**Modelo en cascada**

En [Ingeniería de software](https://es.wikipedia.org/wiki/Ingenier%C3%ADa_de_software) el desarrollo en cascada, también llamado secuencial o ciclo de vida de un programa (denominado así por la posición de las fases en el desarrollo de esta, que parecen caer en cascada “por gravedad” hacia las siguientes fases), es el enfoque metodológico que ordena rigurosamente las etapas del [proceso para el desarrollo de software](https://es.wikipedia.org/wiki/Proceso_para_el_desarrollo_de_software), de tal forma que el inicio de cada etapa debe esperar a la finalización de la etapa anterior. Al final de cada etapa, el modelo está diseñado para llevar a cabo una revisión final, que se encarga de determinar si el proyecto está listo para avanzar a la siguiente fase. Este modelo fue el primero en originarse y es la base de todos los demás modelos de ciclo de vida.

La versión original fue propuesta por Winston W. Royce en 1970 y posteriormente revisada por [Barry Boehm](https://es.wikipedia.org/wiki/Barry_Boehm) en 1980 e Ian Sommerville en 1985.

Un ejemplo de una metodología de desarrollo en cascada es:

1. Análisis de requisitos.
2. Diseño del sistema.
3. Diseño del programa.
4. Codificación.
5. Pruebas.
6. Despliegue del programa.
7. Mantenimiento.

De esta forma, cualquier error de diseño detectado en la etapa de prueba conduce necesariamente al rediseño y nueva programación del código afectado, aumentando los costos del desarrollo. La palabra cascada sugiere, mediante la [metáfora](https://es.wikipedia.org/wiki/Met%C3%A1fora) de la fuerza de la gravedad, el esfuerzo necesario para introducir un cambio en las fases más avanzadas de un proyecto.

Si bien ha sido ampliamente criticado desde el ámbito académico y la industria, sigue siendo el [paradigma](https://es.wikipedia.org/wiki/Paradigma) más seguido al día de hoy.

(Fase del modelo)

* Análisis de requisitos del software

En esta fase se analizan las necesidades de los usuarios finales del software para determinar qué objetivos debe cubrir. De esta fase surge una memoria llamada SRD (documento de especificación de requisitos), que contiene la especificación completa de lo que debe hacer el sistema sin entrar en detalles internos.

Es importante señalar que en esta etapa se debe consensuar todo lo que se requiere del sistema y será aquello lo que seguirá en las siguientes etapas, no pudiéndose requerir nuevos resultados a mitad del proceso de elaboración del software.

* Diseño del sistema

Descompone y organiza el sistema en elementos que puedan elaborarse por separado, aprovechando las ventajas del desarrollo en equipo. Como resultado surge el SDD ([Descripción del diseño del software](https://es.wikipedia.org/wiki/Descripci%C3%B3n_del_dise%C3%B1o_del_software)), que contiene la descripción de la estructura relacional global del sistema y la especificación de lo que debe hacer cada una de sus partes, así como la manera en que se combinan unas con otras.

Es conveniente distinguir entre diseño de alto nivel o arquitectónico y diseño detallado. El primero de ellos tiene como objetivo definir la estructura de la solución (una vez que la fase de análisis ha descrito el problema) identificando grandes módulos (conjuntos de funciones que van a estar asociadas) y sus relaciones. Con ello se define la arquitectura de la solución elegida. El segundo define los algoritmos empleados y la organización del código para comenzar la implementación.

* Diseño del programa

Es la fase en donde se realizan los algoritmos necesarios para el cumplimiento de los requerimientos del usuario así como también los análisis necesarios para saber qué herramientas usar en la etapa de Codificación.

* Codificación

Es la fase en donde se implementa el [código fuente](https://es.wikipedia.org/wiki/C%C3%B3digo_fuente), haciendo uso de prototipos así como de pruebas y ensayos para corregir [errores](https://es.wikipedia.org/wiki/Error_de_software). Dependiendo del [lenguaje de programación](https://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n) y su versión se crean las bibliotecas y componentes reutilizables dentro del mismo proyecto para hacer que la programación sea un proceso mucho más rápido.

* Pruebas y error

Los elementos, ya programados, se ensamblan para componer el sistema y se comprueba que funciona correctamente. Se buscan sistemáticamente y se corrigen todos los [errores](https://es.wikipedia.org/wiki/Error_de_software) antes de ser entregado al usuario final.

Despliegue del producto de software

Es la fase en donde el usuario final o el cliente ejecuta el sistema, y se asegura que cubra sus necesidades, por lo que se valida o verifica el sistema.

* Mantenimiento

Una de las etapas más críticas, ya que se destina un 75% de los recursos, es el mantenimiento del software ya que al utilizarlo como usuario final puede ser que no cumpla con todas nuestras expectativas.

(Variantes)

Existen variantes de este modelo; especialmente destacamos la que hace uso de [prototipos](https://es.wikipedia.org/wiki/Desarrollo_con_prototipaci%C3%B3n) y en la que se establece un ciclo antes de llegar a la fase de mantenimiento, verificando que el sistema final esté libre de fallos. Otros ejemplos de variantes del modelo en cascada son el modelo en cascada con fases solapadas, cascada con subproyectos, y cascada con reducción de riesgos.

(Ventajas)

* Realiza un buen funcionamiento en equipos débiles y productos maduros, por lo que se requiere de menos capital y herramientas para hacerlo funcionar de manera óptima.
* Es un modelo fácil de implementar y entender.
* Está orientado a documentos.
* Es un modelo conocido y utilizado con frecuencia.
* Promueve una metodología de trabajo efectiva: Definir antes que diseñar, diseñar antes que codificar

(Desventajas)

* En la vida real, un proyecto rara vez sigue una secuencia lineal, esto crea una mala implementación del modelo, lo cual hace que lo lleve al fracaso.
* El proceso de creación del software tarda mucho tiempo ya que debe pasar por el proceso de prueba y hasta que el software no esté completo no se opera. Esto es la base para que funcione bien.
* Cualquier error de diseño detectado en la etapa de prueba conduce necesariamente al rediseño y nueva programación del código afectado, aumentando los costos del desarrollo.
* Una etapa determinada del proyecto no se puede llevar a cabo a menos de que se haya culminado la etapa anterior.

(Ejemplo en una imagen)

