4 Ciclo de vida del software

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.*

4.1 Ciclo de vida del software

Un ciclo de vida es el conjunto de fases por las que un sistema software pasa desde que surge la idea hasta que muere.

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.

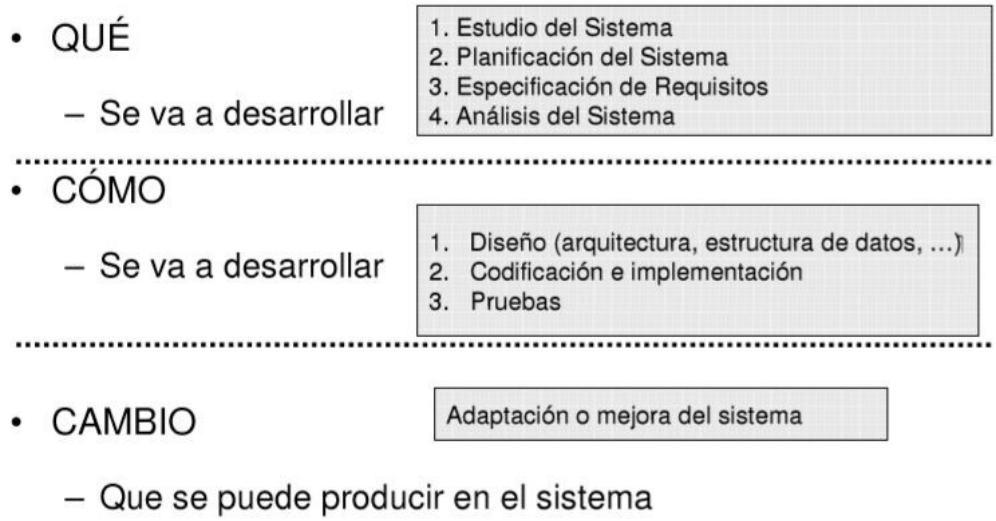
En función de cuáles sean las características del proyecto, se configurará el ciclo de vida de forma diferente en cada caso.

De los ciclos de vida destacan ciertos aspectos claves que son: las fases, las transiciones entre fases y las posibles entradas y salidas que surgen en cada transición.

Un aspecto esencial dentro de las tareas del desarrollo del software es la documentación de todos los elementos y especificaciones en cada fase.

4.2 Etapas principales para realizar en cualquier ciclo de vida

1. Análisis
2. Diseño
3. Codificación
4. Pruebas
5. Documentación
6. Explotación
7. Mantenimiento



4.2.1 Análisis

Construye un modelo de los requisitos.

- Comienza con una entrevista al cliente, que establecerá lo que quiere y dará una idea global de lo que necesita, pero no necesariamente del todo acertada.
- Es necesaria habilidad y experiencia para reconocer requisitos incompletos, ambiguos, contradictorios o incluso innecesarios.
- Es necesario un contraste y un consenso por ambas partes para llegar a definir los requisitos verdaderos del software.
- Para ello se crea un informe ERS (Especificación de Requisitos del Sistema) acompañado del diagrama de clases o de Entidad/Relación.
- Es fundamental, tener claro QUÉ hay que hacer (comprender elproblema).

4.2.2 Diseño

Parte del modelo de análisis.

- Deduce las estructuras de datos, la estructura en la que se descompone el sistema (estructura modular), el formato de entrada/salida de datos, y la interfaz de usuario.
- Determina el funcionamiento del sistema de forma global, sin entrar en detalles. Boceto de interfaz.
- Establece las necesidades de recursos del sistema de software, tanto físicos como lógicos.
- Se crean los diagramas de casos de uso y de secuencia para definir la funcionalidad del sistema. Utiliza lenguajes de modelado para hacer diagramas, por ejemplo UML.

4.2.3 Codificación

Construye el sistema. La salida de esta fase es código ejecutable.

4.2.4 Pruebas

Se comprueba que se cumplen criterios de corrección y calidad.

- Buscan confirmar que la codificación ha sido exitosa y el software no contiene errores.
- Comprueba que el software hace lo que debe hacer.
- También se realizan pruebas después de la etapa de documentación para corroborar que esta es de calidad y satisfactoria para el buen uso de la aplicación.
- Las pruebas las realiza, personal diferente al que codificó la aplicación.

4.2.5 Documentación

1. Documentación disponible para el usuario.
2. Documentación destinada al propio equipo de desarrollo.

Del usuario:

- Muestra información completa y de calidad.
- Indica cómo manejar la aplicación.
- Permite al usuario sin información previa, comprender el propósito y el modo de uso de la aplicación o adicional.

Documentación técnica:

- Explica el funcionamiento interno del programa.
- Explica la codificación del mismo.
- Permite a un equipo de desarrollo cualquiera entender el programa y modificarlo si fuera necesario.

4.2.6 Explotación

Prepara el software para su distribución.

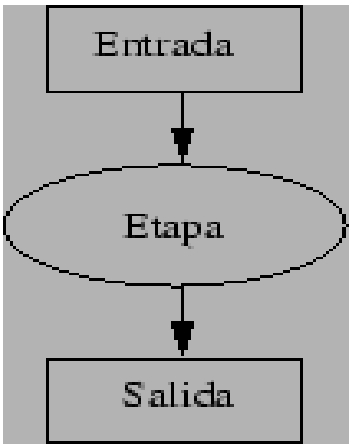
- Implementa el software en el sistema elegido o se prepara para que se implemente por sí solo de manera automática.
- Cuando el software es una versión sustitutiva de otro anterior, hay que valorar si conviene que ambas aplicaciones convivan durante un proceso de adaptación.

4.2.7 Mantenimiento

En esta fase, que tiene lugar después de la entrega, se asegura que el sistema siga funcionando y adaptándose a nuevos requisitos.

- Se arreglan los fallos o errores que suceden cuando el programa ya ha sido implementado en un sistema.
- Se realizan las ampliaciones necesitadas o requeridas.

Las etapas constan de tareas. La documentación es una tarea importante que se realiza en todas las etapas. Cada etapa tiene como entrada uno o varios documentos procedentes de las etapas anteriores y produce otros documentos de salida según se muestra en la figura.



Las formas de organizar y estructurar la secuencia de ejecución de las tareas en las diferentes fases dan lugar a un tipo de ciclo de vida diferente.

Los principales ciclos de vida que se van a presentar a continuación realizan estas tareas.

Cada uno de ellos tiene sus ventajas e inconvenientes.

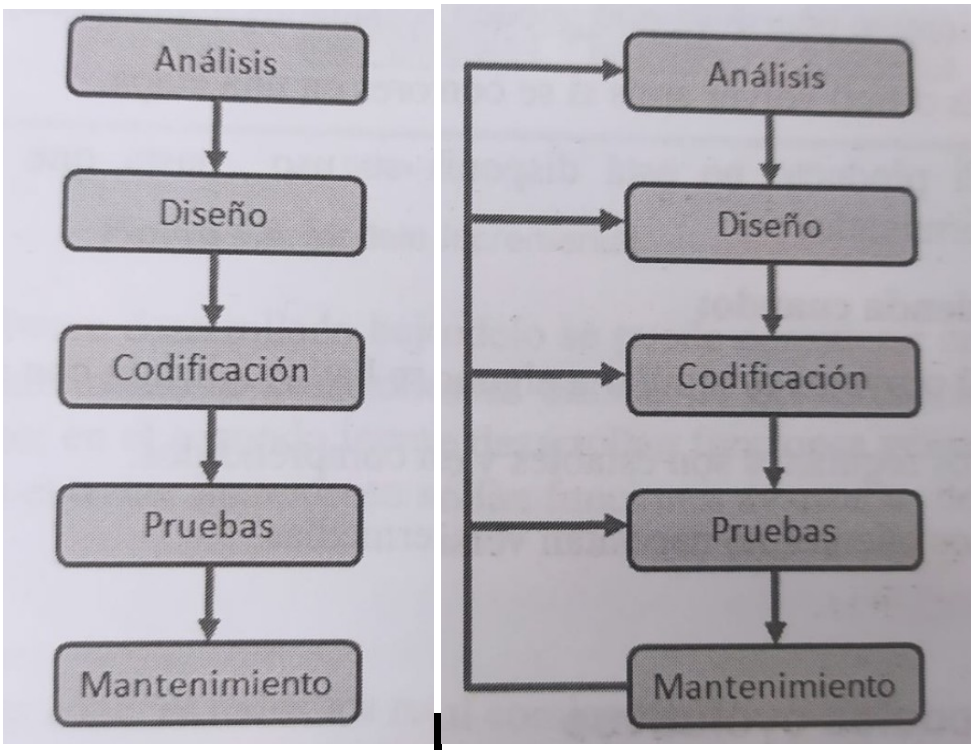
Práctica 1.2:

Desarrolla para exponer los siguientes modelos de ciclos de vida:

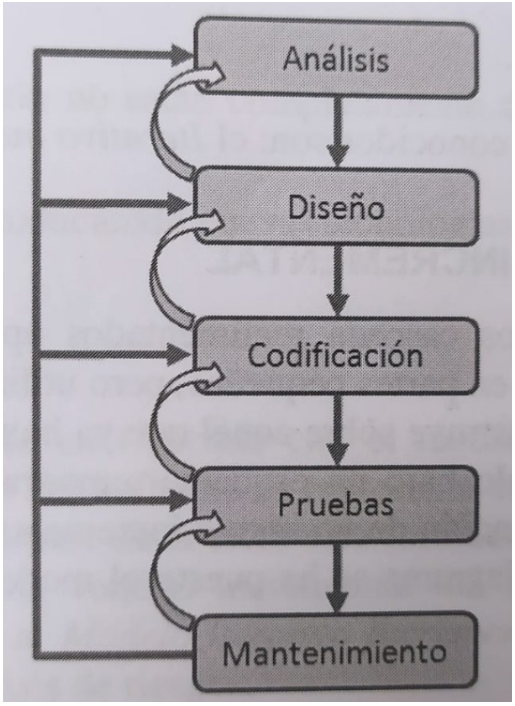
• En cascada • Iterativo incremental • Modelo en espiral • Modelo en V	➢ Características ➢ Ventajas e inconvenientes respecto a otros modelos

4.3 Modelo de 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. 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.



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 estos, 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 varias etapas a la anterior.



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 comenten errores en una etapa.
- El producto no está disponible para su uso hasta que no está completamente terminado.

Se recomienda cuando:

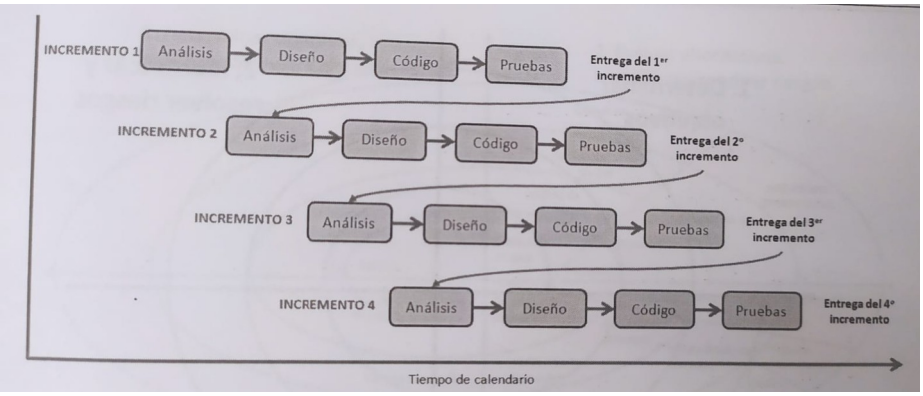
- El proyecto es similar a alguno que ya se haya realizado con éxito anteriormente.
- Los requisitos son estables y están bien comprendidos.
- Los clientes no necesitan versiones intermedias.

Los documentos son:

- Análisis: Toma como entrada una descripción en lenguaje natural de lo que quiere el cliente. Produce el S.R.D. (Software Requirements Document).
- Diseño: Su entrada es el S.R.D. Produce el S.D.D. (Software Design Document).
- Codificación: A partir del S.D.D. produce módulos. En esta fase se hacen también pruebas de unidad.
- Pruebas: A partir de los módulos probados se realiza la integración y pruebas de todo el sistema. El resultado de las pruebas es el producto final listo para entregar.

4.4 Modelo de ciclo de vida 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. En la imagen 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).



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.

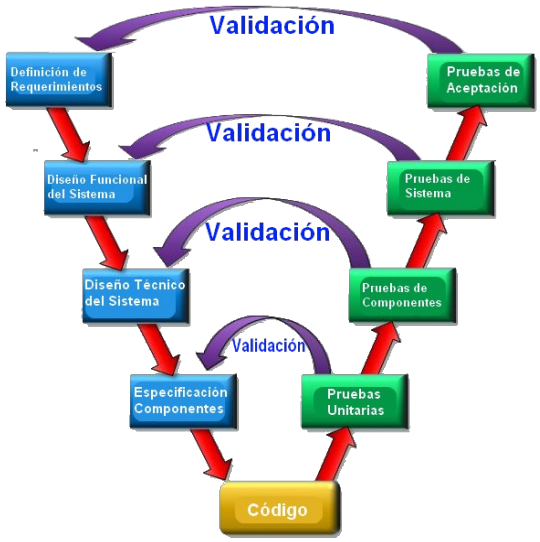
Se recomienda cuando:

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

4.5 Modelo de ciclo de vida en V

El modelo en V viene para corregir los fallos del modelo en cascada, incluyendo explícitamente la retroalimentación.

El problema es que los fallos detectados en fases tardías de un proyecto obligaban a empezar de nuevo, lo cual supone un gran sobrecoste.



Ventajas

- Es un modelo sencillo
- Facilita la localización de fallos con tipos de pruebas específicos.

Desventajas

- No se obtiene el producto hasta el final del ciclo de vida.

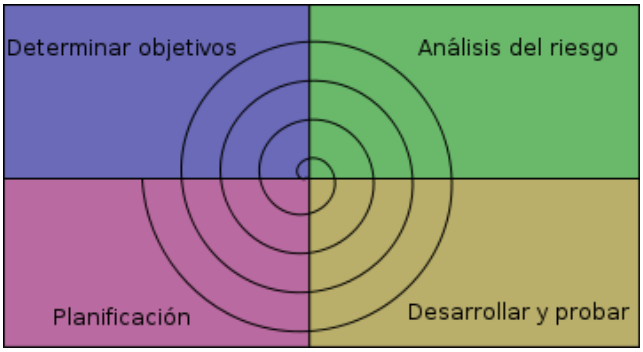
4.6 Modelo de ciclo de vida en espiral

El proceso de desarrollo de software se representa como una espiral, donde en cada ciclo se desarrolla una parte del mismo.

Cada ciclo está formado por cuatro:

- **Determinar objetivos**: Se identifican los objetivos, se ven las alternativas para alcanzarlos y las restricciones impuestas a la aplicación (costos, plazos, interfaz …).

- **Análisis de riesgos:** Un riesgo es por ejemplo, requisitos no comprendidos, mal diseño, errores en la implementación, etc. Utiliza la construcción de prototipos como mecanismo de reducción de riesgos.
- **Desarrollar y probar:** Se desarrolla la solución al problema en este ciclo y se verifica que es aceptable.
- **Planificación:** Revisar y evaluar todo lo realizado y con ello decidir si se continúa, entonces hay que planificar las fases del ciclo siguiente.



En cada iteración se debe recopilar la siguiente información:

- Objetivos:
- Alternativas
- Restricciones
- Riesgos
- Resolución de riesgos
- Resultados
- Planes
- Compromiso

Objetivos: Se hacen entrevistas a los clientes, se les hace rellenar cuestionarios, etc.

Alternativas: Diferentes formas de conseguir los objetivos. Se consideran desde dos puntos de vista:

- Características del producto.
- Formas de gestionar el proyecto.

Restricciones:

- Desde el punto de vista del producto:
  - interfaces de tal o cual manera,
  - rendimiento,
  - etc.
- Desde el punto de vista organizativo:
  - Coste, tiempo, personal, etc.

Riesgos: Lista de riesgos identificados.

Resolución de riesgos: La técnica más usada es la construcción de prototipos.

Resultados: Lo que realmente ha ocurrido después de la resolución de riesgos.

Planes: Lo que se va a hacer en la siguiente fase.

Compromiso: Decisiones de gestión sobre como continuar.

Al terminar una iteración se comprueba que lo que se ha hecho efectivamente cumple con los requisitos establecidos.

Se verifica que funciona correctamente.

El propio cliente evalúa el producto.

No existe una diferencia muy clara entre cuando termina el proyecto y cuando empieza la fase de mantenimiento.

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