# TEMA 086. EL CICLO DE VIDA DE LOS SISTEMAS DE INFORMACIÓN. MODELOS DEL CICLO DE VIDA

**Actualizado a 23/09/2020**

## 1. EL CICLO DE VIDA DE LOS SISTEMAS DE INFORMACIÓN

El **ciclo de vida** de un sistema es la sucesión de etapas por las que pasa el software desde que un nuevo proyecto es concebido hasta que se deja de usar. El ciclo de vida incluye al **ciclo de desarrollo**, que es el tiempo que transcurre desde la concepción del proyecto hasta su instalación una vez desarrollado.

CICLO DE VIDA = CICLO DE DESARROLLO + MANTENIMIENTO.

El ciclo de vida indica QUÉ actividades se realizarán y en qué orden, pero es la **metodología** la que indica CÓMO llevar a cabo esas actividades. Es común encontrar en un ciclo de vida 3 fases genéricas:

- Definición: obtención de requisitos para saber QUÉ hay que hacer.
- Desarrollo: análisis, diseño, codificación, pruebas e implantación.
- Mantenimiento.

## 2. MODELOS DE CICLO DE VIDA

### 2.1. CICLO DE VIDA DE CODIFICACIÓN DIRECTA (CODE AND FIX)

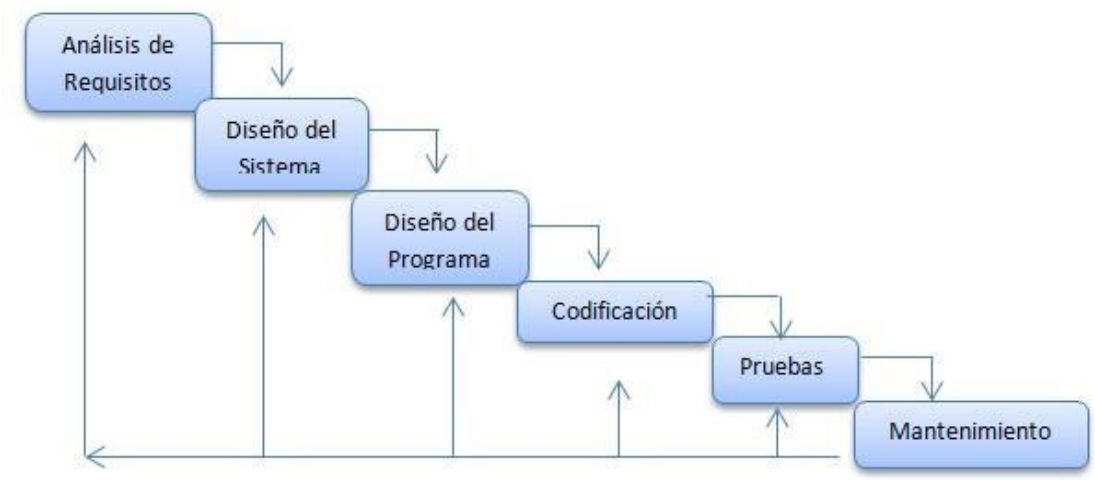
Consiste en **no aplicar ningún modelo de ciclo de vida**, y por lo tanto no es en absoluto recomendable.

Se comienza a codificar inmediatamente, se prueba para detectar errores y éstos se van corrigiendo sobre la marcha según aparecen. No existe documentación y supone grandes costes de mantenimiento a largo plazo.

### 2.2. CICLO DE VIDA EN CASCADA (WATERFALL)

Concebido en los años 70 por Winston Royce como solución a los innumerables problemas que causaba el code and fix: proyectos no finalizados, con importantes retrasos, con sobrecostes etc.

Consta de una serie de **fases sucesivas** (“fases en cascada”), **cada una de las cuales se completa antes de avanzar a la siguiente**. En el modelo original, se contemplaban realimentaciones entre fases, tal y como se observa en la siguiente figura (flechas que permiten la vuelta a una fase previa), pero muchas organizaciones lo han aplicado como si fuese estrictamente lineal.



**Principales ventajas**:

- Marca pautas de trabajo muy claras.
- Facilita la disposición de hitos en el desarrollo del proyecto.
- Facilita la estimación y el seguimiento del progreso.
- Proporciona entregables intermedios que forman el conjunto final del producto (documentos de análisis y diseño etc.).

**Principales inconvenientes:**

- Se comienza estableciendo TODOS los requisitos del sistema, algo que puede ser difícil incluso para el propio usuario. Poco flexible a cambios de requisitos.
- Rigidez: cada actividad es prerrequisito de las que la suceden.
- Los problemas se detectan tarde, incluso aunque se incluyan actividades de verificación y validación.
- "Nada está hecho hasta que todo está hecho", por lo que un cambio debido a un error puede suponer un gran coste.

## 2.3.CICLO DE VIDA EN V

Es similar al ciclo de vida en cascada, pero se caracteriza por tener dos tiempos claramente marcados: **actividades de desarrollo** (rama descendente) y **actividades de prueba** (rama ascendente).

Cada fase de la rama descendente:

- Lleva a cabo las actividades propias de dicha fase.
- Tiene en cuenta y verifica las salidas de la fase previa.
- Sirve de referencia a las actividades de la fase siguiente a través de los productos originados en ella.
- Prepara las pruebas que permitirán validar lo que ha sido definido durante la fase (estas pruebas se ejecutarán en la fase situada al mismo nivel de la rama ascendente de la V).

Cada fase de la rama ascendente:

- Ejecuta las pruebas para validar lo definido durante la fase de su mismo nivel de la rama descendente.
- Sirve de referencia a las actividades en la próxima fase.

**Principales ventajas:**

- Las del ciclo de vida en cascada + favorece la consideración de las pruebas lo antes posible.

**Principales inconvenientes:**

- Rigidez, aunque menos que en el modelo en cascada.
- "Nada está hecho hasta que todo está hecho".


## 2.4.MODELO DE CONSTRUCCIÓN DE PROTOTIPOS

Los prototipos son adecuados cuando no se sabe exactamente qué se quiere construir, por ejemplo, cuando el usuario no sabe exactamente lo que quiere o el desarrollador no está seguro de la eficiencia de una aproximación.

El prototipado tiene 3 fases:

- **Escuchar al usuario**: se corresponde con la recolección de requisitos.
- **Construir y revisar el prototipo**: se realiza un desarrollo rápido centrado en los aspectos del software visibles para el usuario (pantallas etc.).
- **El usuario prueba el prototipo**: esto permite refinar los requisitos y realimentar la siguiente fase de prototipado.

Se aplican varios ciclos de prototipado (cada uno con sus 3 fases), de forma que el refinamiento de requisitos obtenido tras las pruebas del usuario en el ciclo N constituye la entrada para el ciclo N+1. Una vez el prototipo ha servido para su propósito, lo ideal es desecharlo y empezar de nuevo.

**Principales ventajas**:

- Mecanismo ideal para obtener requisitos cuando no están claros.

**Principales inconvenientes**:

- Expectativas infundadas: el usuario puede creer que el trabajo ya está hecho y que en breve dispondrá de un sistema funcional.
- En ocasiones se mantienen en el sistema real decisiones de implementación tomadas con el único objetivo de agilizar el desarrollo del prototipo.

## 2.5. CICLO DE VIDA RAD (RAPID APPLICATION DEVELOPMENT) O DRA (DESARROLLO RÁPIDO DE APLICACIONES)

Desarrollado inicialmente por James Martin en 1980. El método comprende el desarrollo iterativo, la construcción de prototipos y el uso de herramientas CASE.

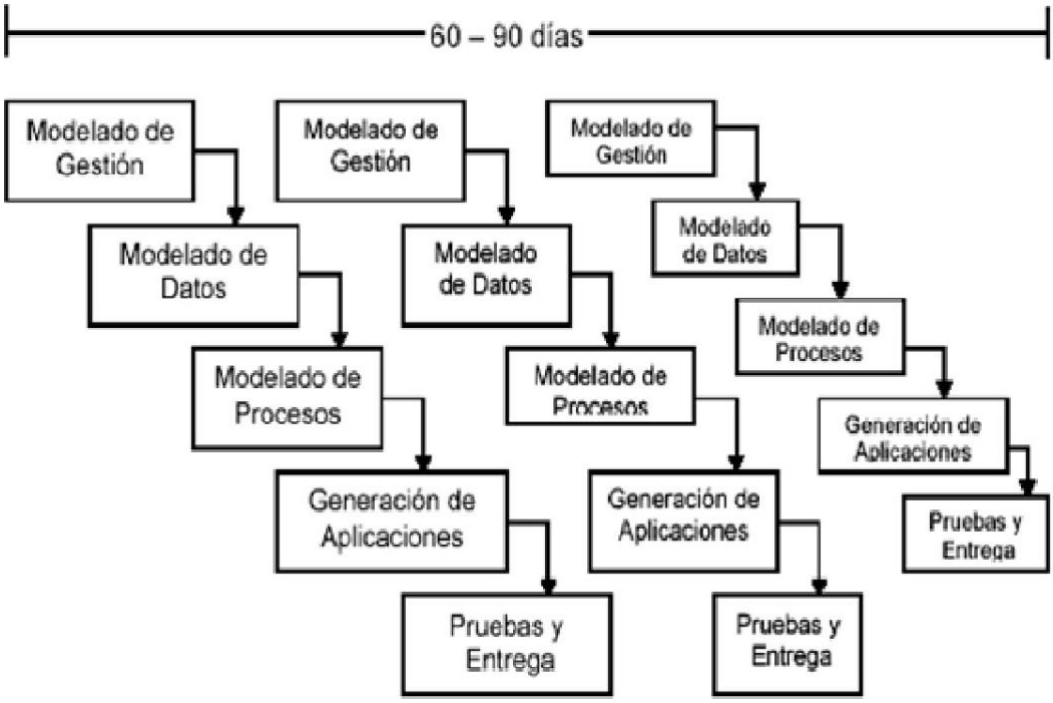
Es un modelo de proceso del desarrollo del software lineal secuencial que enfatiza **un ciclo de desarrollo extremadamente corto**. DRA es una adaptación a "alta velocidad" del modelo en cascada, en la que se logra el desarrollo rápido utilizando un enfoque de construcción basado en componentes. Si se comprenden bien los requisitos y se limita el ámbito del proyecto, el proceso DRA permite crear un "sistema completamente funcional" dentro de periodos cortos de tiempo (usualmente 60-90 días). Normalmente el enfoque DRA comprende las siguientes fases:

- **Modelado de gestión**: el flujo de información entre las funciones de gestión se modela de forma que responda a las siguientes preguntas: ¿Qué información conduce el proceso de gestión? ¿Qué información se genera? ¿Quién la genera? ¿A dónde va? ¿Quién la procesa?
- **Modelado de datos**: el flujo de información definido en la fase anterior se refina como un conjunto de objetos de datos necesarios para apoyar la empresa. Se definen las características (llamadas atributos) de cada uno de los objetos y las relaciones entre ellos.
- **Modelado de proceso**: los objetos de datos definidos en la fase anterior quedan transformados para lograr el flujo de información necesario para implementar una función de gestión. Las descripciones del proceso se crean para añadir, modificar, suprimir, o recuperar un objeto de datos.
- **Generación de aplicaciones**: El DRA asume la utilización de técnicas de cuarta generación. En lugar de crear software con lenguajes de tercera generación, DRA trabaja para volver a utilizar componentes de programas ya existentes (cuando es posible) o a crear componentes reutilizables (cuando sea necesario). En todos los casos se utilizan herramientas automáticas para facilitar la construcción del software.
- **Pruebas y entrega**: Como el proceso DRA enfatiza la reutilización, ya se han comprobado muchos de los
- **Componentes de los programas**. Esto reduce el tiempo de pruebas. Sin embargo, se deben probar todos los componentes nuevos y ejercitar todas las interfaces a fondo.

**Restricciones para su utilización**:

La limitación de tiempo impuesta en un proyecto DRA demanda "ámbito en escalas". Si una aplicación se puede modular de forma que cada función principal se pueda completar en

menos de 3 meses (utilizando el enfoque descrito anteriormente), es un candidato del DRA. Cada una de las funciones puede ser afrontada por un equipo DRA diferente y ser integradas en un solo conjunto.




**Principales ventajas**:

- Reutilización.
- Paralelismo en el desarrollo.
- Adecuado para tecnología orientada a objetos (basada en componentes).

**Principales inconvenientes**:

- Para proyectos grandes, el DRA requiere recursos humanos suficientes para crear los equipos DRA correctos.
- Requiere compromiso (de clientes y desarrolladores) en las rápidas actividades necesarias para completar el sistema.
- No es adecuado para todo tipo de proyectos. Por ejemplo, resulta problemático si el sistema no se puede descomponer modularmente o hay elevados riesgos técnicos.

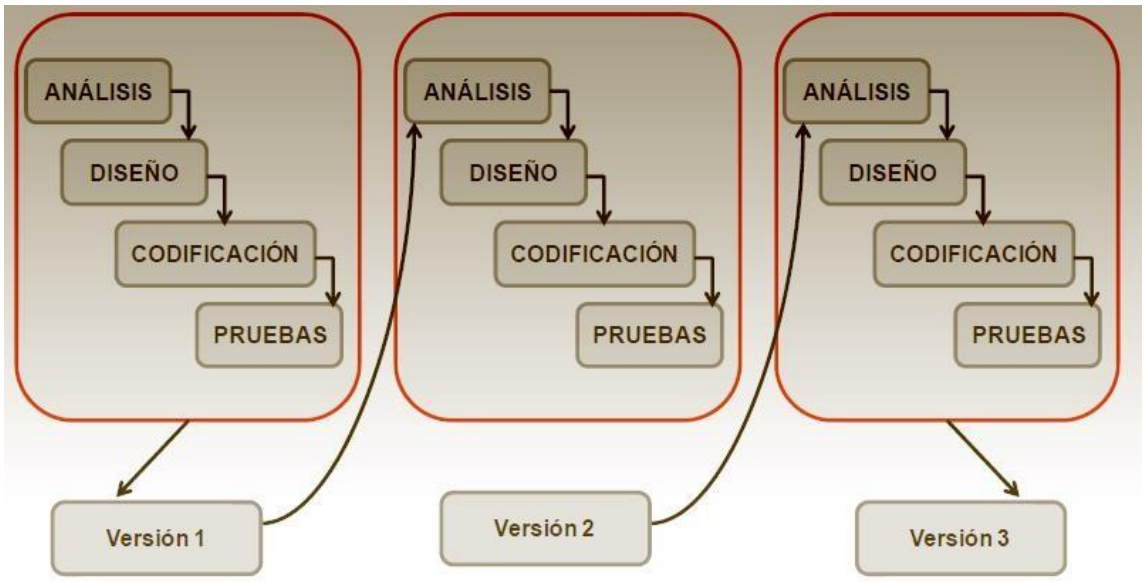
**Características**:

- Equipos Híbridos.
- Herramientas especializadas: control de versiones, componentes reutilizables, herramientas colaborativas y de trabajo en equipo, APIs etc.
- "Timeboxing": cada fase de desarrollo tiene siempre la misma duración temporal.
- Prototipos Iterativos y Evolucionarios:
    - Reunión JAD (Joint Application Development): reunión entre usuarios finales y los desarrolladores para obtener un borrador inicial de los requisitos.

- Iterar hasta acabar:
  - Los desarrolladores construyen y depuran el prototipo basado en los requisitos actuales.
  - Los diseñadores revisan el prototipo.
  - Los clientes prueban el prototipo y depuran los requisitos.
  - Los clientes y desarrolladores se reúnen para revisar juntos el producto, refinar los requisitos y generar solicitudes de cambios.

## 2.6.MODELO INCREMENTAL

Es el proceso de construir una **implementación parcial del sistema, para posteriormente ir aumentando la funcionalidad** con diferentes versiones. Para llevar a cabo los incrementos se pueden aplicar reiteradamente varias secuencias basadas en el modelo en cascada.




**Características de los incrementos**:

- Cada incremento puede ser el refinamiento de un incremento anterior o el desarrollo de una nueva funcionalidad.
- En el primer incremento se debe abordar el núcleo esencial del sistema (sus requisitos fundamentales).
- En incrementos posteriores se abordarán funciones suplementarias (algunas ya conocidas y otras no).
- El usuario evalúa el resultado de un incremento y se elabora un plan para el incremento siguiente. El proceso se repite hasta que se elabore el producto completo.

**Principales ventajas**:

- Al ir desarrollando parte de las funcionalidades, es más fácil determinar si los requerimientos planeados para los niveles subsiguientes son correctos.
- Los errores se detectan antes, lo que implica que se reducen los costes de mantenimiento.
- Mayor flexibilidad, puesto que el software debe ser construido de forma que facilite la incorporación de nuevos requisitos.

**Principales inconvenientes**:

- Dificultad para definir el núcleo que formará parte del primer incremento.
- Dificultad en la definición de los incrementos.
- Las soluciones de incrementos anteriores pueden no ser válidas para incrementos posteriores.

## 2.7.MODELO EVOLUTIVO

Es un ciclo de vida diseñado para acomodarse a la naturaleza evolutiva del software (el software, al igual que todos los sistemas complejos, evoluciona con el tiempo). Se caracteriza por permitir desarrollar versiones cada vez más completas del software. Es útil para introducir en el mercado una versión limitada del software que permita cumplir la presión competitiva (una versión que cumple el conjunto de requisitos principales, pero para la que se tendrán que incluir extensiones en el futuro).

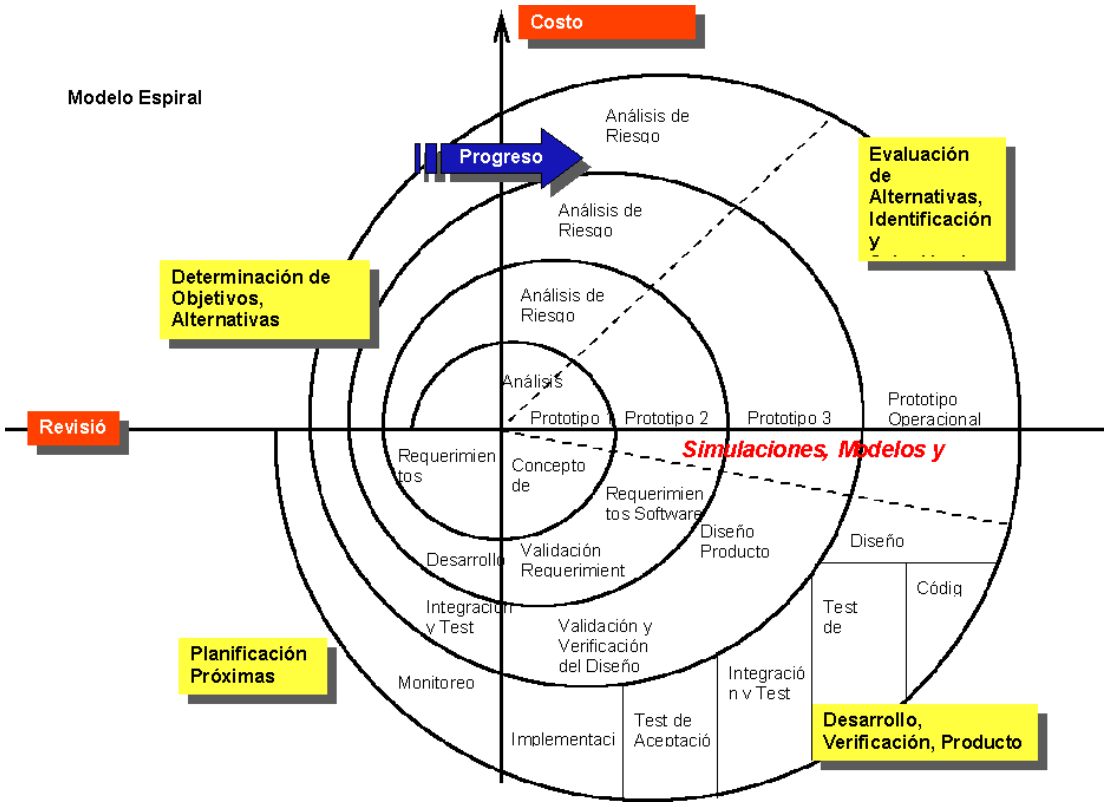
Los modelos de ciclo de vida vistos hasta el momento no tenían en cuenta la naturaleza evolutiva del software, lo que suponía mayores costes de mantenimiento. Es posible combinar los modelos evolutivos con los incrementales.

## 2.8.MODELO EN ESPIRAL

Consiste en aplicar un conjunto sucesivo de ciclos (vueltas en la espiral) hasta completar el software. Cada ciclo representa una fase del proceso de desarrollo, pero estas fases no son fijas, sino que se adaptan a las necesidades del proyecto. El ciclo más interno se suele corresponder con la viabilidad del sistema, el siguiente con el análisis de requisitos y así sucesivamente.

Se definen una serie de pasos para cada ciclo o vuelta de la espiral. La forma más habitual es tener un ciclo dividido en los siguientes 4 pasos:

- Determinación de objetivos, alternativas y restricciones.
- Evaluación de alternativas, considerando el análisis de riesgos.
- Desarrollo del siguiente nivel del producto.
- Planificación del siguiente ciclo.




La dimensión radial en la figura refleja los costes acumulativos en los que incurre el proyecto, y la dimensión angular el progreso del mismo. Durante cada fase se puede seleccionar el ciclo de desarrollo que se desee: cascada, V etc.

**Principales ventajas**:

- Permite adaptar el proceso de desarrollo a las necesidades cambiantes del proyecto y al conocimiento que se va adquiriendo.
- Permite el manejo de prototipos, enlazándolo con el análisis de riesgos.
- Gestiona los riesgos de forma explícita.

**Principales inconvenientes**:

- Requiere de una considerable habilidad para la consideración del riesgo.
- La evaluación de riesgos puede disparar el coste.
- Complejidad del modelo.
- Incertidumbre en el número de iteraciones necesarias.
- Algunos autores consideran que no es operativo en la práctica.