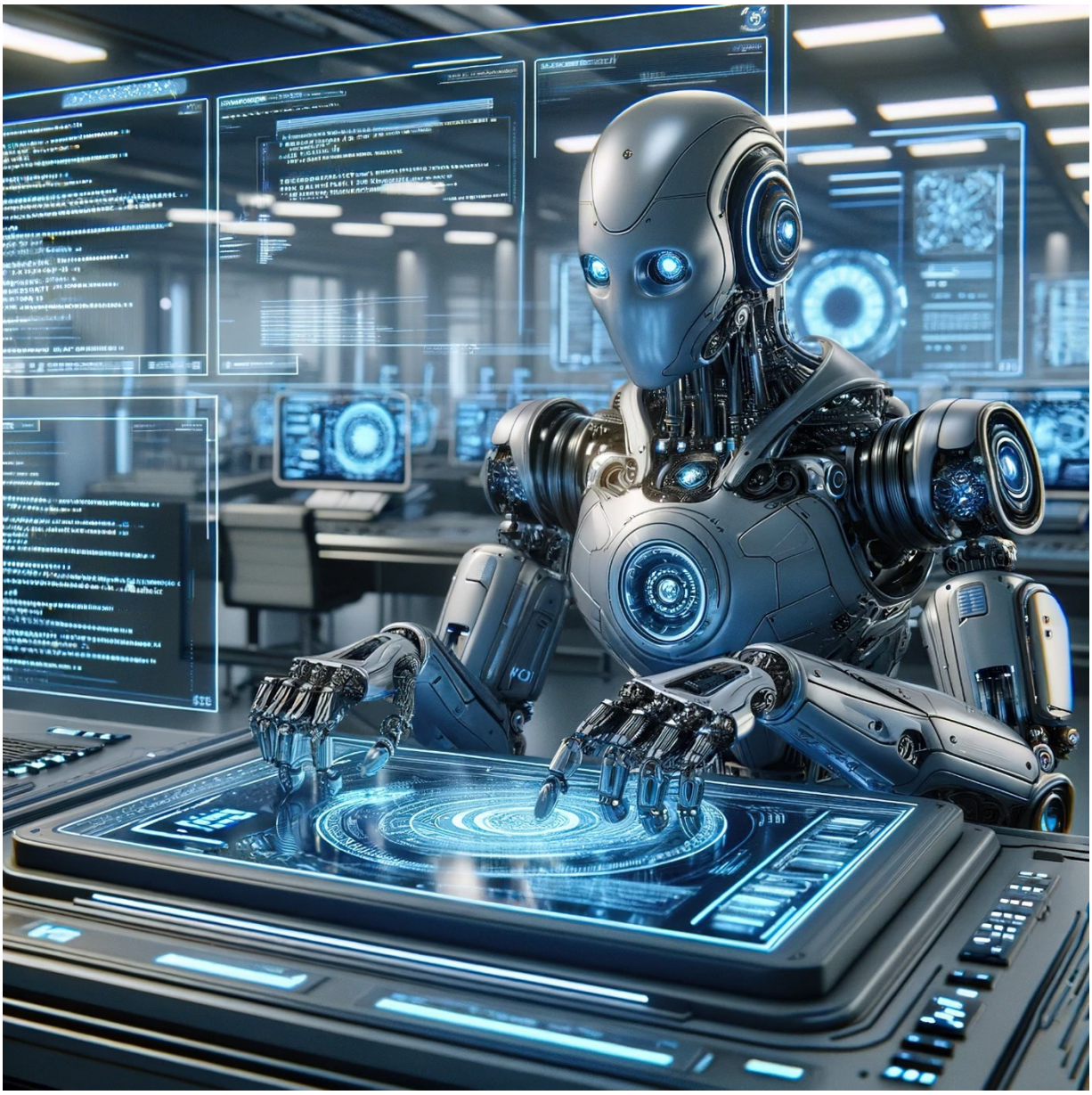
Diego Fermín Pastrana Monzón

CICLO DE VIDA DEL SOFTWARE



Índice:

1. El ciclo de vida del desarrollo de software es muy importante en lo que es en sí la ingeniería de software ya que define los pasos fundamentales a la hora de llevar a cabo proyectos como por ejemplo el desarrollo de un programa. De esta manera se aseguran unos estándares de calidad para el producto final que llega al mercado.

El ciclo empieza en base a los requisitos, donde se establece exactamente que necesitan los usuarios que van a usar el software. En esta fase podríamos decir que es como una entrevista, es decir se recopila la mayor cantidad posible de información relevante para entender y planear como ser. Luego se sigue a la parte de diseño, aquí se planificará la arquitectura del software, y esto es de vital importancia ya que se definirá como será cada parte y como interactuará cada una.

Después, en la fase de implementación, los diseños previos se transforman en el producto final mediante el código que crean los desarrolladores para hacer el programa. Justamente con el código es como se construye el programa, asegurando que cada función se implemente y funcione correctamente. Después en la implementación el software debe pasar por una etapa de pruebas rigurosas de diferentes tipos, todo esto para garantizar que sea robusto, fiable y que cumpla con todas las especificaciones iniciales.

Una vez que ya el software se ha probado con éxito y se han corregido todos los fallos, se procede a lanzarlo al mercado. Después el ciclo de vida se completa con el mantenimiento, en esto entran cosas como actualizaciones del software y mejorarlo en respuesta al feedback que den los usuarios y los cambios en el hardware. Con esto último podríamos poner el ejemplo de que lanzamos un sistema operativo o un videojuego pero no va del todo bien con nuevos procesadores o incluso dan errores.

Se pueden adoptar diferentes modelos durante el ciclo de vida, por ejemplo el modelo en cascada que es estructurado y secuencial, o el modelo ágil que es un poco más flexible y permite hacer iteraciones rápidas y más adaptadas a lo largo del desarrollo del software. El elegir entre un modelo u otro depende de varios factores, pero normalmente lo que se incluye es por la naturaleza del propio proyecto y las necesidades de la organización que lleva a cabo el desarrollo.

Además la gestión de forma eficaz del desarrollo de software implica cosas como enfrentarse a diferentes problemas, tales como los cambios repentinos, la gestión de tiempo y los costes. Pero aplicando una debida planificación, con una buena gestión de los cambios en el software, una buena comunicación que sea clara y concisa entre los participantes del desarrollo facilita el proceso de creación y además asegura que el producto final sea de alta calidad y que cumpla o incluso que exceda las expectativas de los usuarios que vayan a usar dicho software.

Foto generada por DALL·E

2. La planificación en el desarrollo de software es una parte crucial para cualquier proyecto. Aquí es donde definimos la dirección que tomará el propio proyecto, además de establecer los fundamentos con los que haremos dicho software después. De esta manera se garantiza la compresión de todos los objetivos y lo que esperamos lograr cuando ya esté terminado.

En cuanto a los objetivos del proyecto debemos tener en cuenta que al empezar la fase de planificación es muy importante tener los objetivos bien claros para evitar posibles problemas. Dichos objetivos tienen que ser lo más específicos posibles, medibles, alcanzables, relevantes y estar definidos en el tiempo para cumplir con los plazos; a esta metodología se le llama “metodología SMART”. Además tener los objetivos claros no solo ayuda a mantener al equipo que está desarrollando el software enfocado, sino que también facilita el dar prioridad a las tareas más importantes y mejora la capacidad de tomar la mejor estrategia para su desarrollo a lo largo del mismo.

En cuanto a los requisitos del proyecto hay que tener en cuenta que es importante establecerlos correctamente ya que si lo analizamos es una parte crítica de la planificación. En esto podemos incluir los requisitos funcionales y los no funcionales, además de los técnicos y de rendimiento. Dichos requisitos se podrían definir de cierta forma como la base sobre la que construiremos el software, y de esta manera se previene en la mayoría de los casos errores con costes muy altos, como por ejemplo si hay que rehacer la mayor parte del código de un programa por malentendidos o problemas de compatibilidades. Teniendo en cuenta esto ahora sabemos que los requisitos deben incluir detalles sobre las plataformas en las que van a funcionar y las tecnologías que van a usar, además de tener en cuenta los requisitos del sistema que los va a ejecutar. De esta manera nos aseguramos de que nuestro software cumple con los objetivos finales y con las necesidades del usuario final.

Posteriormente hay que tener en cuenta el alcance del proyecto, lo que implica establecer límites bien claros sobre lo que incluirá dicho software que estamos desarrollando y que cosas se descartarán durante el desarrollo. Establecer dichos límites es crucial para mantener al equipo de desarrollo dentro de los parámetros establecidos y evitar el problema conocido como “alcance creciente”, que trata de que los requerimientos del propio proyecto tienden a expandirse más allá de los límites originales del mismo.

También hay que considerar los entregables, ya que es de vital importancia definir con claridad los entregables que se esperan producir por el equipo de desarrollo. En los entregables puede haber varias cosas, desde simples prototipos del software, hasta documentación técnica, y el software ya terminado. Los entregables que están bien definidos no solo establecen las expectativas del usuario sino que también ayudan a planificar de manera más eficaz el proyecto, con los plazos (que es cuando se debe entregar el software) y recursos que sean necesarios para dicho proyecto.

Foto generada por DALL·E

3. Las pruebas de software son esenciales para garantizar que las aplicaciones funcionen como deberían y así garantizar la satisfacción del usuario final. En esto entra la depuración que hasta los años 80 fue el principal método de prueba, ya más tarde se enfocaron más en pruebas más orientadas a entornos del mundo real. Después de esto las pruebas en los años 90 evolucionaron más hacia lo que podríamos llamar la garantía de calidad.

En base a esto ahora podemos hablar de las pruebas de integración, estas se enfocan en la integración de módulos. Estos pueden tener diferentes enfoques, la incremental, donde los módulos se prueban incrementalmente y de manera no incremental, en esto es donde todo el software se prueba a la vez. Teniendo esto en cuenta se podría decir que los métodos incrementales incluyen la integración Top-Down en la que se empieza por los módulos de alto nivel y se sigue hacia los de más bajo nivel que son los Bottom-Up. A lo no incremental se le conoce como enfoque Big-Bang y puede ser útil para sistemas más complejos.

En cuanto a las pruebas unitarias podemos decir que se realizan primero para validar el funcionamiento de las unidades más pequeñas de nuestro software. Después en las pruebas de integración se aseguran de que los módulos trabajen juntos en condiciones. Después seguimos con la verificación y también la validación, que sería la prueba de aceptación del usuario (UAT), que se hace en un entorno similar al de producción final para asegurarse de que el sistema cumpla con los requisitos que nos pide el usuario y esté listo para el uso en un entorno ya de la vida real.

De esta manera, las pruebas continuas se han vuelto últimamente una parte esencial del ciclo de vida del propio desarrollo del software, con los métodos llamados DevOps, que permiten la detección y corrección temprana de estos defectos o problemas en el software en el que estamos trabajando. Esto quiere decir que se harán entregas más rápidas mientras además reducirnos los costos de la corrección de los errores.

Después está el modelo V de desarrollo, que es un enfoque que asigna una actividad de prueba a cada etapa del desarrollo del software. Aquí la verificación y validación son igual de importantes y también se unen en la fase de codificación. Además el proceso comienza con el análisis de los requisitos y continúa con el diseño del sistema, después el diseño arquitectónico y el

diseño de módulos. Es decir cada fase de diseño tiene una fase de prueba asociada que son en este caso las pruebas unitarias, de integración, del sistema y de aceptación del usuario.

Las pruebas deben planificarse con mucho cuidado y de manera meticulosa, y los además hay que tener en cuenta que los recursos se asignan mejor cuando se priorizan las pruebas que proporcionan el mayor valor dentro de las limitaciones de que tenemos de tiempo y recursos disponibles. Además la efectividad de las pruebas se mejora realizando la menor cantidad de pruebas necesarias para encontrar el mayor número de defectos y tener la mayor aceptación posible.

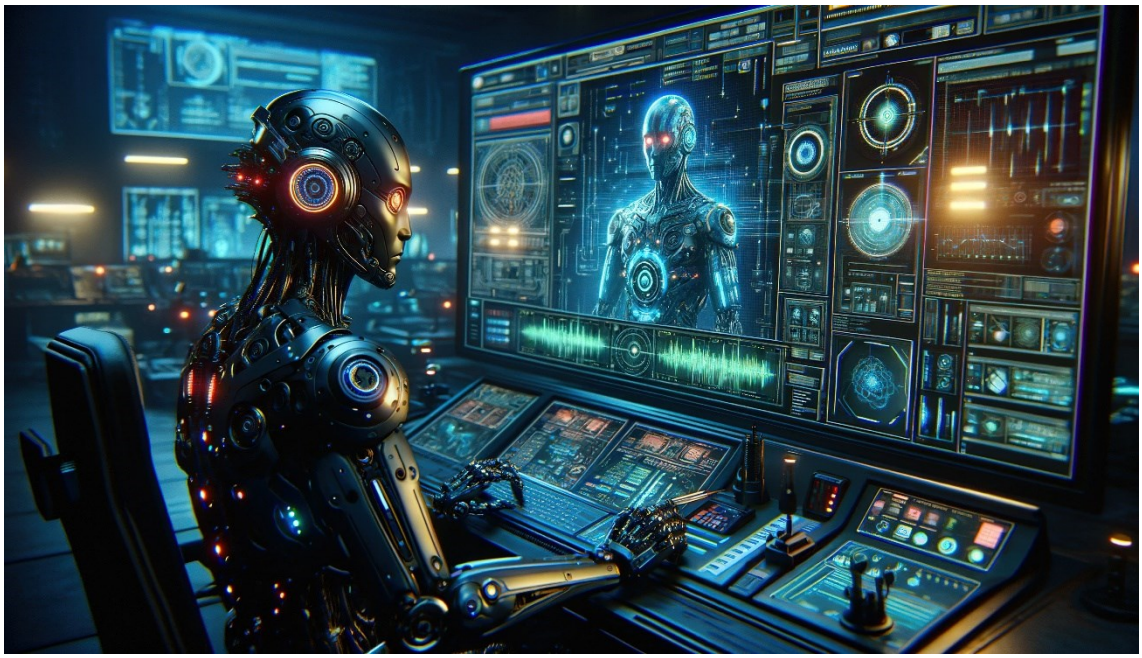


Foto generada por DALL·E

4. El ciclo de vida del desarrollo de software es fundamental para asegurar la calidad del producto final y hacer que el cliente esté satisfecho. Además cada modelo y cada metodología de abarca aspectos más específicos que también son esenciales para cumplir con los estándares de calidad.

Teniendo en cuenta esto podemos hablar del modelo Agile que se destaca por la capacidad para adaptarse a los cambios. Que se esté continuamente retroalimentando en base a los clientes y el ajuste del producto en desarrollo aseguran que el resultado del software final esté acorde a las necesidades del usuario, y esto es clave para la satisfacción del cliente.

Sin embargo, otro modelo que es el modelo en espiral permite gestionar de manera eficaz los riesgos enfocándose de manera iterativa, lo que es super importante para proyectos ya más complejos. Cada iteración del modelo se centra en mejorar de manera gradual el software y en enfrentarse a unos riesgos específicos, resultando en un producto final de calidad mayor y que cumple con las expectativas de seguridad y funcionalidad.

Otro modelo, el de cascada proporciona una estructura clara y secuencial, de esta manera puede ser beneficiosa en proyectos con requisitos que están bien definidos, pero que sea tan rígido puede hacer que si surgen cambios durante el desarrollo este se complique un poco. Sin embargo, cuando se aplica de forma correcta, puede resultar en entregables más predecibles y mucho más confiables que cumplen con los criterios que nos da el cliente.

También el mantenimiento es crucial para la calidad y satisfacción del cliente. Durante la fase, se descubren y corrigen errores y otros fallos que no se habían detectado en etapas anteriores, lo cual mejora la experiencia, y por lo tanto, la sensación final que da el software al usarlo. Además, que se haga un mantenimiento continuo asegura que el software este siempre actualizado para las necesidades del usuario.

Por último la integración de DevSecOps es otro factor importante para la calidad y satisfacción del cliente. Este enfoque integra la seguridad en todas las etapas del desarrollo, para asegurar que el producto no solo cumpla con las expectativas funcionales del cliente, sino que también sea seguro, lo cual es cada vez más importante.

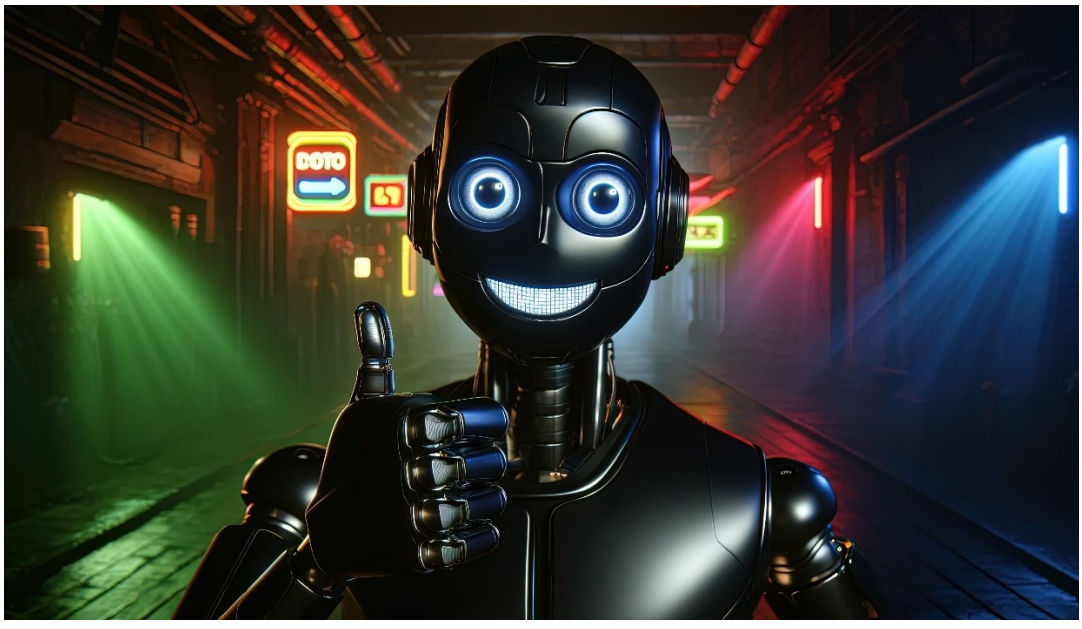


Foto generada por DALL·E

Conclusión:

En conclusión, el ciclo de la vida del software es muy importante para asegurar un mínimo de calidad y que los clientes queden satisfechos con el software. Con cada etapa, desempeñamos un papel fundamental en el éxito del producto que se le entregará al cliente, en este caso podría ser un programa. La planificación necesaria y unas pruebas llevadas a cabo con rigurosidad garantiza también que no solo se cumpla lo anterior sino que cumpla con las necesidades que cambiaran con el tiempo de los usuarios y también del mercado.

Bibliografía:

Parte 1

Ciclo de Vida del Software - Qué es, Etapas y Más (aprendeinformaticas.com)

Ciclo de vida del Desarrollo de Software: Fases y ejemplos - Vidasoft

Ciclo de Vida del Software: Etapas, Uso, Función, Modelos e Importancia - LovTechnology

Parte 2

Planeación del desarrollo de software - Reyner González (reynergonzalez.com)

Requisitos del Software: Cómo Definirlos Correctamente (informatecdigital.com)

Qué es la planificación de un proyecto: una guía completa. (administrarproyectos.com)

8 pasos esenciales para planificar un proyecto de desarrollo de software | (leojimzdev.com)

Proceso de planificación - Qué es, definición y concepto (economipedia.com)

Parte 3

¿Qué es la prueba de software y cómo funciona? | IBM

Ingeniería de Software | SDLC V-Modelo – Barcelona Geeks

Pruebas de Integración: qué son, tipos y ejemplos (qalified.com)

Pruebas de integración, la hora de la verdad para el software | OBS Business School

Verificación, validación y pruebas - Soluciones de MATLAB y Simulink - MATLAB & Simulink (mathworks.com)

Parte 4

Software Development Life Cycle (SDLC): Phases, Models, and Benefits - FutureCode IT Consulting (future-code.dev)

Understanding the SDLC: Software Development Lifecycle Explained | GitHub Resources - GitHub Resources

Software Development Life Cycle (SDLC): Phases, Models, and Benefits - FutureCode IT Consulting (future-code.dev)

The Software Development Life Cycle (SDLC): 7 Phases and 5 Models (theproductmanager.com)

Imágenes: DALL•E – OpenAI