1.A. Desarrollo del software

4. Ciclos de vida del software.

Se puede definir el ciclo de vida de un proyecto como: conjunto de fases o etapas, procesos y actividades requeridas para ofertar, desarrollar, probar, integrar, explotar y mantener un producto software.

Al principio, el desarrollo de una aplicación era un proceso individualizado, carente de planificación donde únicamente se hacía una codificación y prueba/depuración del código a desarrollar. Pero pronto se detectaron muchos inconvenientes:

- Dependencia total de la persona que programó.
- Se desconoce el progreso y la calidad del proyecto.
- Falta de flexibilidad ante cambios.
- Posible incremento del coste o incluso imposibilidad de completarlo.
- Puede no reflejar las necesidades del cliente.

De ahí surgió la necesidad de hacer desarrollos más estructurados, aportando valor añadido y calidad al producto final, apareciendo el concepto de ciclo de vida del software.

Algunas ventajas que se consiguen son:

- En las primeras fases, aunque no haya líneas de código, invertir en pensar el diseño es avanzar en la construcción del sistema, pues facilitará la codificación.
- Asegura un desarrollo progresivo, con controles sistemáticos, que permite detectar defectos con mucha antelación.
- El seguimiento del proceso permite controlar los plazos de entrega retrasados y los costes excesivos.
- La documentación se realiza de manera formal y estandarizada simultáneamente al desarrollo, lo que facilita la comunicación interna entre el equipo de desarrollo y la de éste con los usuarios.
- Aumenta la visibilidad y la posibilidad de control para la gestión del proyecto.
- Supone una guía para el personal de desarrollo, marcando las tareas a realizar en cada momento.
- Minimiza la necesidad de rehacer trabajo y los problemas de puesta a punto.

Diversos autores han planteado distintos modelos de ciclos de vida, pero los más conocidos y utilizados son los citados a continuación:

- Modelo en cascada.

  Es el modelo de vida clásico del software.

  Se pasa de unas etapas a otras sin retorno posible, cualquier error en las fases iniciales ya no será subsanable aunque sea detectado más adelante.

  Este escaso margen de error lo hace prácticamente imposible de utilizar. Sólo es aplicable en pequeños desarrollos.

  Todos los requisitos son planteados para hacer un único recorrido del proyecto.

- Modelo en cascada con realimentación.

  Es uno de los modelos más utilizados. Proviene del modelo anterior, pero se introduce una realimentación entre etapas, de forma que podamos volver atrás en cualquier momento para corregir, modificar o depurar algún aspecto. No obstante, si se prevén muchos cambios durante el desarrollo no es el modelo más idóneo.

  Es el modelo perfecto si el proyecto es rígido (pocos cambios, poco evolutivo) y los requisitos están claros.

- Modelos evolutivos.

  Son más modernos que los anteriores. Tienen en cuenta la naturaleza cambiante y evolutiva del software.

  Distinguimos las siguientes variantes:

  - Modelo iterativo incremental.

    Está basado en el modelo en cascada con realimentación, donde las fases se repiten y refinan, y van propagando su mejora a las fases siguientes.

    Cada iteración cubre una parte de los requisitos requeridos, generando versiones parciales y crecientes para el producto software en desarrollo. Cada versión obtenida será el punto de partida para la siguiente iteración.

    Los incrementos a considerar en cada vuelta ya vienen establecidos desde las etapas iniciales.

  - Modelo en espiral.

Se trata de un modelo iterativo como el anterior, pero en este caso, tras cada vuelta se lleva a cabo un análisis de riesgos y se determinan los objetivos a conseguir en las siguientes iteraciones.

Es un modelo muy abierto a cambios de requisitos, incluso una vez puesto en marcha el proyecto. El modelo en espiral corre el riesgo de alargarse excesivamente en el tiempo y aumentar el coste de su desarrollo.

- Modelos ágiles.

Se trata de modelos que están ganando gran presencia en los desarrollos software. Muy centrados en la satisfacción del cliente, muestran gran flexibilidad a la aparición de nuevos requistitos, incluso durante el desarrollo de la aplicación.
Al tratarse de metodologías evolutivas, el desarrollo es incremental, pero estos incrementos son cortos y están abiertos al solapado de unas fases con otras.
La comunicación entre los integrantes del equipo de trabajo y de éstos con el cliente son constantes.
Un ejemplo de metodología ágil es Scrum.