

1. Introducción

Copyright (c) 2025 Adrián Quiroga Linares Lectura y referencia permitidas; reutilización y plagio prohibidos

1.1 El Software

El software, según **Pressman**, se define en tres grandes conjuntos:

- El **conjunto de instrucciones** de ordenador que cuando se ejecutan proporcionan la función y el rendimiento deseado.
- Las **estructuras de datos** facilitan a los programas manipular la información
- Los **documentos** que describen el funcionamiento y la forma de uso de los programas

Sommerville, en cambio lo define como *los programas de cómputo y su documentación asociada*

La **Ingeniería de Software** es una disciplina de la ingeniería que tiene que ver con todos los aspectos de la producción de software, desde su inicio hasta que se sustituye. Al ser una ingeniería, implica la aplicación de métodos, uso de herramientas, técnicas... de forma sistemática y organizada. Estos se aplican en todos los aspectos de la producción: los técnicos, la gestión de recursos humanos, la gestión del tiempo, el coste la calidad...

Atributos del buen software:

- **Confiable**
- **Eficiente**
- **Usable**
- **Mantenible**

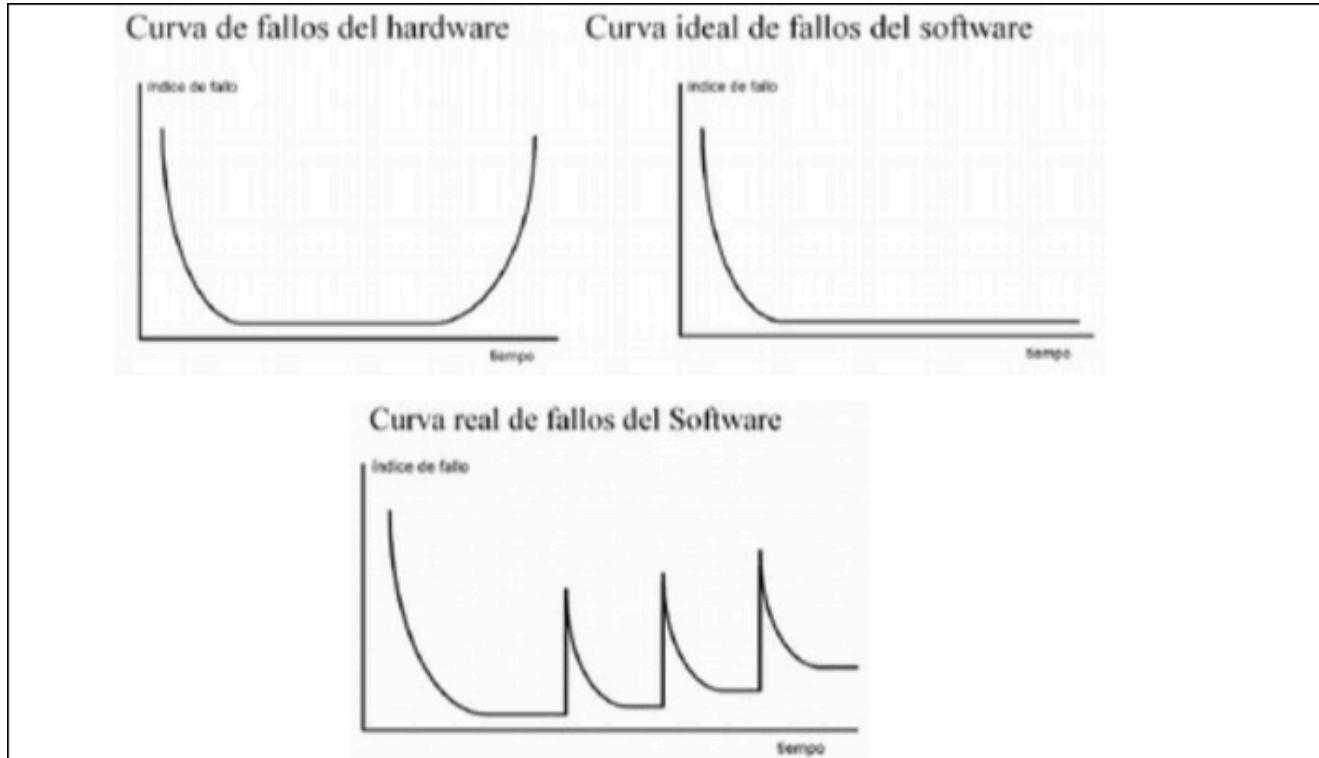
El **desarrollo de software** es la secuencia de actividades que conducen a la elaboración de un producto software. Fundamentalmente se puede subdividir en **especificación** del software, **desarrollo, validación y mantenimiento**.

1.2 Características del software

El software se degrada, pero se desarrolla y se reutiliza

- **No se fabrica, se desarrolla:** esto implica que el coste de desarrollo de la primera unidad es superior a los costes de producción de otras campos, pero el coste unitario decrece hasta hacerse casi imperceptible en comparación con cualquier otra ingeniería.

- **No se estropea, se degrada:** a diferencia de otros productos que empiezan con una alta tasa de fallos y la reducen hasta que empieza a crecer otra vez, el software ideal solo tiene la primera pendiente: empieza con una alta tasa de fallos y la reduce hasta estabilizarse para siempre. Pero se degrada, por lo que con el tiempo y con las actualizaciones y mejoras necesarios, va aumentando y reduciendo su tasa de fallos.



Además, **se puede desarrollar a medida** (personalizado) y se distinguen:

- **Productos genéricos:** software enlatado, producido por una organización para introducir en el mercado dirigido a clientes. Mucho gasto
- **Productos a medida:** son los realizados bajo pedido. Mucho esfuerzo
- **ERPs (Enterprise Resource Plannings):** soluciones genéricas tuneadas para adaptarse a problemas particulares.

Es reutilizable porque dispone de activos, que son secciones de un proyecto que se podrían reutilizar en otros proyectos. Estos bloques reutilizables son:

- **Bibliotecas**, conjunto de subrutinas temáticas de las que disponen la mayoría de lenguajes
- **Módulos**, que se introdujeron con la utilización de técnicas de programación estructurada y modular, pero al dedicarse poco esfuerzo al diseño de módulos suficientemente generales para ser reutilizables, no se suelen reutilizar
- **Frameworks**, se trata de conjuntos de métodos y objetos que también pueden cumplir funciones generales
- **Patrones de diseño**
- Los que no se reutiliza son las **listas de requisitos** que necesita el cliente

Reutilizar reduce los costes, aumenta la productividad, mejora la calidad y facilita el control y la planificación. Sin embargo suele dar lugar a programas menos óptimos, inversiones iniciales cuantiosas ...

1.3 El software heredado

Software generado hace bastante tiempo y que se ha seguido usando sin estar documentado totalmente y sin que el programador que lo desarrolló esté accesible:

- Es **crítico** para las empresas que lo usan debido a su antigüedad
- Ha sido **modificado** de forma **continuada** para ser adaptado a nuevos entornos o para implementar nuevas funcionalidades
- Suelen tener **poca calidad**
 - **Diseño complejo**
 - **Secciones de código oscuro**
 - **Documentación muy pobre

Esto implica:

- **EL mantenimiento es costoso**
- Su evolución representa un **alto riesgo para la empresa**

1.4 Tipos de Software

Tipos de Problemas

- Con solución por **pasos específicos**, mediante el uso de algoritmos o lenguajes procedimentales
- Que **pueden definirse formalmente**, en estos no necesitamos como resolver el problema, si no definir el problema en sí, para que se usan lenguajes declarativos como SQL
- Problemas con **soluciones concretas**, pero cuya solución general desconocemos
- Problemas basado en **conocimiento heurístico**, sistemas expertos basado en el análisis de reglas que pueden llegar a ser contradictorias

Clasificado por categoría (Pressman)

- **Empotrado:** sistemas en la memoria ROM de los sistemas para lavadoras, hornos, coches ...
- **De sistemas:** son los que hacen de interfaz entre programas y programas o entre programas y hardware. SOs, compiladores ...
- **De aplicación:** son los que resuelven una necesidad específica.
- **Científico y de ingeniería:** algoritmos devoradores de números, empleados para cálculo numérico.

- **De línea de productos:** son los que se diseñan para una capacidad específica y la utilización de muchos clientes diferentes
- **De aplicaciones basadas en la web:** empezaron como conjuntos de archivos de hipertexto ligados, pero cada vez evolucionan más.
- **De inteligencia artificial:** utiliza algoritmos no numéricos para resolver problemas inabordables mediante análisis directos.

Clasificaciones de Sommerville

- Por **estructura** (*orientados a*): función, componente, listas y objetos
- Por **función**: programas, interfaces Hombre.Máquina, herramientas de software, librerías, sistemas de uso genérico, bds y sistemas basados en web
- Por **plataforma de cómputo**: sistemas embebidos, sistemas de cómputo distribuido, sistemas de cómputo paralelo, sistemas de tiempo real, sistemas basados en chips, sistemas wereables y sistemas de cómputo ubicuos.

1.5 Problemas del desarrollo del software

- **Planificación y estimación de costes imprecisos:** es complicado estimar costes si no hay realizaciones previas ni mediciones, por eso la **primera norma del CMMI es medir.**
- **Baja productividad:** causada por mayor duración de la esperada. Suele darse por problemas en la especificación. Las soluciones a esto son la **administración de requisitos**, las metodologías ágiles y la gestión de la configuración mediante la gestión del cambio.
- **Mala calidad a la entrega del producto:** para evitar se ideó el proceso del **aseguramiento de la calidad**, así como los procesos de verificación y validación.
- **Rediseño del producto:** en casos en los que el cliente no esté satisfecho, aunque se cumplan los requisitos, puede ser obligado el rediseño. Por lo que hay que hacer que el **cliente entienda los requisitos**.

1.6 Leyes de la evolución del software (Leyes de Lehman)

- **Cambio continuado**
- **Complejidad creciente**
- **Evolución prolongada**
- **Estabilidad organizacional**
- **Crecimiento continuado**
- **Decremento de la calidad**
- **Realimentación del sistema**