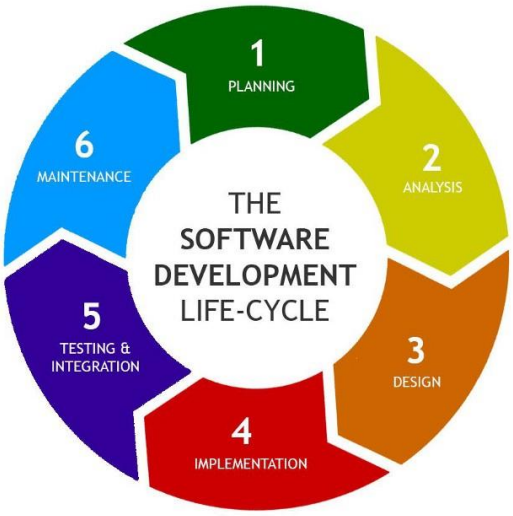
INVESTIGACIÓN: SDLC

Luis Alejandro Gomez

UTXJ 190292 IGDS

System Development Life-Cycle



Ciclo de Vida de Desarrollo de Software y es el proceso utilizado por la industria del software para analizar, desarrollar y probar cualquier pieza de software desarrollado. El objetivo de SDLC es organizar los procesos centrales del proceso de desarrollo de software, de modo que el resultado final sea de la más alta calidad en la situación dada (una combinación de, presupuesto disponible, los casos de negocio para el software, limitaciones de tiempo de desarrollo, más el nivel profesional de los arquitectos e ingenieros de software involucrados en el proceso).

Consiste en 6 partes

1. **Planificación**

   Los requisitos previos del negocio se reconocen en esta fase.

   Se llevan a cabo reuniones con supervisores, socios y clientes para decidir los requisitos previos como:

   - ¿Quién va a utilizar la aplicación de software?
   - ¿Cómo se va a utilizar la aplicación de software?
   - ¿Qué información va a procesar el software?

   Estas son consultas generales que se responden durante la etapa de reconocimiento de requisitos previos.

   A continuación, se crea un documento de especificación de requisitos para servir al propósito de la directriz para la siguiente fase del ciclo. Por lo general, consiste en lo siguiente:

   - Especificación de requisitos funcionales

- Especificación de requisitos comerciales
- Especificación de requisitos de cliente/cliente
- Especificación de requisitos de usuario
- Documento de Diseño de Negocios
- Documento de Negocio

2. Análisis de requisitos

La etapa de análisis de requisitos es la más importante en SDLC. Por lo general, es realizado por los miembros principales del equipo de desarrollo de software junto con expertos en marketing y de la industria. Esta es la parte crucial del proyecto donde el liderazgo del equipo de desarrollo de software debe comprender la esencia del software a desarrollar, los detalles del caso de negocio y el posicionamiento potencial del software que se está desarrollando frente a los productos de los competidores (suponiendo que existan en el mercado).

El resultado de esta etapa, por lo general, es el documento SRS, que significa Especificación de requisitos de software.

En esta etapa, se decide la identificación de los riesgos de desarrollo, así como la línea de base de la metodología de garantía de calidad.

El resultado de la etapa de análisis es definir las soluciones técnicas que conducirán al éxito del proyecto con el mínimo riesgo.

3. Diseño

En nuestro modelo simplificado, asumimos que después de la Etapa 1, el análisis de requisitos, el líder del proyecto escribe las decisiones que se tomaron y desarrolla el llamado documento de Especificación de Requisitos de Software (SRS). El SRS es la guía de referencia para que los arquitectos de software entreguen el mejor resultado posible para el software dado.

El resultado del trabajo de los arquitectos de software es la Especificación de Diseño del Sistema (SDS). Este documento a veces también se denomina Especificación de documento de diseño (DDS). El mejor escenario es que la SDS sea revisada por las partes interesadas importantes del proyecto desde diferentes perspectivas, tales como: evaluación de riesgos, robustez del producto, modularidad del diseño, presupuesto y limitaciones de tiempo. El mejor modelo de diseño se discute y se selecciona para el producto.

El diseño del software describe claramente los módulos arquitectónicos del producto, así como el flujo de datos y los diagramas de comunicación dentro del propio producto, además, especifica cualquier integración de terceros que sea relevante para el producto.

4. **Implementación**

La complejidad de esta etapa depende en gran medida del resultado de las dos etapas anteriores. Cuanto mejor sean los SRS y SDS, más fácil será para los ingenieros de software desarrollar los módulos de software requeridos. No es ningún secreto que si se requieren los primeros prototipos / versiones del software en un plazo de tiempo muy corto, esto puede afectar negativamente la calidad del software final. La calidad también depende en gran medida de las capacidades de análisis de cada individuo que participa en el proceso de codificación, al igual que tener un SRS y SDS debidamente preparados.

Cuanta más experiencia y capacidad intelectual tengan los analistas, arquitectos y desarrolladores, menos tiempo y documentación se necesita en la fase de diseño.

5. **Pruebas e integración**

Aunque esta etapa se llama prueba, en la vida real, la situación es que las fallas encontradas en el software en la fase de prueba, conducen de nuevo a la fase de desarrollo y luego de vuelta a la fase de prueba en círculos, hasta que el software finalmente alcanza la calidad necesaria. La fase de prueba normalmente consta de dos fases internas:

- Pruebas unitarias/funcionales automatizadas
- Pruebas de aceptación realizadas por un ser humano

6. **Mantenimiento**

Ignorando el lado técnico de esta fase, esto es cuando el software se lanza al estado alfa / beta o estable y la retroalimentación comienza a llegar. El análisis de la retroalimentación puede conducir a segundas iteraciones tercera o cuarta del SDLC, lo que significa que el proceso continúa nuevamente con las fases 1, 2, 3, 4 y 5, eliminando gradualmente cualquier problema que se haya encontrado en el software lanzado.

Todo el proceso de desarrollo de software también se puede planificar en varias iteraciones, especialmente, cuando se trata de Agile, donde el enfoque principal es lanzar un producto de trabajo lo antes posible e implementar diferentes características más adelante.

Por lo tanto, no significa que repasar varias iteraciones siempre signifique corregir errores o ajustar. Como acabo de mencionar, también puede ser un proceso planificado previamente.

Agile Development Model

Un modelo es ágil o liviano cuando emplea para su construcción una herramienta o técnica sencilla, apuntando al desarrollo de un modelo aceptable y suficiente en lugar de un modelo perfecto y complejo. Los principales beneficios que se obtienen con la implementación de una metodología ágil son: la capacidad de administrar cambios en las prioridades, mejorar la visibilidad del proyecto, aumentar la productividad.

Los 3 métodos más utilizados son:

1.SCRUM

Scrum es un proceso que se usa para minimizar los riesgos durante la realización de un proyecto de manera colaborativa. Algunas ventajas son la productividad y calidad por medio de un seguimiento diario de los avances del proyecto, realizando entregas parciales del producto final. Este método es ideal para empresas dedicadas al desarrollo de software que trabajan para diferentes clientes a la vez. Con Scrum se aplican un conjunto de mejores prácticas para trabajar colaborativamente y en equipo obteniendo los mejores resultados en el menor tiempo posible.

XP o Extreme Programming

La “programación extrema” es un proceso aplicado en equipos pequeños de programadores que se centran en un solo cliente o desarrollo a la vez. Consiste en diseñar, implementar y programar lo más rápido posible, en ocasiones omitiendo la documentación y los procedimientos regulares.

3. Desarrollo Ligero o “Lean”

El desarrollo ligero o “Lean Programming”, es un conjunto de técnicas orientadas a conseguir exactamente lo que necesita el cliente. Se fundamenta en tener un equipo muy capaz y comprometido al principio del aprendizaje continuo, haciendo uso de ciclos de evolución de software. Este tipo de desarrollo es ideal para empresas que están desarrollando un software B2C orientado a tener éxito en el mercado.

Conclusión

El ciclo de vida de desarrollo de software, marca la pauta para un inicio y un final en el que será desarrollado, como su nombre lo dice, un proyecto de desarrollo de software, existen varios tipos, y se pueden desarrollar de varias formas, adoptar una hace la diferencia entre el éxito y el fracaso de un proyecto.

Algunos tipos son más rápidos y otros se llevan más tiempo para el desarrollo, ya que no comprenden posibles fallas o contratiempos que se presentan al momento del desarrollo, así mismo, un error puede hacer regresar al inicio del proyecto.