tutorialspoint.com

Ciclo de vida del desarrollo Software

9-12 minutes

El ciclo de vida del desarrollo Software (SDLC en sus siglas inglesas), es una secuencia estructurada y bien definida de las etapas en Ingeniería de software para desarrollar el producto sofware deseado.

Actividades del SDLC

El SDLC aporta una serie de pasos a seguir con la finalidad de diseñar y desarrollar un producto software de manera eficiente. El borrador del SDLC incluye los pasos que veremos a continuación:

Comunicación

Este es el primer paso donde l usuario inicia la petición de un producto software determinado. Contacta al proveedor de

servicios e intenta negociar las condiciones. Presenta su solicitud al proveedor de servicios aportando la organización por escrito.

Recolección de solicitudes

A partir de este paso y en adelante el equipo de desarrollo software trabaja para tirar adelante el proyecto. El equipo se reúne con varios depositarios de dominio del problema, e intentan conseguir la máxima cantidad de información possible sobre lo que requieren. Los requisitos se contemplan y agrupan en requisitos del usuario, requisitos funcionales y requisitos del sistema. La recolección de todos los requisitos se lleva a cabo como se especifica a continación -

- Estudiando el software y el sistema actual o obsoleto,
- Entrevistando a usuarios y a desarrolladores de Software,
- Consultando la base de datos o
- Recogiendo respuestas a través de cuestionarios.

Estudio de viabilidad

Después de la recolección de requisitos, el equipo idea un plan para procesar el software. En esta fase, el equipo analiza si el software puede hacerse para cubrir todos los requisitos del usuario y si hay alguna posibilidad de que el software ya no sea necesario. Se investiga si el proyecto es viable a nivel financiero, práctico, y a nivel tecnológico para que la organización acepte la oferta. Hay varios algoritmos disponibles, los cuales ayudan a los desarrolladores a concluir si el proyecto software es factible o no.

Análisis del sistema

En este pas los desarrolladores trazan su plan e intentan crear el mejor y más conveniente modelo de software para el proyecto. El análisis del sistema inclye el entendimiento de las limitaciones del producto Software; el aprendizaje de los problemas relacionados con el sistema; los cambios que se requieren en sistemas ya existentes con antelación, identificando y dirigiendo el impacto del proyecto a la organización y al personal, etc. El equipo del proyecto analiza las posibilidades del proyecto y planifica la

temporalización y los recursos correspondientes.

Diseño de Software

El siguiente paso es diseñar el producto software con la ayuda de toda la información recogida sobre requisitos y análisis. Los inputs (aportacines) de los usuarios y los resultados de la recogida de información hecha en la fase anterior seran las aportaciones base de la fase actual. El output (o resultado) de esta etapa toma la forma de 2 diseños; El diseño lógico y el diseño físico. Los ingenieros crean meta-data (Metadatos), Diagramas dilógicos, diagramas de flujo de datos, y en algunos casos pseudocódigos.

Codificación

Esta fase también se puede denominar 'fase de programación'. La implementación del diseño de software empieza con el lenguaje de programación más conveniente, y desarrollando programas ejecutables y sin errores de manera eficiente.

Pruebas

Se estima que el 50% de todos los procesos de desarrollo de software deberían ser evaluados. Los errores pueden arruinar el software tanto a nivel crítico y hasta el punto de ser eliminado. Las pruebas de Software se hacen mientras se codifica y suelen hacerlo los desarrolladores y otros expertos evaluadores a varios niveles. Esto incluye evaluación de módulos, evaluación del programa, evaluación del producto, evaluación interna y finalmente evaluación con el consumidor final. Encontrar errores y su remedio a tiempo es la llave para conseguir un software fiable.

Integración

El Software puede necesitar estar integrado con las bibliotecas, Bases de datos o con otro u otros programas. Esta fase del SDLC se focaliza en la integración del software con las entidades del mundo exterior.

Implementación

Aquí se instala el software en máquinas de clientes. A veces, el

software necesita instalar configuraciones para el consumidor final con posterioridad. El Software se evalúa por su adaptabilidad y su portabilidad, en cuanto a las cuestiones relacionadas con la integración y conceptos asociados, se resuelven durante la implementación.

Mantenimiento y Funcionamiento

Esta fase confirma el funcionamiento del software en términos de más eficiencia y menos errores. Si se requiere, los usuarios se forman, o se les presta documentación sobre como operar y como mantenerlo en funcionamiento. El software se mantiene de forma temprana actualizando el código en acorde a los cambios que tienen lugar en entornos del usuario o tecnológicos. Esta fase puede que tenga que encarar retos originados por virus ocultos o problemas no identificados del mundo real.

Disposición

Con el paso del tiempo, puede que el software falle en su ejecución. Puede que se vuelva totalmente obsoleto o que necesite actualizaciones. De ahí surge una necesidad urgente de eliminar una parte importante del sistema. Esta fase incluye archivar datos y componentes software requeridos, cierre del sistema, planificación de la actividad de disposición y terminación de sistema en el momento final del sistema.

Paradigma de desarrollo de Software

El Paradigma de desarrollo de Software ayuda al desarrollador a escoger una estrategia para desarrollar el software. El paradigma de desarrollo software tiene su propio set de herramientas, métodos y procedimientos, los cuales son expresados de forma clara, y define el ciclo de vida del desarrollo del software. Algunos paradigmas de desarrollo de software o modelos de proceso se definen a continuación:

Modelo de cascada

El modelo de cascada es el modelo de paradigma más simple en desarrollo de software. Sigue un modelo en que las fases del

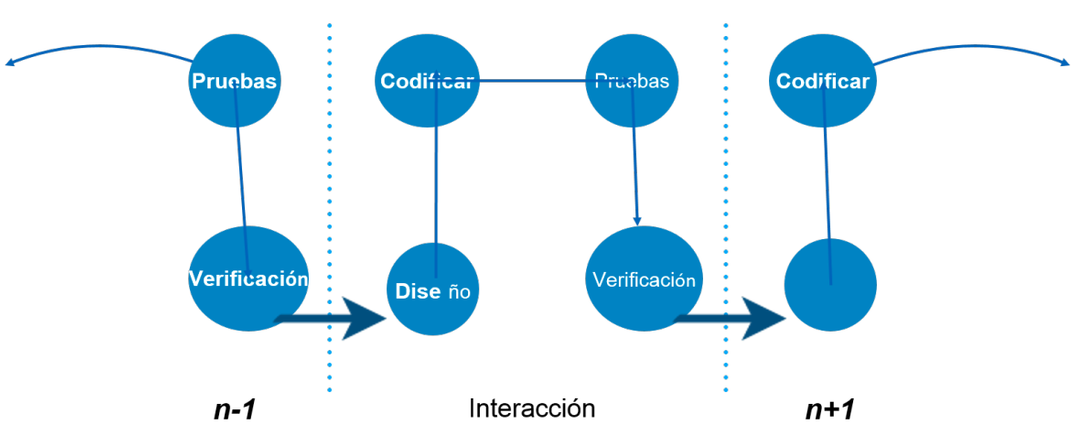
SDLC funcionarán una detrás de la otra de forma lineal. Lo que significa que solamente cuando la primera fase se termina se puede empezar con la segunda, y así progresivamente.

Este modelo asume que todo se lleva a cabo y tiene lugar tal y como se había planeado en la fase anterior, y no es necesario pensar en asuntos pasados que podrían surgir en la siguiente fase. Este modelo no funcionará correctamente si se dejan asuntos de lado en la fase previa. La naturaleza secuencial del modelo no permite volver atrás y deshacer o volver a hacer acciones.

Este modelo es recomendable cuando el desarrollador ya ha diseñado y desarrollado softwares similares con anterioridad, y por eso está al tanto de todos sus dominios.

Modelo repetitivo

Este modelo guía el proceso de desarrollo de software en repeticiones. Proyecta el proceso de desarrollo de forma cíclica repitiendo cada paso después de cada ciclo en el proceso de SDLC.

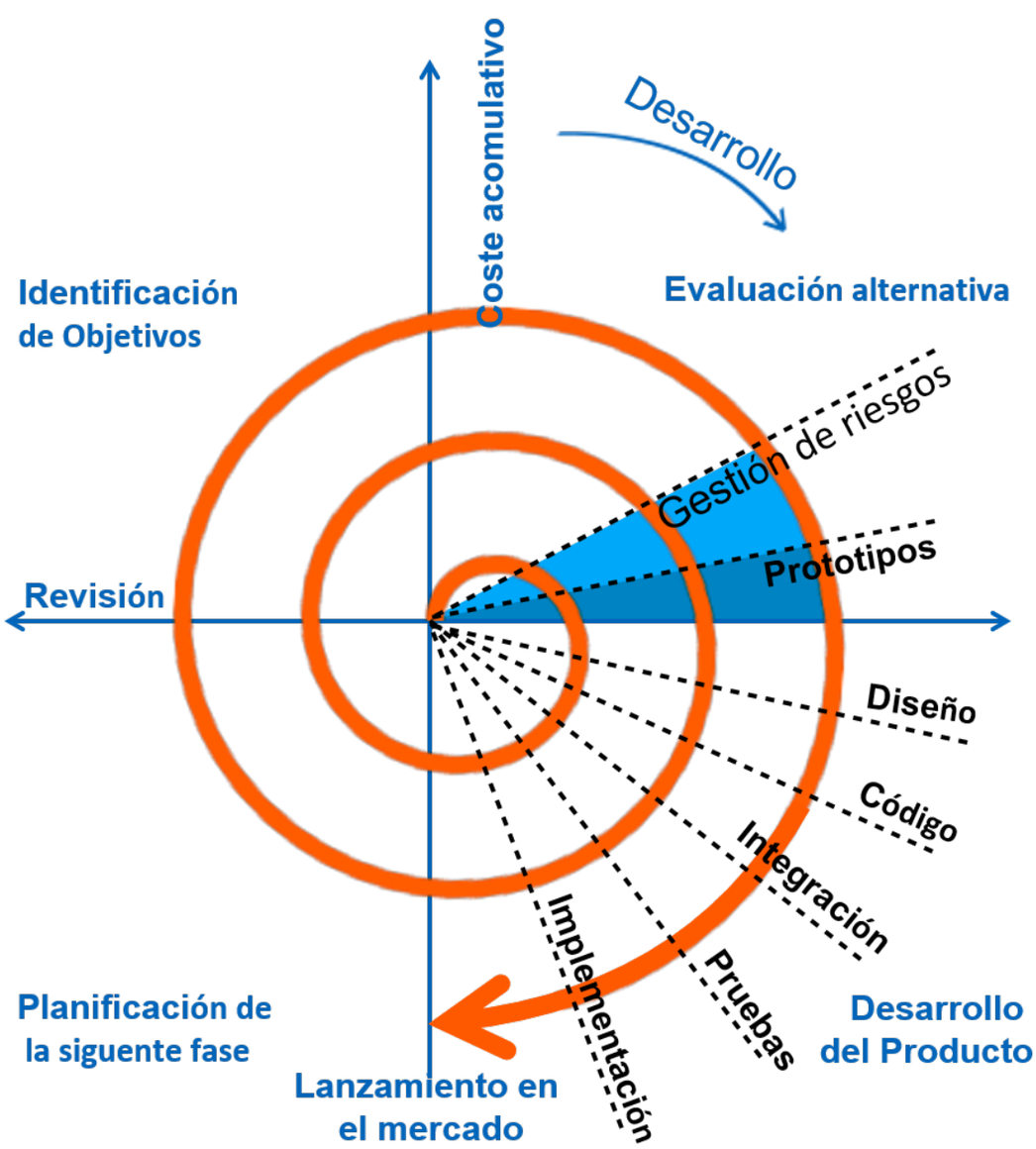


El software primero se desarrolla en menor escala y se siguen y tienen en consideración todos los pasos. Entonces, por cada repetición, más módulos y características son diseñados, codificados, evaluados y añadidos al software. Cada ciclo produce un sotware completo, con más características y capacidad que los previos.

Después de cada repetición, el equipo directivo puede concentrarse en la gestión de riesgos y prepararse para la siguiente repetición. Como el ciclo incluye pequeñas porciones de la totalidad del proceso software, es más fácil gestionar el proceso de desarrollo, pero a la vez se consumen más recursos.

Modelo en espiral

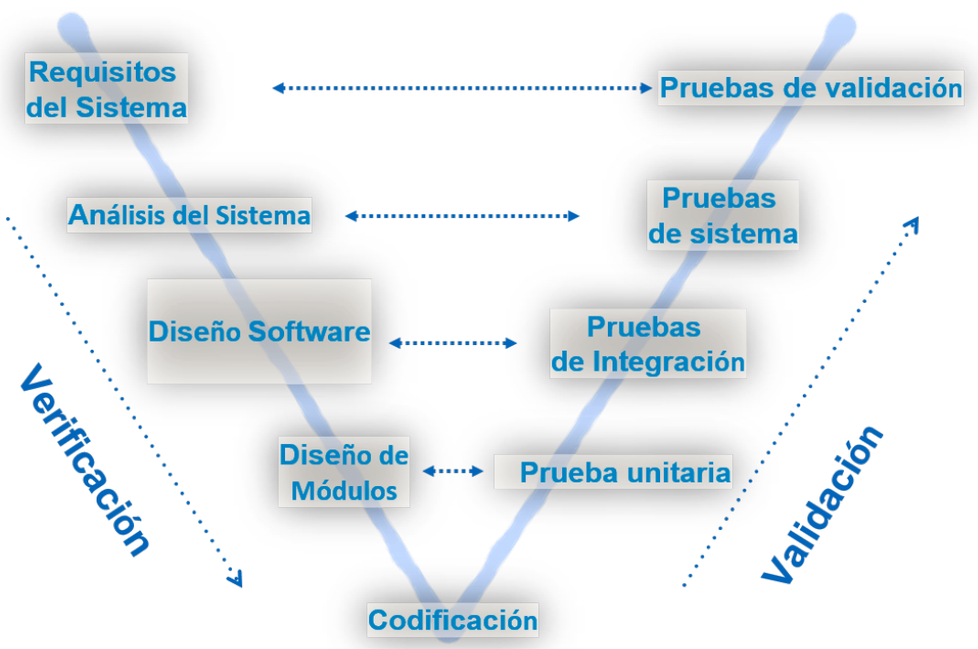
El modelo en espiral es una combinación de ambos modelos, el repetitivo y uno del modelo SDLC. Se puede ver como si se combina un modelo de SDLC combinado con un proceso cíclico (modelo repetitivo).



Este modelo considera el riesgo, factor que otros modelos olvidan o no prestan atención en el proceso. El modelo empieza determinando los objetivos y las limitaciones del software al inicio de cada repetición. En la siguiente etapa se crean los modelos de prototipo del software. Esto incluye el análisis de riesgos. Luego un modelo estándar de SDLC se usa para construir el software. En la cuarta etapa es donde se prepara el plan de la siguiente repetición.

Modelo V

El mayor inconveniente del modelo de cascada es que solo se pasa a la siguiente fase cuando se completa la anterior, por tanto no es posible volver atrás si se encuentra algún error en las etapas posteriores. El Modelo V aporta opciones de evaluación del software en cada etapa de manera inversa.



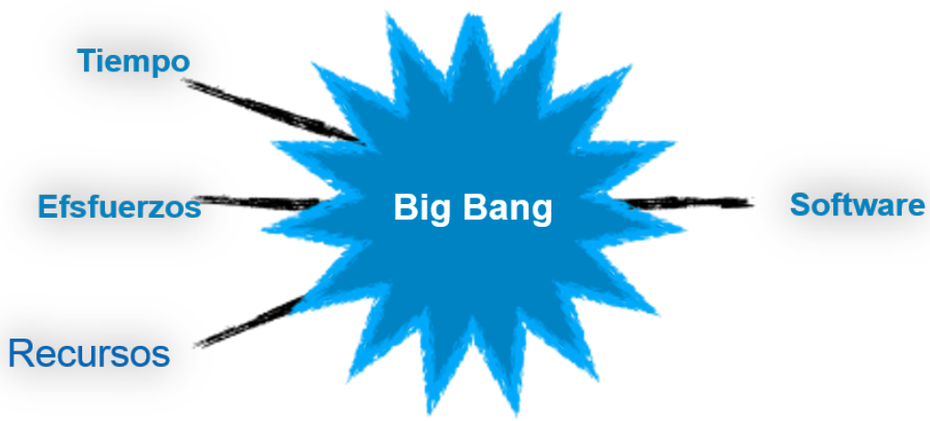
En cada etapa, se crea la planificaión de las pruebas y los casos de pruebas para verificar y validar el producto según los requisitos de la etapa. Por ejemplo, en la etapa de recogida de requisitos, el equipo de evaluadores prepara las pruebas de caso correspondientes a los requisitos. Más tarde, cuando el producto se desarrolla y está preparado para ser evaluado, las pruebas de caso en esta etapa verifican el software y su validez sugún sus requisitos.

Esto hace que tanto la verificación como la validación vayan en paralelo. Este modelo también se conoce como modelo de

validación y verificación.

Modelo Big Bang

Este modelo es el modelo con la forma más simple. Requiere poca planificación, mucha programación y también muchos fondos. Este modelo se conceptualiza alrededor de la teoría de creación del universo 'Big Bang'. Tal como cuentan los científicos, después del big bang muchas galaxias, planetas y estrellas evolucionaron. De la misma manera, si reunimos muchos fondos y programación, quizá podemos conseguir el mejor producto de software.



Para este modelo, se requiere poca planificación. No sigue ningún proceso concreto, y a veces el cliente no está seguro de las futuras necesidades y requisitos. Por tanto la entrada o input respecto a los requisitos es arbitraria.

Este modelo no es recomendable para grandes proyectos de software, pero es bueno para aprender y experimentar.

Para una lectura en profundidad del SDLC y de sus modelos, Pulse aquí.
