Capítulo

2

El Ciclo de Vida del Software

Centro Asociado Palma de Mallorca

Tutor: Antonio Rivero Cuesta

2 El Ciclo de Vida del Software

2.1 Introducción.

Definiremos qué es el ciclo de vida de un producto, cómo es el ciclo de vida del software, qué fases tiene y qué posibles esquemas organizativos pueden tener esas fases, dando lugar a diferentes ciclos de vida.

2.2 Objetivos

Conocer qué es el ciclo de vida de un producto Software.

Conocer las diferentes fases que integran este ciclo de vida, la documentación resultante de cada fase.

Conocer los diferentes enfoques en la organización del ciclo de desarrollo del software.

Conocer la fase de mantenimiento del software.

2.3 El ciclo de vida de un producto

A lo largo de su vida útil el software pasará por una secuencia de fases de su ciclo.

El ciclo de vida en la ingeniería pasa por una serie de etapas.

2.4 El ciclo de vida del software.

El proceso de desarrollo del software, incluyendo el mantenimiento necesario durante su explotación, se denomina **"Ciclo de Vida del Software**".

2.5 Las fases de un ciclo de vida del software

2.5.1 Análisis

Que debe hacer el sistema a desarrollar, se genera el SRD (Documento de especificación de requisitos). Especificación precisa y completa sin detalles internos.

2.5.2 Diseño

Descomponer y organizar sistema para hacer desarrollo en equipo, se genera el SDD (Documento de diseño del software). Describe estructura global, especifica que debe hacer cada parte y como se combinan.

2.5.3 Codificación

Se escribe y prueba código fuente para cada elemento, se genera el código fuente en lenguaje de programación elegido, con comentarios para que esté claro.

2.5.4 Integración

Combinar todos los componentes y hacer pruebas exhaustivas, se genera el sistema software, ejecutable junto con la documentación de las pruebas.

2.5.5 Explotación

No forma parte del ciclo de desarrollo de un producto software, pero influye en el resto de las fases.

Comprende el periodo de funcionamiento de la aplicación.

2.5.6 Mantenimiento

Durante la explotación habrá cambios por defectos no detectados o por mejoras, se generan documentos de cambios que tendrá información del problema, descripción de la solución y modificaciones realizadas.

2.6 Documentos que se generan en el ciclo de vida

Los trabajos realizados en cada una de las fases del ciclo de vida se describen formalmente en cada uno de los siguientes documentos.

2.6.1 Documento de especificación de requisitos.

El SRD (Software Requirements Document).

Especificación precisa y completa de lo que debe hacer el sistema, prescindiendo de los detalles internos de cómo lo debe hacer.

2.6.2 Documento de Diseño de Software (SDD)

El SDD (Software Design Document).

Descripción de la estructura global del sistema y la especificación de qué debe hacer cada una de sus partes así como la combinación de unas con otras.

2.6.3 Código fuente

Es el producto de la fase de codificación.

2.6.4 El sistema software

Es el producto ejecutable de la fase de integración.

2.6.5 Documentos de cambio

En caso que haya modificaciones debe quedar constancia escrita de las acciones realizadas.

2.7 Tipos de ciclo de vida del software

2.7.1 Ciclo de vida en cascada

Cada resultado de una fase es el elemento de entrada de la fase siguiente. Antes de comenzar una fase se establece un proceso de revisión para detectar errores que serian muy costosos de resolver si se pasase a la fase siguiente con ese lastre.

Las diferentes variantes del modelo se diferencian en el reconocimiento o no como fases separadas de ciertas actividades. Las fases de un ciclo de vida en cascada son: Análisis, Diseño, Codificación, Integración y Mantenimiento.

Este modelo trata de aislar cada fase de la siguiente, de manera que cada fase pueda ser desarrollada por grupos de personas diferentes, facilitando así la especialización.

Se hace énfasis en la necesidad de terminar correctamente cada fase antes de comenzar la siguiente. Esto se debe a que los errores producidos en una fase son muy costosos de corregir si ya se ha realizado el trabajo de las fases siguientes.

Para detectar los errores lo antes posible, y evitar que se propaguen a fases posteriores, se establece un proceso de revisión al completar cada fase, antes de pasar a la siguiente. Esta revisión se realiza fundamentalmente sobre la documentación producida en esa fase, y se hace de una manera formal, siguiendo una lista de comprobaciones establecida de antemano.

Si en alguna fase se detectan errores producidos en fases anteriores, será necesario rehacer parte del trabajo volviendo a un punto anterior en el ciclo de vida. Esto se indica con línea discontinua en las Figuras 2.1 y 2.2 de las Páginas 35 y 36.

2.7.2 El modelo en V

Se basa en una secuencia de fases análoga a la del modelo en cascada. Se representa con un diagrama bidimensional nivel (eje vertical) - desarrollo (eje horizontal).

En las actividades situadas en un nivel determinado se trabaja sobre una unidad del nivel de detalle superior, que se organiza en varias unidades del nivel de detalle inferior. Así durante la codificación se trabaja con un modulo que se construye con sentencias, en el diseño e integración se

trabaja con un sistema software que se organiza (durante el diseño) o se construye (durante la integración) con varios módulos.

Si nos fijamos en la Figura 2.3 de la Pagina 37 vemos como las fases iniciales, en la rama izquierda descendente, el sistema software se va descomponiendo en elementos cada vez más sencillos, hasta llegar a las sentencias del lenguaje de programación. A partir de aquí el sistema se va construyendo poco a poco a base de integrar los elementos que lo componen, siguiendo la rama derecha ascendente, hasta disponer del sistema completo listo para ser usado.

En esta misma figura podemos observar que el resultado de una fase no solo sirve como entrada para la fase inmediatamente siguiente, sino que también debe utilizarse en fases posteriores para comprobar que el desarrollo es correcto.

Podemos diferenciar los conceptos de Verificación y Validación:

Verificación: Comprobación de que una parte del sistema cumple con sus especificaciones, se produce a nivel de modulo.

Validación: Comprobación de que un elemento satisface las necesidades del usuario identificadas durante el análisis, se produce a nivel de sistema completo.

Ver Figura 2.4 Pagina 38.

2.8 Uso de prototipos.

En los modelos clásicos cada fase del desarrollo tiene una duración limitada en el tiempo, de forma que una vez terminada una fase pueden dedicarse a otra cosa los recursos humanos o materiales que se han empleado.

Esto significa que no se contempla de manera organizada las vueltas atrás debidas a errores detectados en fases anteriores.

El empleo de prototipos puede solucionar, al menos en parte, este problema.

Se definen como elementos auxiliares que permiten probar experimentalmente ciertas soluciones parciales a las necesidades del usuario o a los requisitos del sistema.

Para limitar el coste del desarrollo del prototipo, debemos de limitar sus funciones, su capacidad (permitiendo que procese solo unos pocos datos), su eficiencia, usar un hardware más potente, usar un software más potente.

2.8.1 Prototipos rápidos.

También se denominan Prototipos de Usar y Tirar y Maquetas (cuando su funcionamiento o capacidad es muy limitada).

Solo afectan a las fases de análisis y diseño. Experimentan algunas alternativas y garantizan en lo posible que las decisiones tomadas son correctas. Una vez finalizadas estas fases, el sistema final se codifica totalmente partiendo de cero. No se aprovecha el código del prototipo. Lo importante en estos prototipos es desarrollarlos en poco tiempo, para evitar alargar excesivamente la duración de las fases de análisis y diseño.Ver Figura 2.5 Pagina 40.

2.8.2 Prototipos evolutivos

En este caso el prototipo inicial se construye tras unas fases parciales de análisis y diseño. La experimentación con el prototipo permitirá avanzar en esas fases parciales, y a continuación ampliar el prototipo inicial para convertirlo en el sistema final mediante adiciones sucesivas. Al mismo tiempo los documentos de especificación, diseño, etc…, van siendo también desarrollados progresivamente.

Ver Figura 2.6 Pagina 41.

Este modelo puede considerarse como un proceso iterativo en bucle sobre el modelo en cascada, de manera que en cada iteración se hace solo una parte del desarrollo, avanzando un poco en cada fase.

Cada iteración utiliza todo lo que se ha generado en la iteración anterior y produce un nuevo prototipo, que es una nueva versión parcial del sistema, hasta llegar finalmente al sistema completo, con lo que se sale del bucle de iteraciones y termina el proceso.

2.8.3 Herramientas para la realización de prototipos

Si el prototipo es evolutivo se usaran las mismas herramientas que en el sistema definitivo, si es rápido se podrán usar diferentes. Las herramientas más idóneas para construir prototipos son:

Las de 4ª generación: Que se apoyan en bases de datos de uso general, formularios de entradas de datos, y formatos de informes.

Los lenguajes de 4ª generación: Tiene un estilo declarativo no operacional, describen el resultado que se desea obtener, en vez de describir las operaciones para conseguirlo, tales como Prolog y SmallTalk.

Los lenguajes de especificación: Tienen como objetivo formalizar la especificación de requisitos de un sistema software, y disponiendo de un compilador tendremos un prototipo ejecutable

El uso extensivo de la reutilización del software: Es otra técnica para la construcción de prototipos, que se basa en la utilización de módulos o subsistemas escritos de antemano.

2.9 El modelo en espiral.

La característica principal del modelo en espiral es la introducción de la actividad de análisis de riesgo, como elemento fundamental para guiar la evolución del proceso de desarrollo.

En la representación de este modelo podemos observar:

- **Unos Ejes Cartesianos:** cada cuadrante contiene una clase de actividad (Planificación, Análisis de Riesgo, Ingeniería y Evaluación).
- **Una Dimensión Radial:** nos indica el esfuerzo total realizado hasta cada momento. Siempre es creciente.
- **Una Dimensión Angular:** nos indica un avance relativo en el desarrollo de las actividades de cada cuadrante.
- **Ciclo de la Espiral:** en cada ciclo de la espiral se realiza una parte del desarrollo total, siguiendo la secuencia de las 4 clases de actividades.

Planificación: Sirve para establecer el contexto del desarrollo y decidir que parte del mismo se abordara en ese ciclo de la espiral.

Análisis de riesgo: Consiste en evaluar diferentes alternativas para la realización de la parte de desarrollo elegida, seleccionando la más ventajosa.

Ingeniería: Son las indicadas en los modelos clásicos: análisis, diseño, codificación, integración y mantenimiento, su resultado será ir obteniendo en cada ciclo una versión más completa del sistema.

Evaluación: Analiza los resultados de la fase de ingeniería con la colaboración del cliente y este resultado se tiene como información de entrada para la planificación del ciclo siguiente.

Tendremos diferentes variantes del modelo en espiral, según que parte del desarrollo se decida hacer en cada ciclo.

Ver Figura 2.7 Pagina 42.

2.10 Programación extrema

Los clientes suelen ser exigentes en cuanto a plazos de entrega del producto.

El objetivo de la programación extrema es ser capaz de responder de una forma rápida y de calidad a las exigencias del cliente.

El equipo de desarrollo se basa en cuatro valores principales:

- **Sencillez**: se debe programar sólo aquello que nos han pedido.
- **Comunicación**: se potencia el trabajo en equipo, dentro del grupo y con el cliente.
- **Retroalimentación**: los ciclos de proceso son muy cortos, en muy poco tiempo se evalúa el diseño y se cambia si no cumple los requisitos del cliente.
- **Valentía**: permite a los programadores afrontar la recodificación continua del programa.

2.11 Mantenimiento del software.

También llamada etapa de explotación y mantenimiento, consiste en repetir o rehacer parte de las actividades de las fases anteriores para introducir cambios en una aplicación que ha sido entregada al cliente.

2.11.1 Evolución de las aplicaciones.

Tenemos tres tipos de mantenimiento que son:

- **Correctivo**: Corrige errores que no se detectaron durante el desarrollo inicial.
- **Adaptativo**: Se da en aplicaciones que tienen un tiempo de vigencia elevado y es necesario ir adaptando la interfaz que corre sobre ellas.
- **Perfectivo**: Se da en aplicaciones en las que existe competencia de mercado, para optimizar las sucesivas versiones. También cuando las necesidades iniciales del usuario han sido modificadas.

2.11.2 Gestión de cambios.

Se pueden tener dos enfoques a la hora de hacer cambios en el software.

Como un nuevo desarrollo: Si afecta a la mayoría de los componentes, aplicando un nuevo ciclo de vida desde el principio, (aunque aprovechando lo ya desarrollado).

Como modificación de algunos elementos si afecta a una parte localizada del producto.

El cambio de código de algunos módulos puede requerir modificar los documentos de diseño o incluso en el caso del mantenimiento perfectivo, modificar el SRD.

La realización de cambios se controla mediante el informe de problema y el informe de cambio.

***Informe de Problema*:** describe una dificultad en la utilización del producto que requiere alguna modificación para subsanarla.

***Informe de Cambio*:** describe la solución dada a un problema y el cambio realizado en el producto software.

El Informe de Problema puede ser originado por los usuarios. Este informe se pasa a un grupo de Ingeniería para la comprobación y la caracterización del problema y luego a un grupo de Gestión para decidir la solución a adoptar.

Este grupo de Gestión inicia el Informe de Cambio, que se pasa de nuevo al grupo de Ingeniería para su desarrollo y ejecución.

2.11.3 Reingeniería.

Las técnicas de ingeniería inversa o reingeniería sirven para ayudar al mantenimiento de aplicaciones antiguas que no fueron desarrolladas siguiendo las técnicas de ingeniería del software, con su respectiva documentación.

Se toma el código fuente y a partir de él, se construye de forma automática alguna documentación, en particular, de diseño, con la estructura modular y las dependencias entre módulos y funciones (a veces será necesario reconstruir a mano la documentación y la modificación del código fuente).

2.12 Garantía de calidad de software

Las actividades del ciclo de vida que inciden sobre la calidad del producto software son las revisiones y pruebas.

2.12.1 Plan de garantía de calidad:

Las técnicas de Ingeniería de Software tratan de formalizar el proceso de desarrollo evitando los inconvenientes de una producción artesanal informal. Esta organización del proceso de desarrollo de software debe materializarse en un documento formal denominado Plan de Garantía de Calidad.

SQAP: Es el Plan de Garantía de Calidad del Software, debe contemplar los siguientes aspectos:

- Organización de los equipos de personas, dirección y seguimiento del desarrollo.
- Modelo de ciclo de vida a seguir (fases y actividades).
- Documentación requerida (con contenido y guión).
- Revisiones y auditorias que se harán durante el desarrollo.
- Organización de las pruebas que se harán sobre el producto a diferentes niveles.
- Organización de la etapa de mantenimiento, especificando como ha de gestionarse la realización de cambios del producto ya en explotación.

2.12.2 Revisiones

Consiste en inspeccionar el resultado de una actividad para determinar si es aceptable o tiene defectos que se deben corregir, se suelen aplicar a la documentación generada en cada fase.

Las revisiones se deben formalizar siguiendo las siguientes pautas:

- Se realizaran por un grupo de personas de 3 a 5.
- No debe ser realizada por los autores del producto.
- No se debe revisar ni el productor ni el proceso de producción, sino el producto.
- Se ha de establecer previamente una lista formal de comprobaciones a realizar, si la decisión ha de decidir la aceptación o no del producto.
- Debe levantarse Acta con los puntos de discusión y las decisiones adoptadas.

Este documento es el producto de la revisión.

2.12.3 Pruebas

Consisten en hacer funcionar un producto software bajo unas condiciones determinadas y comprobar si los resultados son los correctos.

El objetivo es descubrir los errores contenidos en el software.

Si no se descubre ningún error no se garantiza la calidad del producto. Una prueba tiene éxito si se descubre algún error, con lo que se sabe que el producto no cumple con algún criterio de calidad,

por el contrario, si la prueba no descubre ningún error , no se garantiza con ello la calidad del producto, ya que pueden existir otros errores que habrían de descubrirse mediante pruebas diferentes.

2.12.4 Gestión de configuración

La configuración de software es la manera en que diversos elementos se combinan para constituir un producto software bien organizado tanto desde el punto de vista de la explotación por el usuario como de su desarrollo o mantenimiento.

Para cubrir la doble visión del software, desde el punto de vista del usuario y del desarrollador, habremos de considerar como elementos componentes de la configuración todos los que intervienen en el desarrollo y no solo los que forman parte del producto entregado al cliente.

El problema de la Gestión de Configuración es que estos elementos evolucionan a lo largo del desarrollo y la explotación del producto software, dando lugar a diferentes configuraciones particulares, compuestas de diferentes elementos.

Los elementos individuales evolucionan a base de construir sucesivas versiones de los mismos.

Para mantener bajo control la configuración o configuraciones software hay que apoyarse en técnicas particulares de "Control de Versiones" y "Control de Cambios".

El control de versiones: Consiste en almacenar de forma organizada las sucesivas versiones de cada elemento de la configuración

El control de cambios: Consiste en garantizar que las diferentes configuraciones del software se componen de elementos compatibles entre sí. Este control se realiza normalmente usando el concepto de *Linea Base*.

Línea base: Es una configuración particular del sistema, cada línea se construye a partir de otra mediante la inclusión de ciertos cambios, que pueden ser la adición o supresión de elementos, o la sustitución de algunos por versiones nuevas de los mismos.

La aceptación de los cambios y la consiguiente creación de la nueva Línea Base ha de controlarse mediante pruebas o revisiones apropiadas, para garantizar la corrección de la nueva configuración.

Línea base congelada: Se le da este nombre a la antigua línea base que se modificó y no se borró. Se borrará cuando se esté seguro de no necesitarla más.

2.12.5 Normas y estándares

Son normas y recomendaciones sobre los diferentes aspectos del desarrollo del software.

Algunos estándares son:

- IEEE.
- DoD.
- ESA.
- ISO.
- CMMI
- MÉTRICA-2.
- MÉTRICA-3.