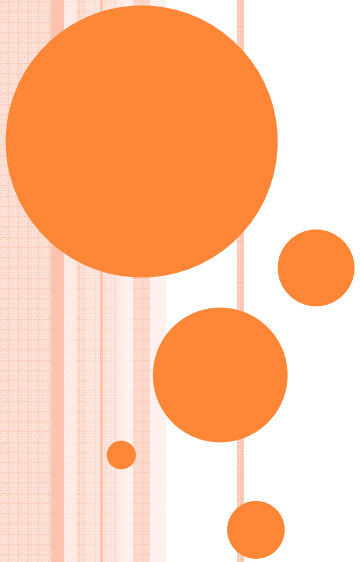


Introducción al desarrollo de Software

“Entornos de Desarrollo”

DAM

ÍNDICE

- Software: Clasificación y licencias
- Hardware.
  - Arquitectura von Neumann
  - Unidad de control
- Sistema operativo
- Sistema de información
- Sistema informático
- Información y datos
- Programa informático
- Lenguajes de programación. Clasificación
- Compilación
- Introducción a Ingeniería del Software
  - Ciclo de Vida: Fases del desarrollo de una aplicación
  - Modelos de proceso de desarrollo de software
  - Metodologías de desarrollo software
- Herramientas Case

SOFTWARE

- RAE: “Conjunto de programas, instrucciones y reglas informáticas para ejecutar ciertas tareas en una computadora”
- Estándar 729 IEEE: “Conjunto de los programas de cómputo, procedimientos, reglas, documentación y datos asociados que forman parte de las operaciones de un sistema de computación”
- Categorías:
  - Por el tipo de tarea que realiza
  - Método de distribución

SOFTWARE

- Según el tipo de tarea que realiza
  - Software de sistema: permite que el hardware funcione. Sistemas operativos, controladores de dispositivo…
    - BIOS (basic input/output system)
      - software que localiza y reconoce todos los dispositivos necesarios para cargar el sistema operativo en la memoria RAM;
      - es un software muy básico instalado en la placa base que permite que ésta cumpla su cometido
  - Software de aplicación: Programas que ayudan a realizar tareas específicas en cualquier campo susceptible de ser automatizado. Ej. Aplicaciones ofimáticas, ERP, CAD,…
  - Software de programación o desarrollo: proporcional al programador herramientas para ayudares a escribir programas informáticos . Ej. IDEs

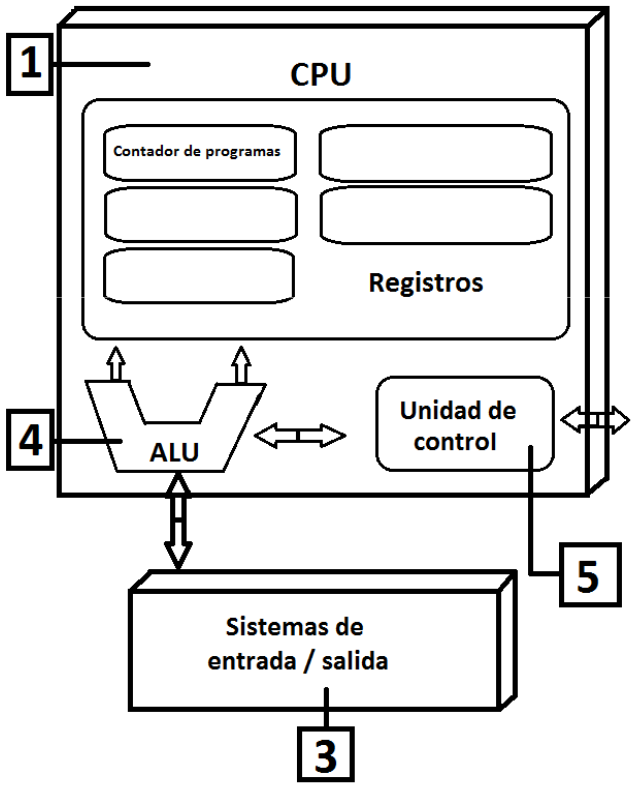
SOFTWARE

- Según el método de distribución
  - Shareware: Evaluación de forma gratuita por un tiempo. Ej. WinRar
  - Freeware: Software que se distribuye sin cargo. Suele incluir licencia de uso que permite su redistribución pero con algunas restricciones, como no modificar la aplicación en sí, venderla …
  - Adware: Suelen ser programas Shareware que de forma automática descargan publicidad en nuestro ordenador cuando se ejecutan o instalan !!
  - Software multimedia: presentan de forma integrada textos, gráficos, sonidos y animaciones (ej. Enciclopedias multimedia)
  - Software de uso específico: Software desarrollado a medida para resolver un problema específico. Ej. Software de un banco

SOFTWARE

- Licencias
  - Contrato entre el desarrollador y el usuario
  - Software libre: el autor cede una serie de libertades a usuario en el marco de una licencia
    - Libertad de uso con cualquier fin en cuantos ordenadores quiera
    - Libertad de adaptar el código a necesidades específicas
    - Libertad de distribuir copias a otros usuarios
    - Libertad de mejorar el programa y hacer públicas y distribuir la versión mejorada
- Software de dominio público : carece de licencia
- Licencia GPL (GNU General Public License)
  - Da derecho al usuario a usar y modificar el programa. Ej. Moodle
- http://www.gnu.org/licenses/license-list.es.html#SoftwareLicences

HARDWARE
Arquitectura von Neumann



1.CPU

Interpreta las instrucciones contenidas en los programas y procesa los datos.

2. Memoria Principal

Retienen datos informáticos durante algún intervalo de tiempo.

3.Dispositivos de entrada/salida

Son los dispositivos con los cuales el ordenador se comunica con el mundo exterior

4.Unidad aritmético-lógica

Circuito digital que calcula operaciones aritméticas y operaciones lógicas entre dos números.

5.Unidad de control

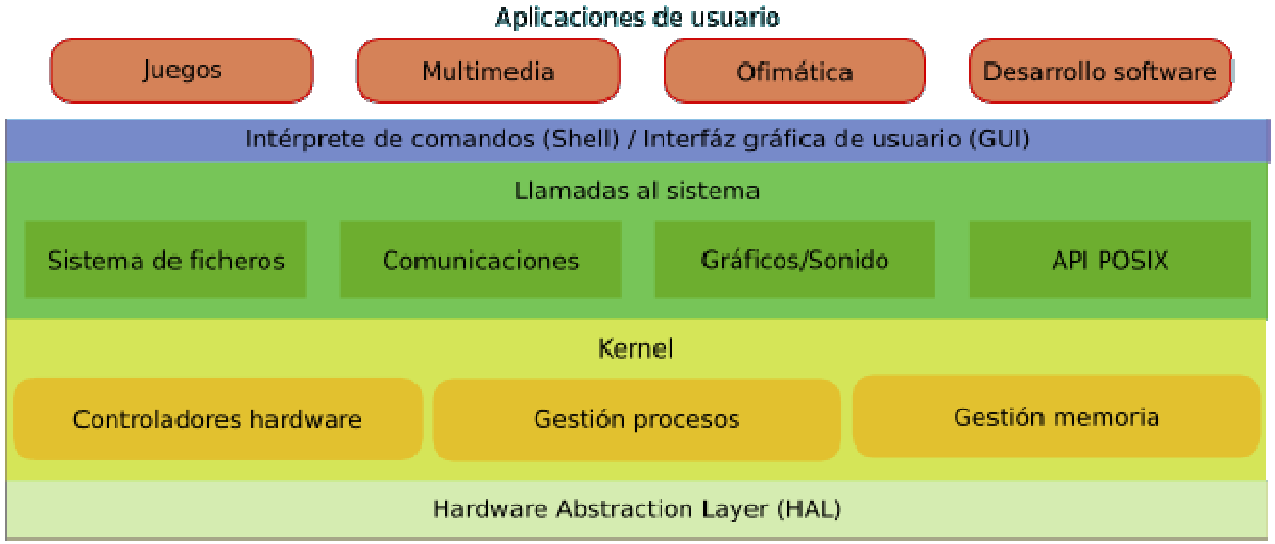
Busca las instrucciones en la memoria principal,las interpreta y las ejecuta.

HARDWARE

- Parte física de un sistema informático, por ejemplo, un disco duro, un monitor, una tarjeta gráfica, cables, etc.
- Arquitectura Von Neumann describe los componentes hw
  - Unidad central de proceso (CPU) : encargada de controlar, coordinar y realizar todas las operaciones de un sistema informático.
    - Unidad de control (UC): detecta por medio de señales eléctricas el estado de cada uno de los elementos conectados al ordenador y gobierna las unidades de entrada, salida y entrada / salida, además de interpretar y ejecutar las instrucciones que constituyen los programas.
    - Unidad aritmético lógica (U.A.L): Es la parte del procesador encargada de realizar todas aquellas operaciones de tipo aritmético y tipo lógico.
    - Registros: de trabajo o de propósito general. Almacenamiento interno de la CPU
  - Memoria
    - Principal o RAM : parte del sistema donde se almacenan temporalmente los programas que se van a ejecutar junto con los datos que queremos procesar
    - Memoria auxiliar, son dispositivos de almacenamiento masivo de información.

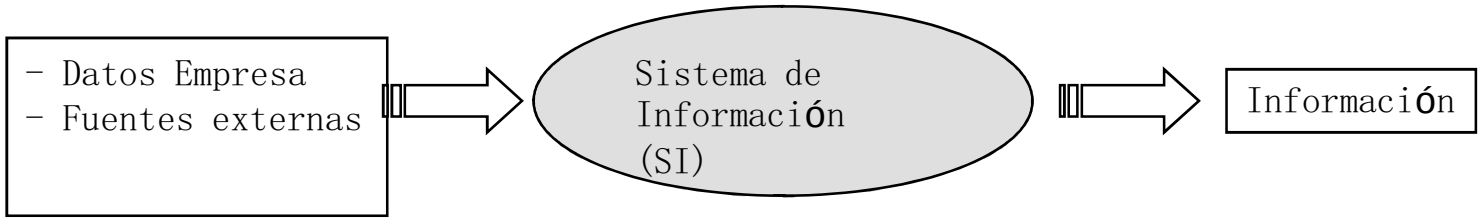
SISTEMA OPERATIVO

- El objetivo fundamental de los sistemas operativos es gestionar y administrar los recursos del Sistema convirtiéndose en el mediador entre el usurario y el Hardware.



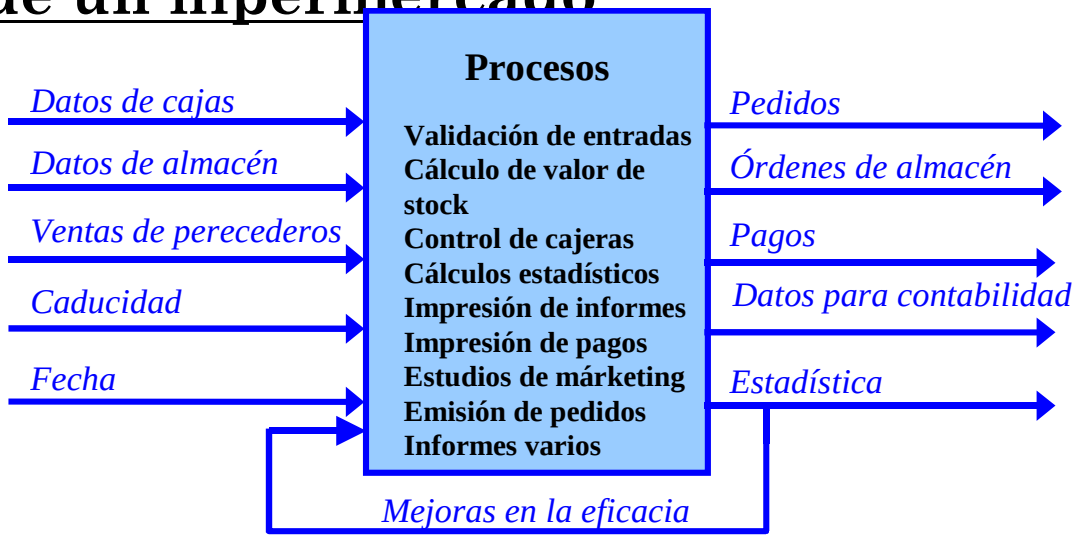
Sistema de Información

- Proporcionan servicio a todos los demás sistemas de una organización.
- El SI toma los datos :
  - de la empresa
  - de fuentes externas
  - Elabora la información que ha de servirles para:
    - la gestión
    - toma de decisiones en esta empresa.



Sistema de Información

- **Sistema de Información (SI).**
  - "conjunto de elementos, entre los que se encuentran los procedimientos, las personas y la propia información, que tiene como finalidad suministrar la información necesaria en el lugar adecuado y en el momento oportuno para contribuir al buen funcionamiento de la empresa y la adecuada toma de decisiones".
- **Ej. SI de un hipermercado**



Sistema de Información

- Elementos de un sistema de información
  - Relación con un sistema de informatización automatizado o sistema informático

SI	SIA
Procedimientos	**Software**
Información	**Datos**
Personas	**Usuarios**
Soporte para guardar información	**Hardware**

Sistema Informático

- Sistema de información automatizado
- Es un conjunto de elementos que permiten procesar información por medio de equipos informáticos y cuya finalidad es la de obtener nueva información a partir de la ya existente. Todo sistema informático queda dividido de forma global en cuatro capas o niveles que son:
  - Hardware
  - Sistema operativo
  - Programas de aplicación
  - Recursos humanos, que son aquellas personas encargadas del desarrollo, implantación, explotación y mantenimiento de un sistema informático.

Información y datos

- La información debe de ser:
  - **exacta**: estar libre de errores de cálculo o transmisión
  - **precisa**: no debe diferir entre el valor obtenido y el real
  - **oportuna**: no debe de tardar mucho tiempo entre que se emite y llega al usuario
  - **plena**: debe ser completa
  - **significativa**: máximo contenido semántico posible
  - **coherente**: con el resto de información disponible
  - **segura**: protegida del deterioro o acceso no autorizado. (*LOPD "Ley Orgánica de Protección de Datos"*)

Programa Informático

- Conjunto de instrucciones escritas en un lenguajes de programación que aplicadas sobre un conjunto de datos resuelven un problema o parte del mismo
- Para ejecutarlo necesita ser traducido a un lenguaje máquina → Compilador o intérprete
- Proceso
  - Programa en ejecución
    - Un programa puede desencadenar varios procesos

LENGUAJES DE PROGRAMACIÓN

- Conjunto de caracteres, las reglas para la combinación de esos caracteres y las reglas que definen sus efectos cuando los ejecuta un ordenador
- Elementos
  - Alfabeto o vocabulario (léxico): conjunto de símbolos permitidos
  - Sintaxis: reglas que indican cómo realizar las construcciones con los símbolos del lenguaje
  - Semántica: reglas que determinan el significado de cualquier construcción del lenguaje
- Clasificación. Según diferentes criterios
  - Nivel de abstracción: Alto nivel, nivel medio, bajo nivel
  - Forma de ejecución: Compilados, interpretados
  - Paradigma de programación:
    - Imperativos, funcionales, lógicos, estructurados, OO

LENGUAJES DE PROGRAMACIÓN

- Clasificación atendiendo a nivel de abstracción
  - **Lenguajes de bajo nivel**
    - Se encuentran más próximos a la arquitectura de la maquina
    - Lenguaje máquina o código ejecutable
      - Lenguaje inteligible directamente por un ordenador.
      - Se basa en la combinación de dos únicos símbolos, el 0 y el 1, denominados bits.
      - Es propio de un determinado procesador, es decir, que cada procesador tiene su propio y particular lenguaje maquina, que no podrá ser entendido por cualquier otro.
    - Lenguaje ensamblador
      - Se programa utilizado nombres nemotécnicos y las instrucciones trabajan directamente con registros de la memoria física de la máquina
      - Los programas escritos en ensamblador no son transportables, es decir, un programa escrito para un microprocesador concreto no funcionará para un microprocesador diferente
      - Ejemplo. La instrucción a = a + b; en un lenguaje de alto nivel vista en ensamblador :

```
LDR rb,b
LDR rc,c
ADD ra,rb,rc
STR ra,a
```

LENGUAJES DE PROGRAMACIÓN

- Clasificación atendiendo a nivel de abstracción
  - **Lenguajes de nivel medio**
    - Con algunas características que los acercan a os de bajo nivel, pero también de alto nivel
    - Ej. Lenguaje C. Se suelen utilizar para aplicaciones como la creación de sistemas operativos
  - **Lenguajes de alto nivel**
    - Suelen estar formaos por palabras del lenguaje natural (inglés)
    - Son independientes de la máquina. No dependen del hardware del ordenador.
    - Requieren ser compilados o interpretados
    - Ej. C++, C#, Java, Cobol, PL/SQL, Pascal, PHP, Python,…

LENGUAJES DE PROGRAMACIÓN

- Clasificación atendiendo a forma de ejecución
  - **Lenguajes Compilados**
    - Compilador:
      - Es un programa informático que traduce un programa escrito en un lenguaje de programación a otro lenguaje de programación, generando un programa equivalente que la máquina será capaz de interpretar. Usualmente el segundo lenguaje es lenguaje de máquina, pero también puede ser simplemente texto.
    - Enlazador (linker)
      - Es un programa que toma los objetos generados en los primeros pasos del proceso de compilación, la información de todos los recursos necesarios (biblioteca), quita aquellos recursos que no necesita, y enlaza el código objeto con su(s) biblioteca(s) con lo que finalmente produce un fichero ejecutable o una biblioteca. En el caso de los programas enlazados dinámicamente, el enlace entre el programa ejecutable y las bibliotecas se realiza en tiempo de carga o ejecución del programa.
    - Código objeto
      - Código que resulta de la compilación del código fuente
      - Consiste en lenguaje máquina o bytecode y se distribuye en varios archivos que corresponden a cada código fuente compilado. Para obtener un programa ejecutable se han de enlazar todos los archivos de código objeto con un programa llamado enlazador (linker).

LENGUAJES DE PROGRAMACIÓN

- Clasificación atendiendo a forma de ejecución
  - **Lenguajes interpretados**
    - Los intérpretes se encargan de que cada vez que se ejecuta una instrucción, se debe interpretar y traducir a lenguaje máquina
      - Son más lentos en ejecución que los compilados
    - Ej. PHP, Java,
    - Java incluye un proceso de compilación generando un formato intermedio llamado bytecode
      - No es entendible por el sistema operativo
      - Requiere la máquina virtual de java (JRE) para poder ejecutarlo

LENGUAJES DE PROGRAMACIÓN

- Según el paradigma de programación
  - Es un enfoque particular para la construcción del software que incluye un conjunto de reglas, patrones y estilos de programación
  - Un lenguaje de programación puede usar más de un paradigma
  - **Lenguajes imperativos**
    - La sentencia principal es la asignación. Las estructuras de control permiten establecer orden de ejecución y cambiar el flujo del programa.
    - Dentro de esta categoría se engloba la **programación estructurada, la programación modular y la programación orientada a objetos.**
    - Ejemplo en C

```
int main() {
      int x;
      int fact = 1;
      scanf ("%i", &x);
      for (int i= 2; i <= x; i++){
          fact = fact * i;
      }
      printf ("Factorial = %i", fact);
}
```

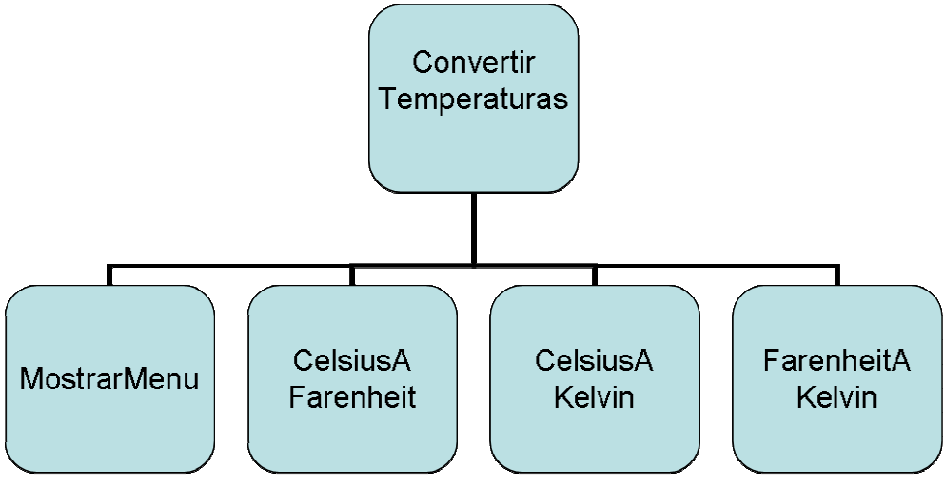
LENGUAJES DE PROGRAMACIÓN

- Según el paradigma de programación
  - **Lenguajes funcionales**
    - Está basado en el concepto matemático de función
    - En los funcionales
      - No existe la operación de asignación
      - Las variables almacenan definiciones o referencias a expresiones
      - La operación fundamental es la aplicación de una función a una serie de argumentos
      - La computación se realiza mediante la evaluación de expresiones
    - Ejemplos: Lisp, Scheme, ….
    - Ejemplo en lisp

```
(defun factorial (N)
    (if  (= N 1)
      1
      ( * N (factorial ( - N 1)))))
```

LENGUAJES DE PROGRAMACIÓN

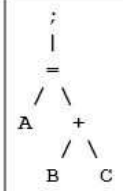
- Según el paradigma de programación
  - **Lenguajes lógicos**
    - En estos lenguajes un cálculo es el proceso de encontrar qué elementos de un dominio cumplen determinada relación definida sobre dicho dominio o el proceso de determinar si un elemento cumple o no dicha relación
    - Ejemplo Prolog. (para sistemas expertos…)
  - **Lenguajes de programación estructurados**
    - Estructuras de control fundamento de la programación estructurada: secuencial, condicional, repetitiva
    - Un programa se dividirá en módulos
      - No usar variables globales



LENGUAJES DE PROGRAMACIÓN

- Según el paradigma de programación
  - **Lenguajes orientados a objetos**
    - Compuesto por un conjunto de objetos, que consta de una estructura de datos y de una colección de métodos u operaciones que manipulan esos datos.
    - Los objetos se comunican unos con otros a través de paso de mensajes.
    - Un clase es la visión genérica del objeto. Un objeto es la instanciación de la clase
    - Ejemplos : C++, C#, Java, …

The Phases of a Compiler

Phase	Output	Sample
Programmer (source code producer)	Source string	`A=B+C;`
Scanner (performs *lexical analysis*)	Token string	`'A','=','B','+','C',';'` And *symbol table* with names
Parser (performs *syntax analysis* based on the grammar of the programming language)	Parse tree or abstract syntax tree	
Semantic analyzer (type checking, etc)	Annotated parse tree or abstract syntax tree	

Intermediate code generator	Three-address code, quads, or RTL	`int2fp B t1`
		`+ t1 C t2`
		`:= t2 A`

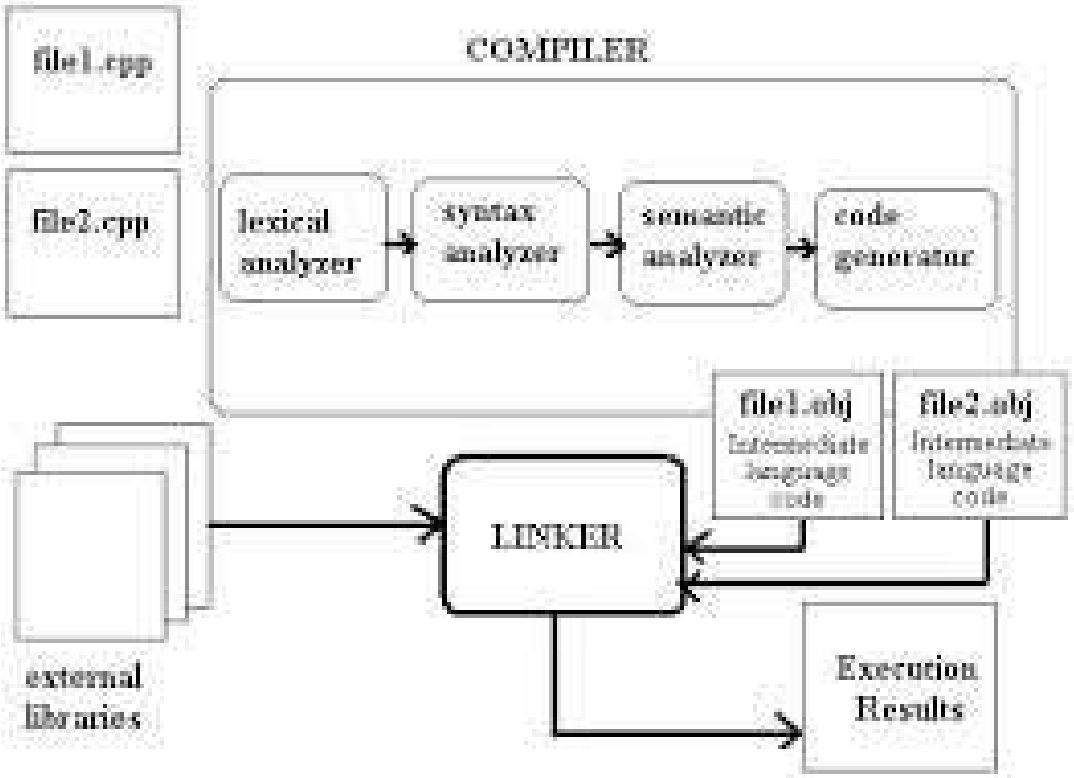
Optimizer	Three-address code, quads, or RTL	`int2fp B t1` `+ t1 #2.3 A`

Code generator	Assembly code	`MOVF #2.3,r1`
		`ADDF2 r1,r2`
		`MOVF r2,A`

Peephole optimizer	Assembly code	`ADDF2 #2.3,r2` `MOVF r2,A`

COMPILACIÓN

- Etapas en la creación de un programa
  - Edición
  - Compilación
  - Linkado o montaje
  - Ejecución



CONCEPTOS Y DEFINICIONES

- Clasificación de lenguajes de alto nivel
  - Lenguajes estructurados
    - Utilizan instrucciones como unidad de trabajo de los programas (Cobol, Pascal, C, Ada).
  - Lenguajes orientados a objetos
    - La unidad de proceso es el objeto (instanciación de Clase la cual pertenece a una Jerarquía) y en el se incluyen los datos (variables) y las operaciones que actúan sobre ellos (Smalltalk, C++, Java)
  - Lenguajes declarativos
    - Los programas se construyen mediante descripciones de funciones o expresiones lógicas (Lisp, Prolog). Dentro de este punto tenemos los lenguajes lógicos basados en la lógica de primer orden y los lenguajes funcionales basados en *lambda-Calculo*.

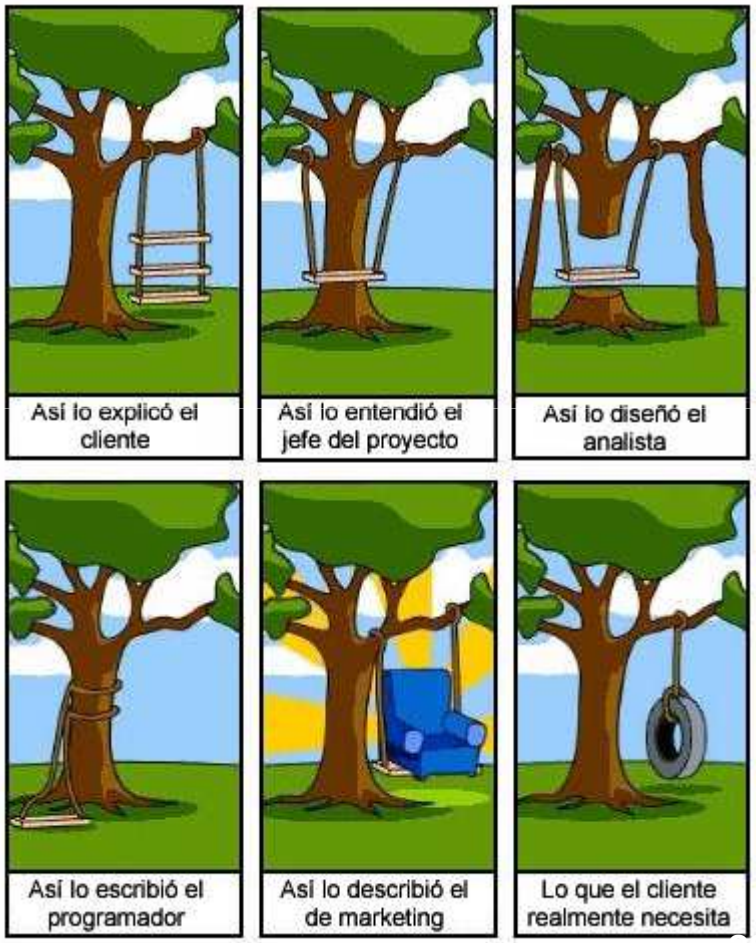
INTRODUCCIÓN A LA INGENIERÍA DEL SOFTWARE

- Software =

  Instrucciones + Estructuras de Datos + Documentación

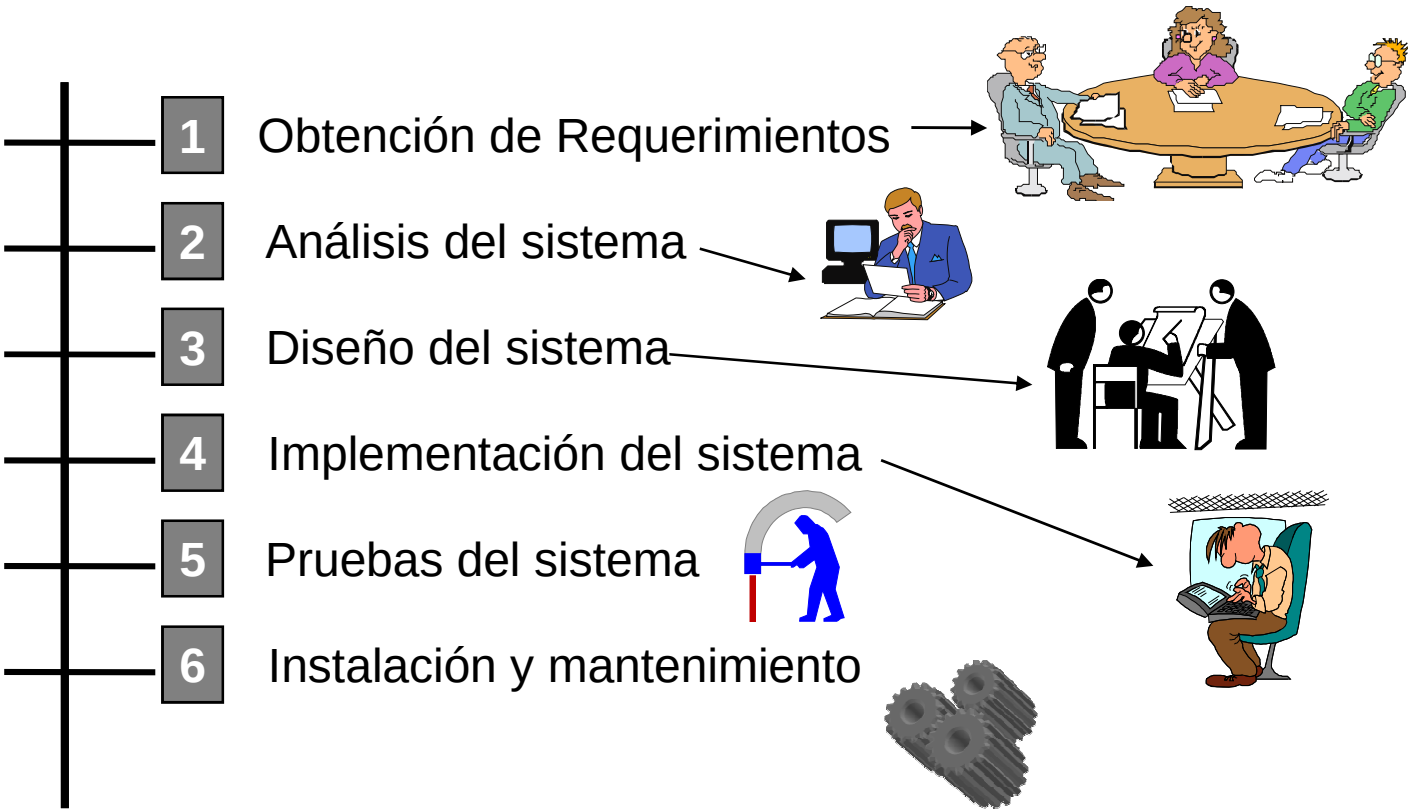
- Crisis del software

1. Aumento de la necesidad de CREACIÓN de SW
2. Se desarrolla software de manera desorganizada
3. Software de mala calidad. SW Funciona mal



INTRODUCCIÓN A LA ING. SW.

- **Ciclo de vida:** "Todas las etapas por las que pasa un proyecto de desarrollo de software desde que se inicia hasta que se finaliza y se considera una solución completa, correcta y estable"



CICLO DE VIDA : FASES

- QUÉ
  - Se va a desarrollar

1. Estudio del Sistema
2. Planificación del Sistema
3. Especificación de Requisitos
4. Análisis del Sistema

- CÓMO
  - Se va a desarrollar

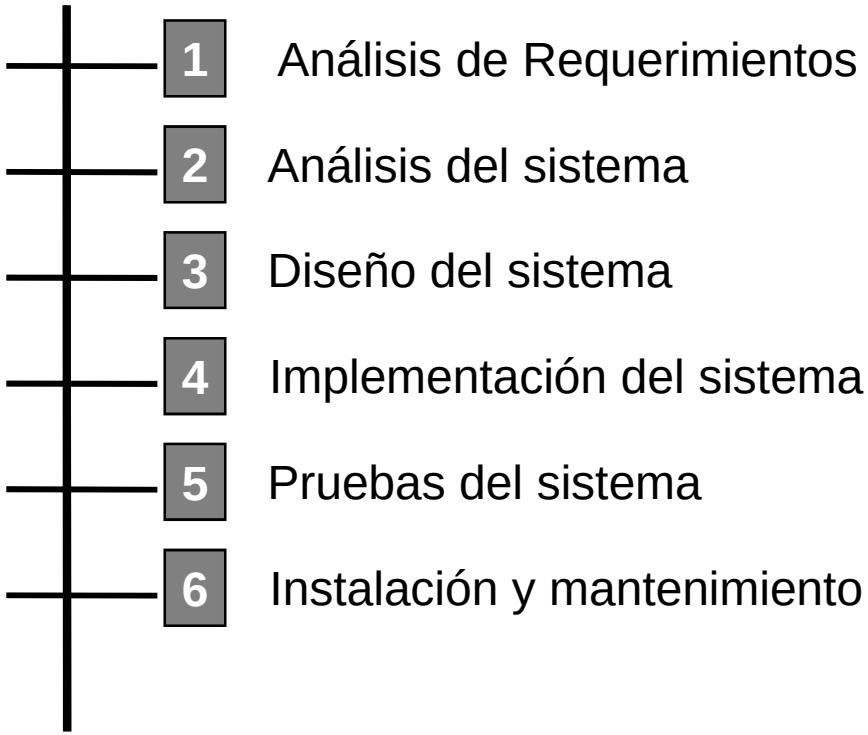
1. Diseño (arquitectura, estructura de datos, …)
2. Codificación e implementación
3. Pruebas

- CAMBIO
  - Que se puede producir en el sistema

Adaptación o mejora del sistema

Ciclo de vida: Fases

- Fases del ciclo de vida:



Objetivo: Conocer las:
1. Características
2. Detalles
3. Limitaciones del problema a resolver.

Ciclo de vida: Fases

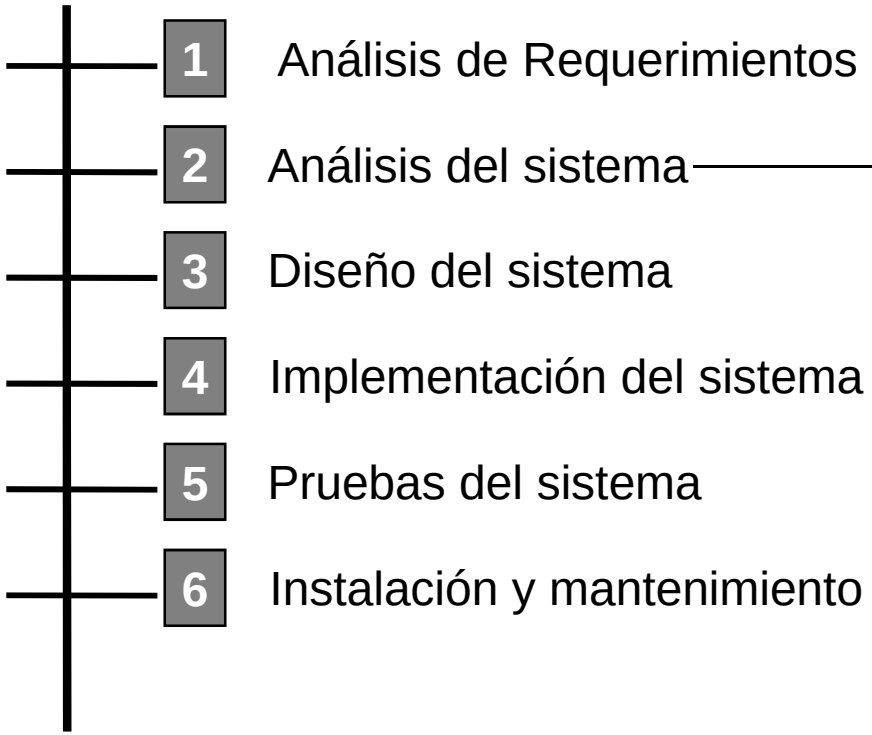
- Análisis
  - Posibles técnicas para obtener y representar requisitos
    - Entrevista
    - Brainstorming
    - Prototipos. Versión inicial del sistema. Luego se tira.
    - Casos de uso. Técnica definida en UML
  - Se especifican dos tipos de requisitos
    - Funcionales
      - Describen con detalle las distintas funcionalidades del sistema
    - No funcionales
      - Trata características del sistema que no afectan a la funcionalidad:
        - Sistema operativo, tiempos de respuesta, Fiabilidad..
    - Pueden soparse requisitos de interfaz de usuario
  - Actividad
    - Discutir ejemplos de requisitos funcionales y no funcionales (garceta pág. 13)
  - Para representar los requisitos funcionales se utilizan diferentes técnicas
    - Diagramas de Flujo de Datos (DFD's)
    - Diagramas de transición de estados (DTE's)
    - Diagrama Entidad / Relación DER
    - Diccionario de datos

Ciclo de vida: Fases

- Análisis: Documento Especificaciones de Requisitos Software
  - Estructura según estándar 830 [IEEE ,1998]
    1. Introducción
       1. Propósito
       2. Ámbito del sistema
       3. Definiciones, Acrónimos y Abreviaturas
       4. Referencias
       5. Visión general del documento
    2. Descripción General
       1. Perspectiva del producto
       2. Funciones del producto
       3. Características de los usuarios
       4. Restricciones
       5. Suposiciones y dependencias
       6. Requisitos Futuros
    3. Requisitos Específicos
       1. Interfaces Externas
       2. Funciones
       3. Requisitos de Rendimiento
       4. Restricciones de Diseño
       5. Atributos del sistema
       6. Otros requisitos
    4. Apéndices
  - Consulta documento para más detalle
    - http://www.fdi.ucm.es/profesor/gmendez/docs/is0809/ieee830.pdf

Ciclo de vida: Fases

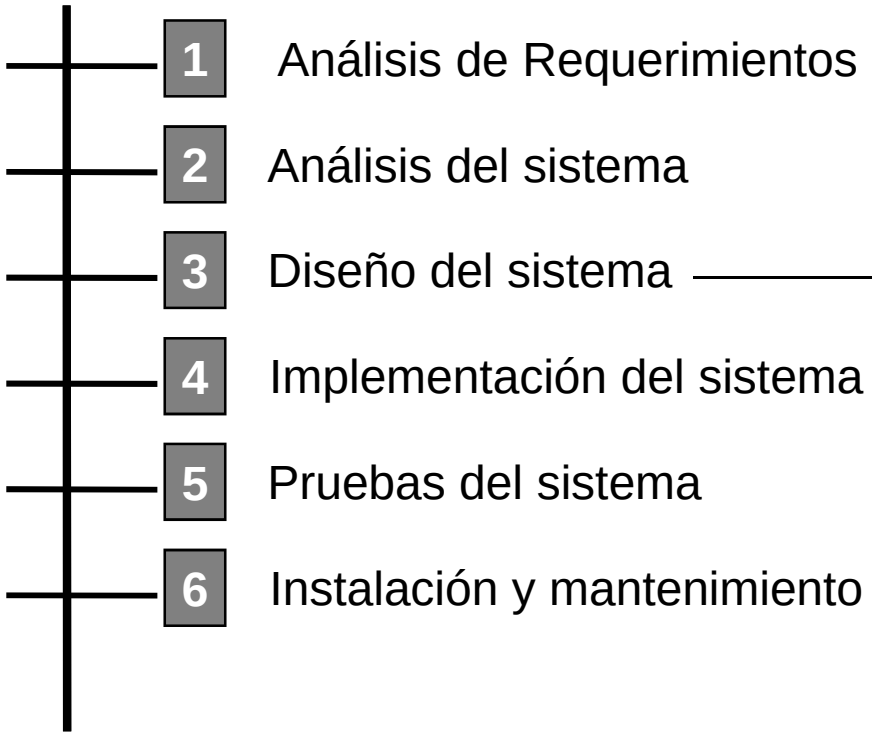
- Fases del ciclo de vida:



Objetivo: Descomponer el problema inicial en problemas más pequeños.
- Se identifican las funciones que el sistema debe realizar

Ciclo de vida: Fases

- Fases del ciclo de vida:



Objetivo: Cómo resolver los subproblemas identificados en la fase de análisis.

CICLO DE VIDA: FASES

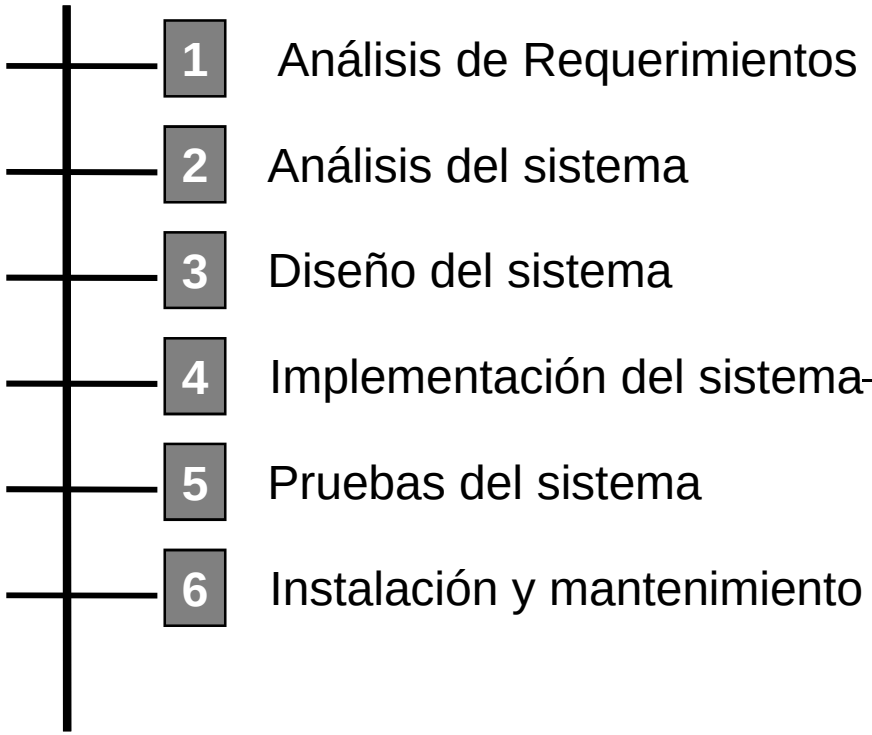
- Diseño. Principalmente hay dos tipos (garceta pág. 15)
  - **Diseño estructurado**
    - Se basa en el flujo de datos a través del sistema
    - El diseño estructurado clásico produce un modelo de diseño con 4 elementos
      - Diseño de datos
      - Diseño arquitectónico partiendo de los dfd's
      - Diseño de interfaz de usuario
      - Diseño a nivel de componentes. El resultado es una especificación con el suficiente detalle para que la pueda implementar un programador
        - Diagramas de flujo, pseudocódigo, tablas de decisión….

CICLO DE VIDA: FASES

- **Diseño orientado a objetos**
  - El sistema se entiendo como un conjunto de objetos que tienen propiedades y comportamientos, además de eventos que activan operaciones que modifican el estado de los objetos, que a su vez, interactúan con otros objetos
  - Define 4 capas de diseño [Pressman]
    - Subsistema: se centra en el diseño de los subsistemas que implementan las funciones principales del sistema
    - Clases y objetos : Especifica a arquitectura de objetos global y la jerarquía de clases requeridas para implementar un sistema
    - Mensajes: indica como se realizan la colaboración entre los objetos
    - Responsabilidades: Operaciones y atributos que caracterizan cada clase
  - Para el análisis y diseño orientado a objetos → UML

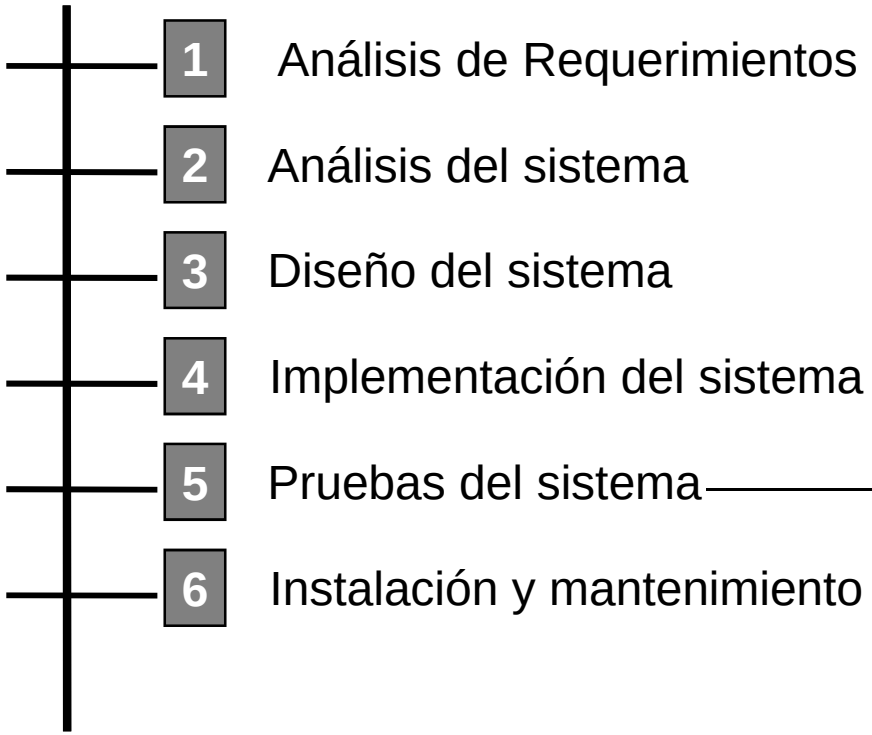
Ciclo de vida: Fases

- Fases del ciclo de vida:



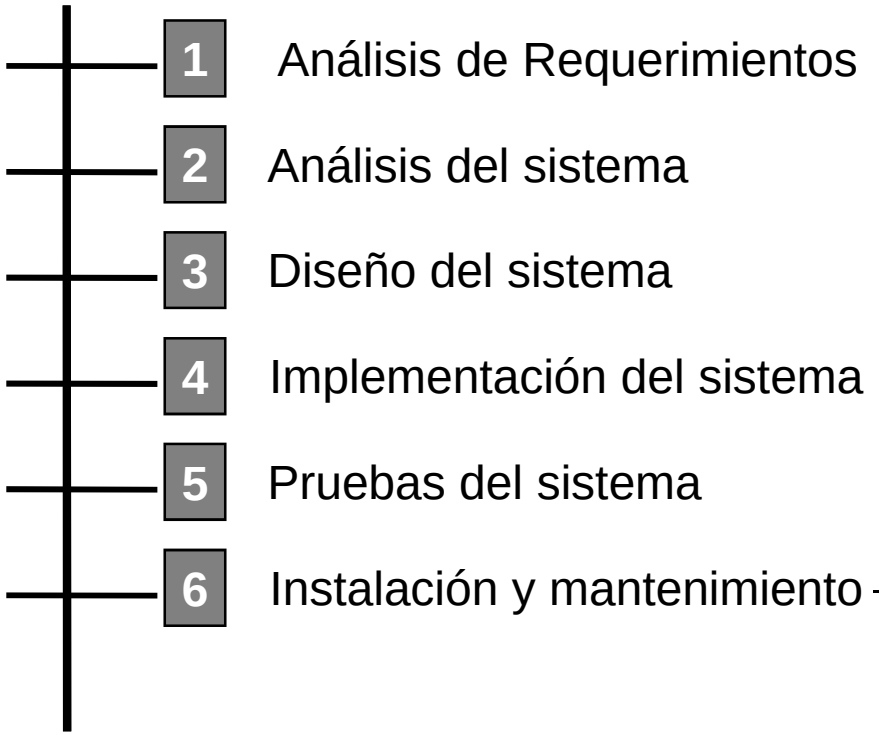
Objetivo: Codificar el sistema. Pasar los subproblemas a un lenguaje de programación

Ciclo de vida: Fases



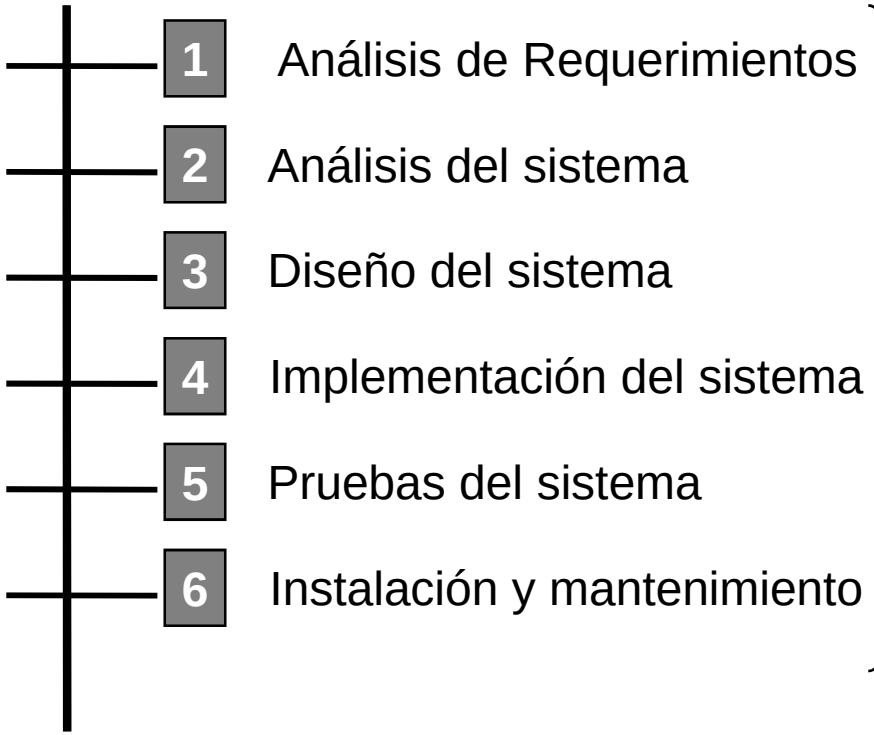
Objetivo: Comprobar el correcto funcionamiento de la aplicación.

Ciclo de vida: Fases



Objetivo: Instalar la aplicación al cliente y empezar a mantenerla(corregir errores, mejoras, ampliaciones,…)

Ciclo de vida: Fases

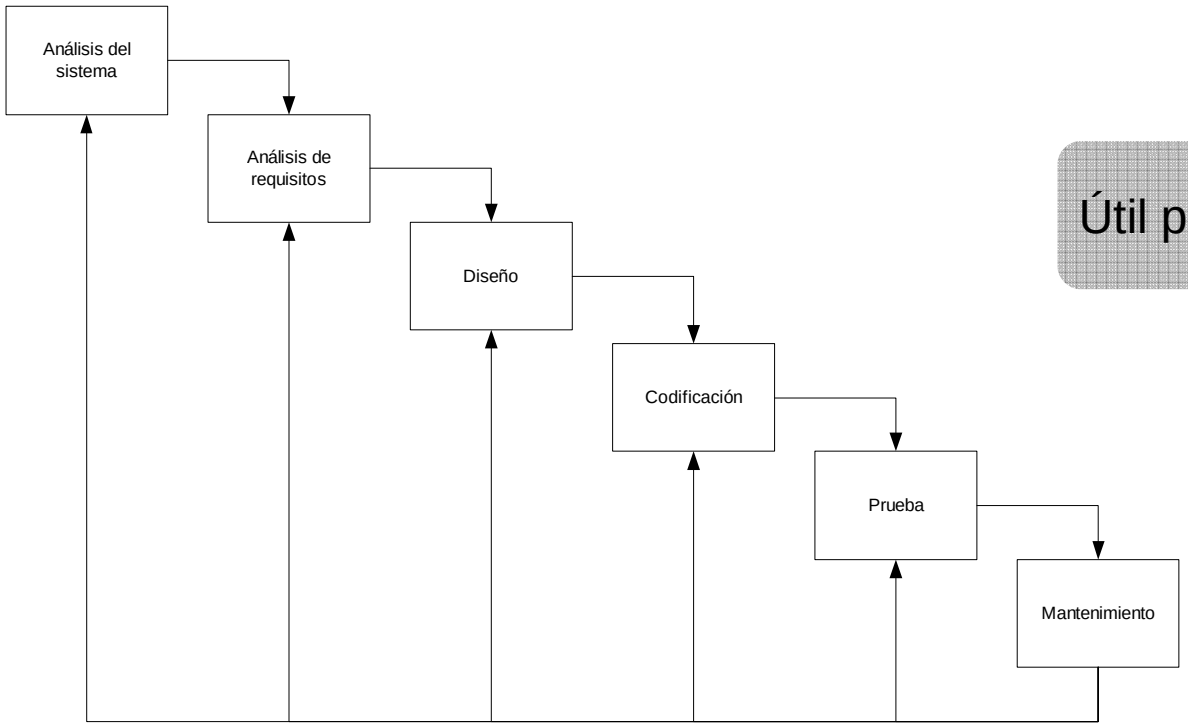


DOCUMENTAR: se realiza durante todas las fases

Modelo en Cascada

LINEAL, SECUENCIAL ó EN CASCADA

- Pasa de una fase a la siguiente de manera lineal, secuencialmente.
- Hasta que no se finaliza una fase no se pasa a la siguiente



Útil para Proyectos Cortos

Modelo en Cascada

LINEAL, SECUENCIAL ó EN CASCADA

VENTAJAS

- Fácil de comprender, planificar y seguir
- La calidad del producto resultante es alta

PROBLEMAS:

- Difícil de ajustar a proyectos reales, porque en los reales se producen realimentaciones.
- Difícil que el usuario exponga sus requisitos al principio
- No está disponible versión operativa del proyecto hasta las últimas etapas
- Se retrasa la localización y corrección de errores

SE RECOMIENDA CUANDO:

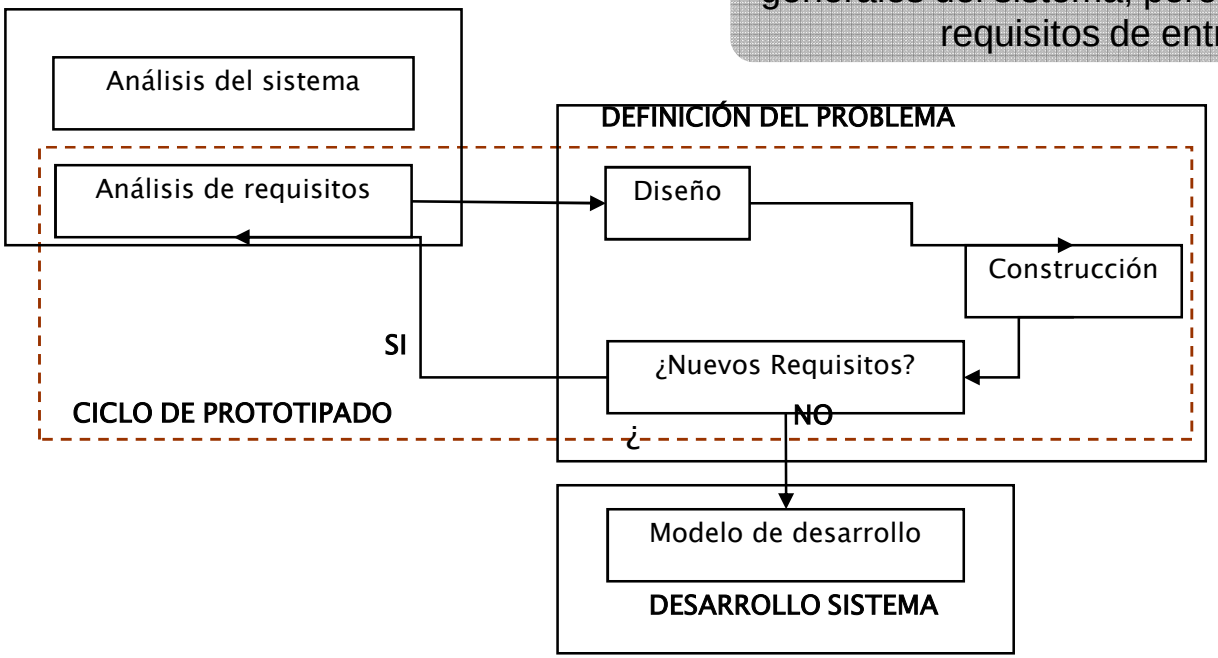
- El proyecto es similar a alguno anterior
- Los requisitos son estables y están bien definidos
- No se requiere obtener versiones intermedias

MODELO PROTOTIPOS

ORIENTADO A PROTOTIPOS

- Prototipo es una primera propuesta que se irá refinando para la solución final.
- Se va realizando una solución provisional sin tener que llegar al final de todos los procesos

Útil cuando se tienen los objetivos generales del sistema, pero falta definir los requisitos de entrada



Modelo Prototipos

ORIENTADO A PROTOTIPOS

PROBLEMAS:

- La gestión de desarrollo del software es muy lenta. El cliente, como ve que ya funciona algo ya piensa que se está cerca del producto final, con unos "pequeños ajustes"

- Los primeros prototipos es posible que no se reutilicen y no presentan calidad o fiabilidad. Se ha hecho algo para que el cliente lo "pueda tocar" pero está "cogido con hilos".

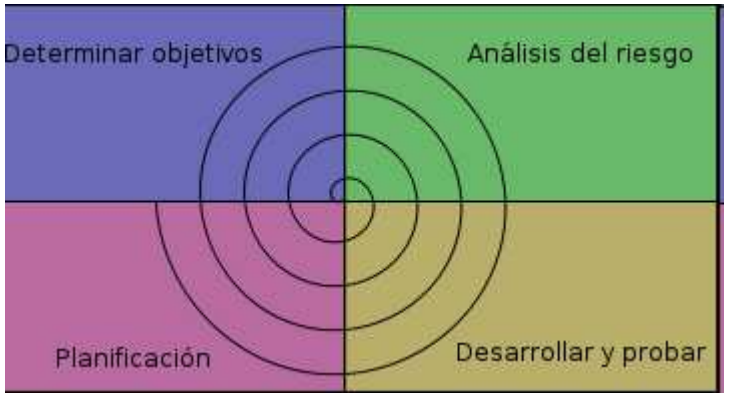
- Cuando el prototipo está preparado, es necesario comunicación desarrolladores con el cliente => produce datos para **refinar el diseño**

Modelo en Espiral

En Espiral

- Las actividades de este modelo se conforman en una espiral
- En cada bucle o iteración representa un conjunto de actividades
    - Determinar Objetivos
    - Análisis del riesgo
    - Planificación
    - Desarrollar y probar

Útil cuando se tienen los objetivos generales del sistema, pero falta definir los requisitos de entrada



Modelo en Espiral

En ESPIRAL

VENTAJAS

- No requiere una definición completa de los requisitos para empezar
- Análisis de riesgo en todas lasa etapas
- Reduce los riesgos del proyecto
- Incorpora objetivos de calidad

PROBLEMAS:

- Genera mucho tiempo en el desarrollo del sistema
- Modelo costoso a medida que la espiral pasa por sucesivas iteraciones
- Es difícil la identificación y evaluación de riesgos
- Planificar un proyecto con esta metodología es a menudo imposible, debido a la incertidumbre en el número de iteraciones que serán necesarias.

SE RECOMIENDA CUANDO:

- Proyectos de gran tamaño y que necesitan constantes cambios
- Proyectos donde sea importante el factor riesgo

METODOLOGÍAS DE DESARROLLO

- Se Indica:
  - Cómo se divide un proyecto en etapas
  - Qué tareas se realizan en cada etapa
  - Qué salidas se producen y cuando se deben producir
  - Que restricciones se aplican
  - Qué herramientas hay que utilizar
  - Cómo se gestiona y controla un proyecto

METODOLOGÍAS DE DESARROLLO

- **Características deseables**
  - Existencia de reglas predefinidas
  - Cobertura total del ciclo de desarrollo
  - Verificaciones intermedias
  - Enlaces con procesos de gestión
  - Comunicación efectiva
  - Utilización en proyectos diferentes
  - Fácil formación
  - ¡¡¡ Herramientas !!!
  - ¡¡¡ Soporte de la reutilización de software !!!

- **OBJETIVO:**
  - Crear mejores aplicaciones
  - Realizar un mejor proceso de desarrollo
  - Aplicar un proceso estándar

METODOLOGÍAS DE DESARROLLO

SSADM

- Desarrollada en el Reino Unido
- Objetivo: ser usada por la Administración Pública

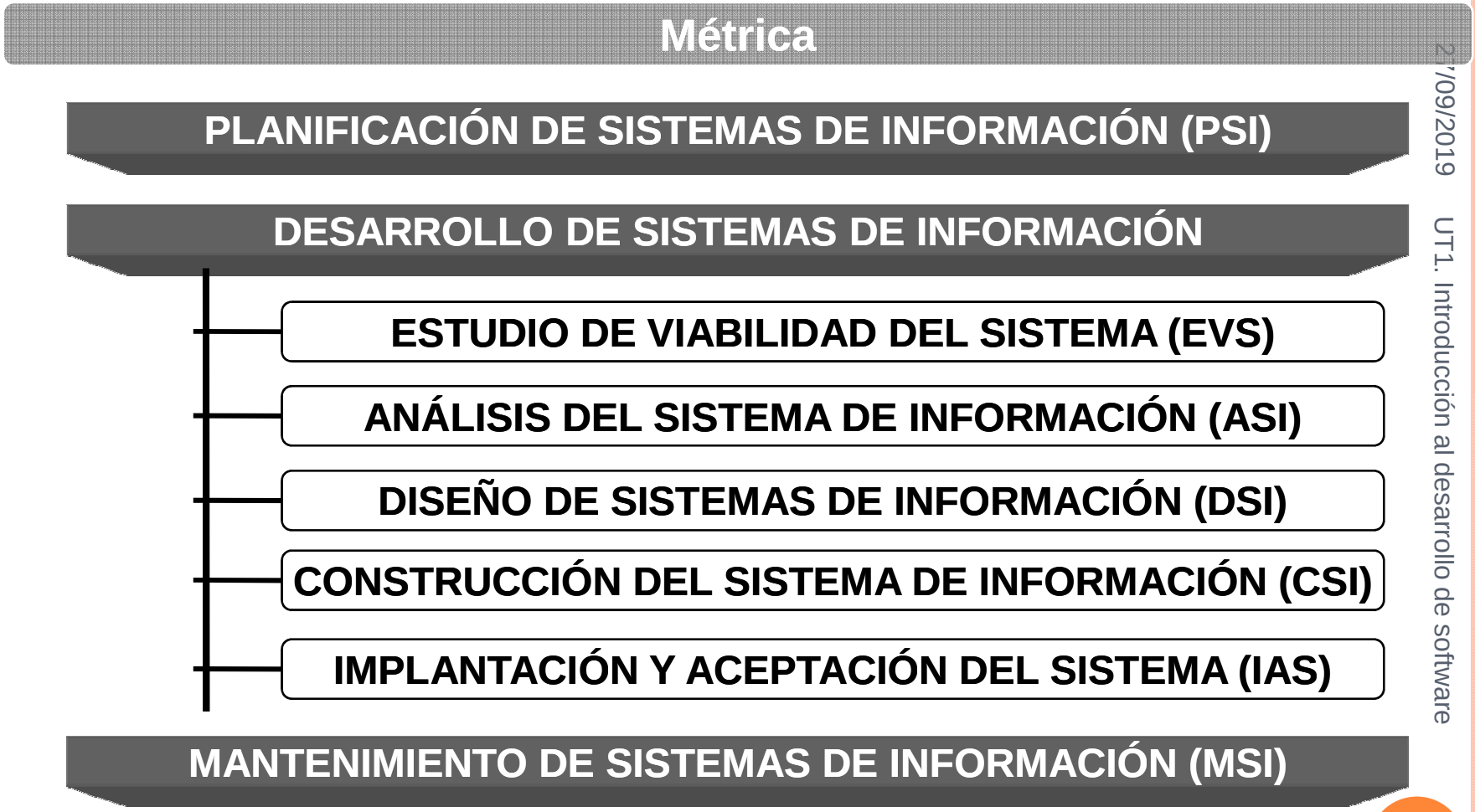
Merise

- Desarrollada en Francia

Métrica

- Desarrollada en ESPAÑA
- Objetivo: ser usada en el desarrollo de productos Software por parte de la administración.
- Hoy en día se usa en todo tipo de proyectos

METODOLOGÍAS DE DESARROLLO



METODOLOGÍAS DE DESARROLLO

METODOLOGÍAS DE DESARROLLO

Métrica

PLANIFICACIÓN DE SISTEMAS DE INFORMACIÓN (PSI)

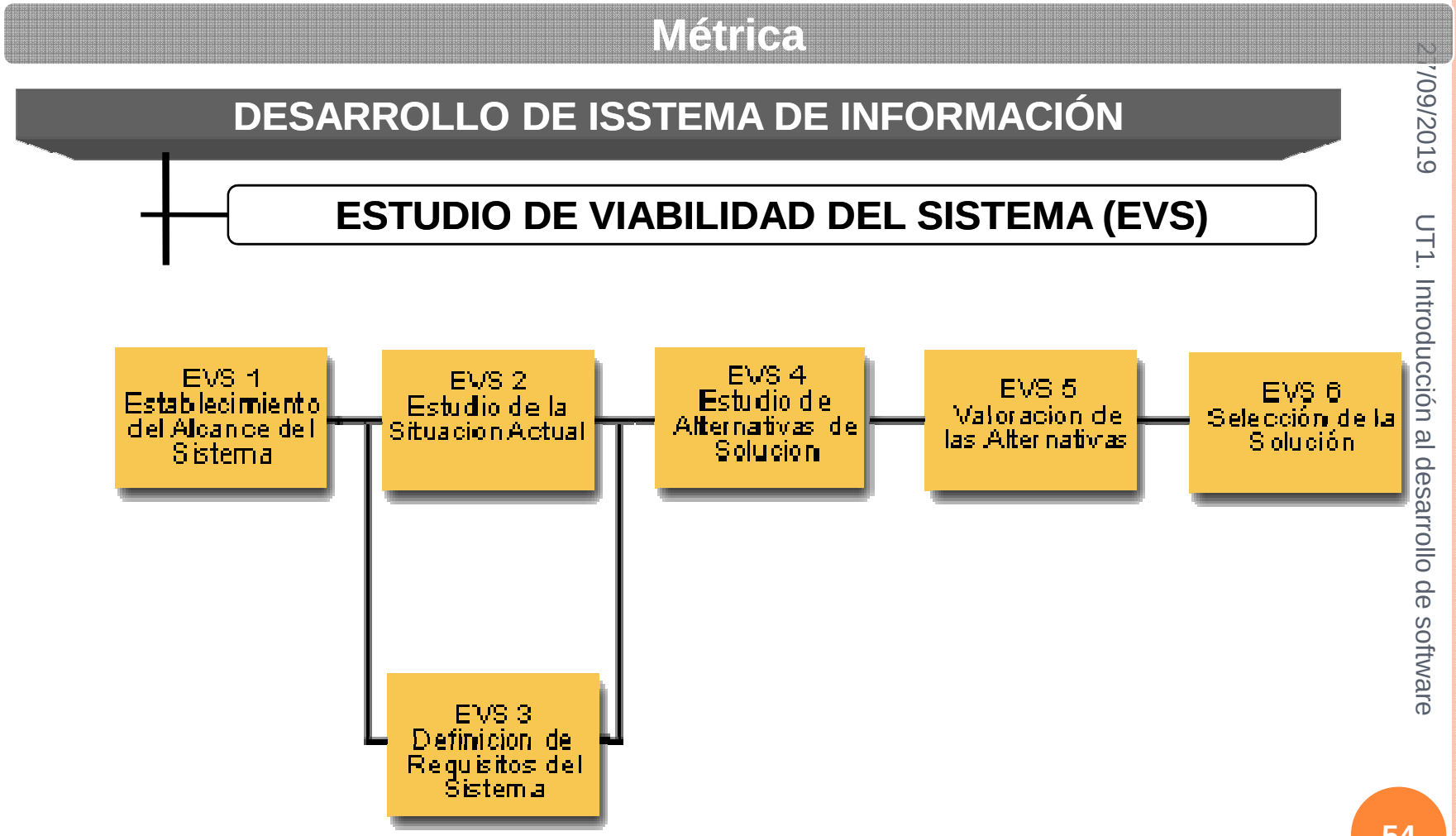
Objetivo:

1. Obtener información acerca de los sistemas de información actualmente implantados en la empresa
2. Evaluar las necesidades de nuevas funcionalidades en dichos sistemas
3. Establecer proyectos que permitirán su mejora

Salida (documentación):

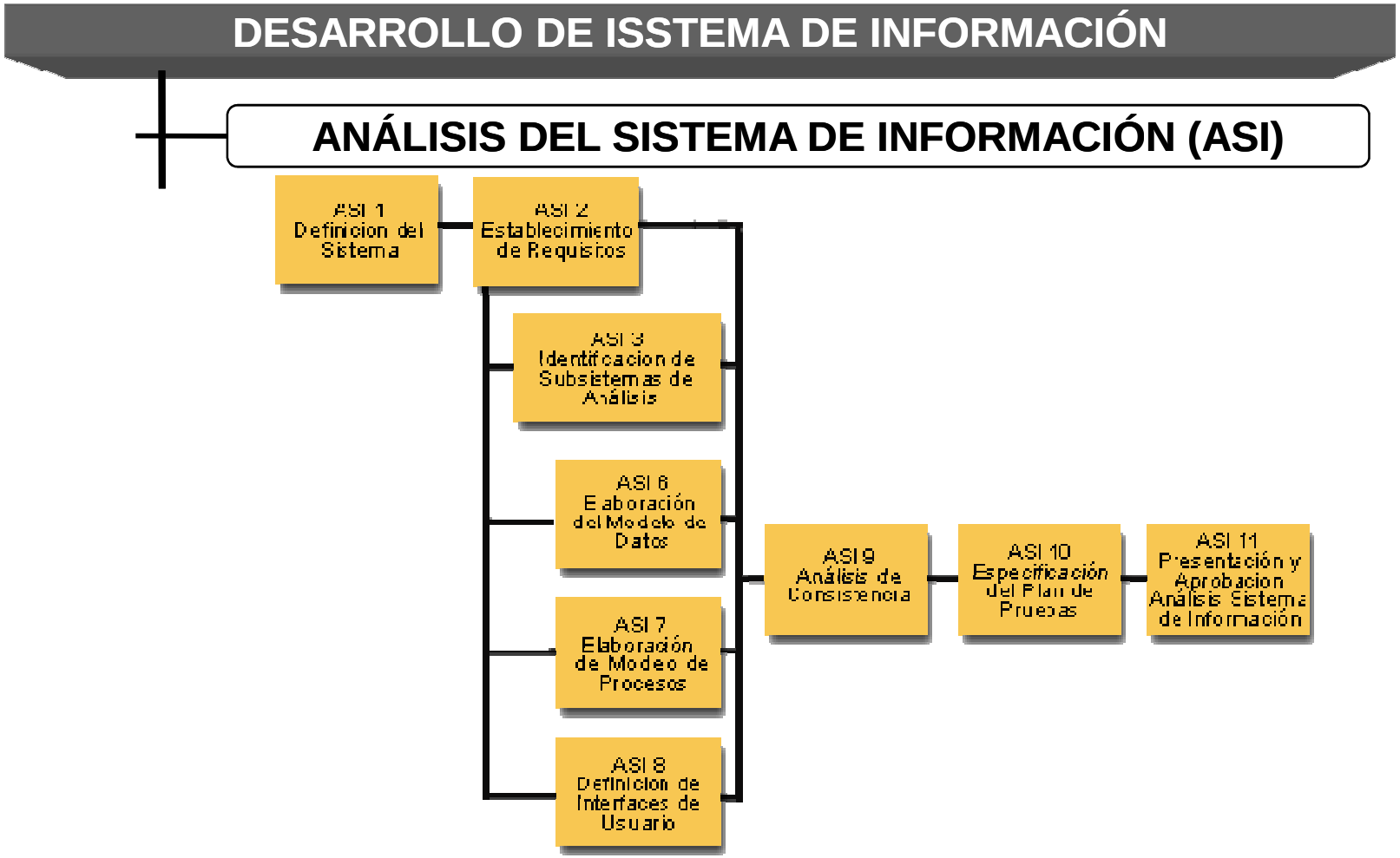
1. Descripción de la situación actual de los SI de la empresa
2. Propuesta de proyectos para mejorar los SI
3. Evaluación de los recursos para desarrollar los proyectos
4. Definición de un plan de seguimiento y control

METODOLOGÍAS DE DESARROLLO

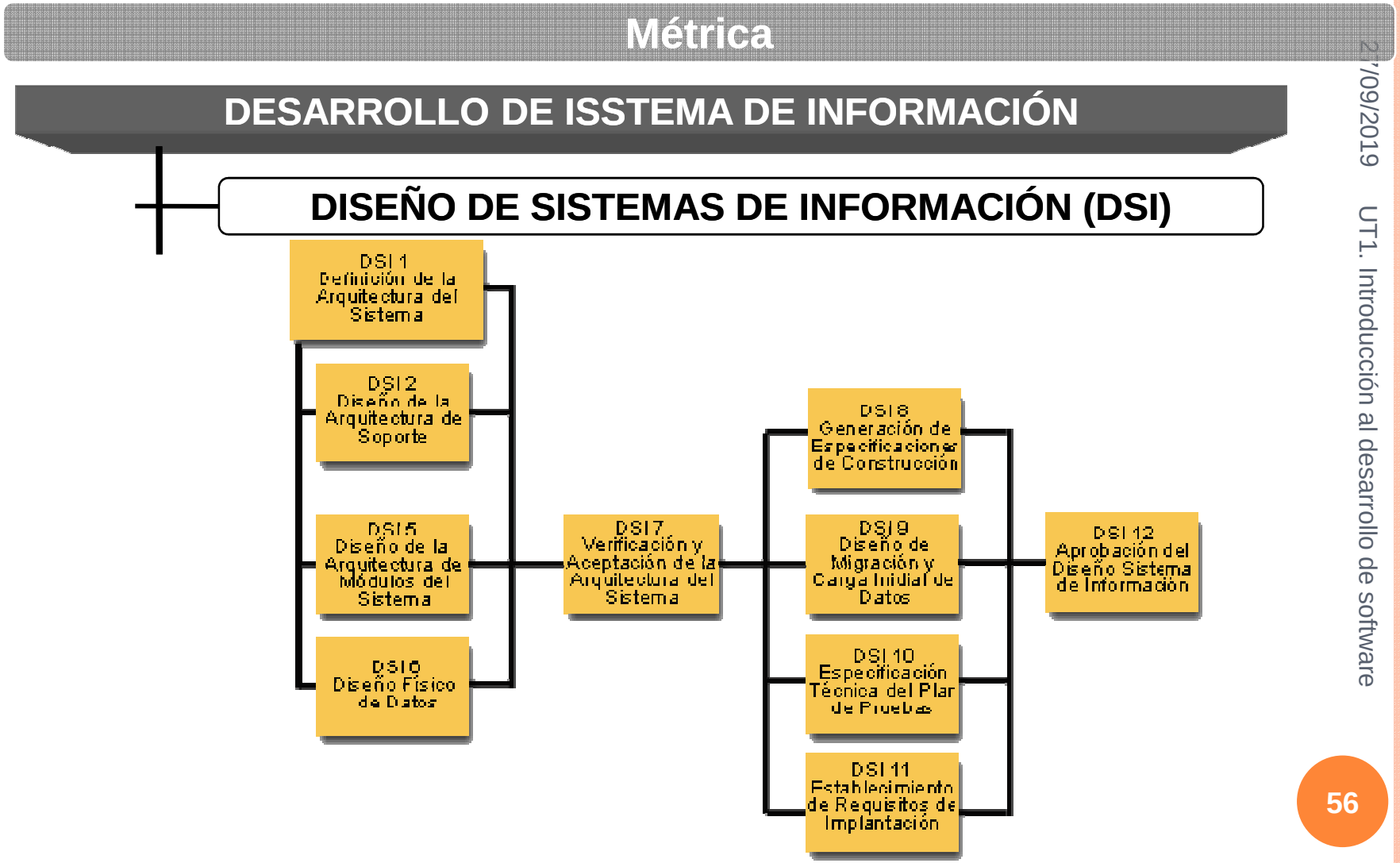


METODOLOGÍAS DE DESARROLLO

Métrica

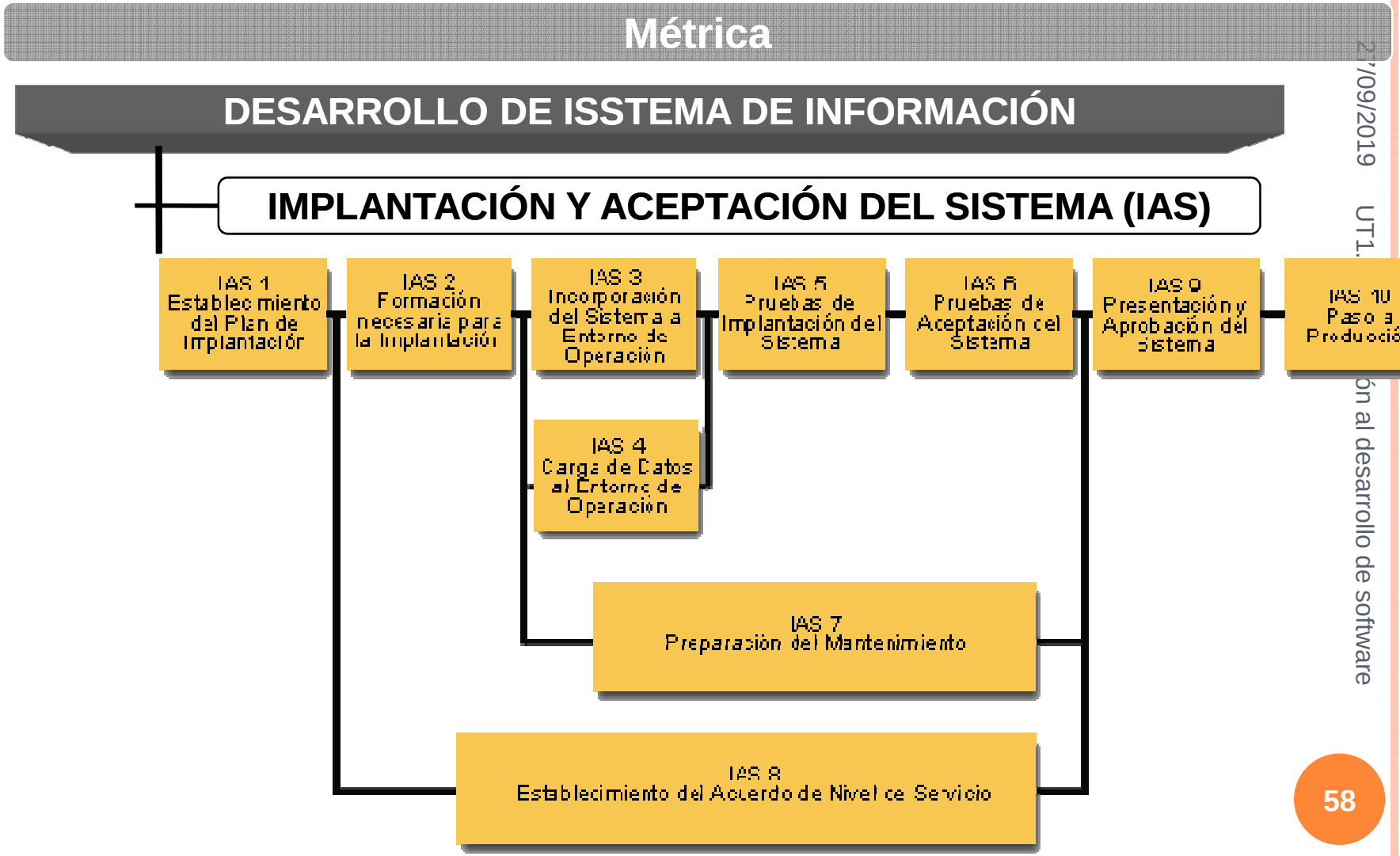


METODOLOGÍAS DE DESARROLLO



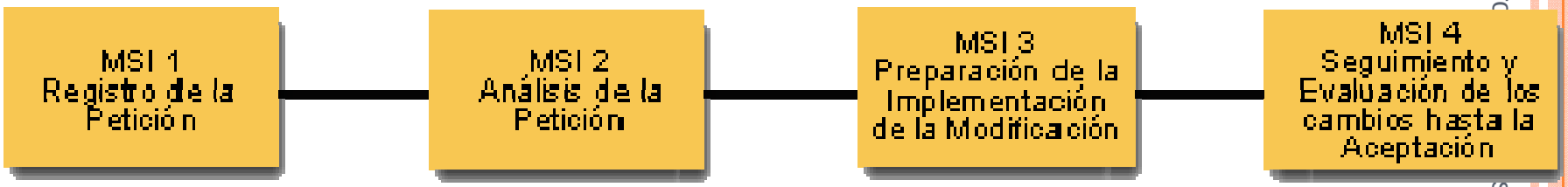
METODOLOGÍAS DE DESARROLLO

METODOLOGÍAS DE DESARROLLO



METODOLOGÍAS DE DESARROLLO

MANTENIMIENTO DE SISTEMAS DE INFORMACIÓN (MSI)



HERRAMIENTAS CASE

- Las **herramientas CASE** (***C**omputer **A**ided **S**oftware **E**ngineering*) son diversas aplicaciones informáticas destinadas a aumentar la productividad y calidad en el desarrollo de software.
- Objetivos de la tecnología CASE
  - INCREMENTAR
    - Productividad del equipo.
    - Calidad del Software.
    - Reusabilidad del software.
  - REDUCIR
    - Costes de desarrollo y mantenimiento.
  - AUTOMATIZAR
    - Gestión del proyecto.
    - Desarrollo del software.
    - mantenimiento del software (Incluyendo la automatización y estandarización de la documentación y de su mantenimiento)

HERRAMIENTAS CASE

- Clasificación según la fase del ciclo de vida que abordan:
  - CASE frontales (*front-end*) o ***Upper CASE***: Herramientas de apoyo a las primeras fases:
    - Análisis, diseño.
  - CASE dorsales *(back-end) o* ***Lower CASE***: Herramientas de apoyo a las últimas fases:
    - Implementación (generación de código).
    - Pruebas (caja blanca y caja negra).
    - Mantenimiento.
  - ICASE (*Integrated-CASE*)
    - Contienen elementos de *Upper* y *Lower* CASE: comtemplan todo el ciclo de desarrollo.
  - *Reverse Engineering* (Ingeniería inversa)

CASE VS IDE VS RAD

- **IDES:**
  - El acrónimo **IDE** significa en inglés **Integrated Development Environment** o lo que viene a ser lo mismo Entorno integrado de desarrollo. Un **IDE** consiste en un **editor de código**, un **compilador**, un **depurador** y un **constructor de interfaz gráfica GUI**. Los **IDEs** pueden ser **aplicaciones solas o pueden formar parte de otras aplicaciones**. El lenguaje Visual Basic por ejemplo puede ser usado dentro de las aplicaciones de Microsoft Office, lo que hace posible escribir sentencias de Basic en forma de macros para Word. Es posible que un **IDE** pueda **funcionar para varios lenguajes** como es el caso de Eclipse y NetBeans que permite plugins para lenguajes adicionales o Visual Studio .NET que permite programar en C++, C# y VB.Net.

- RAD
  - El acrónimo **RAD** significa en inglés **Rapid Application Development** o lo que viene a ser lo mismo desarrollo rápido de aplicaciones. La **herramientas RAD** son aquellas que **proporcionan herramientas que automatizan algunas tareas del desarrollo de la aplicación haciéndola transparente para el programador**, por ejemplo, **WebSphere** dispone de un **generador visual de JSPs** también permite **generar Servicios Web de manera gráfica** o **editar los archivos XML de configuración de manera gráfica**, para que no se tengan que introducir los tags XML directamente.