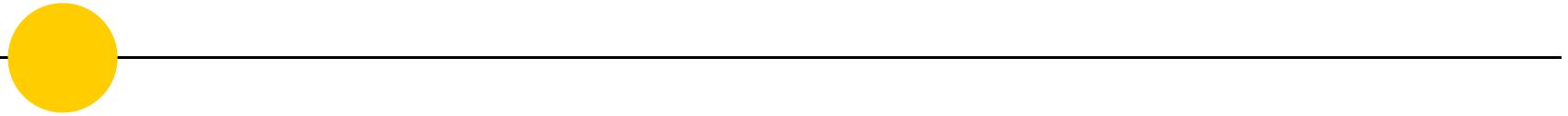
UD1 – Desaroolo del software



Índice

1 Software y Hardware

1. Software y Hardware

1.1 Definición HW y SW

El ordenador se compone de dos partes: Software y Hardware.

- **Software**: Instrucciones que el ordenador necesita para funcionar. Parte lógica del ordenador.

  Sistemas operativos → Gestionan el ordenador de manera eficiente. Permiten la comunicación con el usuario.

  Aplicaciones → Programas informáticos que tratan de resolver necesidades concretas del usuario.

- **Hardware**: Componentes físicos del ordenador. Todo lo que se puede ver y tocar.

  Dentro de la torre → placa base, memoria RAM, microprocesador, disco duro….

  Fuera de la torre → teclado, ratón, torre, impresora….

¿Porqué Windows es el S.O. que más se usa?
¿Porqué cada vez se usa más Linux?

1. Software y Hardware

1.2 Software

Hace referencia a:

- Los programas y datos almacenados en un ordenador.
- Los programas encargados de dar instrucciones para realizar las tareas con el hardware o para comunicarse con otro software.
- Los datos necesarios para la ejecución de los programas.

Clasificación

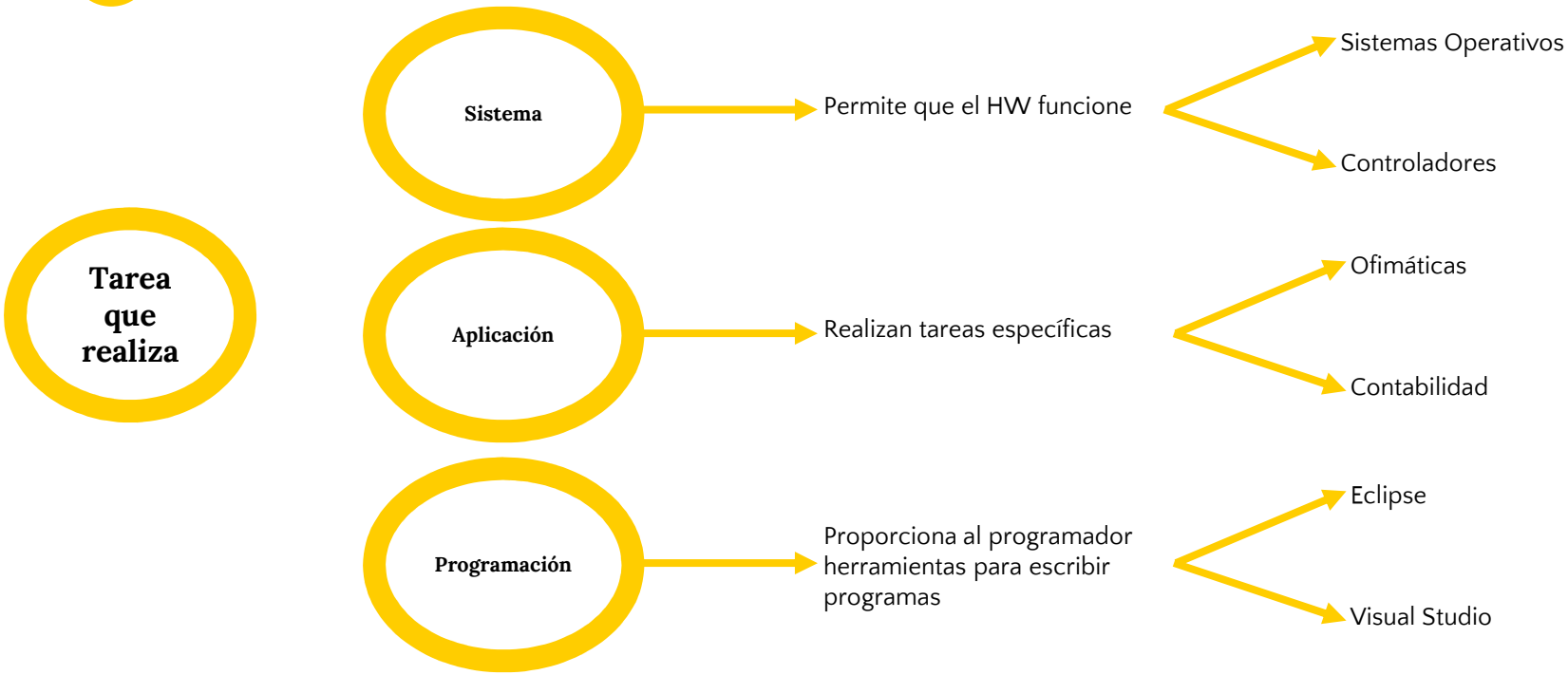
Trabajo que realiza
- De sistema
- De aplicación
- De programación

Método que realiza
- Shareware
- Freeware
- Adware
- De uso específico
- Multimedia

Licencia
- Software libre
- Software propietario
- Software de dominio público

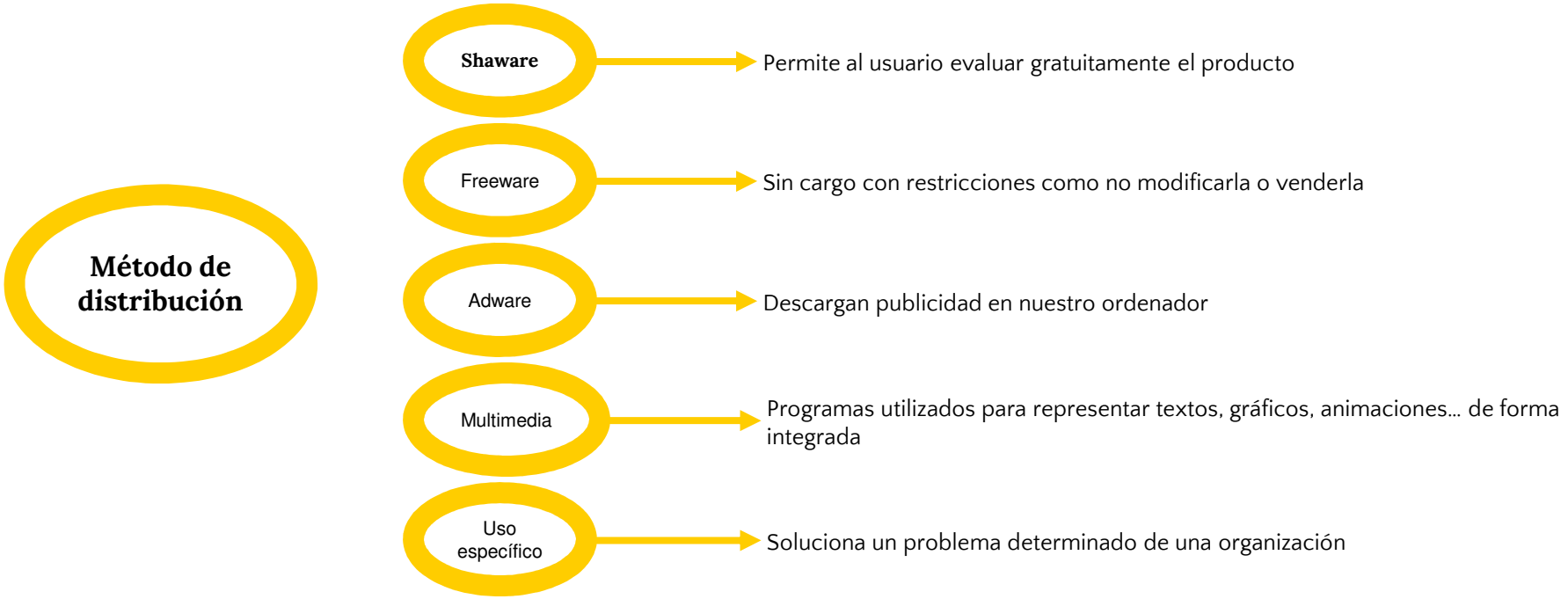
1. Software y Hardware

1.2 Software - Clasificación



1. Software y Hardware

1.2 Software - Clasificación



1. Software y Hardware

1.2 Software - Clasificación

Licencias

Contrato entre el desarrollador del software y el usuario.

Libre

El autor cede una serie de libertades básicas al usuario, permitiendo

1. Libertad de utilizar el programa con cualquier fin en cuantos ordenadores desee.
2. Libertad de estudiar y adaptar el código a sus necesidades.
3. Libertad de distribuir copias.
4. Libertad de mejorar el programa y hacer públicas y distribuir al público las modificaciones.

Propietario

Se distribuye en formato binario para evitar el acceso al código fuente

Público

Carece de licencia por desconocimiento del autor

1. Software y Hardware

1.3 Concepto de programa

Un **programa informático** es un conjunto de instrucciones escritas en lenguaje de programación que con unos datos de entrada resuelven un problema o parte de un problema.

Un programa se escribe en un lenguaje de programación, pero para que el ordenador sea capaz de ejecutarlo es necesario traducirlo a **lenguaje máquina**. **Compilador**

Este programa, escrito en lenguaje máquina, se carga en la **memoria principal** del procesador para que este **ejecute** todas las **instrucciones** que lo componen.

2 Lenguajes de programación

2. Lenguajes de programación

2.1 Definición

Conjunto de caracteres, reglas para la combinación de dichos caracteres y reglas que definen los efectos que se producirán cuando se ejecutan en un ordenador.

Consta de:

- **Alfabeto o vocabulario (léxico):** conjunto de símbolos predeterminados.
- **Sintaxis:** reglas que indican cómo realizar las construcciones con los símbolos del lenguaje.
- **Semántica:** reglas que determinan el significado de cualquier construcción del lenguaje.

2. Lenguajes de programación

2.2 Tipos

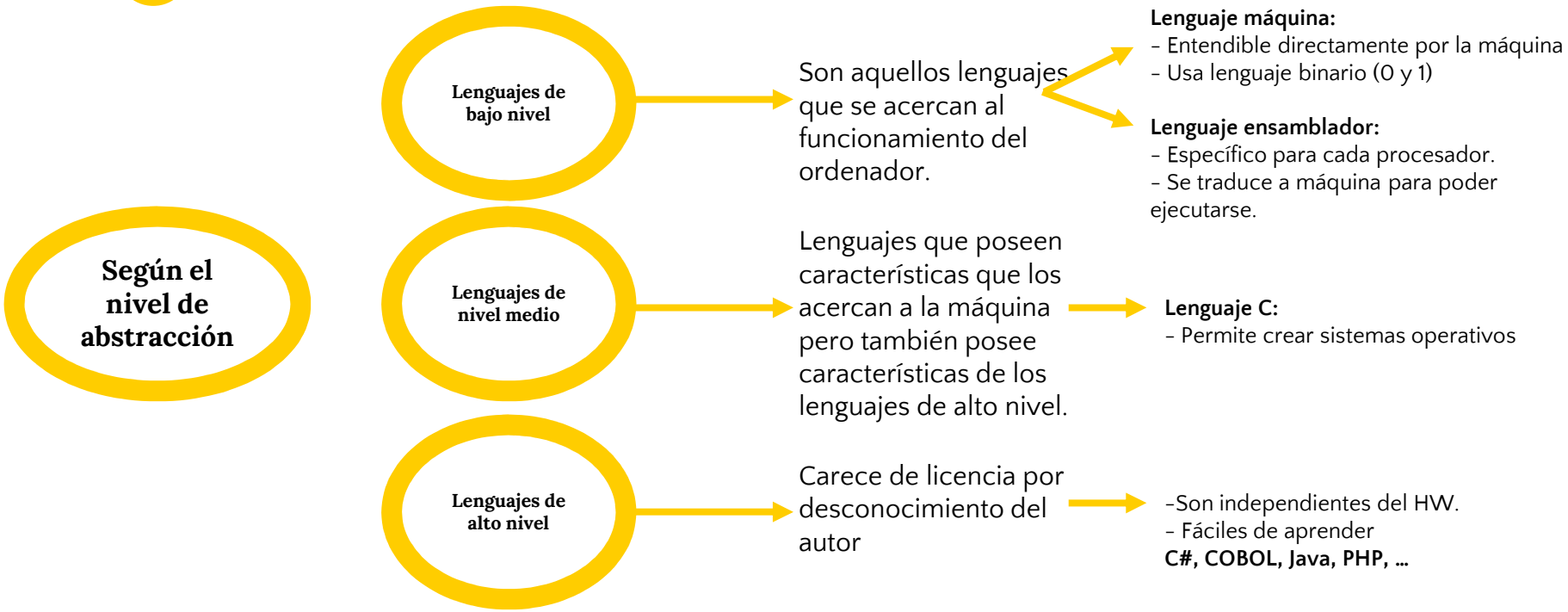
Según el nivel de abstracción
- Lenguaje de bajo nivel.
- Lenguaje de nivel medio.
- Lenguaje de alto nivel.

Según la forma de ejecución
- Lenguajes compilados.
- Lenguajes interpretados.

Según el paradigma de programación
- Lenguajes imperativos.
- Lenguajes funcionales.
- Lenguajes lógicos.
- Lenguajes estructurados.
- Lenguajes orientados a objetos.

2. Lenguajes de programación

2.2 Tipos



2. Lenguajes de programación

2.2 Tipos

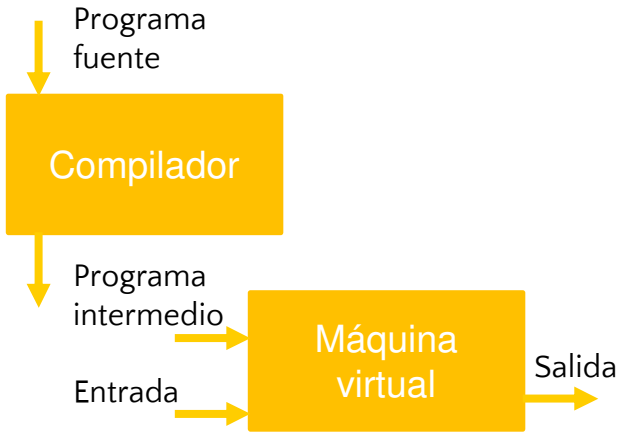
Lengauejs compilados

Lenguaje resultante de la compilación de un programa fuente.

Según la forma de ejecución

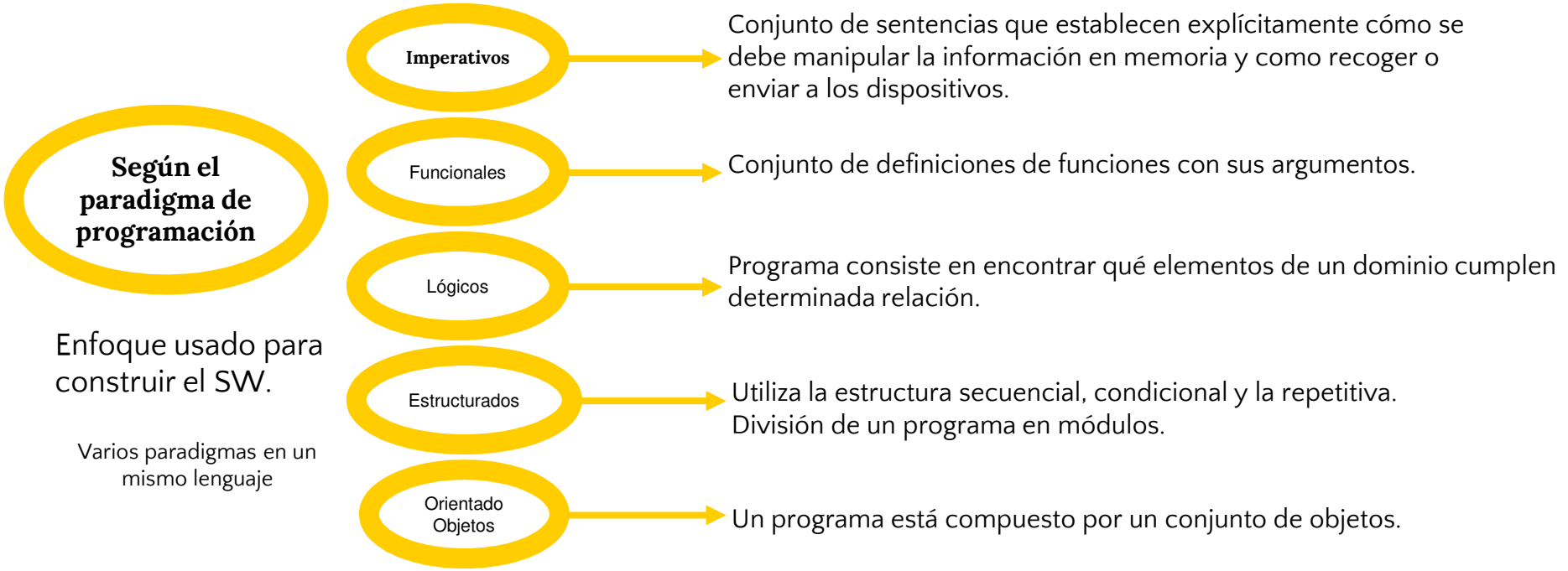
Lenguajes interpretados

Lenguaje compilado que luego es interpretado.



2. Lenguajes de programación

2.2 Tipos



Enfoque usado para construir el SW.

Varios paradigmas en un mismo lenguaje

2. Lenguajes de programación

2.3 Herramientas utilizadas

Para desarrollar un programa informático se utilizan entornos de programación.

IDE (Integrated Development Environment)

Es un programa informático formado por un conjunto de herramientas de programación que ayudan en las tareas de creación, modificación, compilación, implementación y depuración del software.

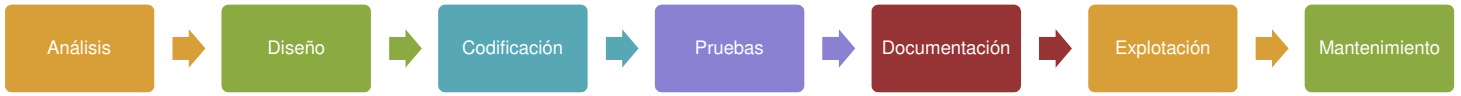
Tareas que realiza:

- Crear, editar y modificar el código fuente.
- Compilar, montar y ejecutar el programa.
- Examinar el código fuente.
- Ejecutar en modo depuración.
- Realización de pruebas.
- Generar documentación.
- ...

3

Fases del desarrollo de una aplicación

Entendemos por desarrollo del software todo el proceso que ocurre desde que se concibe una idea hasta que un programa está implementado en el ordenador y funcionando.



3. Fases del desarrollo de una aplicación

3.1 Análisis

Es la fase más importante ya que en ella se definen los requisitos que debe cumplir la aplicación y de ella depende el éxito o fracaso de esta. **Analista.**

Se especifican y analizan los requisitos **funcionales** y **no funcionales** del sistema

- **Funcionales**: Se especifíca el comportamiento de la aplicación ante cualquier situación, incluyendo los casos de error o excepciones.

- **No funcionales**: Se especifíca todos aquelllos comportamientos en los que no participa el usuario: tiempos de respuesta del programa, legislación aplicable...

¿Crees que debemos tener completamente cerrada una etapa para pasar a la siguiente?

3. Fases del desarrollo de una aplicación

3.1 Análisis

Para ayudar a la **toma de requisitos** se usan los siguientes medios:

- **Entrevistas.** Hablar con el cliente y obtener todos los datos necesarios.
- **Desarrollo conjunto de aplicaciones (JAD).** Dinámicas de grupo, cada participante posee un rol.
- **Planificación de requisitos (JRP).** Subconjunto JAD, está dirigido a la alta dirección.
- **Brainstorming.** Reuniones de grupo para generar ideas desde distintos tipos de vista.
- **Prototipos.** Versión inicial del sistema que clarifica algunos puntos y conceptos.
- **Casos de uso.** Técnica que define UML, especificación de los distintos escenarios que describen como será el SW.

3. Fases del desarrollo de una aplicación

3.1 Análisis

Para **representar los requisitos** se utilizan las siguientes técnicas:

- **Diagrama de transición de estados.** Representa como responde el sistema como consecuencia de sucesos externos.
- **Diagrama entidad/Relación.** Usado para representar los datos y como se relacionan entre ellos.
- **Diccionario de datos.** Descripción de los datos utilizados por el sistema
- **Diagrama de flujo.** Representación gráfica de un algoritmo, de forma que se describen las diferentes tareas y la secuencia en la que se ejecutan para alcanzar una solución.Debe cumplir las siguientes reglas:
  - Debe ser independiente del lenguaje de programación elegido para la codificación.
  - Diseño normalizado.
  - Debe ser intuitivo.
  - No debe ser rígido.

Al finalizar esta fase ya sabemos ***¿Qué hay que hacer?***

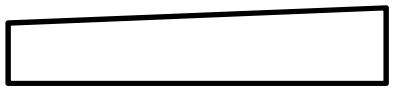
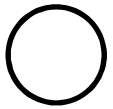
3. Fases del desarrollo de una aplicación

3.1 Análisis – Diagrama de flujo

Terminador	Marca el inicio, final o una parada necesaria a realizar en la ejecución del programa	
Entrada/Salida	Se utiliza para mostrar la introducción de datos desde un periférico a la memoria del ordenador y la salida de resultados desde la memoria del ordenador a un periférico	
Pantalla	Representa la salida por pantalla	
Impresora	Salida de datos por impresora	

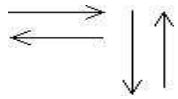
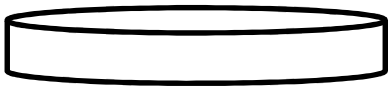
3. Fases del desarrollo de una aplicación

3.1 Análisis – Diagrama de flujo

Teclado	Símbolo de entrada por teclado	
Conector	Es utilizado para el reagrupamiento de líneas de flujo.	
Subrutina	Se utiliza para realizar una llamada a un subprograma o proceso, es decir, un módulo independiente cuyo objetivo es realizar una tarea y regresar el control de ejecución del programa al módulo principal	
Décisión de dos salidas	Indica operaciones lógicas o comparativas seleccionando en función del resultado entre dos caminos alternativos que se pueden elegir.	

3. Fases del desarrollo de una aplicación

3.1 Análisis – Diagrama de flujo

Flechas	Indicadores de la dirección del flujo de datos.	
Disco magnético	Representa una función de entrada/salida para soporte en un disco magnético.	

- Todos los símbolos deben estar conectados.
- Diseño claro de arriba-abajo e izquierda-derecha.
- Prohibido el cruce de líneas → mal diseño.
- A un símbolo de proceso pueden llegarle varias líneas pero solo puede salir una de él.
- A un símbolo de decisión pueden llegarle varias líneas pero solo puede salir una de entre las posibilidades.
- A un símbolo de inicio de proceso no le llega ninguna línea.
- A un símbolo de final de proceso pueden llegarle muchas líneas pero de él no sale ninguna.

Programas para crear diagramas de flujo Dia Diagram Editor

3. Fases del desarrollo de una aplicación

3.1 Análisis – DF- Ejemplo 1

Descripción del problema

Alquiler de videojuegos 

- *Gestión de videojuegos*: adquisiciones, retiradas, disponibilidad...

- Gestión de socios; altas, bajas, modificación de datos, sanciones, autorizados, cuentas...

- Gestión alquiler: entrega, devolución, devolución tardía, reclamación, disponibilidad...

3. Fases del desarrollo de una aplicación

3.1 Análisis – DF- Ejemplo 1

-***Gestión de videojuegos***

• Alta videojuego:

1. El empleado solicita al sistema comenzar con el proceso de alta de un nuevo videojuego.
2. El sistema solicita los datos del nuevo videojuego: título, tipo, fases, versión, creador y año de lanzamiento.
3. El empleado proporciona los datos requeridos y solicita que los almacene.
4. El sistema almacena los datos e informa de que han sido almacenados con éxito.

•Baja videojuego.

•Consulta videojuego.

Requisitos funcionales

- Copias de seguridad: el sistema deberá incorporar algún mecanismo que permita realizar copias de seguridad de los datos almacenados.
- El sistema no deberá tardar más de 5 segundos en mostrar el resultado de una búsqueda.
- el sistema deberá visualizarse correctamente en cualquier navegador: Explorer, Mozilla,...

Requisitos no funcionales

3. Fases del desarrollo de una aplicación

3.1 Análisis – DF - Ejemplo 1

Diagrama de flujo

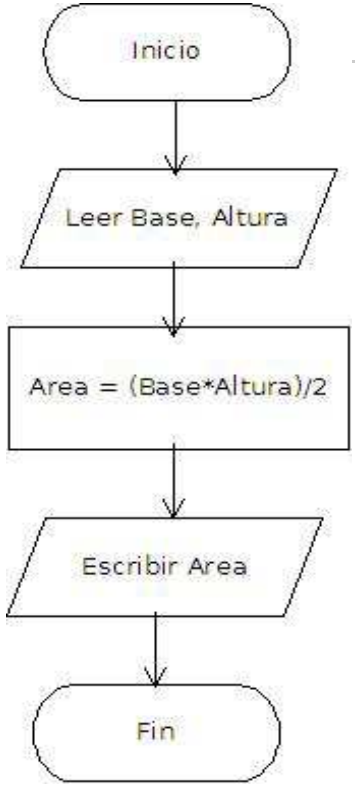
-*Gestión de videojuegos*

3. Fases del desarrollo de una aplicación

3.1 Análisis – DF – Ejemplo 2

Diseñe un algoritmo calcule el área de un triángulo.

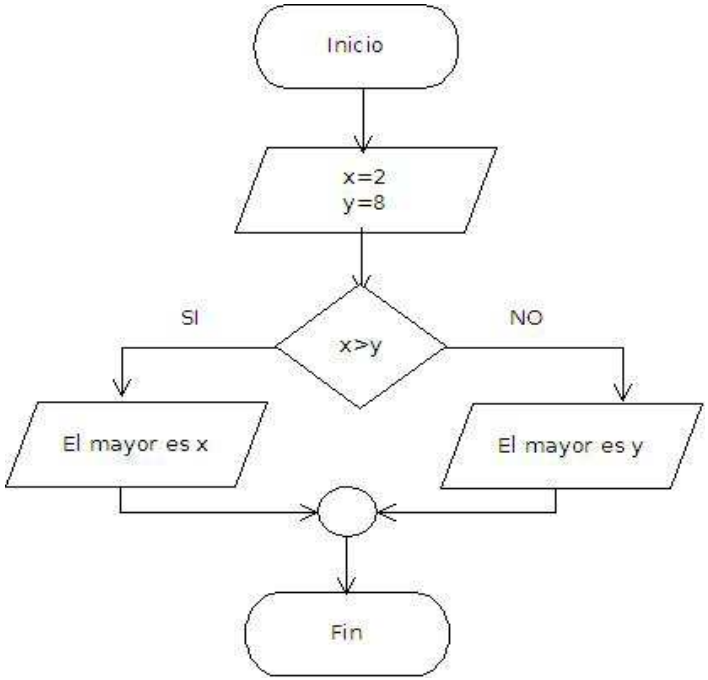
Las entradas y salidas de varios elementos se hacen con una sola representación, separándolos con comas.



3. Fases del desarrollo de una aplicación

3.1 Análisis – DF – Ejemplo 3

Diseñe un algoritmo escriba el número mayor de dos valores x e y proporcionados.



3. Fases del desarrollo de una aplicación

3.1 Análisis - pseudocódigo

La notación pseudocódigo se puede definir como un lenguaje intermedio entre el lenguaje natural o lenguaje humano y el lenguaje de programación seleccionado.

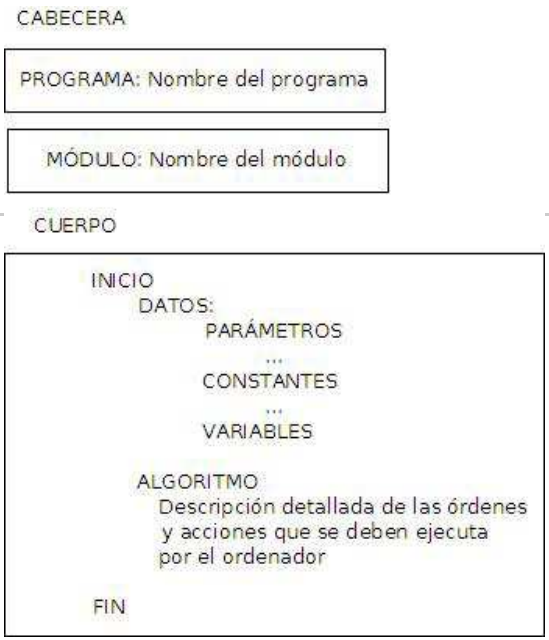
Características:

- No se puede ejecutar directamente sobre un ordenador.
- Permite el diseño y desarrollo de algoritmos independientemente del lenguaje de programación.
- Es sencillo de aprender y utilizar.
- Facilita al programador el paso del algoritmo al lenguaje de programación.
- Permite gran flexibilidad de diseño del algoritmo a la hora de expresar acciones concretas.
- Permite la realización de futuras correcciones o actualizaciones.
- Exige la utilización de sangrías en su escritura.
- Permite obtener la solución de un problema mediante aproximaciones sucesivas

3. Fases del desarrollo de una aplicación

3.1 Análisis - Pseudocódigo

Todo algoritmo representado en notación pseudocódigo deberá reflejar las siguientes partes:

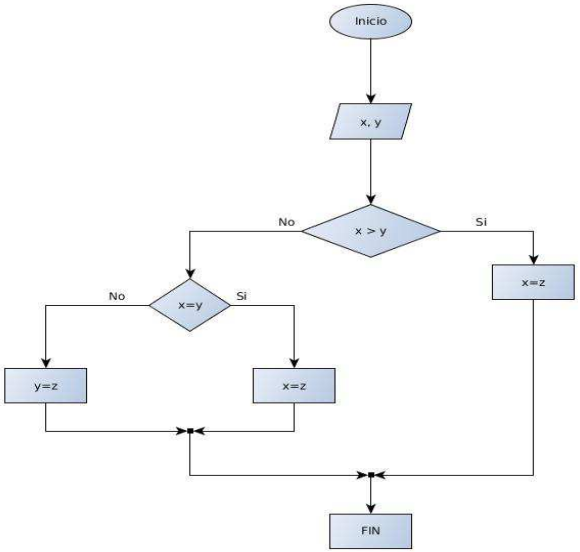


1. **Cabecera:** bloque informativo donde quedará reflejado el nombre del algoritmo y el nombre del programa al que pertenece dicho algoritmo. Se especificará primero el Módulo y después el Programa.
2. **Cuerpo:** el resto. Está compuesto por dos bloques:
   a) **Bloque de datos**, lugar donde deberán quedar descritos todos aquellos elementos de trabajo necesarios en la ejecución del programa, entendiendo como tal parámetros, constantes y variables de todo tipo.
   b) **Bloque de acciones**, donde se describen todas las acciones que el ordenador deberá realizar durante la ejecución del programa cuando éstas sean convertidas a código ejecutable.

3. Fases del desarrollo de una aplicación

Diseñe un algoritmo que lea dos valores numéricos distintos, x e y, y determine cuál es el mayor dejando el resultado en una tercera variable z.

3.1 Análisis – Pseudocódigo – Ejemplo 1

Pseudocódigo	Diagrama de flujo
Inicio Leer x, y Si x > y Entonces z=x Si no Si x=y Entonces z=x Si no z=y Fin si Fin si Fin	

3. Fases del desarrollo de una aplicación

3.1 Análisis – Pseudocódigo – Ejemplo 2

Diseñe un algoritmo que calcule el área de un rectángulo.

Pseudocódigo

Inicio
  Leer base
  Leer altura
  area = base * altura
  Escribir “El valor del área es: “ area
Fin

3. Fases del desarrollo de una aplicación

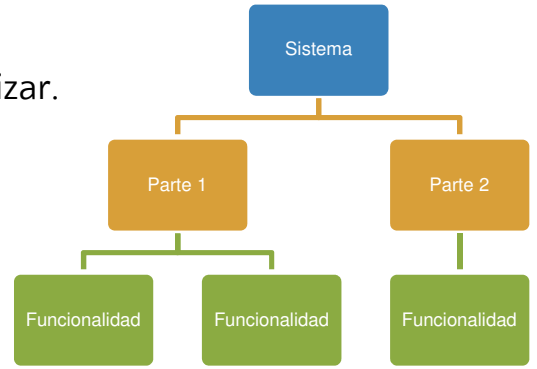
3.2 Diseño

Ya sabemos que hacer, ahora nos planteamos **¿Cómo se hace?**

Dividir el sistemas en partes y establecer las relaciones entre ellas.

Como resultado de esta fase se debe establecer:

- Entidades y relaciones que conforman la base de datos.
- Seleccionar el lenguaje de programación que se va a utilizar.
- Seleccionar el SSBD
- ...



3. Fases del desarrollo de una aplicación

3.3 Codificación

Ya sabemos como se hace, ahora **lo hacemos** → **Programación**

Esta tarea la realiza el **programador** y debe cumplir con las especificaciones que se han establecido en las fases anteriores.

Las características deseables de todo código son:

- **Modularidad:** que esté dividido en trozos más pequeños.
- **Corrección:** que realice la tarea para la que se implementó.
- **Fácil de leer:** para facilitar su desarrollo y mantenimiento.
- **Eficiencia:** buen uso de los recursos.
- **Portabilidad:** implementación en cualquier equipo.

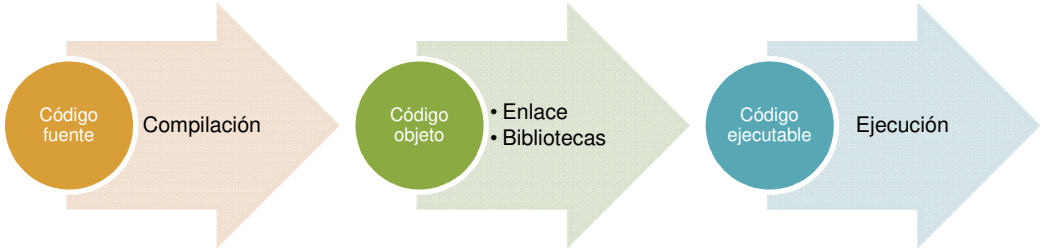
3. Fases del desarrollo de una aplicación

3.3 Codificación - Obtención del código ejecutable

Código fuente Conjunto de instrucciones, escrito por los programadores en lenguaje de alto nivel, que la computadora deberá ejecutar.

Código objeto Código que se obtiene tras compilar el código fuente. Código intermedio entre el código fuente y el ejecutable. No es legible ni por el ordenador ni por los humanos.

Código ejecutable Código que puede ejecutar directamente el procesador. Se obtiene tras enlazar el código objeto con una series de librerías y rutinas.



3. Fases del desarrollo de una aplicación

3.4 Pruebas

Realizado el software se debe probar para evitar fallos antes de lanzarlo a ejecución.

La realización de las pruebas es imprescindible para asegurar la validación y verificación del software construido.

Existen dos tipos de pruebas:

- **Unitarias:** prueban cada una de las partes del software comprobando así su funcionamiento. JUnit
- **Integración :** tras las pruebas unitarias, se comprueba el funcionamiento del sistema al completo, interrelacionando todas sus partes.

3. Fases del desarrollo de una aplicación

3.5 Documentación

Es imprescindible documentar todas las fases del proyecto para que el cliente posea toda la información además de mantener y abarcar futuras modificaciones.

Se deben elaborar los siguientes documentos:

Guía técnica
Debe recoger el diseño de la aplicación, la codificación de los programas y las pruebas realizadas.

Manual de usuario
Recoge como se debe utilizar la aplicación a nivel de usuario. Todas las funcionalidades deben quedar reflejadas.

Manual de instalación
Recoge todos los pasos que se deben seguir para puesta en marcha del software.

3. Fases del desarrollo de una aplicación

3.6 Explotación

Consiste en la instalación, puesta en marcha y funcionamiento de la aplicación en producción.

Los usuarios finales conocen la aplicación y comienzan a utilizarla.

Podemos establecer las siguientes subfases:

1. Instalación.
2. Configuración.
3. Paso a producción.
4. Uso por parte de los usuarios.

3. Fases del desarrollo de una aplicación

3.7 Mantenimiento

Una vez que se ha instalado la aplicación y esta se encuentra funcionando, el trabajo no ha terminado.

Aún quedan por solventar problemas que puedan surgir tras el uso de la aplicación o cambios que solicite el cliente de mejoras o modificaciones.

Tipos de cambio:

- **Perfectivos:** se centran en la mejor del software.
- **Evolutivos:** nuevas necesidades que le surjan al cliente.
- **Adaptativos:** modificaciones relacionadas con los cambios en el mercado.
- **Correctivos:** eliminación de errores.

El mantenimiento se define como proceso de control, mejora y optimización del Software.

4 Ciclos de vida del software

Existen varios modelos de ciclo de vida. Dependiendo de las características del proyecto se debe elegir un modelo u otro.

4. Ciclos de vida del software

4.1 Ciclo de vida cascada (clásico)

- Cada proceso comienza cuando termina el anterior.
- Los desarrollos reales presentan iteraciones.
- Es difícil obtener todos los requisitos al comienzo.
- Se tarda mucho en disponer del software.
- Es mejor que no seguir ningún ciclo de vida.
- Es más fácil de planificar, es el ciclo ideal.

Modelo en cascada con retroalimentación

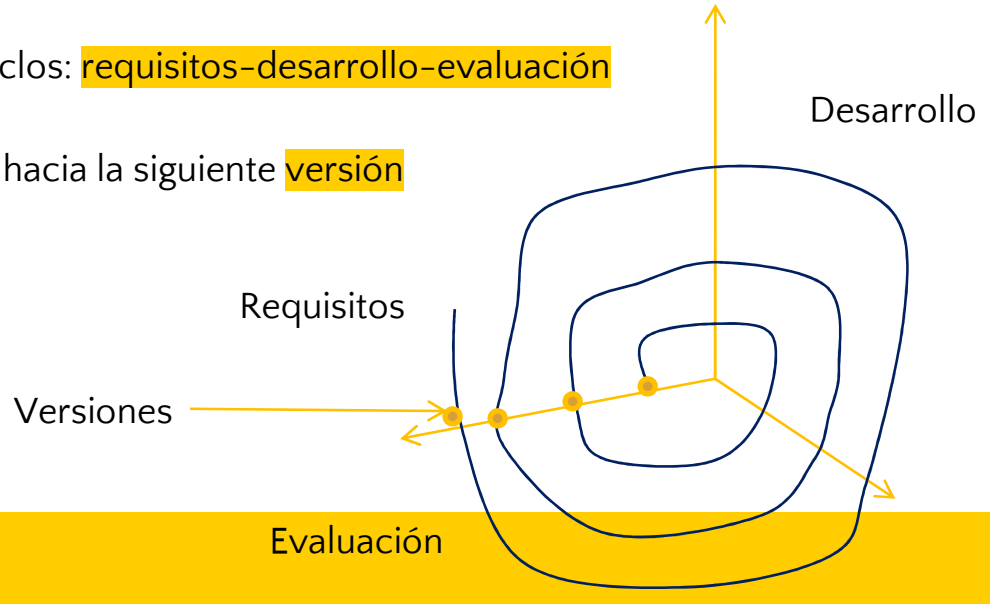
4. Ciclos de vida del software

4.2 Ciclo de vida evolutivos

- Mientras mayor sea un proyecto, menor es su probabilidad de éxito.
- Conseguir todos los requisitos al principio es prácticamente imposible.
- Las necesidades de clientes y usuarios evolucionan durante el desarrollo y nacen nuevos requisitos.

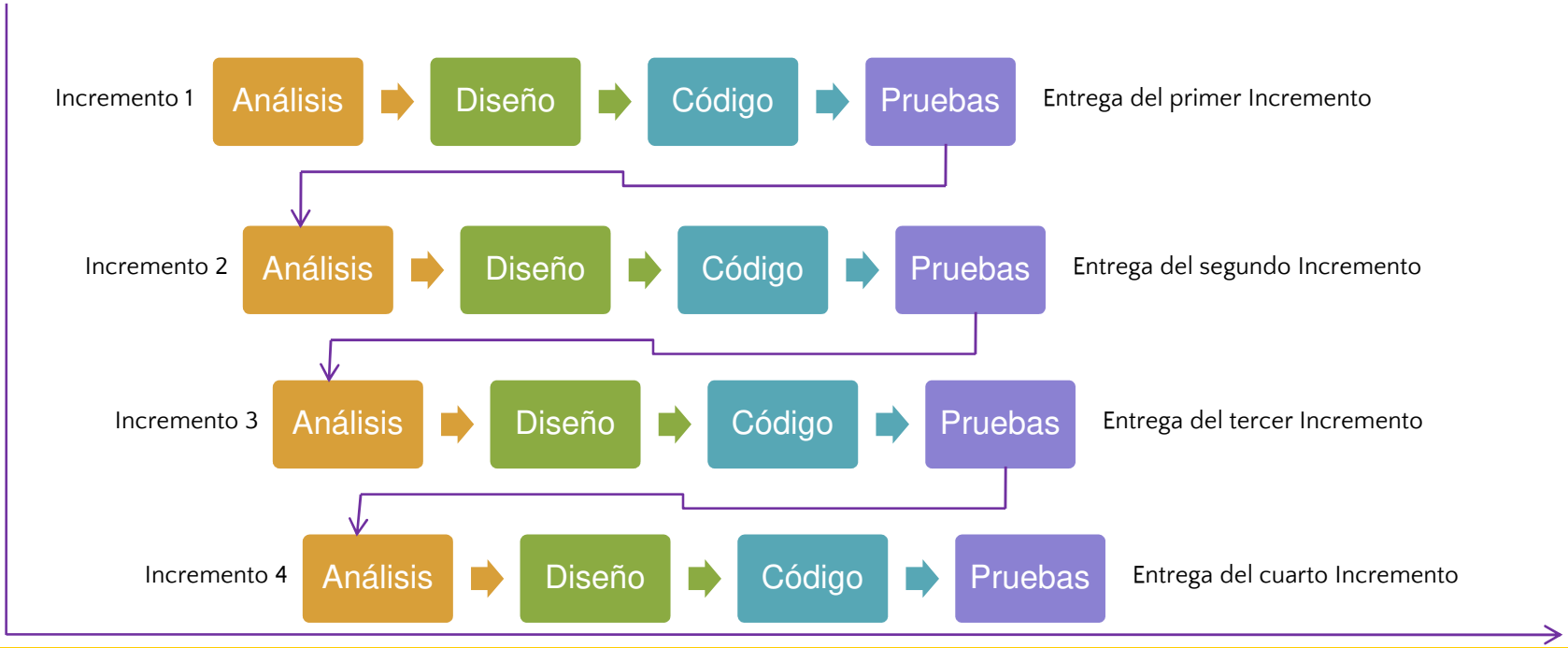
Este modelo afronta estos problemas mediante ciclos: requisitos-desarrollo-evaluación

El resultado de la evaluación permite evolucionar hacia la siguiente versión



4. Ciclos de vida del software

4.2.1 Ciclo de vida evolutivos Iterativo Incremental



4. Ciclos de vida del software

4.2.1 Ciclo de vida evolutivos. Iterativo Incremental

- Repetición de varios ciclos de vida en cascada.
- Al final de cada ciclo se entrega una versión parcial del SW incrementada con cierta funcionalidad nueva respecto a las entregas anteriores.
- Los ciclos se repiten hasta obtener el producto completo.
- Los usuarios disponen antes del SW, aunque no esté completo, por lo que pueden surgir mejoras.
- Se suele aplicar a desarrollos de gran tamaño, cuando los requisitos o el diseño no están completamente definidos y es posible que haya grandes cambios.
- Se tiene el riesgo de no acabar nunca.

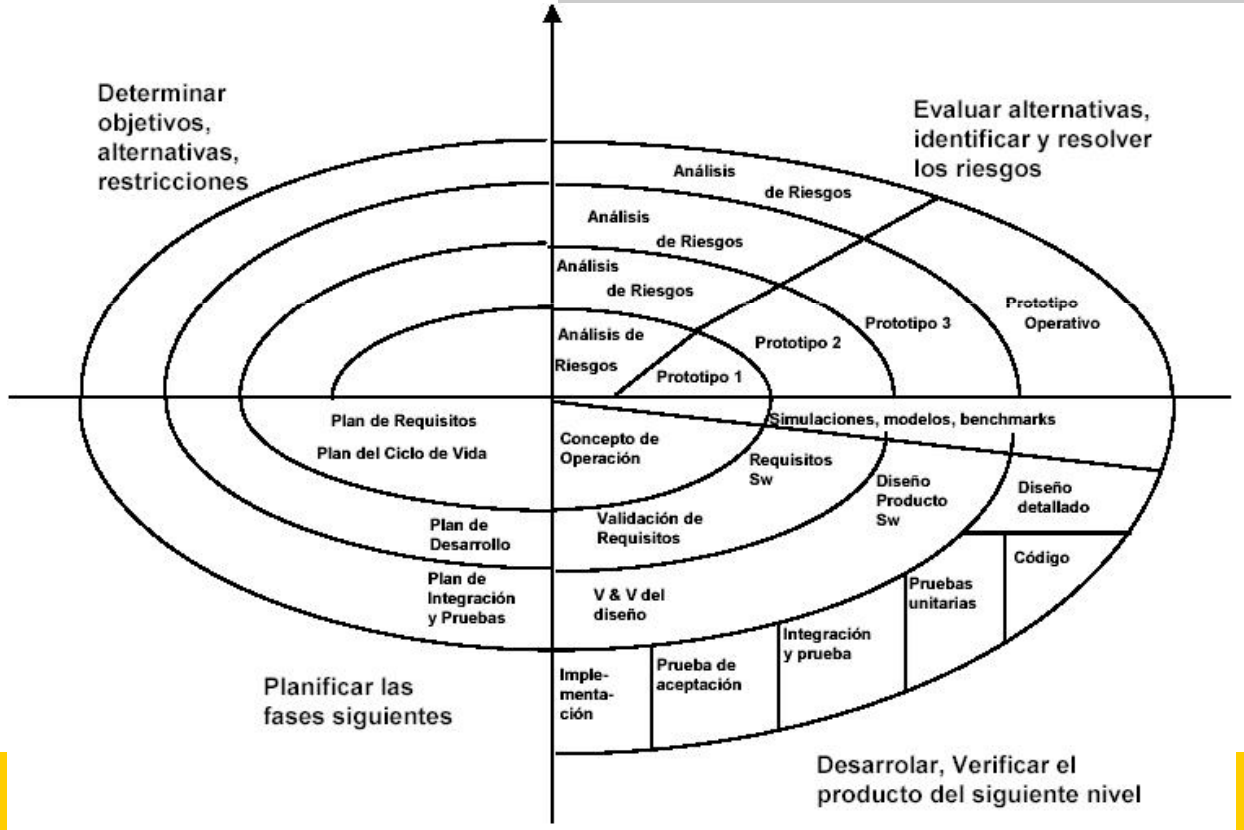
4. Ciclos de vida del software

4.2.2 Ciclo de vida evolutivos. Espiral

- Modelo en cascada unido al modelo iterativo de construcción de prototipos.
- Cada ciclo está formado por cuatro fases:
  1. **Determinar objetivos:** cada ciclo de la espiral inicia con la identificación de los objetivos y las restricciones impuestas a la aplicación.
  2. **Identificar y resolver riesgos:** evaluación de las alternativas en relación con los objetivos y limitaciones. Identificación de los riesgos (mal diseño, errores implementación, ...)
  3. **Desarrollar y probar:** desarrollo de la solución al problema y verificación de que es correcto.
  4. **Planificación:** revisar y evaluar todo lo que se ha hecho. Planificación del siguiente ciclo.
- En cada ciclo se tiene el cuenta el análisis de riesgos.
- No requiere la definición completa de los requisitos para empezar a funcionar.
- Se suele usar en proyectos de gran tamaño donde se necesitan cambios constantes.

4. Ciclos de vida del software

4.2.2 Ciclo de vida evolutivos. Espiral



5 Máquinas virtuales

5. Máquinas virtuales

5.1 Introducción

Una máquina virtual es un software que permite separar el funcionamiento del ordenador de los componentes hardware instalados.

Permite ejecutar una aplicación sobre cualquier equipo, independientemente de sus componentes físicos → portabilidad.

Actúa de puente entre la aplicación y el hardware del equipo.

Funciones:

- Hace portables a las aplicaciones.
- Reserva memoria para los objetos que se crean y libera la memoria no utilizada.
- Comunica el sistema donde se instala la aplicación con los componentes hardware que necesite del equipo.
- Vigilar el cumplimiento de las normas de seguridad.

5. Máquinas virtuales

5.2 La máquina virtual de Java

Permite ejecutar un programa compilado en Java en cualquier plataforma: Mac, Windows, Linux, Ubuntu...

JMV Java Virtual Machine

Download