4. Desarrollo de software. Ciclos y CASE.

Caso práctico
En BK programación ya están manos a la obra. Ada reúne toda su plantilla para desarrollar el nuevo proyecto.
Ella sabe mejor que nadie que no será sencillo y que habrá que pasar por una serie de etapa. Ana no quiere perderse la reunión, quiere descubrir por qué hay que tomar tantas anotaciones y tantas molestias antes incluso de empezar.

Entendemos por Desarrollo de Software todo el proceso que ocurre desde que se concibe una idea hasta que un programa está implementado en el ordenador y funcionando.

El proceso de desarrollo, que en un principio puede parecer una tarea simple, consta de una serie de pasos de obligado cumplimiento, pues sólo así podremos garantizar que los programas creados son eficientes, fiables, seguros y responden a las necesidades de los usuarios finales (aquellos que van a utilizar el programa).

Como veremos con más detenimiento a lo largo de la unidad, el desarrollo de software es un proceso que conlleva una serie de pasos. Genéricamente, estos pasos son los siguientes:

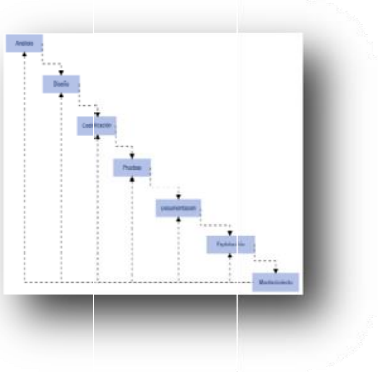
Etapas en el desarrollo de software:

Como vamos a ver en el siguiente punto, según el orden y la forma en que se lleven a cabo las etapas hablaremos de diferentes ciclos de vida del software.

La construcción de software es un proceso que puede llegar a ser muy complejo y que exige gran coordinación y disciplina del grupo de trabajo que lo desarrolle.

Reflexión
Según estimaciones, el 26% de los grandes proyectos de software fracasan, el 48% deben modificarse drásticamente y sólo el 26% tienen rotundo éxito. La principal causa del fracaso de un proyecto es la falta de una buena planificación e las etapas y mala gestión de los pasos a seguir. ¿Por qué el porcentaje de fracaso es tan grande? ¿Por qué piensas que estas causas son tan determinantes?

4.1 Ciclos de vida del software.



La serie de pasos a seguir para desarrollar un programa es lo que se conoce como Ciclo de Vida del Software.

Cada etapa vendrá explicada con más detalle en el punto de la presente unidad dedicado a las fases del desarrollo y ejecución del software.

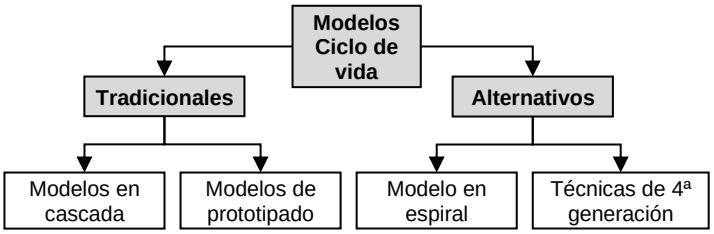
Diversos autores han planteado distintos modelos de ciclos de vida, pero los más conocidos y utilizados son los que aparecen a continuación.

Siempre se debe aplicar un modelo de ciclo de vida al desarrollo de cualquier proyecto software.

Se puede definir como el **conjunto de etapas por las que atraviesa el sistema desde su concepción hasta su retirada**, pasando por su desarrollo y explotación.
Ya que puede haber grandes diferencias entre un proyecto y otro, existen varios modelos de ciclo de vida. No obstante, todos ellos cubren estos objetivos básicos:

- Definir y determinar el orden de las **etapas** del desarrollo.
- Establecer los **criterios de transición** para pasar de una etapa a la siguiente.
- Proporcionar **puntos de control** para la gestión del proyecto.

Muchos autores clasifican los modelos de ciclo de vida en dos grandes grupos:



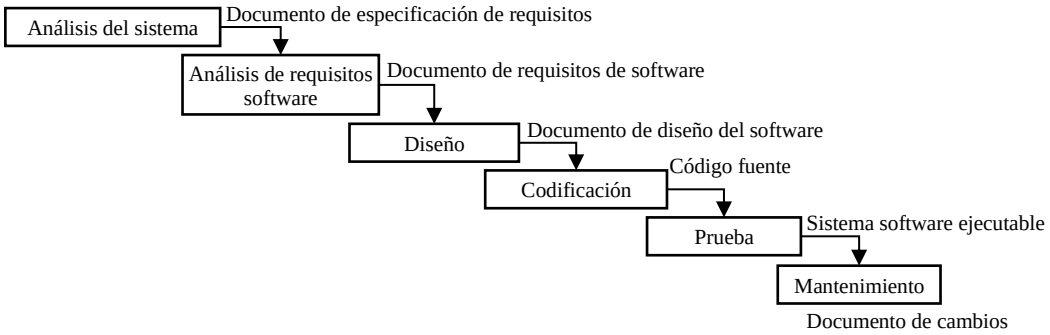
Para seleccionar el modelo más apropiado al proyecto en cuestión, se tendrán en cuenta una serie de factores como:

- Cultura de la organización.
- Deseo de asumir riesgos.
- Área de aplicación.
- Volatilidad de los requisitos.
- Comprensión de dichos requisitos.

Ya que cada modelo tiene sus ventajas y sus inconvenientes, en la práctica no se suelen seguir los modelos en su forma pura, sino que suelen combinarse con otros modelos para adecuarse mejor al proyecto en cuestión.

Modelo clásico o en cascada

Modelo creado por Royce en 1970, todavía sigue siendo el más utilizado.
Se compone de una serie de etapas secuenciales, de tal manera que es necesario haber finalizado una etapa para pasar a la siguiente, ya que los resultados de una etapa serán necesarios para iniciar la siguiente.
El número de etapas puede variar según el autor, siendo común este **modelo de seis etapas**:



En la actualidad se suele utilizar una variante de este modelo llamado **modelo en cascada realimentado**, que permite corregir las deficiencias detectadas durante una etapa volviendo a etapas anteriores del ciclo de vida.

Ventajas:

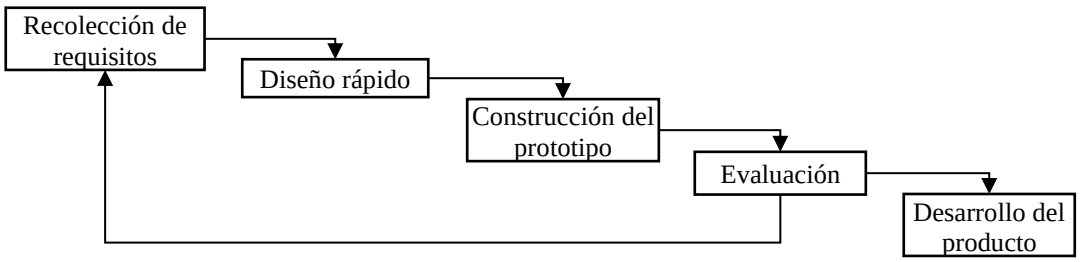
- Fácil de entender e implementar.
- Identifica entregables e hitos en cada etapa.
- Recomendado para productos maduros y equipos débiles.

Inconvenientes:

- Existen proyectos para los cuales el orden secuencial de este modelo es inviable.
- Obtener la totalidad de requerimientos al inicio del proyecto es poco realista.
- El software se entrega muy tarde, por lo que cualquier detección errores derivados de las etapas iniciales conllevará un gran coste.

Modelos de prototipado

Este modelo consta de **cinco etapas**:



Se realiza un rápido análisis del sistema y se diseña y construye un prototipo con los requisitos obtenidos. El usuario evalúa el prototipo y con él, afina más los requisitos. Este proceso se repite hasta que termine la definición de requisitos, momento en el que se pasará al desarrollo del producto final.

Existen distintos enfoques para la realización del prototipo, que van desde los prototipos en papel, maquetas que aporten al usuario un ejemplo visual del funcionamiento del sistema o prototipos evolutivos que satisfagan unos requisitos básicos iniciales y se vayan acercando al sistema final a medida que se refinan requisitos o se definen nuevos.

Ventajas:

- Proporciona software en fases tempranas.
- Facilita la identificación de requisitos.
- Desarrollo más rápido.
- Implantación más sencilla, ya que los usuarios conocen el producto.
- El sistema resultante necesita pocos cambios.

Inconvenientes:

- A pesar de todo puede crear falsas expectativas al usuario, ya que el prototipo y el sistema final no tienen por qué ser idénticos.

Modelo en espiral

Es un **modelo incremental** ya que con cada vuelta a la espiral se añaden nuevos requisitos, de manera que con cada iteración se crea una nueva versión del software más próxima al producto final.

En este modelo se definen **cuatro etapas** que se representan gráficamente mediante cuatro cuadrantes:

- **Planificación**: se determinan los objetivos, requisitos y restricciones del proyecto.
- **Evaluación de alternativas**: se evalúa el riesgo de cada alternativa y se elige la mejor.
- **Desarrollo**: se desarrolla el producto correspondiente a la etapa, siguiendo otros modelos (modelo clásico, prototipado…).
- **Revisión**: se valoran los resultados obtenidos y, si no son adecuados o se necesita implementar mejoras, se planifica la siguiente vuelta a la espiral.

Ventajas:

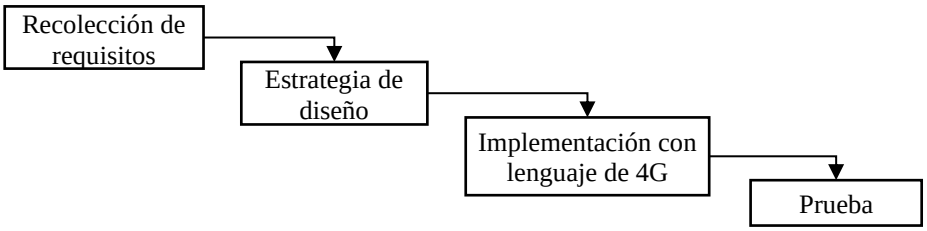
- Se disminuyen los riesgos.
- Proporciona software en fases tempranas.
- Se pueden acomodar otros modelos.
- Integra el desarrollo con el mantenimiento.
- Recomendado para proyectos largos, caros y complejos.

Inconvenientes:

- Coste elevado.
- Gran dependencia de la experiencia identificando y evaluando de riesgos.

Técnicas de cuarta generación

Engloba un amplio conjunto de herramientas de software que facilitan la especificación de características software a alto nivel para, más tarde, realizar automáticamente su conversión a código fuente mediante un **lenguaje de cuarta generación** (no procedimental).
Las etapas de este modelo son:



Estas herramientas han evolucionado hasta lo que conocemos como **herramientas CASE**, que ayudan a automatizar gran parte de las actividades de ingeniería.
Ventajas:

- Reducen el tiempo y coste de desarrollo.
- Elimina la codificación.
- Recomendado para proyectos pequeños y medianos.

Inconvenientes:

- Código ineficiente.
- Limitadas a **software de gestión** (generación de informes a partir de BBDD…).

4.2 Herramientas de apoyo al desarrollo del software.

En la práctica, para llevar a cabo varias de las etapas vistas en el punto anterior contamos con herramientas informáticas, cuya finalidad principal es automatizar las tareas y ganar fiabilidad y tiempo.

Esto nos va a permitir centrarnos en los requerimientos del sistema y el análisis del mismo, que son las causas principales de los fallos del software.

Las herramientas **CASE** son un conjunto de aplicaciones que se utilizan en el desarrollo de software con el objetivo de reducir costes y tiempo del proceso, mejorando por tanto la productividad del proceso.

¿En qué fases del proceso nos pueden ayudar?

En el diseño del proyecto, en la codificación de nuestro diseño a partir de su apariencia visual, detección de errores…

El desarrollo rápido de aplicaciones o **RAD** es un proceso de desarrollo de software que comprende el desarrollo iterativo, la construcción de prototipos y el uso de utilidades CASE. Hoy en día se suele utilizar para referirnos al desarrollo rápido de interfaces gráficas de usuario o entornos de desarrollo integrado completos.

La tecnología CASE trata de automatizar las fases del desarrollo de software para que mejore la calidad del proceso y del resultado final.

En concreto, estas herramientas permiten:

- Mejorar la planificación del proyecto.
- Darle agilidad al proceso.
- Poder reutilizar partes del software en proyectos futuros.
- Hacer que las aplicaciones respondan a estándares.
- Mejorar la tarea del mantenimiento de los programas.
- Mejorar el proceso de desarrollo, al permitir visualizar las fases de forma gráfica.

CLASIFICACIÓN

Normalmente, las herramientas CASE se clasifican en función de las fases del ciclo de vida del software en la que ofrecen ayuda:

- **U-CASE**: ofrece ayuda en las fases de planificación y análisis de requisitos.
- **M-CASE**: ofrece ayuda en análisis y diseño.
- **L-CASE**: ayuda en la programación del software, detección de errores del código, depuración de programas y pruebas y en la generación de la documentación del proyecto.

Ejemplos de herramientas CASE libres son: ArgoUML, Use Case Maker, ObjectBuilder…

Para saber más:
http://temariotic.wikidot.com/tema-58-boe-13-02-1996