Ciclo de Vida de Desarrollo Software

El ciclo de vida de desarrollo de software (SDLC) es el proceso por el cual se **busca diseñar una aplicación** de software bajo unos **estándares de rentabilidad y eficiencia** en términos de tiempo y para crear software de alta calidad.

El objetivo es **minimizar los riesgos** que puedan surgir en el proyecto por medio de una **planificación anticipada** y que cumpla con los requisitos del cliente durante la fase de producción y posteriores fases.

Los modelos de ciclo de vida de desarrollo de software presentan de manera conceptual y organizada un ciclo de vida de desarrollo.

Los conceptos de ciclo de vida de desarrollo y ciclo de desarrollo están muy relacionados y a menudo se intercambian, no obstante existen diferencias entre ellas

El término **ciclo de desarrollo** se refiere más específicamente a las fases **dentro del proceso de desarrollo** de software, principalmente a la etapa de **construcción del software**. Se enfoca en las actividades técnicas necesarias para llevar a cabo el desarrollo o la programación del software.

Este ciclo incluye:

- **Codificación o programación**: El desarrollo real del código del software.
- **Pruebas unitarias o de integración**: Evaluación de pequeños componentes de código para asegurarse de que funcionan según lo previsto.
- **Depuración**: Identificación y corrección de errores.

¿Por qué es importante el SDLC?

El desarrollo de software puede ser difícil de administrar debido a los requisitos cambiantes, los avances de la tecnología y la colaboración interfuncional. La metodología del SDLC ofrece un marco de administración sistemático con entregas específicas en cada etapa del proceso de desarrollo de software.

Ventajas del SDLC:

- **Mayor visibilidad** del proceso de desarrollo para todas las partes interesadas implicadas
- Una estimación, **planificación** y programación eficientes
- Mejoras en la administración de riesgos y **estimación de costos**
- Entregas de software sistemáticas y **mayor satisfacción** de los clientes

¿Cómo funciona el SDLC?

El ciclo de vida del desarrollo de software (SDLC) describe varias **tareas necesarias** para crear una aplicación de software. El proceso de desarrollo pasa por varias **etapas** a medida que los desarrolladores **agregan nuevas características y corrigen errores** del software.

Los detalles del proceso SDLC varían para equipos diferentes. Sin embargo, a continuación se describen algunas fases comunes del SDLC.

Planificación

La fase de planificación incluye normalmente tareas como **análisis de costos y beneficios, programación, estimación de recursos y asignación.** El equipo de desarrollo recopila requisitos de varias partes interesadas, como clientes, expertos internos y externos, así como directivos, para crear un documento de **especificaciones con los requisitos** del software.

Diseño

En la fase de diseño, los ingenieros de software analizan los requisitos e **identifican las mejores soluciones** para crear el software. Decidirán la mejor manera de **integrar** el nuevo software en cualquier **infraestructura de TI existente** que la organización pueda tener.

Implementación

En la fase de implementación, el equipo de desarrollo **codifica el producto**. Se analizan los requisitos para identificar tareas de codificación más pequeñas que puedan hacerse diariamente para conseguir el resultado final.

Pruebas

El equipo de desarrollo combina las **pruebas automáticas y manuales** para comprobar si el software tiene errores. Los análisis de calidad incluyen probar el software para detectar errores y comprobar si cumple los requisitos del cliente.

Despliegue

Cuando los equipos desarrollan software, lo codifican y prueban en una copia diferente que no es a la que acceden los usuarios. El software que los clientes usan se llama **producción**, mientras que las otras copias están en el entorno de compilación o **entorno de pruebas**.

Disponer de un entorno de compilación y de un entorno de producción diferenciados garantiza que los clientes puedan seguir usando el software incluso cuando se modifica o actualiza. La fase de despliegue incluye varias tareas para llevar la última copia compilada al entorno de producción, como empaquetado, configuración del entorno e instalación.

Mantenimiento

En la fase de mantenimiento, entre otras tareas, el equipo **corrige errores**, resuelve problemas de los clientes y **administra los cambios** hechos en el software. Además, el equipo **supervisa el rendimiento** general del sistema, la seguridad y la experiencia del usuario para identificar nuevas maneras de mejorar el software existente.

¿Qué son los modelos SDLC?

Un modelo de ciclo de vida del desarrollo de software se presenta de manera **conceptual y organizada** para permitir que las organizaciones lo implementen. Diferentes modelos disponen las fases del SDLC en diversos órdenes cronológicos para optimizar el ciclo de desarrollo. A continuación, se muestran algunos modelos SDLC conocidos.

Cascada

El modelo de cascada dispone de manera **secuencial** de modo que las nuevas fases **dependen del resultado de la anterior**. Desde un punto de vista conceptual, el diseño fluye desde una fase a otra inferior, como en una cascada.

Ventajas y desventajas:

El modelo de cascada hace que la administración del proyecto sea muy **estricta** y proporciona un **resultado tangible** al final de cada fase. Sin embargo, hay poco margen de cambio una vez que una fase se considera completa, ya que pueden afectar al tiempo de entrega, al costo y a la calidad del software. Por lo tanto, el modelo es más adecuado para **pequeños proyectos** de desarrollo de software, donde las tareas se pueden organizar y administrar fácilmente y los requisitos se pueden definir con precisión.

Iterativo

El proceso iterativo sugiere que los equipos comienzan el desarrollo de software con un **pequeño subconjunto de requisitos**. Posteriormente, se **mejoran las versiones** de manera iterativa a lo largo del tiempo hasta que el software final esté listo para pasar a producción. El equipo produce una nueva versión de software al final de cada iteración.

Ventajas y desventajas

Es fácil identificar y **administrar riesgos**, ya que los requisitos pueden cambiar entre cada iteración. Sin embargo, la repetición de los ciclos puede dar lugar a que cambien los objetivos y se subestimen los recursos.

Espiral

El modelo de espiral combina los **pequeños ciclos repetidos** del modelo iterativo con el **flujo secuencial y lineal** del modelo de cascada para dar prioridad al análisis de riesgos. Puede usar el modelo de espiral para garantizar la actualización y mejora graduales del software mediante la creación de prototipos en cada fase.

Ventajas y desventajas

El modelo de espiral es adecuado para **proyectos grandes y complejos que requieren cambios frecuentes**. Sin embargo, puede ser costoso para proyectos pequeños con objetivos muy concretos.

Ágil

El modelo ágil dispone las fases del SDLC en varios **ciclos de desarrollo**. El equipo itera a través de las fases rápidamente y solo se hacen **pequeños cambios progresivos** de software en cada ciclo. Los requisitos, planes y resultados se **evalúan continuamente** para responder con rapidez a los cambios.

Ventajas y desventajas

Los ciclos rápidos de desarrollo permiten a los equipos identificar y abordar problemas en proyectos complejos desde el principio y antes de que se conviertan en problemas graves. También promueven la **participación de los clientes** y las partes interesadas para que den su opinión en todo el ciclo de vida del proyecto. Sin embargo, depender en exceso de la opinión de los clientes puede hacer que los objetivos cambien drásticamente o dejar el proyecto a medias.

Scrum

Se basa en el principio ágil de desarrollo iterativo e incremental. Al período de trabajo para desarrollar un incremento de producto se lo denomina “sprint” (1 a 4 semanas). Durante cada sprint, el equipo crea un incremento de software potencialmente entregable. Scrum establece una reunión al inicio de cada sprint para determinar el trabajo que se va a realizar, otra al final para evaluar el resultado, y revisiones diarias que realiza el equipo en su auto-gestión.

- **Product owner**: que corresponde con los responsables funcionales. Será el encargado de definir las prioridades del backlog de forma conjunta con el scrum master
- **Scrum master**: Perfil encargado de solucionar las necesidades del equipo de desarrollo y coordinar la correcta ejecución de la iteración.
- **Equipo de desarrollo**: los profesionales encargados de las labores de codificación.

Ventajas y desventajas

Los ciclos de sprints, reuniones diarias (daily) y reuniones semanales (weekly) permiten ajustes rápidos y constantes al proyecto manteniendo la posibilidad de tener un proyecto entregable en cualquier momento. Sin embargo puede resultar contraproducente el esfuerzo administrativo elevado en equipos grandes o en proyectos con requisitos inestables

Kanban

Es un modelo muy relacionado con las metodologías ágiles. Consiste en la elaboración de un cuadro o diagrama en la que se reflejan tres columnas de tareas, divididas en pendientes, proceso o terminadas. Este diagrama ha de estar al alcance de todos los miembros del equipo para evitar la repetición de tareas y o la posibilidad de que alguna sea olvidada.

Ventajas y desventajas

- Ofrecer un panorama que permite ver de un vistazo el trabajo de tu equipo. Como un método visual de gestión de proyectos, Kanban puede ayudarte a poner en marcha el trabajo y obtener una visión clara de los flujos de trabajo de tu equipo.

- Aumentar la claridad, especialmente en los equipos remotos. Al centralizar las tareas y reducir la cantidad de trabajo en curso en un momento dado, los diagramas Kanban pueden ayudar al equipo a obtener información de un vistazo sobre quién está haciendo qué.
- Si hay demasiado trabajo en curso, puede resultar abrumador.

Fuente: https://aws.amazon.com/es/what-is/sdlc/

https://www.preparatic.org/material/20141011/Presentacion_Metodologias.pdf

https://asana.com/es/resources/what-is-kanban

https://www.atlassian.com/work-management/project-management/project-life-cycle

Atlassian: Software development life cycle (SDLC)

IBM: What is the Software Development Life Cycle?