Qué es el ciclo de vida del software

El ciclo de vida del desarrollo del software (también conocido como SDLC o *Systems Development Life Cycle*) contempla las fases necesarias para validar el desarrollo del software y así garantizar que este cumpla los requisitos para la aplicación y verificación de los procedimientos de desarrollo, asegurándose de que los métodos usados son apropiados para el cliente.
Su origen radica en que es muy costoso rectificar los posibles errores que se detectan tarde en la fase de implementación. Utilizando metodologías apropiadas, se podría detectar a tiempo para que los programadores puedan centrarse en la calidad del software, cumpliendo los plazos y los costes asociados.

Aunque existen diferentes ciclos de desarrollo de software, la normativa **ISO/IEC/IEEE 12207:2017** establece:
"Un marco común para los procesos del ciclo de vida de los programas informáticos, con una terminología bien definida, a la que pueda remitirse la industria del software. Contiene procesos, actividades y tareas aplicables durante la adquisición, el suministro, el desarrollo, el funcionamiento, el mantenimiento o la eliminación de sistemas, productos y servicios informáticos. Estos procesos del ciclo de vida se llevan a cabo mediante la participación de los interesados, con el objetivo final de lograr la satisfacción del cliente".

Fases de desarrollo de software

La metodología para el desarrollo de software es un modo sistemático de realizar, gestionar y administrar un proyecto para llevarlo a cabo con grandes posibilidades de éxito. Esta sistematización indica cómo se divide un proyecto en módulos más pequeños para normalizar cómo se administra el mismo.

Así, una metodología para el desarrollo de software son los procesos a seguir sistemáticamente para idear, implementar y mantener un producto de software desde que surge la necesidad del producto hasta que se cumple el objetivo por el cual fue creado.

De esta forma, las etapas del desarrollo de software son las siguientes:

Planificación

Antes de empezar un proyecto de desarrollo de un sistema de información, es necesario hacer ciertas tareas que influirán decisivamente en el éxito del mismo. Dichas tareas son conocidas como el *fuzzy front-end* del proyecto, puesto que no están sujetas a plazos.

Algunas de las tareas de esta fase incluyen actividades como la determinación del ámbito del proyecto, la realización de un estudio de viabilidad, el análisis de los riesgos asociados, la estimación del coste del proyecto, su planificación temporal y la asignación de recursos a las diferentes etapas del proyecto.

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.

Implementación

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

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.

Uso 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).

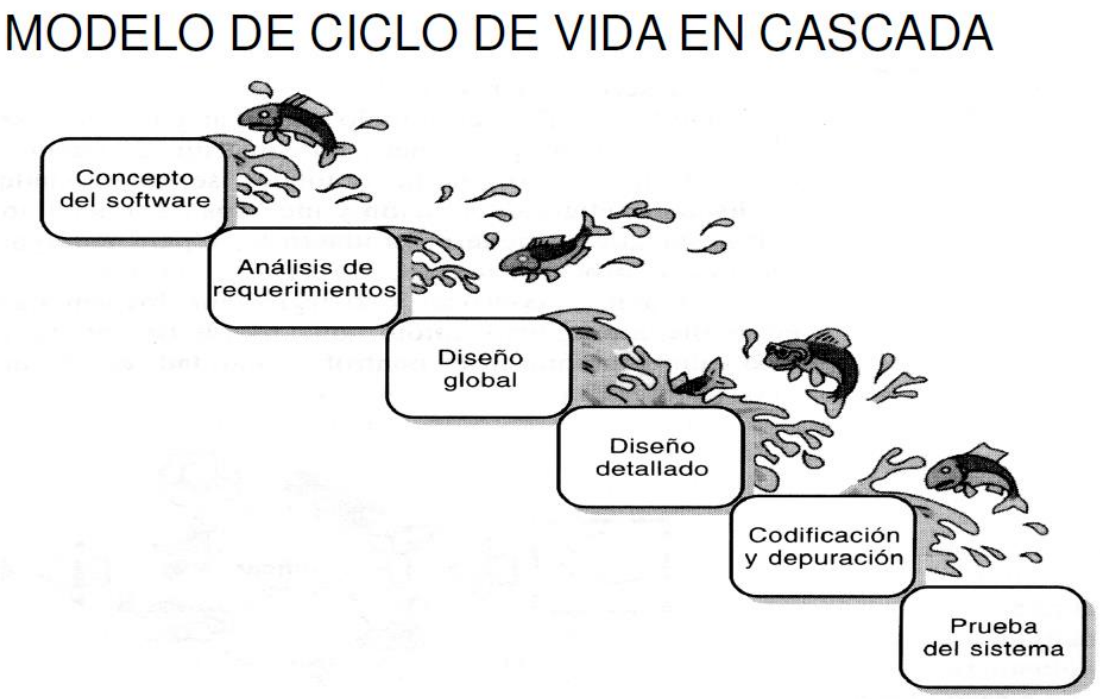
Aunque suene contradictorio, cuanto mejor es el software más tiempo hay que invertir en su mantenimiento. La principal razón es que se usará más (incluso de formas que no se habían previsto) y, por ende, habrá más propuestas de mejoras.

Modelos de ciclos de vida del software

Con el fin de facilitar una metodología común entre el cliente y la compañía de software, los modelos de ciclo de vida (o paradigmas de desarrollo de software) se han actualizado para plasmar las etapas de desarrollo involucradas y la documentación necesaria, de forma que cada fase se valide antes de continuar con la siguiente.

Modelo en cascada

En el modelo de ciclo de vida en cascada las fases anteriores funcionarán una detrás de la otra de manera lineal. De este modo, solo cuando una fase termine se podrá continuar con la siguiente, y así progresivamente.

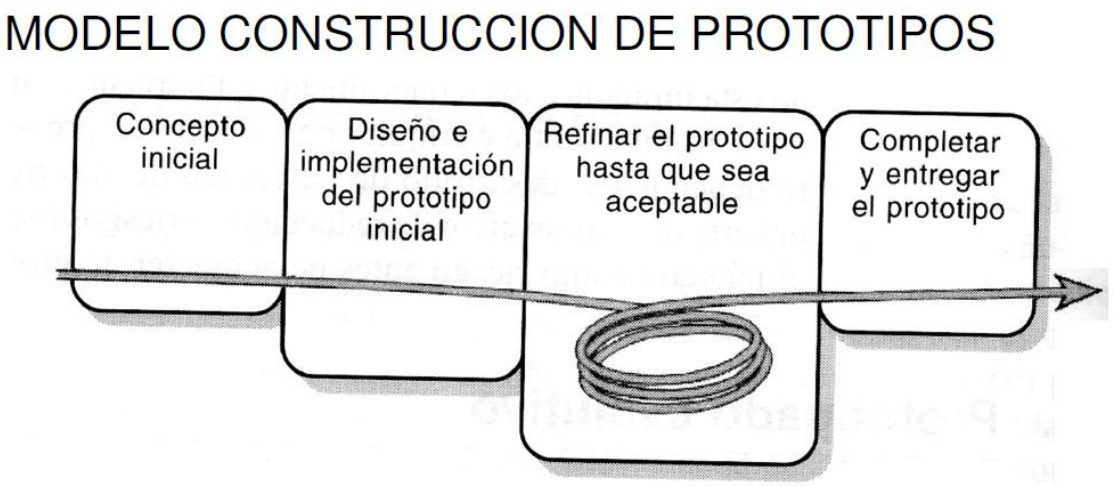


CRÍTICAS AL MODELO DE CICLO DE VIDA EN CASCADA

- Acentúa el fracaso de la industria software frente al usuario final.
- Se tarda mucho tiempo en pasar por todo el ciclo, dado que hasta que no se finalice una fase no se pasa a la siguiente.
- No refleja el proceso real de desarrollo software. Los proyectos reales raramente siguen este flujo secuencial, puesto que siempre hay iteraciones.

Modelo por prototipos

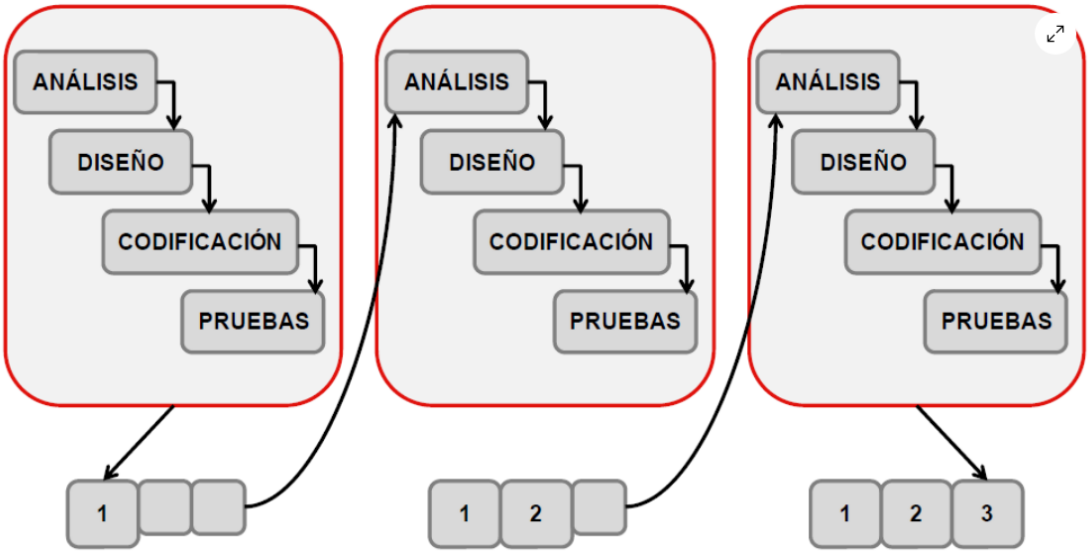
Este modelo va creando una serie de modulos presentables al cliente para que vaya dando su visto bueno en cada paso



Esta manera se asegura de que el resultado será muy similar al esperado pero es el que más recursos gastará, solo debería usarse en caso de que el precio no sea un problema y el propio cliente no sepa muy bien que es lo que quiere

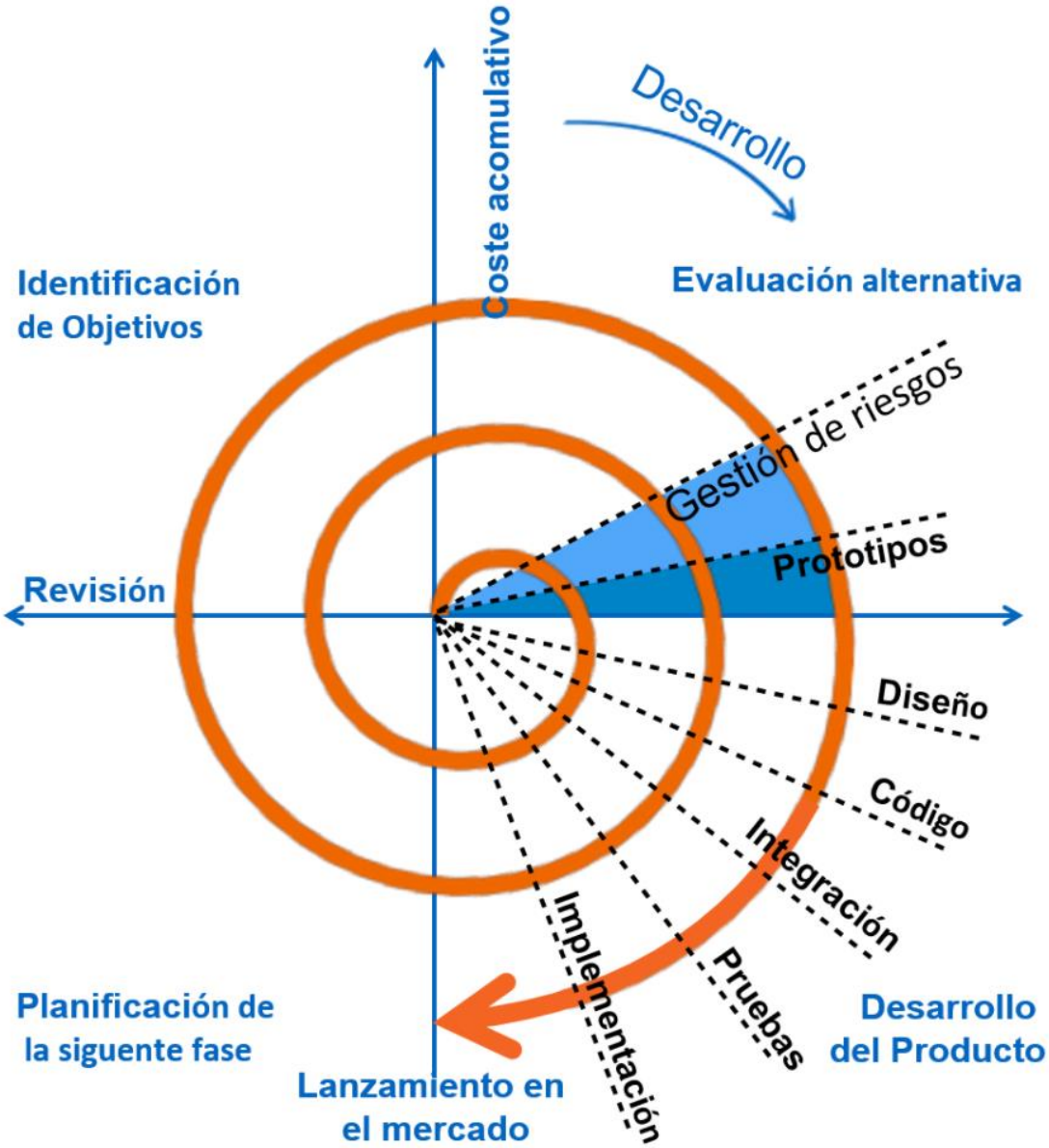
Modelo incremental

El modelo incremental combina elementos del modelo en cascada con la filosofía interactiva de construcción de prototipos. Se basa en la filosofía de construir incrementando las funcionalidades del programa. Este modelo aplica secuencias lineales de forma escalonada mientras progresa el tiempo en el calendario. Cada secuencia lineal produce un incremento del software.



Modelo en espiral

El modelo en espiral es una combinación de los modelos anteriores donde se tiene en cuenta el riesgo. De esta forma, se comienza fijando los objetivos y las limitaciones al empezar cada repetición. En la etapa siguiente se crean los modelos de prototipo del software, que incluye el análisis de riesgo. Posteriormente se usa un modelo estándar para construir el software y finalmente se prepara el plan de la próxima repetición.

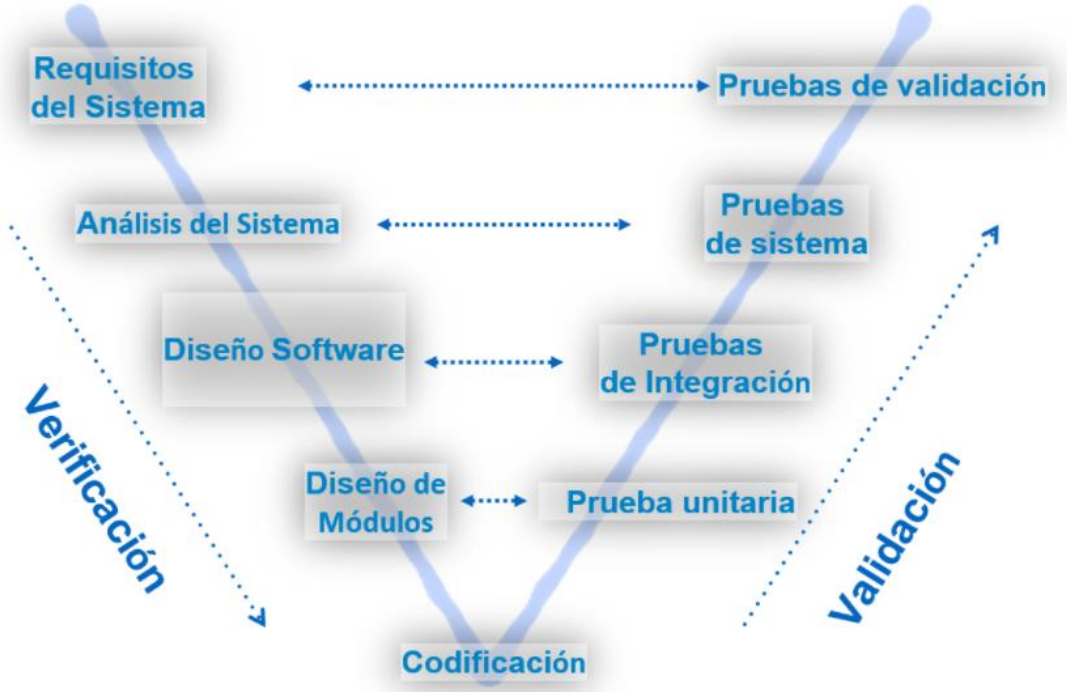


Otros modelos

Modelo en V

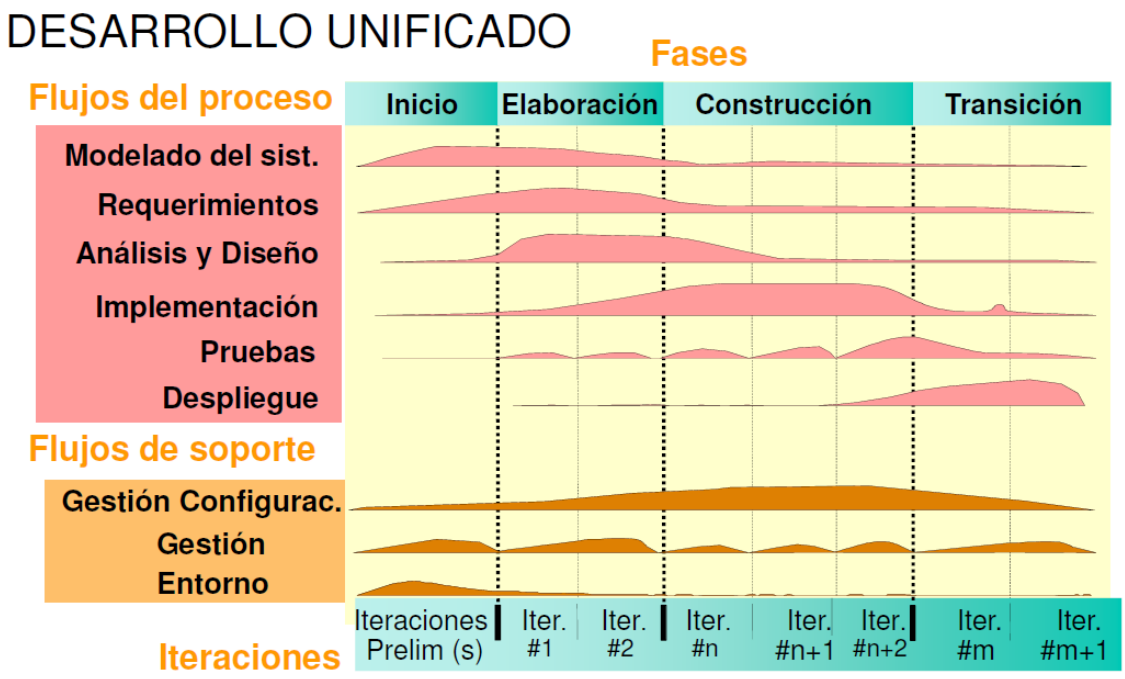
Uno de los grandes problemas del modelo en cascada es que solo se pasa a la siguiente fase si se completa la anterior y no se puede volver atrás si hay errores en etapas posteriores. Así, el modelo en V da más opciones de evaluación del software en cada etapa.

En cada fase se crea la planificación de las pruebas y los casos de pruebas para verificar y validar el producto en función de los requisitos de la misma. De esta manera, verificación y validación van en paralelo.



Modelo Big Bang

Probablemente este sea el modelo más simple, ya que necesita poca planificación, mucha programación y muchos fondos. Este modelo tiene como concepto principal la creación del universo; así, si se reúnen fondos y programación, se consigue el mejor producto de software.



Bibliografía:

https://www.tutorialspoint.com/es/software_engineering/software_development_life_cycle.htm

https://www.efectodigital.online/single-post/2018/04/23/ciclo-de-vida-de-desarrollo-de-software

https://intelequia.com/blog/post/2083/ciclo-de-vida-del-software-todo-lo-que-necesitas-saber

universidad de alicante