirlo en un Sprint.

Burndown Chart:

El Burndown Chart es un gráfico de trabajo pendiente a lo largo del tiempo que muestra la velocidad a la que se están completando los objetivos, requisitos, o historias de usuarios. Permite extrapolar si el equipo podrá completar el trabajo en el tiempo estimado.

Eventos o ceremonias

Sprint Planning:

- Duración: Hasta 8 horas.
- Se define el objetivo del Sprint.
- Product Owner presenta requerimientos.
- Team estima el esfuerzo.
- Se genera Sprint Backlog a entregar.
- Asisten: Scrum Master – Dev Team - Product Owner.

Daily Meeting:

- Duración: 15 minutos.
- Mismo lugar y hora.
- ¿Qué hice ayer? ¿Qué voy a hacer hoy? ¿Qué impedimentos tengo?

- Team actualiza Sprint Backlog.
- Asisten: Dev Team, Scrum Master. El resto es opcional.

Sprint Review:

- Duración: De 2 a 4 horas.
- Se presenta incremento producto.
- Informal, informativa, se obtiene feedback.
- Asisten: Todos.

Sprint Retrospective:

- Duración: De 1 a 2 horas.
- Revisión del proceso de trabajo, identificar oportunidades de mejora.
- Asisten: Dev Team, Scrum Master y Product Owner.

Además, se incluye un tablero en el cual se pueden distinguir las siguientes columnas:

- To Do: Requerimientos que se incluyeron en el Sprint.
- In Progress: Requerimientos que se están desarrollando actualmente.
- To Verify: Requerimientos listos para ser probados.
- Done: Requerimientos finalizados y listos para entregar.

¿Cómo funciona?

Para comenzar se identifican los requerimientos, se valoran, se estima el esfuerzo de alto nivel y se priorizan. Estos requerimientos serán los que conformen el Product Backlog.

Luego se definen los requerimientos que se incluirán en el Sprint (mediante la Sprint Planning) y conformarán entonces el Sprint Backlog.

Una vez definidas las tareas se comienza a trabajar en el desarrollo del producto según los requerimientos incluidos, actualizando el tablero. Diariamente se realizan las Daily Meetings, en lo posible antes de comenzar la jornada laboral, con el objetivo de sincronizar al equipo.

Al finalizar el Sprint se realiza la Sprint Review con el objetivo de presentar el incremento a todo el equipo y posteriormente se realiza la Sprint Retrospective donde se suele realizar un ejercicio donde se visualiza:

- ¿Qué hicimos bien? y lo vamos a mantener.

- ¿Qué hicimos mal? para lo cual, una vez identificados los problemas, se votan para obtener el/los temas más conflictivos y se adopta una solución a estos. En general no es posible atacarlos a todos, razón por la cual aquellos más votados serán los que se tendrán en cuenta, sin descartar los demás que pueden ser atacados en otro momento. Además, se asigna un responsable, quien revisará que se aplique la solución adoptada.

- ¿Qué podemos mejorar? mismo mecanismo de identificación, soluciones, votos y asignación de responsable para verificar que se aplique lo pactado.

Una vez finalizado el sprint se actualiza el tablero, en general se puede conservar una foto de este para futuras revisiones y se vuelve a comenzar la iteración.