# Tema 1- Elementos de desarrollo del software

## —Tipos de software:

* De sistema (Sistema operativo, drivers -controladores)

En informática, el software de sistema, denominado también software de base, es el conjunto de instrucciones que permiten el manejo de la computadora. Consiste en un software que sirve de soporte o base para controlar e interactuar con el hardware y otros programas. (Ejemplos Windows, Linux)

* De aplicación (Suite ofimática, Navegador, Edición de imagen, ...)

En informática, el software de aplicación es un tipo de software de computadora diseñado para realizar un grupo de funciones, tareas o actividades coordinadas para el beneficio del usuario. Ejemplos de una aplicación serían un procesador de textos, una hoja de cálculo, una aplicación de contabilidad, un navegador web, un reproductor multimedia, un simulador de vuelo aeronáutico, una consola de juegos o un editor de fotografías. «Software de aplicación» hace referencia colectivamente a todas las aplicaciones, como analogía con el software del sistema, que está principalmente relacionado con aquel ejecutado por el sistema operativo.

* De desarrollo (Editores, compiladores, intérpretes, ...)

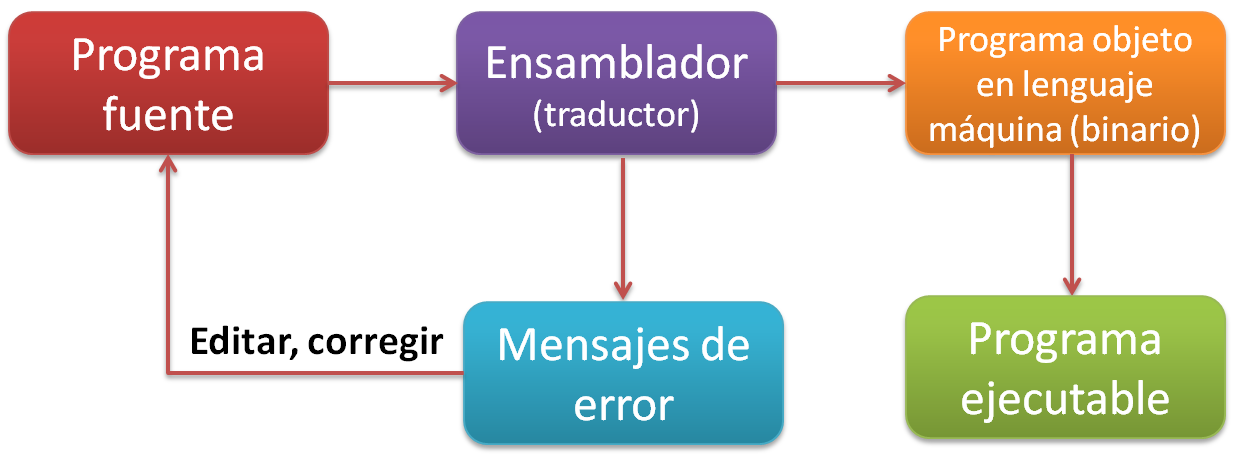
Es un tipo de software muy característico que destaca por ser el utilizado para desarrollar otros software, es decir, son herramientas y aplicaciones a los que se le da uso para desarrollar otras herramientas y programas. Suelen verse equipados con todo lo que un usuario informático puede necesitar para realizar ciertas actividades. Entre los beneficios que se integran se logra encontrar por ejemplo procesadores de texto, un depurativo, diferentes tipos de colectores, entre muchos otros.

## —Relación Hardware-software

* Hardware y software son mutuamente dependientes el uno del otro. Los dos deben trabajar juntos para hacer una computadora producen una salida útil.
* Software no puede utilizarse sin soporte hardware.
* Hardware sin conjunto de programas para operar sobre no puede ser utilizada y es inútil.
* Para realizar un trabajo particular en el equipo, se debe cargar software relevante en el hardware
* Aplicaciones diferentes pueden ser cargadas en un hardware para ejecutar trabajos diferentes.
* Un software actúa como una interfaz entre el usuario y el hardware.
* Si el hardware es el 'corazón' de un sistema informático, a continuación, el software es su 'alma'. Ambos son gratuitos para el otro.

1. Disco duro : almacena de forma permanente los archivos ejecutables y los archivos de datos.
2. Memoria RAM : almacena de forma temporal el código binario de los archivos ejecutables y los archivos de datos necesarios.
3. CPU : lee y ejecuta instrucciones almacenadas en memoria RAM, así como los datos necesarios.
4. E/S: recoge nuevos datos desde la entrada, se muestran los resultados, se leen/guardan a disco, ...
5. El disco duro se considera un periférico de E/S (Entrada/Salida).
6. La CPU se llama también UCP (en inglés), procesador o microprocesador.

## —Codigo fuente, código objeto y ejecutable



* Código Fuente
  + Es el escrito por los programadores en algún editor de texto.
  + Utiliza sentencias y órdenes derivadas del inglés
  + Más cercanos al razonamiento humano.
  + Lenguaje de programación de alto nivel y contiene el conjunto de instrucciones.
  + java, C, C++, HTML
* Código Objeto
  + Es el código binario resultado de compilar el código fuente.
  + La compilación es la traducción de una sola vez del programa y se realiza utilizando un compilador.
  + La interpretación es la traducción y ejecución simultánea del programa línea a línea
  + El código objeto no es directamente inteligible por el ser humano, tampoco por la computadora
  + Es un código intermedio entre el código fuente y el ejecutable y sólo existe si el programa se compila, ya que si se interpreta (traducción línea a línea del código) se traduce y se ejecuta en un solo paso.
    - Linker o enlazador:
      * Es un programa que genera código máquina (llamado también código objeto) traduciendo instrucciones de un programa fuente escrito en ensamblador a códigos ejecutables
      * Interpreta los nombres simbólicos para direcciones de memoria y otros elementos
* Codigo ejecutable
  + Es el objeto binario resultante de enlazar los archivos de código objeto con ciertas rutinas y bibliotecas necesarias
  + El sistema operativo será el encargado de cargar el código ejecutable en memoria RAM y proceder a ejecutarlo
  + También es conocido como código máquina y ya sí es directamente inteligible por la computadora

## 

## —Ciclo de vida del software

Ingeniería de software: Disciplina que estudia los principios y metodologías para el desarrollo y mantenimiento de sistemas software.

* Algunos autores consideran que "desarrollo de software" es un término más apropiado que "ingeniería de software" puesto que este último implica niveles de rigor y prueba de procesos que no son apropiados para todo tipo de desarrollo de software.

La Ingeniería de Software estudia la creación de software confiable y de calidad, basándose en métodos y técnicas de ingeniería. Brindando soporte operacional y de mantenimiento, el campo de estudio de la ingeniería de software.Integra ciencias de la computación, ciencias aplicadas y las ciencias básicas en las cuales se encuentra apoyada la ingeniería.

La creación del software es un proceso intrínsecamente creativo y la ingeniería del software trata de sistematizar este proceso con el fin de acotar el riesgo de fracaso en la consecución del objetivo, por medio de diversas técnicas que se han demostrado adecuadas sobre la base de la experiencia previa.

La ingeniería de software se puede considerar como la ingeniería aplicada al software, esto es, por medios sistematizados y con herramientas preestablecidas, la aplicación de ellos de la manera más eficiente para la obtención de resultados óptimos; objetivos que siempre busca la ingeniería. No es solo de la resolución de problemas, sino más bien teniendo en cuenta las diferentes soluciones, elegir la más apropiada.

La ingeniería de software requiere llevar a cabo numerosas tareas agrupadas en etapas, al conjunto de estas etapas se le denomina ciclo de vida. Las etapas comunes a casi todos los modelos de ciclo de vida son las siguientes:

### Análisis

* 1. Se determina y define claramente las necesidades del cliente y se especifica los requisitos que debe cumplir el software a desarrollar. La especificación de requisitos debe:
     1. Ser completa y sin omisiones
     2. Ser concisa y sin trivialidades
     3. Evitar ambigüedades. Utilizar lenguaje formal.
     4. Evitar detalles de diseño o implementación
     5. Ser entendible por el cliente
     6. Separar requisitos funcionales y no funcionales
     7. Dividir y jerarquizar el modelo
     8. Fijar criterios de validación

### 

### Diseño

* 1. Se descompone y organiza el sistema en elementos componentes que pueden ser desarrollados por separado.
  2. Se especifica la interrelación y funcionalidad de los elementos componentes.
  3. Las actividades habituales son las siguientes:
     1. Diseño arquitectónico
     2. Diseño detallado
     3. Diseño de datos
     4. Diseño de interfaz de usuario

### Codificación

* 1. Se escribe el código fuente de cada componente.
  2. Pueden utilizarse distintos lenguajes informáticos:
  3. Lenguajes de programación: C, C++, Java, Javascript, ...
  4. Lenguajes de otro tipo: HTML, XML, JSON, ...

### Pruebas

* 1. El principal objetivo de las pruebas debe ser conseguir que el programa funcione incorrectamente y que se descubran defectos.
  2. Deberemos someter al programa al máximo número de situaciones diferentes.

### Mantenimiento

* 1. Durante la explotación del sistema software es necesario realizar cambios ocasionales.
  2. Para ello hay que rehacer parte del trabajo realizado en las fases previas.
  3. Tipos de mantenimiento:
     1. **Correctivo:** se corrigen defectos.
     2. **Perfectivo:** se mejora la funcionalidad.
     3. **Evolutivo:** se añade funcionalidades nuevas.
     4. **Adaptativo:** se adapta a nuevos entornos.

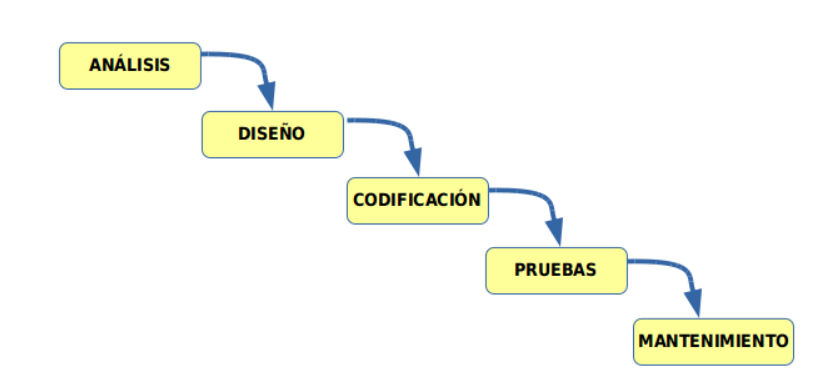
## Resultado tras cada fase (I)

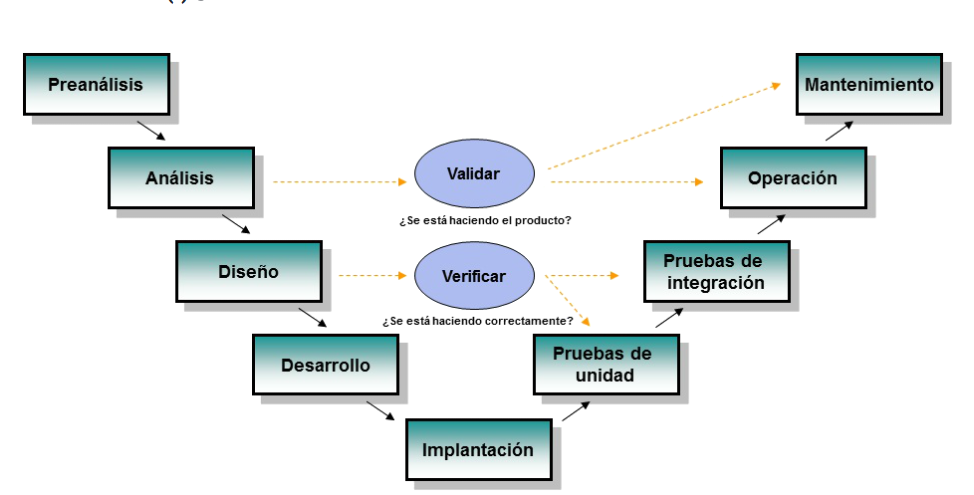
* Ingeniería de sistemas: Especificación del sistema
* ANÁLISIS: Especificación de requisitos del software
* DISEÑO arquitectónico: Documento de arquitectura del software
* DISEÑO detallado: Especificación de módulos y funciones
* CODIFICACIÓN: Código fuente

## Resultado tras cada fase (II)

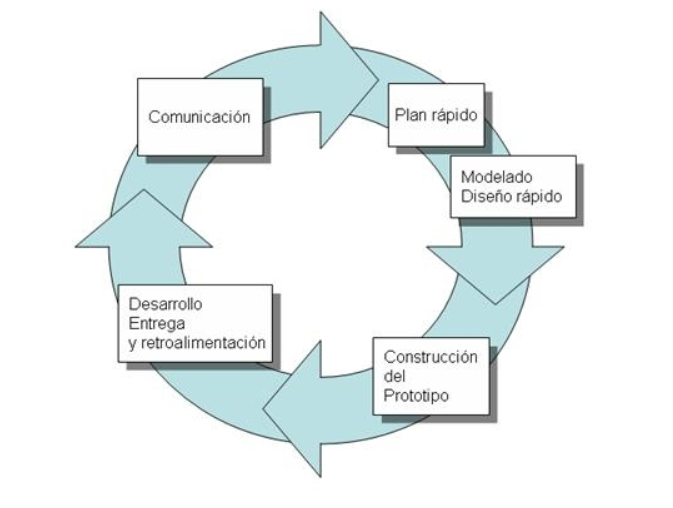
* PRUEBAS de unidades: Módulos utilizables
* PRUEBAS de integración: Sistema utilizable
* PRUEBAS del sistema: Sistema aceptado
* Documentación: Documentación técnica y de usuario
* MANTENIMIENTO: Informes de errores y control de cambios

## Modelos de desarrollo de software

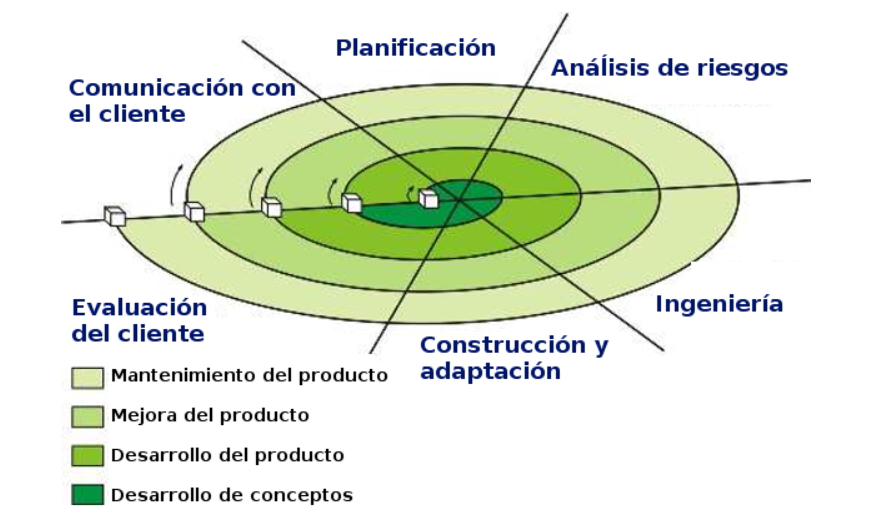
* Modelos clásicos (predictivos)
  + Modelo en cascada
    - Modelo de mayor antigüedad.
    - Identifica las fases principales del desarrollo software
    - Las fases han de realizarse en el orden indicado.
    - El resultado de una fase es la entrada de la siguiente fase.
    - Es un modelo bastante rígido que se adapta mal al cambio continuo de especificaciones.
    - Existen diferentes variantes con mayor o menor cantidad de actividades.
  + Modelo en V



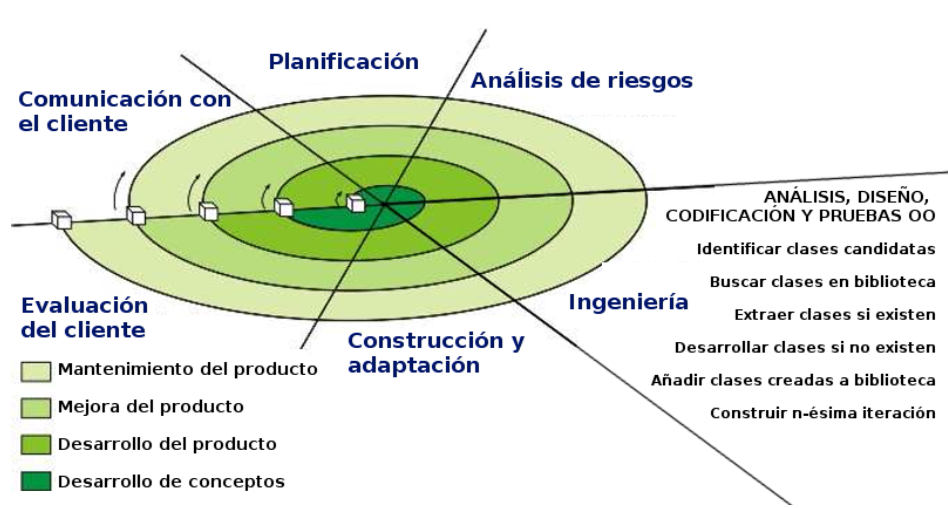
* + - Modelo muy parecido al modelo en cascada.
    - Visión jerarquizada con distintos niveles.
    - Los niveles superiores indican mayor abstracción.
    - Los niveles inferiores indican mayor nivel de detalle.
    - El resultado de una fase es la entrada de la siguiente fase.
    - Existen diferentes variantes con mayor o menor cantidad de actividades.
  + Modelo de construcción de prototipos



* + - A menudo los requisitos no están especificados claramente:
      * por no existir experiencia previa.
      * por omisión o falta de concreción del usuario/cliente.
    - Proceso:
      * Se crea un prototipo durante la fase de análisis y es probado por el usuario/cliente para refinar los requisitos del software a desarrollar.
      * Se repite el paso anterior las veces necesarias.
    - Tipos de prototipos:
      * Prototipos rápidos
        + El prototipo puede estar desarrollado usando otro lenguaje y/o herramientas
        + Finalmente el prototipo se desecha.
      * Prototipos evolutivos
        + El prototipo está diseñado en el mismo lenguaje y herramientas del proyecto.
        + El prototipo se usa como base para desarrollar el proyecto.
  + Modelos evolutivos o incrementales
    - Modelo en espiral (iterativos)



* + - * Desarrollado por Boehm en 1988.
      * La actividad de ingeniería corresponde a las fases de los modelos clásicos: análisis, diseño, codificación
      * Aplicado a la programación orientada a objetos
        + En la actividad de ingeniería se da gran importancia a la reutilización de código



* + - Metodologías ágiles (adaptativos)
      * Son métodos de ingeniería del software basados en el desarrollo iterativo e incremental.
      * Los requisitos y soluciones evolucionan con el tiempo según la necesidad del proyecto.
      * El trabajo es realizado mediante la colaboración de equipos auto-organizados y multidisciplinarios, inmersos en un proceso compartido de toma de decisiones a corto plazo.
      * Las metodologías más conocidas son:
        + Kanban

También se denomina "sistema de tarjetas"

Desarrollado inicialmente por Toyota para la industria de fabricación de productos

Controla por demanda la fabricación de los productos necesarios en la cantidad y tiempo necesarios.

Enfocado a entregar el máximo valor para los clientes, utilizando los recursos justos.

Lean manufacturing:

La producción Lean es un modelo de gestión que se enfoca en minimizar las pérdidas de los sistemas de manufactura al mismo tiempo que maximiza la creación de valor para el cliente final. Para ello utiliza la mínima cantidad de recursos

La creación de flujo se focaliza en la reducción de los ocho tipos de "desperdicios" en productos manufacturados

Es un sistema de producción por demanda, generalmente mediante el uso de una tarjeta física.

El objetivo de Kanban es minimizar el TEP (Trabajo En Progreso), o stock, entre los procesos. Para lograr esto Kanban se asegura que el proceso superior produzca partes sólo si el proceso inferior las necesita. "Por demada" significa que los trabajadores del proceso inferior consumen las partes que necesitan de los procesos superiores.

* + - * + Scrum

Modelo de desarrollo incremental.

Iteraciones (sprint) regulares cada 2 a 4 semanas.

Al principio de cada iteración se establecen sus objetivos priorizados (sprint backlog)

Al finalizar cada iteración se obtiene una entrega parcial utilizable por el cliente.

Existen reuniones diarias para tratar la marcha del sprint.

* + - * + XP (eXtreme Programming)

Valores

Simplicidad

Comunicación

Retroalimentación

Valentía o coraje

Respeto o humildad

Características

Diseño sencillo

Pequeñas mejoras continuas

Pruebas y refactorización

Integración continua

Programación por parejas

El cliente se integra en el equipo de desarrollo

Propiedad del código compartida

Estándares de codificación

40 horas semanales

* + - Metodologías ágiles (II)
      * Método basado en procesos creado en contraposición con las metodologías formales las cuales consideraban lentas y pesadas.
      * Algunos de sus valores son:
        + *Valorar más a los individuos y sus interacciones que a los procesos y las herramientas:* Los procesos deben ser una ayuda y un soporte para guiar el trabajo. Deben adaptarse a la organización, a los equipos y a las personas; y no al revés.
        + *Valorar más el software funcionando que la documentación exhaustiva:* Los documentos no pueden sustituir, ni pueden ofrecer la riqueza y generación de valor que se logra con la comunicación directa entre las personas y a través de la interacción con los prototipos.
        + *Valorar más la colaboración con el cliente que la negociación contractual:* Las prácticas ágiles están especialmente indicadas para productos difíciles de definir con detalle en el principio, o que si se definieran así tendrían al final menos valor que si se van enriqueciendo con retro-información continua durante el desarrollo. En el desarrollo ágil el cliente es un miembro más del equipo, que se integra y colabora en el grupo de trabajo. Los modelos de contrato por obra no encajan.
        + *Valorar más la respuesta ante el cambio que seguir un plan:* Para un modelo de desarrollo que surge de entornos inestables, que tienen como factor inherente el cambio y la evolución rápida y continua, resulta mucho más valiosa la capacidad de respuesta que la de seguimiento y aseguramiento de planes pre-establecidos

## —Lenguajes de programación

### Obtención de código ejecutable

* Para obtener código binario ejecutable tenemos 2 opciones:
  + Compilar
  + Interpretar

## Proceso de compilación/interpretación

* La compilación/interpretación del código fuente se lleva a cabo en dos fases:
  + Análisis léxico
  + Análisis sintáctico
* Si no existen errores, se genera el código objeto correspondiente.
* Un código fuente correctamente escrito no significa que funcione según lo deseado.
* No se realiza un análisis semántico.

## —Lenguajes compilados

* Ejemplos: C, C++
* Principal ventaja: Ejecución muy eficiente.
* Principal desventaja: Es necesario compilar cada vez que el código fuente es modificado.

## —Lenguajes interpretados

* Ejemplos: PHP, Javascript
* Principal ventaja: El código fuente se interpreta directamente.
* Principal desventaja: Ejecución menos eficiente.

## JAVA

* Lenguajes compilado e interpretado.
* El código fuente Java se compila y se obtiene un código binario intermedio denominado bytecode.
* Puede considerarse código objeto pero destinado a la máquina virtual de Java en lugar de código objeto nativo.
* Después este bytecode se interpreta para ejecutarlo.
* Ventajas:
  + Estructurado y Orientado a Objetos
  + Relativamente fácil de aprender
  + Buena documentación y base de usuarios
* Desventajas
  + Menos eficiente que los lenguajes compilados

## Tipos lenguajes

* Según la forma en la que operan:
  + Declarativos: indicamos el resultado a obtener sin especificar los pasos.
    - Tipos de lenguajes declarativos:
      * Lógicos: Utilizan reglas. Ej: Prolog
      * Funcionales: Utilizan funciones. Ej: Lisp, Haskell
      * Algebraicos: Utilizan sentencias. Ej: SQL
    - Normalmente son lenguajes interpretados.
  + Imperativos: indicamos los pasos a seguir para obtener un resultado.
    - Tipos de lenguajes imperativos:
      * Estructurados: C
      * Orientados a objetos: Java
      * Multiparadigma: C++, Javascript
    - Los lenguajes orientados a objetos son también lenguajes estructurados.
    - Muchos de estos lenguajes son compilados.
* Tipos de lenguajes según nivel de abstracción:
  + Bajo nivel: ensamblador
  + Medio nivel: C
  + Alto nivel: C++, Java

## Evolución

1. Código binario
2. Ensamblador
3. Lenguajes estructurados
4. Lenguajes orientados a objetos