
CIPFP Mislata

Centre Integrat Públic
Formació Professional Superior

# Ingeniería del software

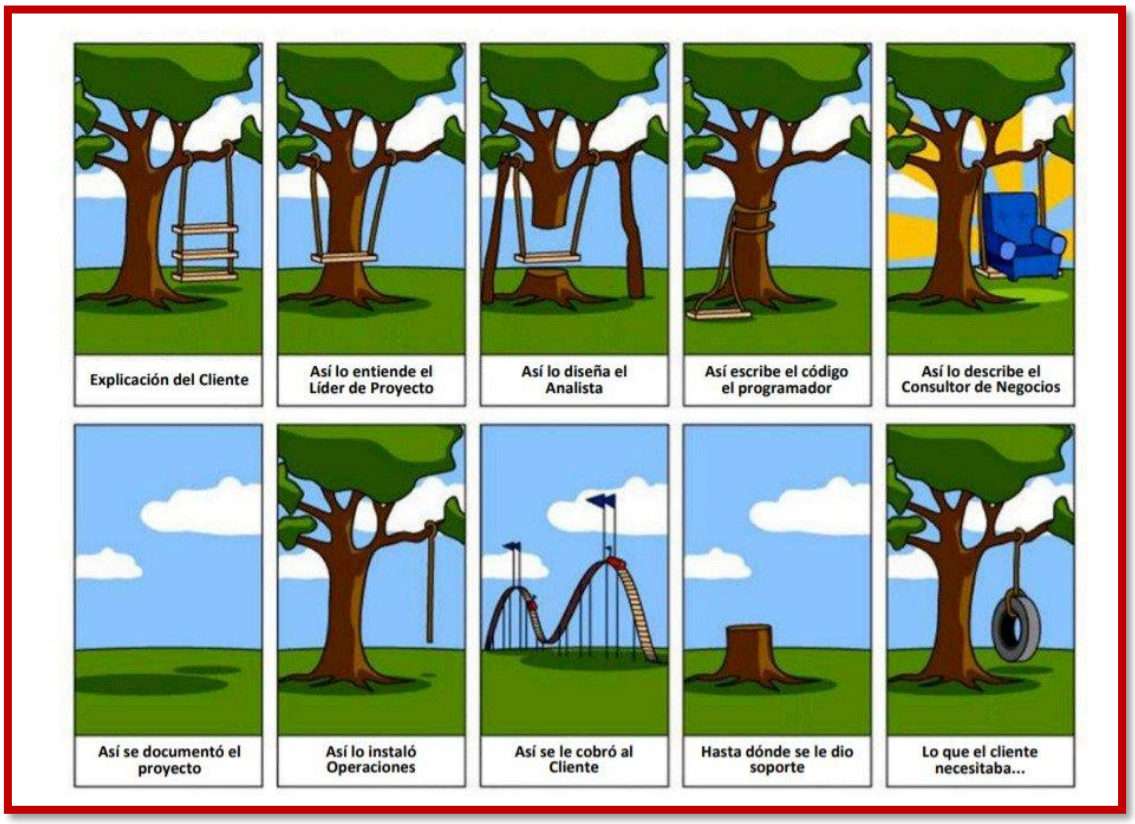
**Autor:** Joan Puigcerver Ibáñez

**Correo electrónico:** j.puigcerveribanez@edu.gva.es

**Curso:** 2022/2023

Este material es una obra derivada a partir del material de: **Sergio Badal**

# 1.Ingeniería del software




## 1.1. ¿Qué es la ingeniería del software?

La **ingeniería del software** es una disciplina que engloba herramientas, técnicas, recomendaciones y métodos que se utilizan en la creación de una aplicación/sistema/aplicativo/plataforma informática o, más genéricamente, al desarrollo de un proyecto de software que resuelve un problema o suple una necesidad.

El desarrollo de una aplicación se entiende como un intercambio comercial de un producto (el software) entre un *desarrollador* y un *cliente*, pudiendo ambas partes ser empresas, organizaciones o personas físicas y ser una o varias.

Esta disciplina tiene la codificación o programación como pilar fundamental, pero incluye otras tareas antes y después de que son tan importantes o más que la codificación. El ingeniero de software, por tanto, se encarga de toda la gestión del proyecto para que se pueda desarrollar a tiempo y forma, es decir, en un plazo determinado y con presupuesto y requisitos previstos.

La ingeniería del software incluye *el análisis previo de la situación* (qué y cuándo lo haremos), *el diseño del proyecto* (cómo lo haremos), *el desarrollo del software*, *las pruebas necesarias* para comprobar su correcto funcionamiento, *la documentación* (qué se ha hecho, como se utiliza), y *la implantación o desplazamiento del sistema* (puesta en marcha).

## 1.2. ¿Por qué es importante?

El término **Crisis del Software** se definió a principios de los años 70, cuando la planificación de un proyecto software era prácticamente inexistente. Seguían un proceso de desarrollo bastante artesanal, sin una metodología o un camino a seguir para su desarrollo.

El análisis previo a la codificación abarcaba el 8% del tiempo total de desarrollo de software. El 80% de los errores se producían en esta fase, incrementando el coste de corrección de errores a medida evolucionando el proyecto. Con estos indicadores estaba claro que algo estaba fallando y que el proceso de desarrollo de software necesitaba un cambio radical.

El Informe CHAOS2 de Standish sobre el éxito de los proyectos de software concluyó en 2015 que:

- **36% tuvieron éxito**: Se entregaron a tiempo, en presupuesto, con las características requeridas.
- **45% fueron modificados**: Con retraso, por encima del presupuesto, y/o con menos características.
- **19% fracasaron**: Se cancelaron antes de su finalización o se entregaron y nunca se utilizaron.

Cada año, desde 1994, el Grupo Standish crea una lista de 10 atributos y su relativo peso que ellos llaman los factores de éxito. Los tres primeros para 2018 son:

- **Latencia de decisión**: Las decisiones deben tomarse lo antes posible.
- **Alcance mínimo**: A mayor tamaño del proyecto, mayor índice de fracaso.
- **Responsables del proyecto**: Mejor una única persona que una Junta Directiva, por ejemplo.

**MODERN RESOLUTION FOR ALL PROJECTS**

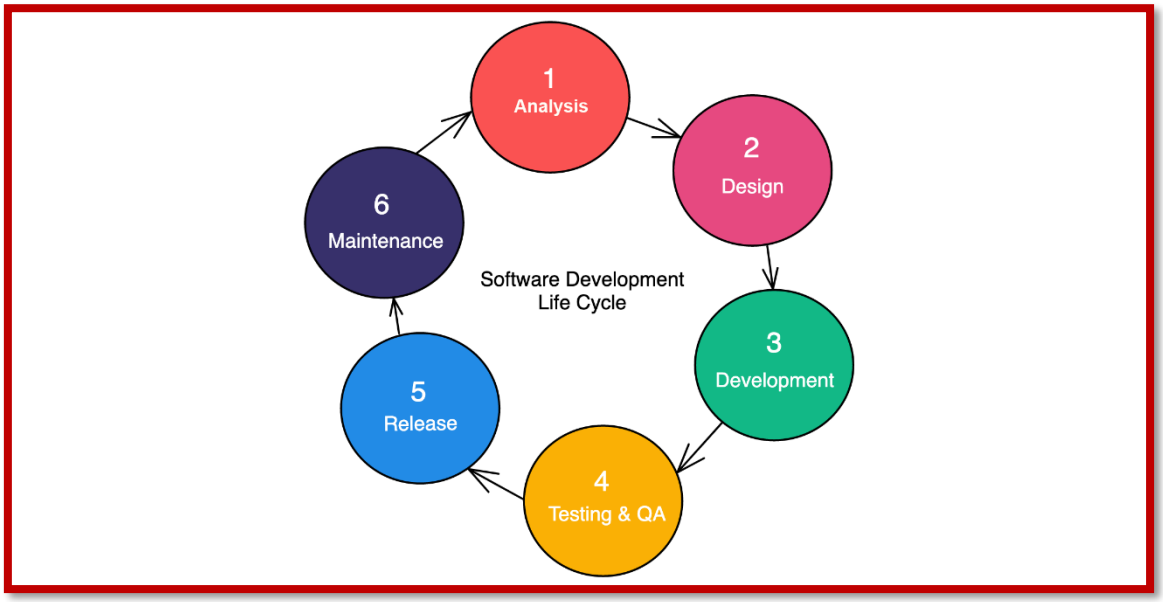
| | 2011 | 2012 | 2013 | 2014 | 2015 |
|---|---|---|---|---|---|
| SUCCESSFUL | 29% | 27% | 31% | 28% | 29% |
| CHALLENGED | 49% | 56% | 50% | 55% | 52% |
| FAILED | 22% | 17% | 19% | 17% | 19% |

*The Modern Resolution (OnTime, OnBudget, with a satisfactory result) of all software projects from FY2011–2015 within the new CHAOS database. Please note that for the rest of this report CHAOS Resolution will refer to the Modern Resolution definition not the Traditional Resolution definition.*

# 2.Ciclo de vida del software

## 2.1. Fases más comunes

Las fases del desarrollo de software, conocidas como **ciclo de vida del software**, representan las etapas por las que debe pasar una aplicación para ser entregada al destinatario con todas las garantías.




### *2.1.1. Análisis*

| **Análisis** | | |
|---|---|---|
| **Descripción** | **Perfil professional** | **¿Qué se genera?** |
| Descripción del problema. **¿Qué haré?** | • Jefe de proyecto<br>• Arquitecto software<br>• Consultor | • Especificación de los requisitos<br>• Prototipo<br>• Diagrama de casos de uso |

El análisis de una aplicación pretende determinar las necesidades que debe cubrir. Esta fase se realiza en contacto directo con el cliente. Se especifican los requisitos funcionales y no funcionales del proyecto.

La obtención de requisitos se plasma en estos documentos:

- Una **especificación de los requisitos del sistema**, con los requisitos funcionales y no funcionales.
- Uno o varios **prototipos**, que pueden ser desde un esbozo en papel hasta una versión muy básica del aplicativo. También existen herramientas de prototipado, que normalmente utilizan diseñadores.

En cuanto a los requisitos, se agrupan en:

- **Funcionales**: describen con detalle la función del sistema, es decir, qué debe hacer la aplicación.
- **No funcionales**: trata sobre características del sistema, unos plazos de entrega concretos, tiempos mínimos de ejecución o detalles más técnicos que exija el cliente por contrato (lenguaje de programación, plataforma, sistema operativo...)

### *2.1.2. Diseño*

| **Diseño** | | |
|---|---|---|
| **Descripción** | **Perfil professional** | **¿Qué se genera?** |
| Descripción de la solución. **¿Cómo lo haré?** | • Analista funcional<br>• Arquitecto software<br>• Analista programador | • Diagramas de estructura<br>• Diagramas de comportamiento |

En el **diseño** (o moldeado del software) se plantea una solución para la aplicación y se hace un modelo utilizando diferentes técnicas. Este diseño suele hacerse por niveles, empezando con una idea general que se va dividiendo en componentes para después moldear cada uno (diseño descendente/drop-down).

No hay que confundir la fase de DISEÑO en términos de ISW con el DISEÑO GRÁFICO (aspecto de nuestra aplicación). Son cosas completamente distintas.

### 2.1.3. Codificación

| **Codificación o implementación** | | |
|---|---|---|
| **Descripción** | **Perfil professional** | **¿Qué se genera?** |
| Implementación de la solución | • Desarrollador<br>• Administrador de BBDD | • Código fuente / Librerías<br>• Ejecutables<br>• Bases de datos (sistemas de información) |

En la **codificación** se realiza el programa atendiendo a todos sus componentes; esto incluye elementos como la base de datos, servidores o comunicaciones.

En muchos proyectos se comete el error de suponer que, por un lado, el programador es la "persona para todo" en el desarrollo del software y que, por otro, cualquier analista o consultor es también un buen programador. No todo profesional está capacitado para tratar directamente con el cliente.

### 2.1.4. prueba

| **prueba** | | |
|---|---|---|
| **Descripción** | **Perfil professional** | **¿Qué se genera?** |
| Prueba de la solución | - Probador | - Verificación de la solución |
| | - Programador | - Pruebas |

Las **pruebas** son revisiones que deben realizar personas distintas de las que codificaron el aplicativo para detectar errores de usabilidad o errores no detectados en la fase anterior.

Se pueden realizar pruebas de diversos tipos: individuales, de integración y sobre todo de aceptación con el cliente, siempre con la especificación de requisitos (análisis) delante.

### 2.1.5. Documentación

| Documentación | | |
|---|---|---|
| **Descripción** | **Perfil professional** | **¿Qué se genera?** |
| Documentar trabajo realizado y usabilidad | • - Administrativo | • - Manuales de uso |
| | • - Programador | • - Documentación código |

La fase de **documentación** consiste en dejar por escrito las decisiones técnicas que se han tomado. Desde por qué un framework y no otro hasta por qué se han creado 10 tablas en la base de datos y no 200.

Esta fase no debe realizarse necesariamente después de la de codificación. De hecho, se considera una buena práctica documentar a la vez que codifica.

### 2.1.6. Implantación o despliegue

| Implantación o despliegue | | |
|---|---|---|
| **Descripción** | **Perfil professional** | **¿Qué se genera?** |
| Publicar la solución | - Programador (sénior) | - Solución publicada / entregada |

**La implantación o despliegue** (no confundir con implementación) consiste en publicar la solución final en la plataforma destino o entregar al cliente el producto final en el formato acordado.

Muy pocas personas en la organización tienen (o deben tener) acceso a los servidores donde residen las aplicaciones que están publicadas (servidores de producción). Esta fase recae en los más veteranos y experimentados (senior).

### *2.1.7. Mantenimiento*

| **Mantenimiento** | | |
|---|---|---|
| **Descripción** | **Perfil professional** | **¿Qué se genera?** |
| Actualizar el aplicativo y corrección de errores | - Programador (júnior) | - Evolutivo, correctivo y adaptativo |
| | - Técnicos de apoyo | |

La fase de **mantenimiento** suele tener una sección específica en los contratos de desarrollo donde se detalla cómo se facturará este servicio y en qué términos.

El mantenimiento puede ser:

- **Evolutivo**: Modificaciones y/o eliminaciones en el producto por un cambio de las necesidades del cliente o por cambios en requisitos externos (normativos, legales, ...).
- **Correctivo**: Acciones para mejorar la calidad del producto en sistemas en cualquiera de sus aspectos: Corrección de errores, reestructuración y optimización del código, definición más clara y mejora del rendimiento.
- **Adaptativo**: Modificaciones necesarias para adaptarse a variaciones del entorno (contexto) a los que el sistema opera, por ejemplo, cambios de hardware, de base de datos, comunicaciones, ...

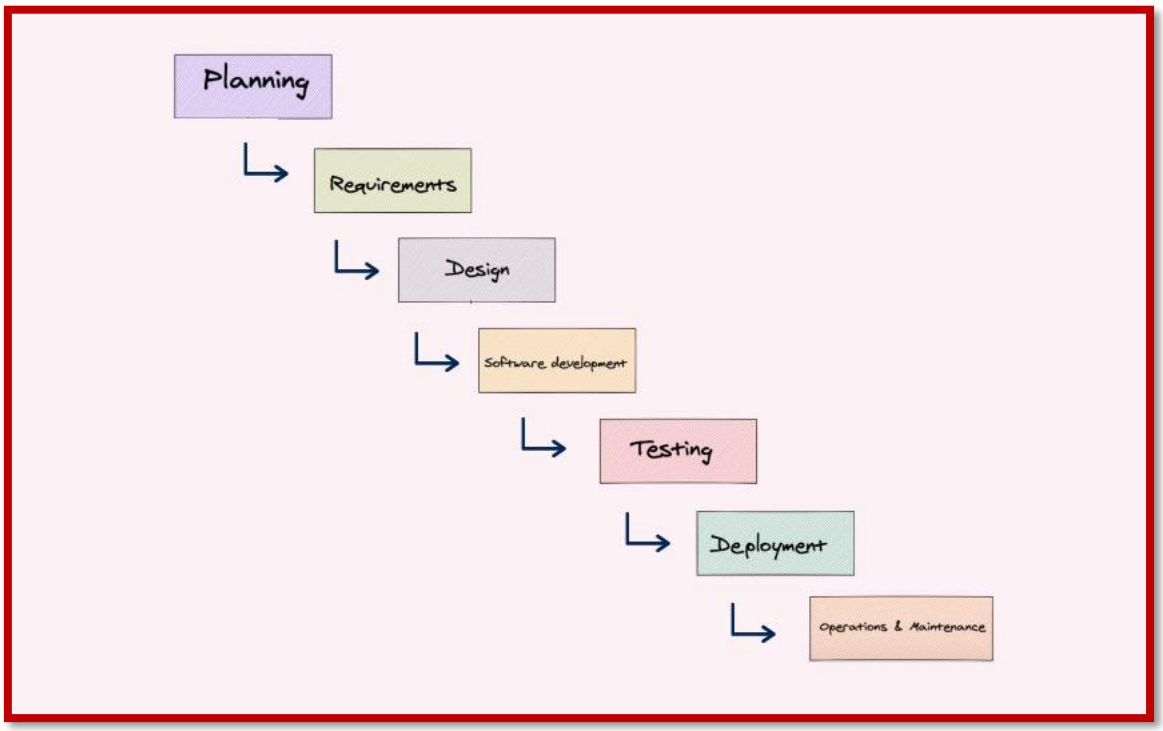
En este punto el software está entregado y facturado, por lo que se le resta importancia a pesar de que muchas empresas de desarrollo de software pagan gran parte de las nóminas de sus empleados con las cuotas de mantenimiento de todos los clientes. Esta fase suele recaer en los recién llegados (junior).

## 2.2. Tipo de ciclos de vida

En función de cómo se secuencian las fases que hemos visto antes, tendremos estos tipos:

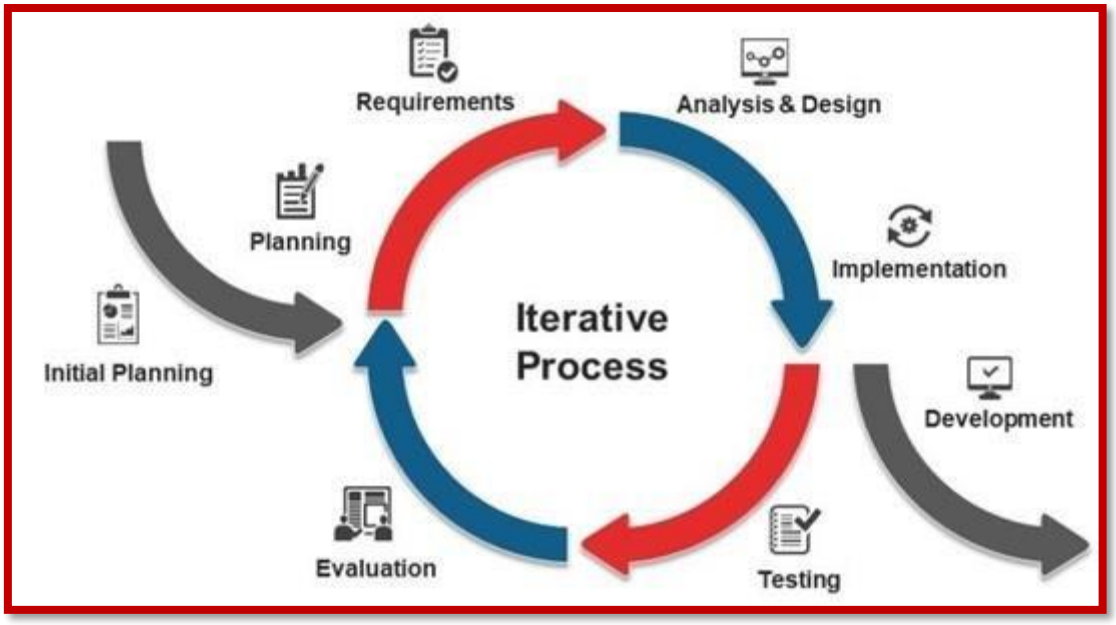
### *2.2.1. Clásico o cascada*

Todas las fases se distribuyen en una estructura lineal, donde no existe retroalimentación ni la posibilidad de volver atrás. Sólo válido en grandes proyectos con requisitos muy claros.



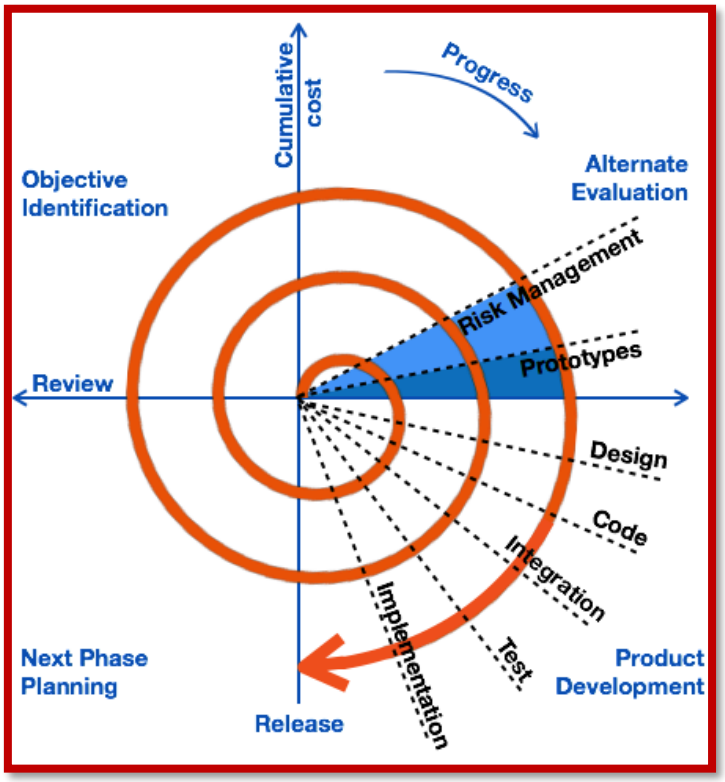
### 2.2.2. Ciclo de vida iterativo

Todas las fases se distribuyen en una estructura iterativa, donde se repite en bucle el sistema lineal. Una vez realizado el despliegue de la aplicación, se prueba y se evalúa el sistema para ver si es necesario realizar alguna modificación.




### 2.2.3. Ciclo de vida a espiral

A partir del modelo iterativo, añade la gestión de riesgo, para determinar si es viable detener o no el desarrollo de la aplicación.



# 3.Metodologías

Los ciclos de vida se combinan y explotan para crear metodologías que ofrecen técnicas concretas para cada etapa o fase de los ciclos de vida escogidos. Hay tres enfoques:

- **Metodologías tradicionales**: Basadas en combinaciones de los ciclos de vida anteriores. Fases demasiado largas, feedback insuficiente.
    - Ejemplos: METRICA3, RUP, MERISE
- **Metodologías ágiles**: Fomentan el reajuste continuo de los objetivos de desarrollo con las necesidades y expectativas del cliente, y proporcionan procesos de desarrollo de software "más ligeros", más rápidos y "ágiles" que pueden adaptarse a los inevitables cambios en los requisitos del cliente
    - Ejemplos: SCRUM, KANBAN, PROGRAMACIÓN XP

**CHAOS RESOLUTION BY AGILE VERSUS WATERFALL**

| SIZE | METHOD | SUCCESSFUL | CHALLENGED | FAILED |
|---|---|---|---|---|
| All Size Projects | Agile | 39% | 52% | 9% |
| | Waterfall | 11% | 60% | 29% |
| | | | | |
| Large Size Projects | Agile | 18% | 59% | 23% |
| | Waterfall | 3% | 55% | 42% |
| Medium Size Projects | Agile | 27% | 62% | 11% |
| | Waterfall | 7% | 68% | 25% |
| Small Size Projects | Agile | 58% | 38% | 4% |
| | Waterfall | 44% | 45% | 11% |

The resolution of all software projects from FY2011-2015 within the new CHAOS database, segmented by the agile process and waterfall method. The total number of software projects is over 10,000.