1.3. CICLO DE VIDA DEL SOFTWARE

El proceso de desarrollo del software implica un conjunto de actividades que se tienen que planificar y gestionar de tal manera que aseguren un producto final que dé solución a las necesidades de todas aquellas personas que lo van a utilizar.

1.3.1. Definición

El estándar ISO/IEC 12207-1 define ciclo de vida del software como: *Un marco de referencia que contiene los procesos, las actividades y las tareas involucradas en el desarrollo, la explotación y el mantenimiento de un producto de software, abarcando la vida del sistema desde la definición de los requisitos hasta la finalización de su uso.*

El ciclo de vida de un producto software comprende el periodo que transcurre desde que el producto es concebido hasta que deja de estar disponible o es retirado. Normalmente, se divide en etapas y en cada etapa se realizarán una serie de tareas. Usualmente se consideran las siguientes etapas: especificación y análisis de requisitos, diseño del sistema, implementación del software, aplicación y pruebas, entrega y mantenimiento:

1. **Análisis.** Construye un modelo de los requisitos. En esta etapa se debe entender y comprender de forma detallada el problema que se va a resolver. Es muy importante producir en esta etapa una documentación entendible, completa y fácil de verificar y modificar.
2. **Diseño.** En esta etapa ya sabemos qué es lo que hay que hacer, ahora hay que definir cómo se va a resolver el problema. Se deducen las estructuras de datos, la arquitectura de software, la interfaz de usuario y los procedimientos. Por ejemplo, en esta etapa hay que seleccionar el lenguaje de programación, el Sistema Gestor de Bases de Datos, etc.
3. **Codificación.** En esta etapa se traduce lo descrito en el diseño a una forma legible por la máquina. La salida de esta fase es código ejecutable.
4. **Pruebas.** Se comprueba que se cumplen criterios de corrección y calidad. Las pruebas deben garantizar el correcto funcionamiento del sistema.
5. **Mantenimiento.** Esta fase tiene lugar después de la entrega del software al cliente. En ella hay que asegurar que el sistema pueda adaptarse a los cambios. Se producen cambios porque se han encontrado errores, es necesario adaptarse al entorno (por ejemplo se ha cambiado de sistema operativo) o porque el cliente requiera mejoras funcionales.

Cada etapa tiene como entrada uno o varios documentos procedentes de las etapas anteriores y produce otros documentos de salida, por ello una tarea importante a realizar en cada etapa es **la documentación**.

1.3.2. Modelos de ciclo de vida

Existen varios modelos de ciclo de vida, es importante tener en cuenta las características del proyecto software para elegir un modelo u otro. Los modelos más importantes son: *Cascada*, *Incremental* y *Evolutivo*.

1.3.2.1. Ciclo de vida en cascada

En este modelo las etapas para el desarrollo del software tienen un orden, de tal forma que para empezar una etapa es necesario finalizar la etapa anterior; después de cada etapa se realiza una revisión para comprobar si se puede pasar a la siguiente, véase Figura 1.4 (1). Este modelo permite hacer iteraciones, por ejemplo, durante la etapa de mantenimiento del producto el cliente requiere una mejora, esto implica que hay que modificar algo en el diseño, lo cual significa que habrá que hacer cambios en la codificación y se tendrán que realizar de nuevo las pruebas, es decir, si se tiene que volver a una de las etapas anteriores hay que recorrer de nuevo el resto de las etapas, véase Figura 1.4 (2).

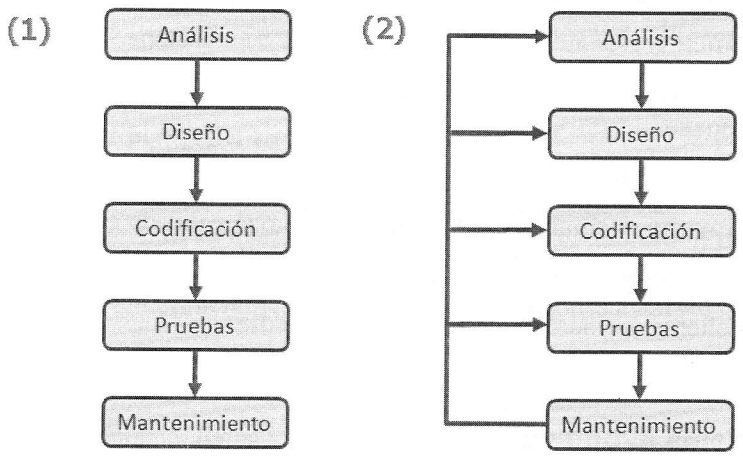

Figura 1.4. Modelo en Cascada.

Tiene varias variantes, una de la más utilizada es la que produce una realimentación entre etapas, se la conoce como *Modelo en Cascada con Realimentación*. Por ejemplo, supongamos que la etapa de *Análisis* (captura de requisitos) ha finalizado y se puede pasar a la de *Diseño*. Durante el desarrollo de esta etapa se detectan fallos (los requisitos han cambiado, han evolucionado, ambigüedades en la definición de los mismos, etc.), entonces será necesario retornar a la etapa anterior, realizar los ajustes pertinentes y continuar de nuevo con el *Diseño*. A esto se le conoce como realimentación, pudiendo volver de una etapa a la anterior o incluso de varias etapas a la anterior, véase Figura 1.5.

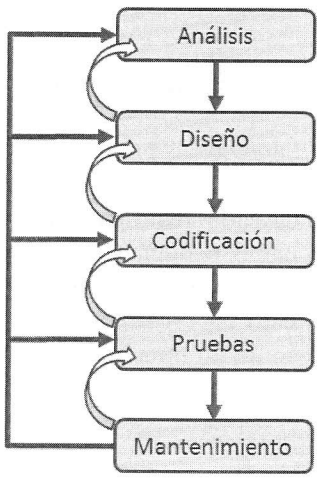

Figura 1.5. Modelo en Cascada con Realimentación.

Ventajas:

- Fácil de comprender, planificar y seguir.
- La calidad del producto resultante es alta.
- Permite trabajar con personal poco cualificado.

Inconvenientes:

- La necesidad de tener todos los requisitos definidos desde el principio (algo que no siempre ocurre ya que pueden surgir necesidades imprevistas).
- Es difícil volver atrás si se cometen errores en una etapa.
- El producto no está disponible para su uso hasta que no está completamente terminado.

Se recomienda cuando:

- El proyecto es similar a alguno que ya se haya realizado con éxito anteriormente.
- Los requisitos son estables y están bien comprendidos.
- Los clientes no necesitan versiones intermedias.

1.3.2.2. Modelos evolutivos

El software evoluciona con el tiempo, es normal que los requisitos del usuario y del producto cambien conforme se desarrolla el mismo. La competencia en el mercado del software es tan grande que las empresas no pueden esperar a tener un producto totalmente completo para lanzarlo al mercado, en su lugar se van introduciendo versiones cada vez más completas que de alguna manera alivian las presiones competitivas.

El modelo en cascada asume que se va a entregar un producto completo, en cambio los modelos evolutivos permiten desarrollar versiones cada vez más completas hasta llegar al producto final deseado. En estos modelos se asume que las necesidades del usuario no están completas y se requiere una vuelta a planificar y diseñar después de cada implantación de los entregables.

Los modelos evolutivos más conocidos son: el *Iterativo incremental* y el *Espiral*.

MODELO ITERATIVO INCREMENTAL

Está basado en varios ciclos cascada realimentados aplicados repetidamente. El modelo incremental entrega el software en partes pequeñas, pero utilizables, llamadas «incrementos». En general, cada incremento se construye sobre aquél que ya ha sido entregado[2]. En la Figura 1.6 se muestra un diagrama del modelo bajo un esquema temporal, se observa de forma iterativa el modelo en cascada para la obtención de un nuevo incremento mientras progresa el tiempo en el calendario (por simplificar el diagrama se ha puesto el modelo en cascada más básico, el lineal secuencial).

[2] Ingeniería del software. Un enfoque práctico. Roger S. Pressman.

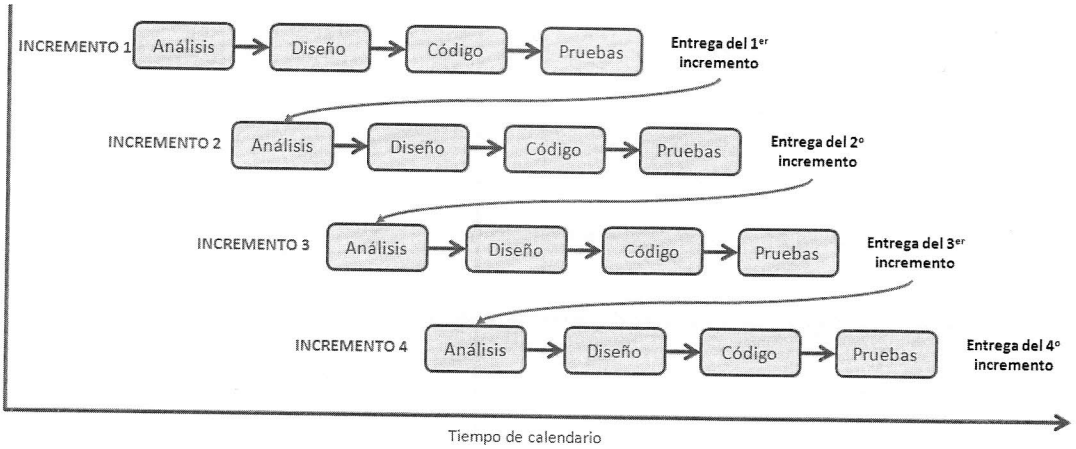

Figura 1.6. Modelo Iterativo Incremental.

Como ejemplo de software desarrollado bajo este modelo se puede considerar un procesador de textos, en el primer incremento se desarrollan funciones básicas de gestión de archivos y de producción de documentos; en el segundo incremento se desarrollan funciones gramaticales y de corrección ortográfica, en el tercer incremento se desarrollan funciones avanzadas de paginación, y así sucesivamente.

Ventajas:

- No se necesitan conocer todos los requisitos al comienzo.
- Permite la entrega temprana al cliente de partes operativas del software.
- Las entregas facilitan la realimentación de los próximos entregables.

Inconvenientes:

- Es difícil estimar el esfuerzo y el coste final necesario.
- Se tiene el riesgo de no acabar nunca.
- No recomendable para desarrollo de sistemas de tiempo real, de alto nivel de seguridad, de procesamiento distribuido, y/o de alto índice de riesgos.

Se recomienda cuando:

- Los requisitos o el diseño no están completamente definidos y es posible que haya grandes cambios.
- Se están probando o introduciendo nuevas tecnologías.

MODELO EN ESPIRAL

Este modelo combina el *Modelo en Cascada* con el modelo iterativo de construcción de prototipos. El proceso de desarrollo del software se representa como una espiral, donde en cada ciclo se desarrolla una parte del mismo. Cada ciclo está formado por cuatro fases, véase Figura 1.7, y cuando termina produce una versión incremental del software con respecto al ciclo anterior. En este aspecto se parece al *Modelo Iterativo Incremental* con la diferencia de que en cada ciclo se tiene en cuenta el análisis de riesgos.

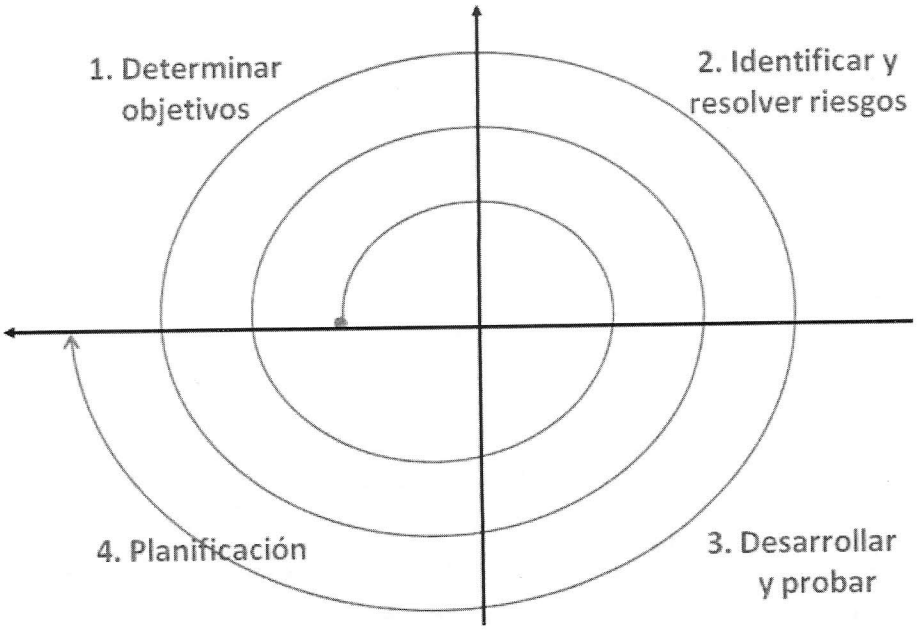

Figura 1.7. Modelo en espiral.

Durante los primeros ciclos la versión incremental podría ser maquetas en papel o modelos de pantallas (prototipos de interfaz); en el último ciclo se tendría un prototipo operacional que implementa algunas funciones del sistema. Para cada ciclo, los desarrolladores siguen estas fases:

1. **Determinar objetivos.** Cada ciclo de la espiral comienza con la identificación de los objetivos, las alternativas para alcanzar los objetivos (diseño A, diseño B, reutilización, compra, etc.), y las restricciones impuestas a la aplicación de las alternativas (costos, plazos, interfaz, etc.)
2. **Análisis del riesgo.** A continuación hay que evaluar las alternativas en relación con los objetivos y limitaciones. Con frecuencia, en este proceso se identifican los riesgos involucrados y (si es posible) la manera de resolverlos. Un riesgo puede ser cualquier cosa: requisitos no comprendidos, mal diseño, errores en la implementación, etc. Utiliza la construcción de prototipos (representación limitada de un producto) como mecanismo de reducción de riesgos.
3. **Desarrollar y probar.** Desarrollar la solución al problema en este ciclo, y verificar que es aceptable.
4. **Planificación.** Revisar y evaluar todo lo que se ha hecho, y con ello decidir si se continúa, entonces hay que planificar las fases del ciclo siguiente.

La Figura 1.8 muestra un ejemplo de este modelo con cuatro ciclos. En el ciclo más interno se comienza con los requisitos y un plan inicial de desarrollo; se evalúan los riesgos y se construyen prototipos de las alternativas. Para terminar, se construye un documento con «el concepto de las operaciones» que describen la funcionalidad del sistema. Al final del ciclo se genera el plan de requisitos del sistema y el plan para todo el ciclo de vida útil.

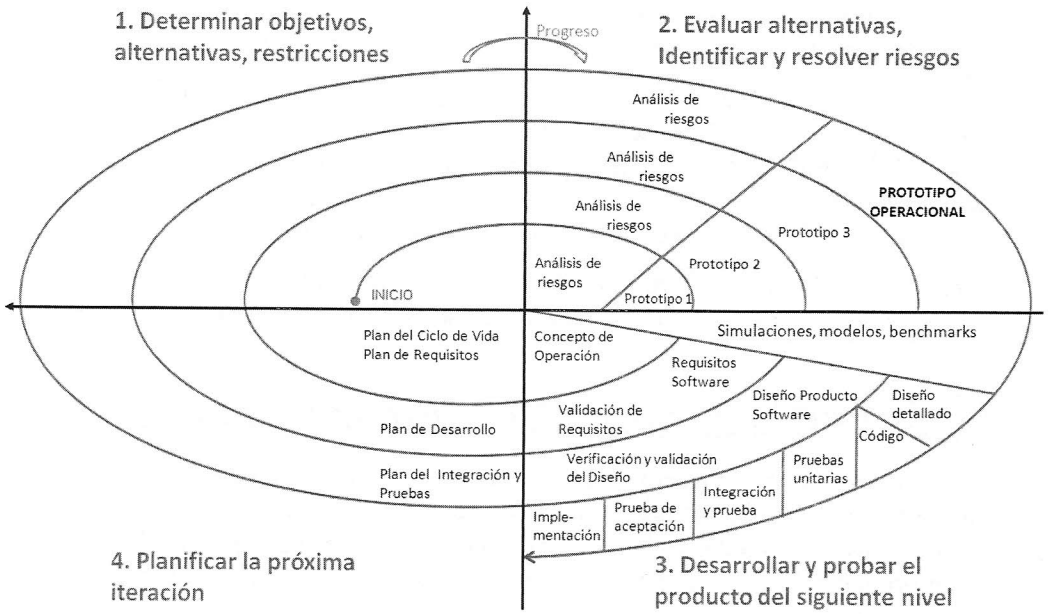

Figura 1.8. Ejemplo de Modelo en espiral.

A partir de aquí comienza el segundo ciclo, el resultado será la especificación y validación de los requisitos del software y la elaboración del plan de desarrollo. En el tercer ciclo se hace el diseño del producto software, la verificación y validación del diseño y se produce el plan de integración y pruebas. En el cuarto ciclo los desarrolladores producen un diseño detallado; implementan los módulos, realizan pruebas unitarias, de integración y aceptación. En este punto del desarrollo se puede haber alcanzado el éxito o no, en cuyo caso será necesario otro ciclo.

En todos los ciclos se hace un análisis de riesgo, donde se evalúan las alternativas según los requisitos y restricciones, y se construyen prototipos para analizarlas y seleccionar una. Estos prototipos pueden ser modelos en papel, modelos de pantallas, simulaciones del sistema, prototipos operacionales, y dependerán del riesgo a evaluar, del ciclo en el que se esté y del tipo de aplicación.

Ventajas:

- No requiere una definición completa de los requisitos para empezar a funcionar.
- Análisis del riesgo en todas las etapas.
- Reduce riesgos del proyecto
- Incorpora objetivos de calidad

Inconvenientes:

- Es difícil evaluar los riesgos.
- El costo del proyecto aumenta a medida que la espiral pasa por sucesivas iteraciones.
- El éxito del proyecto depende en gran medida de la fase de análisis de riesgos.

Se recomienda para:

- Proyectos de gran tamaño y que necesitan constantes cambios.
- Proyectos donde sea importante el factor riesgo.