

**Asignatura: Entornos de desarrollo**

Desarrollo del software

# ÍNDICE

## PRESENTACIÓN

---

1. Concepto de Software y tipos
2. Relación hardware-software
3. Desarrollo de Software
  1. Ciclos de vida del software
  2. Herramientas de apoyo para el desarrollo del software
4. Lenguajes de programación
  1. Concepto y características de los lenguajes mas difundidos
  2. Lenguajes de programación estructurados
  3. Lenguajes de programación orientados a objetos
5. Fases en el desarrollo y ejecución del software
  1. Análisis
  2. Diseño
  3. Codificación y tipos de código
  4. Pruebas
  5. Documentación
  6. Exploración
  7. Mantenimiento
6. Máquinas virtuales
  1. Frameworks
  2. Entornos de ejecución
  3. Java Runtime environment

# 1. Concepto de software y tipos

## Algoritmo

- Conjunto ordenado y finito de operaciones para conseguir una solución a un problema



# 1. Concepto de software y tipos

## Algoritmo



# 1. Concepto de software y tipos

Es un conjunto de programas informáticos que actúan sobre el hardware para ejecutar lo que el usuario desea obtener



# 1. Concepto de software y tipos

## TIPOS DE SOFTWARE

- **Sistema operativo**

- Instalado y configurado, permite que nuestras aplicaciones se puedan ejecutar y funcionen

- **Software de programación**

- Conjunto de herramientas que nos permiten desarrollar programas informáticos

- **Aplicaciones informáticas**

- Conjunto de programas con una finalidad concreta.
- Programa: conjunto de instrucciones escritas en un lenguaje de programación.

## 2. Relación hardware-software

- **Dos puntos de enfoque:**

- Desde el S.O.

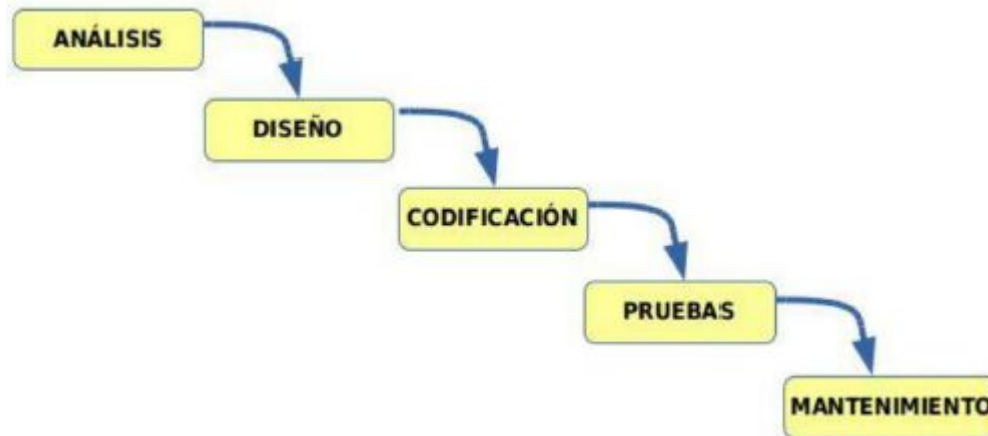
- El encargado de coordinar el hardware durante el funcionamiento del ordenador
    - Actúa como intermediario entre el hardware y las aplicaciones

- Desde el enfoque de las aplicaciones

- Aplicaciones como conjunto de programas
    - Programa como conjunto de instrucciones escritas en un lenguaje
    - Lenguaje: es interpretado y ejecutado por el hardware
      - Gran variedad de lenguajes; pero todos tienen en común que el ser humano es capaz de aprenderlos
      - Hardware: solo es capaz de interpretar señales eléctricas (0, 1)
      - Proceso traducción del Código: para que el hardware sea capaz de ejecutar las instrucciones

### 3. Desarrollo de Software

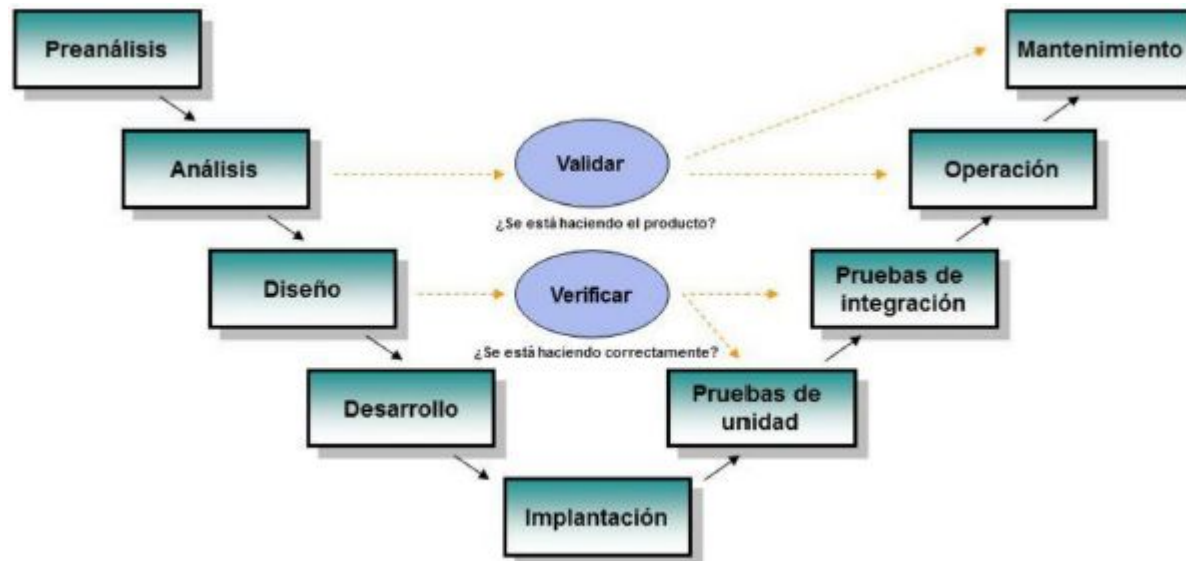
- Desarrollo: proceso desde la idea hasta el programa instalado.
- Ciclos de vida del software
  - Modelo cascada o clásico:





### 3. Desarrollo de Software

- **Modelo Cascada con Realimentación (Modelo V)**
  - Existe realimentación entre etapas (corregir, modificar, depurar, etc.)
  - Perfecto para proyectos rígidos, requisitos claros y poca evolución.

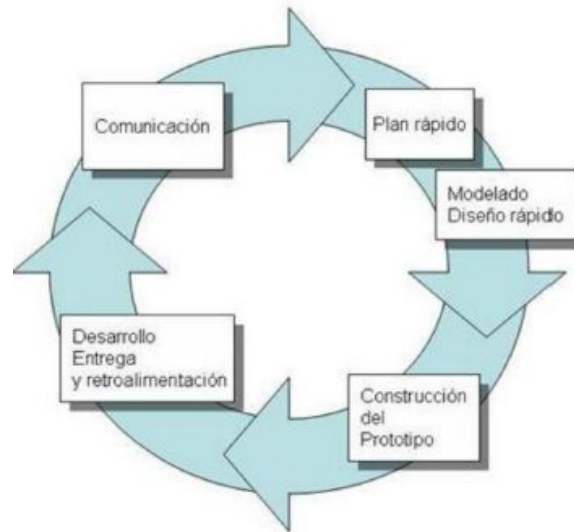


## 3. Desarrollo de Software

- **Modelos Evolutivos**

- **Modelo Iterativo Incremental**

- Cada etapa se mejora y propaga al resto de las fases



### 3. Desarrollo de Software

- **Modelos Espiral**

- El software se crea por versiones, añadiendo funcionalidad en cada versión



### 3. Desarrollo de Software

- **Herramientas de apoyo para el Desarrollo de software**

- **Herramientas CASE**

- Conjunto de aplicaciones para el Desarrollo de software, automatizando las fases del Desarrollo
    - RAD: Desarrollo rápido de aplicaciones
      - Desarrollo iterativo
      - Construcción de prototipos
      - Uso de herramientas CASE
    - Mejoras:
      - Mejora la planificación del Proyecto
      - Aporta agilidad al proceso
      - Se puede reutilizar software en otros proyectos
      - Las aplicaciones cumplen los estándares
      - Mejora el mantenimiento de los programas
      - Mejora el proceso de Desarrollo al permitir visualizar las fases de forma gráfica

### 3. Desarrollo de Software

- **Clasificación herramientas CASE**
  - **U-CASE**
    - Fase de planificación y análisis de requisitos
  - **M-CASE**
    - Fase de análisis y diseño
  - **L-CASE**
    - Fase de programación
    - Detección de errores
    - Generación de pruebas
    - Generación de documentación
  - **Ejemplos: UMLPad, Use Case Market**

## 4. Lenguajes de programación

- Es un idioma creado de forma artificial cuyo objetivo es conseguir “código hardware”
  - Clasificación
    - Lenguaje máquina
    - Lenguaje Ensamblador
    - Lenguaje de alto nivel basado en Código
    - Lenguajes visuales

## 4. Lenguajes de programación

- **Lenguaje MÁQUINA**

- Sus instrucciones son combinaciones de ceros y unos
- Es el único lenguaje que entiende directamente el ordenador (no necesita traducción)
- Fue el primer lenguaje utilizado
- Es único para cada procesador (no es exportable de un equipo a otro)
- Hoy ya nadie programa en éste lenguaje

## 4. Lenguajes de programación

- **Lenguaje ENSAMBLADOR**

- Sustituyó al lenguaje máquina para facilitar la labor de programación
- En lugar de unos y ceros se programa usando mnemotécnicos (instrucciones complejas)
- Necesita traducción al lenguaje máquina para poder ejecutarse
- Sus instrucciones son sentencias que hacen referencia a la ubicación física de los archivos en el equipo
- Es difícil de utilizar



## 4. Lenguajes de programación

- **Lenguaje de ALTO NIVEL basados en código**

- Sustituyeron al lenguaje ensamblador para facilitar más la labor de programación
- En lugar de mnemotécnicos, se utilizan sentencias y órdenes derivadas del idioma inglés, si bien en último caso serán traducidos al lenguaje máquina
- Son más cercanos al razonamiento humano
- Son utilizados hoy día, aunque la tendencia es descendente

## 4. Lenguajes de programación

### CARACTERÍSTICAS

- La elección de un determinado lenguaje dependerá del problema que deseamos resolver
- Clasificación por características
  - Según lo cerca que esté del lenguaje humano
    - LP de ALTO NIVEL: más próximos al razonamiento humano
    - LP de BAJO NIVEL: más próximos al funcionamiento del ordenador (ensamblador, máquina)
  - Según la técnica de programación utilizada
    - LP ESTRUCTURADOS: usan la programación estructurada: Pascal, C, Cobol, etc.
    - LP ORIENTADOS A OBJETOS: C++, Java, Ada, Delphi, etc.
    - LP VISUALES: basados en las técnicas anteriores, permiten programar gráficamente, y el código se genera de forma automática: Visual Basic.net

## 4. Lenguajes de programación

### LENGUAJES DE PROGRAMACIÓN ESTRUCTURADOS

- La programación estructurada se define como una técnica para escribir lenguajes de programación, que permite solo el uso de tres tipos de sentencias o estructuras de control:
  - Sentencias SECUENCIALES
  - Sentencias SELECTIVAS (condicionales)
  - Sentencias REPETITIVAS (iteraciones o bucles)
- **Ventajas**
  - Los programas son fáciles de leer, sencillos y rápidos
  - Fácil mantenimiento de los programas
  - Estructura del programa fácil y clara
- **Inconvenientes**
  - Programa concentrado en un bloque (si grande, difícil manejarlo)
  - No permite reutilizar su código. Por eso le sustituyó la prog. MODULAR

**Ejemplos de programas estructurados: Pascal, C, Fortran, Cobol...**

## 4. Lenguajes de programación

### LENGUAJES DE PROGRAMACIÓN ORIENTADOS A OBJETOS

- Trata el programa como un conjunto de objetos interrelacionados
- Los objetos tienen atributos que los hace diferenciarse
- Clase: colección de objetos con características similares
- **Ventajas**
  - Código reutilizable
  - Errores fácilmente localizables
- **Inconvenientes**
  - No es una programación intuitiva

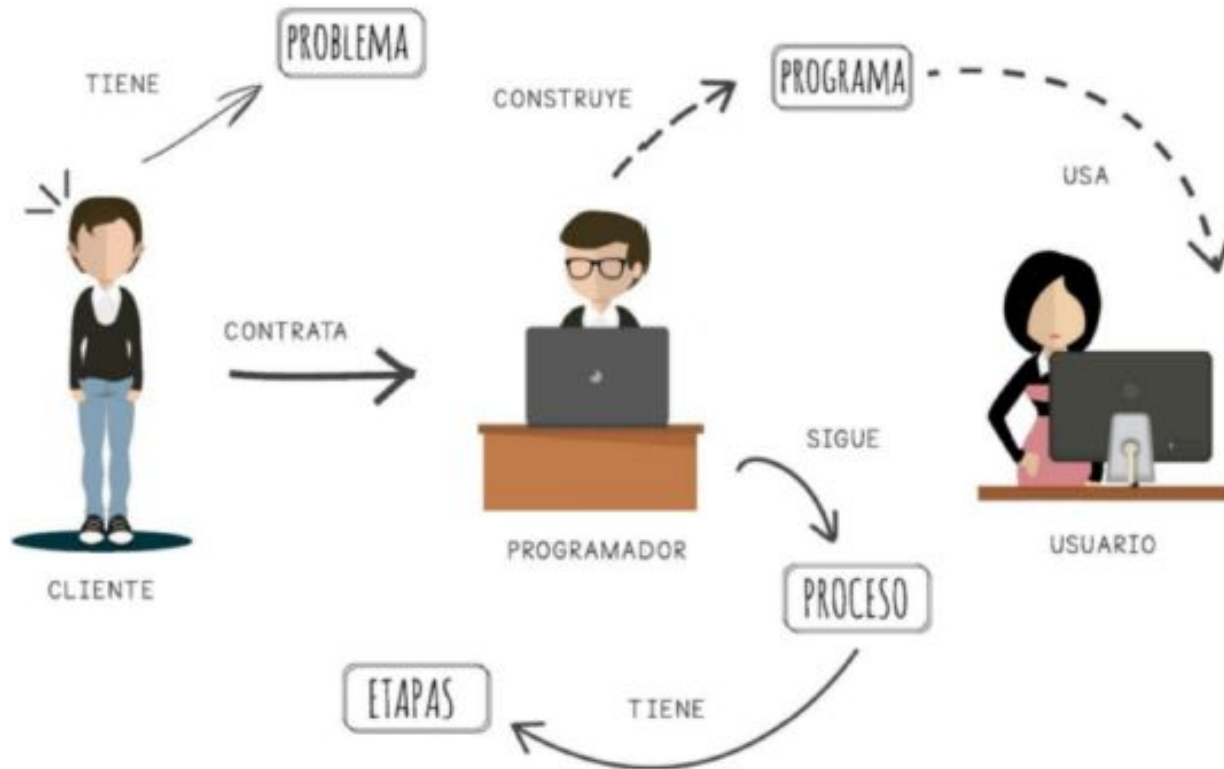
Ejemplos de programas estructurados: Ada, C++, Delphi, Java...

## 5. Etapas en el desarrollo software

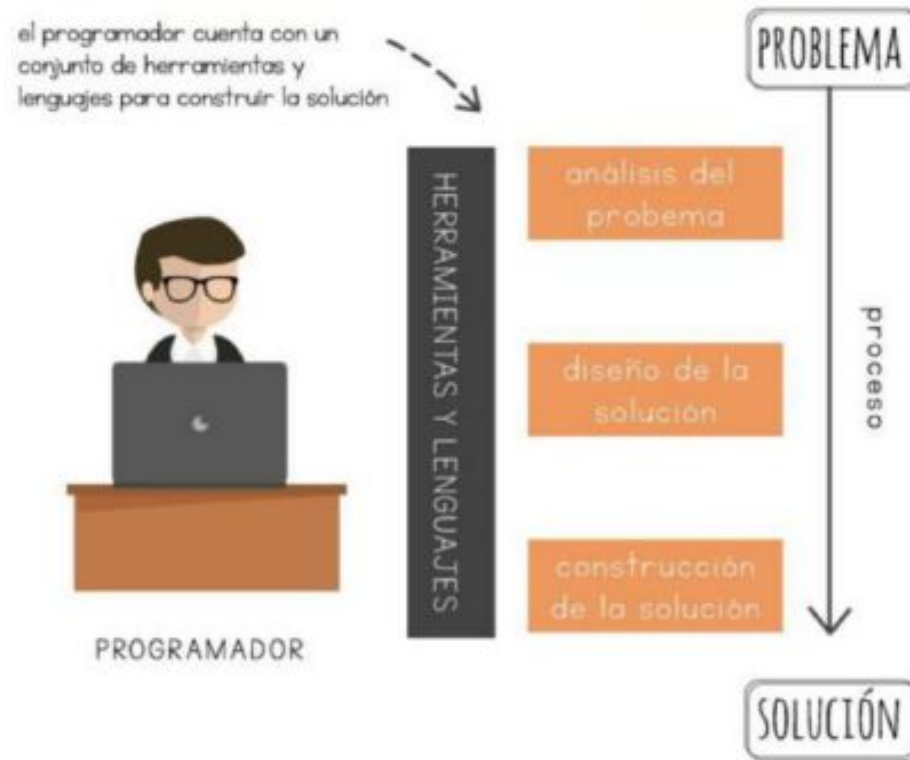
**Objetivo: dividir el Desarrollo en etapas para construir un software fiable y de calidad**

1. **Análisis de los requisitos:** se especifican los requisitos funcionales y no funcionales del Sistema
2. **Diseño:** se divide el sistema en partes y se determina la función de cada una
3. **Codificación:** se elige un lenguaje de programación y se codifican los programas
4. **Pruebas:** se prueban los programas para detectar errores y se depuran
5. **Documentación:** se documenta y guarda toda la información
6. **Explotación:** instalamos, configuramos y probamos la aplicación en los equipos del cliente
7. **Mantenimiento:** se mantiene el contacto con el cliente para actualizar y modificar la aplicación en el futuro.

## 5. Etapas en el desarrollo software






## 5. Etapas en el desarrollo software



## 5. Etapas en el desarrollo software

### ROLES en un proyecto de desarrollo software

	<b>Director de proyecto</b>	Responsable de la ejecución del proyecto con capacidad ejecutiva para tomar decisiones sobre el mismo de acuerdo con el cliente.
	<b>Ingeniero de requisitos</b>	También denominado <i>analista</i> . Responsable de interactuar con clientes y usuarios para obtener sus necesidades y de desarrollar y gestionar los requisitos.
	<b>Equipo de desarrollo</b>	Conjunto de personas implicadas en el desarrollo del software: arquitecto software, diseñador de IU, programador, responsable de pruebas, administrador de BD, etc.
	<b>Equipo de calidad</b>	Conjunto de personas responsables de la calidad de los productos obtenidos, tanto documentación como software. Suelen ocuparse también de la calidad de los procesos.
	<b>Cliente</b>	Responsable de la financiación del proyecto con capacidad ejecutiva para tomar decisiones sobre el mismo. Suele tener una visión global del modelo de negocio.
	<b>Usuario</b>	Usuario potencial del software a desarrollar en el proyecto con una visión detallada, aunque puede que parcial, del modelo de negocio.
	<b>Responsable TIC del Cliente</b>	Responsable del entorno tecnológico del cliente, sobre el que se debe integrar el sistema a desarrollar.



## 5. Etapas en el desarrollo del software

### 1. ANÁLISIS

- Es la fase de mayor importancia en el proyecto. Se especifican y analizan los requisitos funcionales y no funcionales del sistema
- **FUNCIONALES**
  - Qué funciones debe realizar la aplicación
  - Qué respuesta dará la aplicación ante todas las entradas
  - Cómo se comportará la aplicación en situaciones inesperadas
- **NO FUNCIONALES**
  - Tiempo de respuesta del programa
  - Legislación aplicable
  - Tratamiento antes la simultaneidad de peticiones, etc.

**FUNDAMENTAL:** buena comunicación entre el analista y el cliente

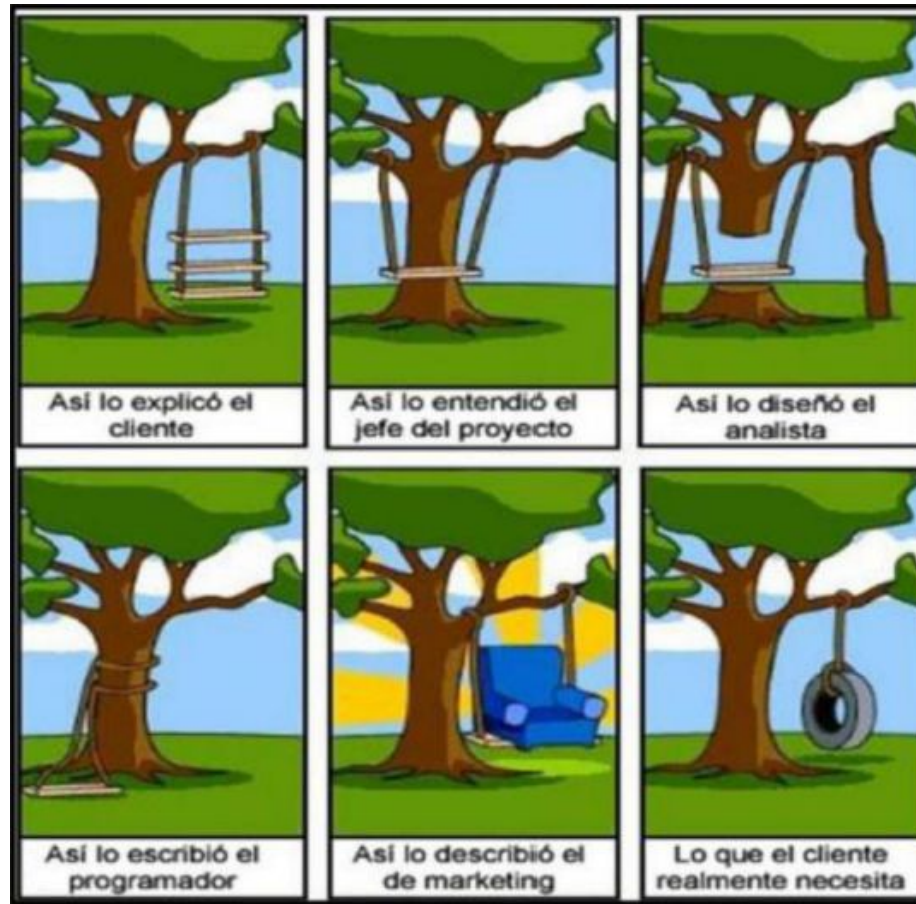
**La culminación de ésta fase es el documento ERS (Especificación de Requisitos Software) [SIGUE...]**

## 5. Etapas en el desarrollo del software

### ERS (Especificación de Requisitos Software)

- Planificación de las reuniones que van a tener lugar
- Relación de los objetivos del usuario cliente y del sistema
- Relación de los requisitos funcionales y no funcionales del sistema
- Relación de objetivos prioritarios y temporización
- Reconocimiento de requisitos mal planteados o que implican contradicciones, etc.

## 5. Etapas en el desarrollo del software



## 5. Etapas en el desarrollo del software

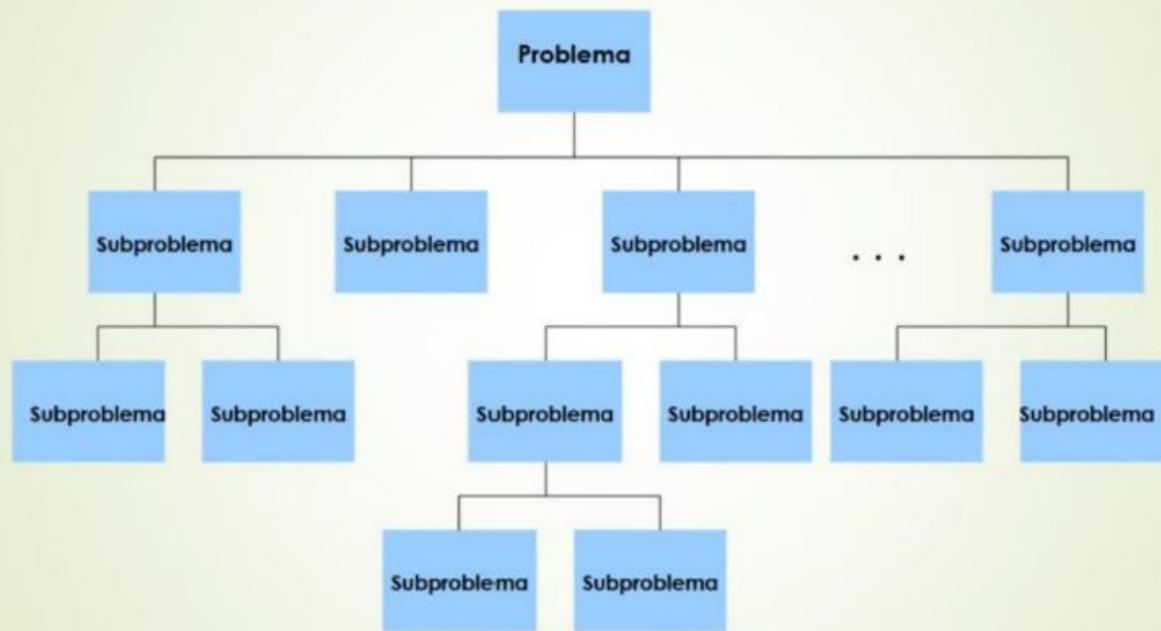
### 2. DISEÑO

- Durante esta fase donde ya sabemos lo que hay que hacer, el siguiente paso es: ¿cómo hacerlo?
- Debemos crear un modelo funcional-estructural de los requerimientos del sistema global para poder dividirlo y afrontar las partes por separado. En éste punto se deben tomar decisiones importantes, tales como:
  - Entidades y relaciones de las bases de datos
  - Selección del lenguaje de programación que se va a utilizar
  - Selección del Sistema Gestor de Base de Datos
  - Etc.

## 5. Etapas en el desarrollo del software

### 2. DISEÑO

#### Idea del diseño descendente



## 5. Etapas en el desarrollo del software

### 2. DISEÑO

- Ejemplo: se necesita una aplicación que calcule la superficie de un rectángulo.
- Dividimos el problema en 3 subproblemas:
  1. Entrada de datos
  2. Cálculo de la superficie
  3. Salida de resultados
- Hemos dejado dividido el problema en tres subproblemas más fáciles

## 5. Etapas en el desarrollo del software

### 2. DISEÑO

- Refinamos:

Subproblema 1: entrada de datos

1. Pedir altura del rectángulo
2. Pedir la base del rectángulo

Subproblema 2: Cálculo de la superficie

- Superficie = base x altura

Subproblema 3: Salida de resultados

1. Mostrar la base y la altura introducidos por teclado
2. Mostrar la superficie calculada

## 5. Etapas en el desarrollo del software

### 2. DISEÑO (otro ejemplo)

#### Un controlador de velocidad de un coche

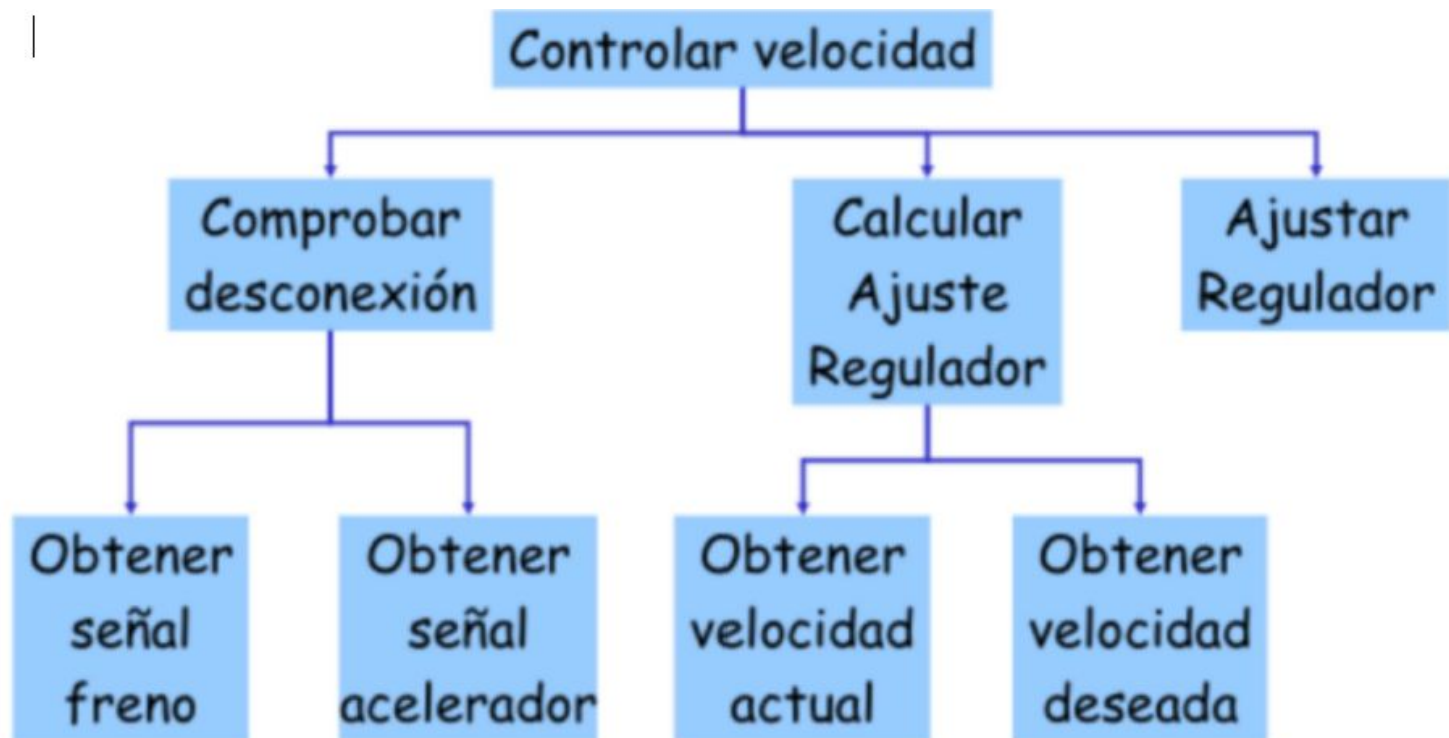
Objetivo: mantener una velocidad constante





## 5. Etapas en el desarrollo del software

### 2. DISEÑO (otro ejemplo)



## 5. Etapas en el desarrollo del software

### 3. CODIFICACIÓN. TIPOS DE CÓDIGO

- Elige el lenguaje de programación
- Codificar toda la información anterior y genera código fuente
- Tarea implementada por programador
- Cumplir exactamente con el análisis y en el diseño de la aplicación.
- Las características deseables de todo código son:
  - ▶ MODULARIDAD: que esté dividido en trozos más pequeños
  - ▶ CORRECCIÓN: que haga lo que realmente se le pide
  - ▶ FÁCIL DE LEER: para facilitar su desarrollo y mantenimiento futuros
  - ▶ EFICIENCIA: que haga un buen uso de los recursos
  - ▶ PORTABILIDAD: que se puede implementar en cualquier equipo

## 5. Etapas en el desarrollo del software

### FASES EN LA PROGRAMACIÓN

- **Código FUENTE**

- Lenguaje de programación alto nivel
- Conjunto de instrucciones necesarias

- **Código OBJETO**

- Compilación
  - Traducción una sola vez del programa mediante el compilador, que genera código binario a partir del fuente
- Interpretación
  - Traducción y ejecución simultánea del programa, línea a línea
  - Es el código Objeto, que no es legible por la máquina ni por un humano. Es interpretado línea a línea

- **Código EJECUTABLE**

- Es el código binario resultante de enlazar los archivos de código objeto con ciertas rutinas y bibliotecas necesarias
- Es conocido como código máquina y ya sí es directamente inteligible por el ordenador

**Los programas interpretados no producen código objeto, el paso al ejecutable es directo**

## 5. Etapas en el desarrollo del software

### 4. PRUEBAS

- Se hacen sobre un conjunto de datos predefinidos
- Imprescindibles para validar y verificar el software
- Tipos:
  - Unitarias
    - Se prueba una a una las diferentes partes del software de forma independiente
    - junit (prueba de aplicaciones Java)
  - De Integración
    - Comprueba el funcionamiento del sistema completo
    - Se prueba la interrelación de los diferentes componentes
    - Prueba final BETA TEST: se prueba en entorno de producción del cliente

## 5. Etapas en el desarrollo del software

### 5. DOCUMENTACIÓN

- Debe documentarse cada una de las etapas
- Ayuda a los usuarios a conocer el software
- Facilita la revisión

## 5. Etapas en el desarrollo del software

### 5. DOCUMENTACIÓN

	GUÍA TÉCNICA	GUÍA DE USO	GUÍA DE INSTALACION
QUEDAN REFLEJADOS	<ul style="list-style-type: none"><li>• El diseño de la aplicación.</li><li>• La codificación de los programas.</li><li>• Las pruebas realizadas.</li></ul>	<ul style="list-style-type: none"><li>• Descripción de la funcionalidad de la aplicación.</li><li>• Forma de comenzar a ejecutar la aplicación.</li><li>• Ejemplos de uso del programa.</li><li>• Requerimientos software de la aplicación.</li><li>• Solución de los posibles problemas que se pueden presentar.</li></ul>	<p>Toda la información necesaria para:</p> <ul style="list-style-type: none"><li>• Puesta en marcha.</li><li>• Explotación.</li><li>• Seguridad del sistema.</li></ul>
¿A QUIÉN VA DIRIGIDO?	Al personal técnico en informática (analistas y programadores).	A los usuarios que van a usar la aplicación (clientes).	Al personal informático responsable de la instalación, en colaboración con los usuarios que van a usar la aplicación (clientes).
¿CUÁL ES SU OBJETIVO?	Facilitar un correcto desarrollo, realizar correcciones en los programas y permitir un mantenimiento futuro.	Dar a los usuarios finales toda la información necesaria para utilizar la aplicación.	Dar toda la información necesaria para garantizar que la implantación de la aplicación se realice de forma segura, confiable y precisa.

## 5. Etapas en el desarrollo del software

### 6. EXPLORACIÓN

- Previamente
  - Se demuestra que el software es fiable
  - Se documentan todas las fases
  - Se verifica que se cumplen los requisitos
- Fase en la que los usuarios finales conocen la aplicación y comienzan su uso
- Fase de INSTALACIÓN
  - El programa es transferido al ordenador del cliente, luego configurado
  - En éste momento se hace la prueba Beta Test
- Fase de CONFIGURACIÓN
  - Se asignan los parámetros de configuración de la empresa
- Fase de PRODUCCIÓN
  - La aplicación pasa a manos de los usuarios finales
  - Comienza la explotación del software

**Momento más CRÍTICO del proyecto: presentarle el producto final al cliente**

## 5. Etapas en el desarrollo del software

## 6. MANTENIMIENTO

- La etapa más larga del proyecto
- Por definición, el software debe cambiarse, actualizarse y evolucionar en el tiempo
  - Evolución del hardware
  - Corrección de errores
  - Evolución y nuevas necesidades en la empresa
- Se pacta con la empresa un servicio de mantenimiento
- Tipos de cambios
  - Perfectivos: mejoras en el funcionamiento del software
  - Evolutivos: nuevas necesidades del cliente
  - Adaptativos: actualizaciones por hardware, nuevas tecnologías del mercado
  - Correctivos: se solucionan errores del programa



## 5. Etapas en el desarrollo del software

### RESULTADOS DE CADA FASE

- **ANÁLISIS**
  - Especificaciones del sistema
  - Especificaciones de requisitos software
- **DISEÑO**
  - Documentación de arquitectura del software
  - Especificación de módulos y funciones
- **PRUEBAS**
  - Pruebas unitarias: módulos utilizables
  - Pruebas de integración sistemas utilizables
- **DOCUMENTACIÓN**
  - Documentación Técnica
  - Documentación de Usuario
- **MANTENIMIENTO**
  - Informes de errores
  - Control de cambios

## 6. Máquinas virtuales

- Software 'especial' que separa el 'funcionamiento' del ordenador de sus componentes hardware
- Permite crear diferentes entornos de desarrollo (SO+Aplicaciones deDesarrollo), para ejecutar aplicaciones en diferentes entornos
- Funciones principales:
  - Conseguir que las aplicaciones sean portables
  - Reservar memoria para los objetos que se crean, y liberar la memoria no utilizada
  - Comunicarse con el sistema donde se instala la aplicación (huésped), para el control de los dispositivos hardware implicados en los procesos
  - Cumplimiento de las normas de seguridad de las aplicaciones

## 6. Máquinas virtuales

### CARACTERÍSTICAS

- La máquina virtual actúa de puente entre la aplicación y el hardware concreto del equipo donde se instala
- Elementos:
  - Framework
  - Entornos de ejecución
- Java Runtime Environment

## 6. Máquinas virtuales

### 1. FRAMEWORKS

- Estructura de la cual se parte para desarrollar proyectos
- Plataforma software que almacena:
  - Programas de soporte
  - Bibliotecas
  - Lenguajes de programación
  - Módulos ya desarrollados
- Ventajas
  - Desarrollo rápido de software
  - Reutilización de partes de código para otras aplicaciones
  - Diseño uniforme del software
  - Portabilidad de aplicaciones de un ordenador a otro

## 6. Máquinas virtuales

- **Inconvenientes**

- Dependencia del código respecto al framework (si cambiamos de framework hay que reescribir gran parte de la aplicación)
- Consumo elevado de recursos del sistema

- **Ejemplos de frameworks**

- .NET es un framework para desarrollar aplicaciones sobre Windows. Visual Studio .net nos da facilidades para construir aplicaciones, y su motor es el '.net framework'
- Spring, de Java. Con conjuntos de aplicaciones (APIs), para el desarrollo y ejecución de aplicaciones

## 6. Máquinas virtuales

### 2. ENTORNOS DE EJECUCIÓN

- Servicio de la máquina virtual que sirve como base software para la ejecución de programas
- Conjunto de utilidades que permite la ejecución de programas
- Forma parte del SO o puede ser un servicio independiente
- Durante la ejecución de un programa, los entornos se encargan de:
  - Configurar la memoria principal disponible en el sistema
  - Enlazar los archivos del programa con:
    - Bibliotecas: subprogramas que sirven para desarrollar componentes software
    - Subprogramas: los que se han creado a propósito para el programa
  - Depurar programas: comprobar la existencia de errores semánticos
- Funcionamiento: está formado por MÁQUINA VIRTUAL + APIs
- El entorno sirve de intermediario entre el lenguaje fuente y el sistema operativo para ejecutar aplicaciones

## 6. Máquinas virtuales

### 2. ENTORNOS DE EJECUCIÓN

- Servicio de la máquina virtual que sirve como base software para la ejecución de programas
- Conjunto de utilidades que permite la ejecución de programas
- Forma parte del SO o puede ser un servicio independiente
- Durante la ejecución de un programa, los entornos se encargan de:
  - Configurar la memoria principal disponible en el sistema
  - Enlazar los archivos del programa con:
    - Bibliotecas: subprogramas que sirven para desarrollar componentes software
    - Subprogramas: los que se han creado a propósito para el programa
  - Depurar programas: comprobar la existencia de errores semánticos
- Funcionamiento: está formado por MÁQUINA VIRTUAL + APIs
- El entorno sirve de intermediario entre el lenguaje fuente y el sistema operativo para ejecutar aplicaciones

## 6. Máquinas virtuales

### JAVA RUNTIME ENVIRONMENT (JRE)

- Entorno en tiempo de ejecución Java
- Se compone de un conjunto de utilidades que permitirán la ejecución de programas Java sobre cualquier tipo de plataforma (Windows, Ubuntu, etc.)
- Componentes
  - Máquina virtual Java (JVM), que interpreta el código escrito en lenguaje Java
  - Bibliotecas estándar que implementa la API de Java



