Entornos de desarrollo

1 Conceptos fundamentales del desarrollo de software

IES Nervión
Miguel A. Casado Alías

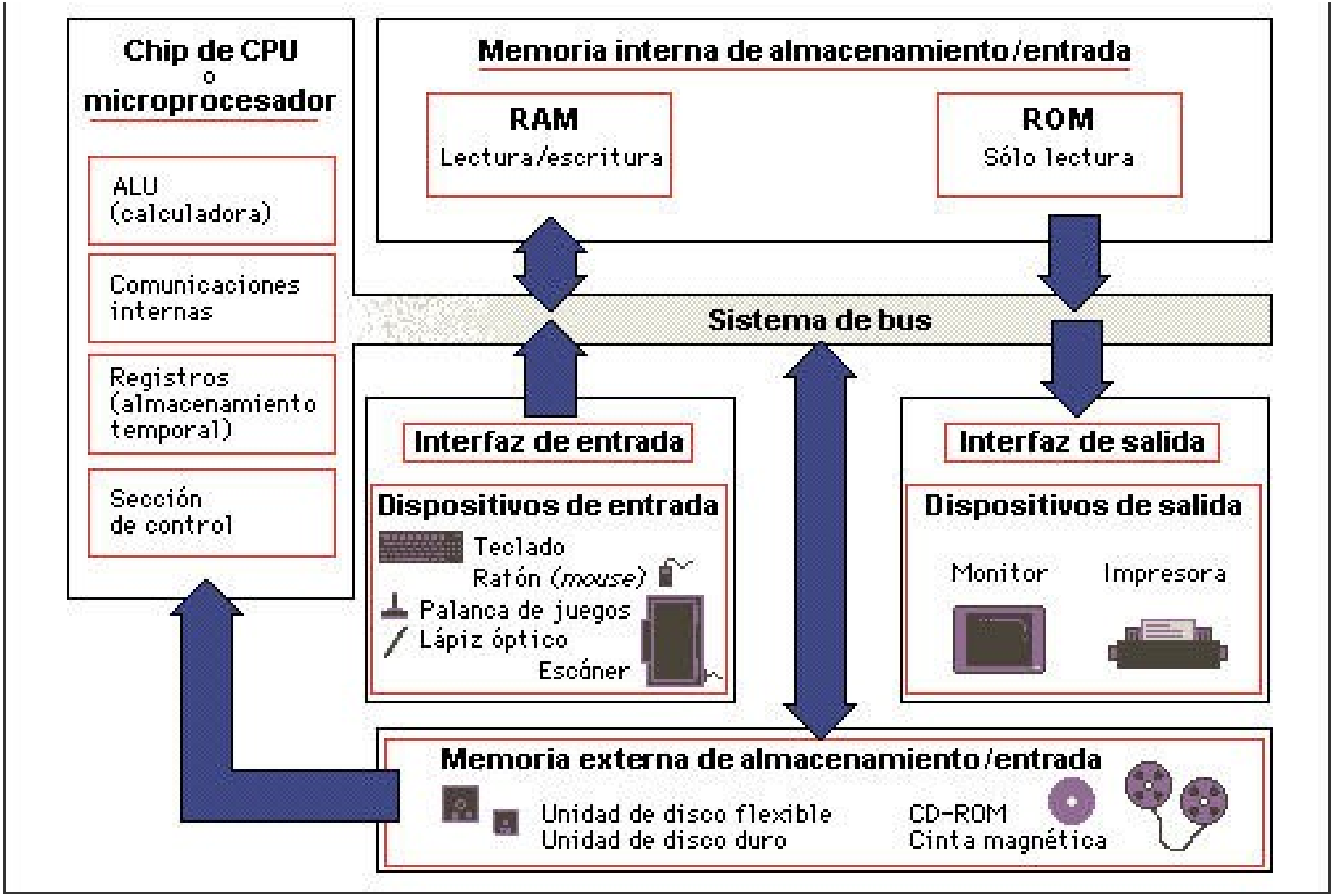
¿Qué es un ordenador?

- Es un dispositivo electrónico programable capaz de almacenar y procesar información, aunque no por sí solo.

Sistema informático

- Conjunto de recursos necesarios para procesar y almacenar información
- Elementos de un sistema informático:
  - Componente físico (hardware)
    - Ordenadores (memorias, procesadores...), discos, teclados...
  - Componente lógico (software)
    - Sistemas operativos, firmwares, aplicaciones...
  - Componente humano
    - Usuarios
    - Programadores, analistas...
- Todos los elementos se comunican y cooperan entre sí para llevar a cabo las tareas deseadas.

Hardware


Chip de CPU
o
microprocesador
ALU
(calculadora)
Comunicaciones
internas
Registros
(almacenamiento
temporal)
Sección
de control
Memoria interna de almacenamiento/entrada
RAM
Lectura/escritura
ROM
Sólo lectura
Sistema de bus
Interfaz de entrada
Dispositivos de entrada
Teclado
Ratón (mouse)
Palanca de juegos
Lápiz óptico
Escáner
Interfaz de salida
Dispositivos de salida
Monitor
Impresora
Memoria externa de almacenamiento/entrada
Unidad de disco flexible
Unidad de disco duro
CD-ROM
Cinta magnética

Software

- Es una colección de programas y datos relacionados que proporcionan las instrucciones necesarias para decirle al hardware qué hacer y cómo hacerlo.
- Sistemas operativos. P.ej: GNU/Linux, Mac OS X, Ms Windows XP...
- Aplicaciones de desarrollo (IDEs, compiladores, etc...)
- Aplicaciones ofimáticas: procesadores de texto, hojas de cálculo...
- SGBD
- Videojuegos
- Y muchas más!!!!

Programas y lenguajes de programación

- **Programar** consiste en planificar y concretar la secuencia de órdenes necesarias para la realización de una tarea
- Un **programa** es un conjunto de instrucciones, en un orden determinado, que se le dan al ordenador para que lleve a cabo una tarea
- Un **lenguaje de programación** es un “idioma” (conjunto de reglas, notaciones, símbolos y/o caracteres) en el que comunicarle instrucciones a un ordenador. Es decir, un “idioma” con el que se puede hacer un programa

Lenguaje máquina (código máquina)

- Es el lenguaje usado directamente por la CPU
- Instrucciones codificadas en binario (0,1)
- Cada CPU tiene su conjunto de instrucciones específico
- Problemas:
  - Programas dependientes de la máquina
  - Programas difíciles de leer y modificar
  - Instrucciones poco intuitivas (secuencias de 0 y 1) y difíciles de recordar
  - Muchos errores en los programas

Ejemplo lenguaje máquina

- El código de máquina en hexadecimal se resalta en rojo, el equivalente en lenguaje ensamblador en magenta, y las direcciones de memoria donde se encuentra el código, en azul[1]

```
-u 100 1a
0CFD:0100 BA0B01      MOV     DX,010B
0CFD:0103 B409        MOV     AH,09
0CFD:0105 CD21        INT     21
0CFD:0107 B400        MOV     AH,00
0CFD:0109 CD21        INT     21
-d 10b 13f
0CFD:0100                                               48 6F 6C 61 2C
0CFD:0110 20 65 73 74 65 20 65 73-20 75 6E 20 70 72 6F 67
0CFD:0120 72 61 6D 61 20 68 65 63-68 6F 20 65 6E 20 61 73
0CFD:0130 73 65 6D 62 6C 65 72 20-70 61 72 61 20 6C 61 20
0CFD:0140 57 69 6B 69 70 65 64 69-61 24
                                                Hola,
 este es un prog
rama hecho en as
sembler para la
Wikipedia$
```

1. Ejemplo extraído de www.wikipedia.com

Lenguaje ensamblador

- Intento de acercar el lenguaje natural al lenguaje de los procesadores
- Uso de palabras mnemotécnicas[1] en lugar de secuencias de ceros y unos para las instrucciones
- Un programa, denominado **ensamblador**, se encarga de traducir el programa a código máquina
- Los programas siguen siendo dependientes de la máquina

1. La mnemotecnia es un procedimiento de asociación mental para facilitar el recuerdo de algo

Lenguajes de alto nivel

- Mucho más cercanos al lenguaje natural que los lenguajes ensamblador y máquina (lenguajes de bajo nivel)
- Cada instrucción escrita en un lenguaje de alto nivel se transforma en una o varias instrucciones de código máquina
- Ejemplos: Basic, Java, Pascal, Cobol, Ada, C (algunos autores lo consideran de nivel intermedio)
- Ventajas:
  - Mayor independencia de la máquina
  - Programas más fáciles de leer y mantener

Compilador

- Un compilador es un programa informático que traduce un programa escrito en un lenguaje de programación de alto nivel a otro lenguaje (generalmente código máquina)
- Las instrucciones escritas en un lenguaje de programación constituyen el **código fuente**
- El compilador procesa el código fuente y genera lo que se denomina **código objeto**

Fases en la creación de un programa

- Fase de edición => **Código fuente**
- Fase de compilación: A partir del código fuente se genera el **código objeto** (en lenguaje máquina)
- Fase de enlazado (linkage): Al código objeto se le añaden los módulos o librerías externas que sean necesarios y se genera el **código ejecutable**
  - Los programas encargados de realizar el enlazado se llaman **enlazadores** o **linkers**

Intérpretes

- Son programas que ejecutan instrucciones escritas en un lenguaje de programación.
- A diferencia de los compiladores, los intérpretes van traduciendo el programa sobre la marcha y generalmente no guardan el resultado de esa traducción
- Algunos lenguajes se compilan para ser traducidos a un código intermedio que después es interpretado (p. ej. Java)
- En el caso de Java, el intérprete se denomina **Máquina Virtual de Java**
- Los programas interpretados suelen ser más lentos que los compilados.
- Los programas interpretados no dependen de la máquina, sino del intérprete

Errores en los programas

- En tiempo de compilación
  - Warnings: No son errores sino advertencias
  - Errores léxicos, sintácticos y semánticos
- En tiempo de ejecución
  - Defectos (Bugs)
  - Herramientas de depuración (Debuggers)
  - Sistemas de seguimiento de bugs (p.ej: Bugzilla)

Enfoque tradicional del desarrollo de SW

- En los inicios de la programación:
  - Codificar y probar.
  - No se planificaba.
  - No se documentaba.
  - Sólo la persona que hacía la aplicación era capaz de hacer modificaciones o correcciones.
- Esa forma de trabajar tiene inconvenientes:
  - Poca flexibilidad. Modificaciones costosas.
  - Se desconoce el progreso del proyecto.
  - No resuelve correctamente las necesidades del cliente.

Concepto de Ciclo de Vida

- Conjunto de actividades de analistas, diseñadores, y usuarios que necesitan llevarse a cabo para desarrollar y poner en marcha un Sistema de Información.
- El Ciclo de Vida debe abarcar toda la vida del sistema, desde su concepción hasta su finalización (cuando deje de utilizarse).

Funciones del Ciclo de Vida

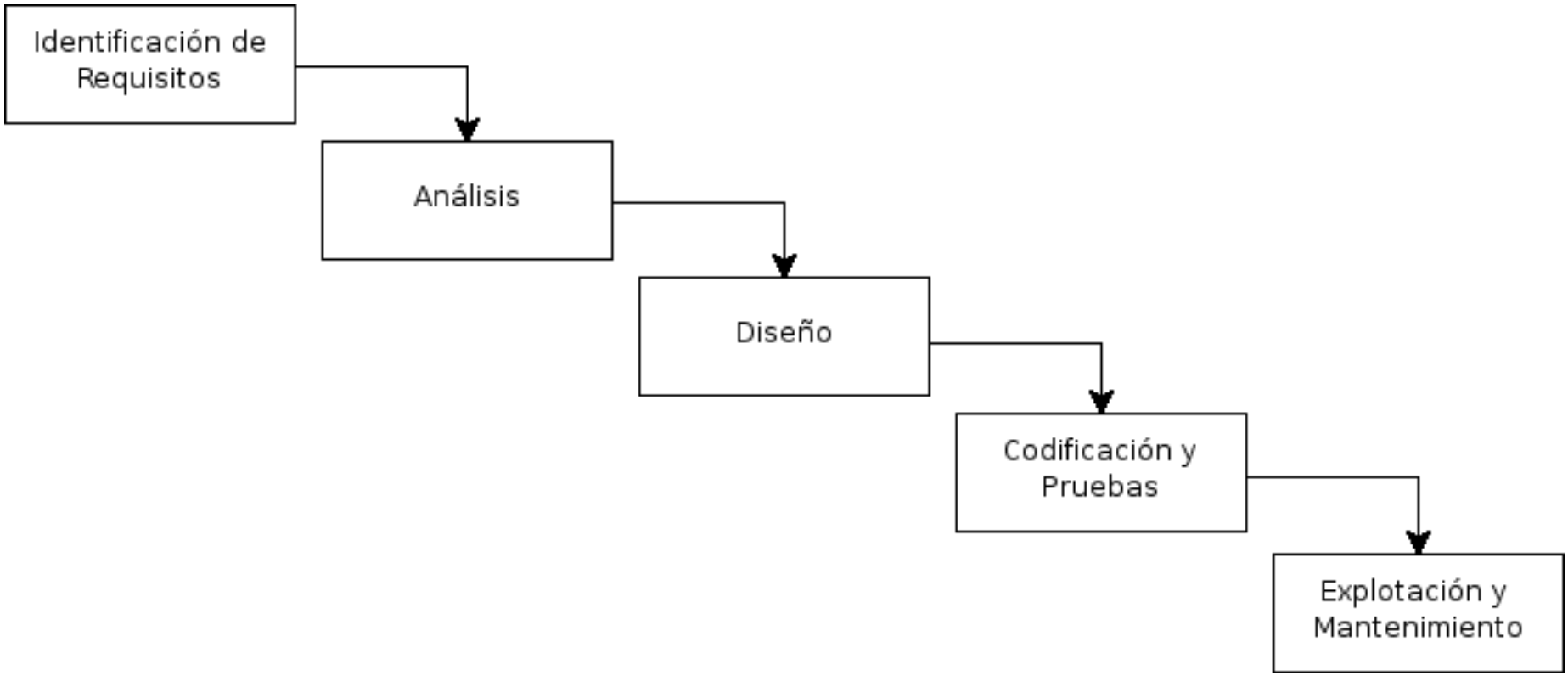
- Determinar el orden de las fases y procesos involucrados en el desarrollo del software.
- Establecer los criterios de transición para pasar de una fase a la siguiente.

Ventajas del enfoque de Ciclo de Vida

- Pensar el diseño es avanzar en la construcción del sistema. Posteriormente la codificación será más sencilla
- Menos defectos y detección precoz de los mismos
- Seguimiento del progreso. Se controla el sobrepasar plazos y costes
- Documentación estandarizada simultáneamente al desarrollo
- Sirve de guía al personal de desarrollo, marcando las tareas a realizar

Ciclo de Vida en Cascada

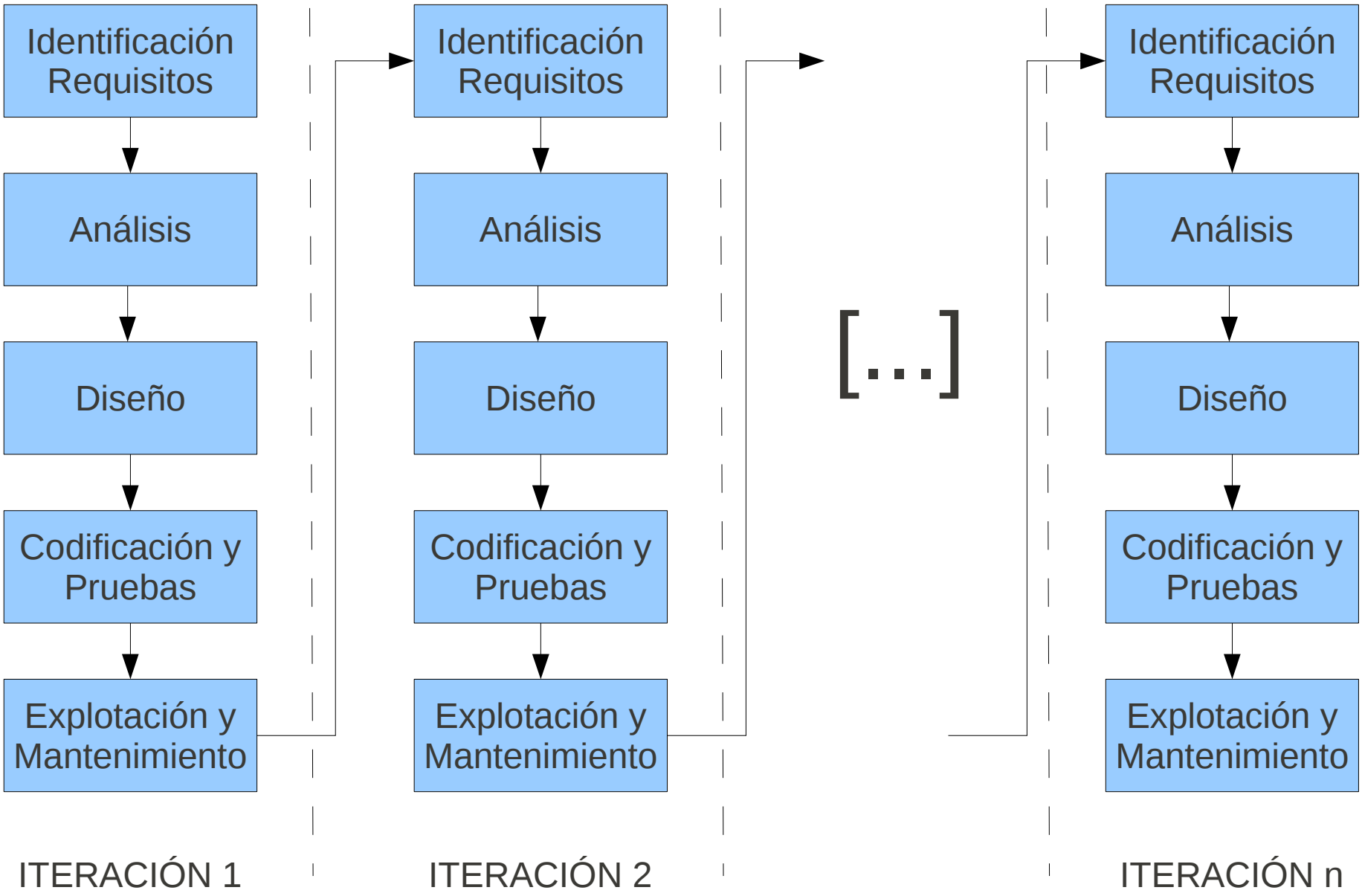
- Cada fase empieza cuando ha terminado la fase anterior.



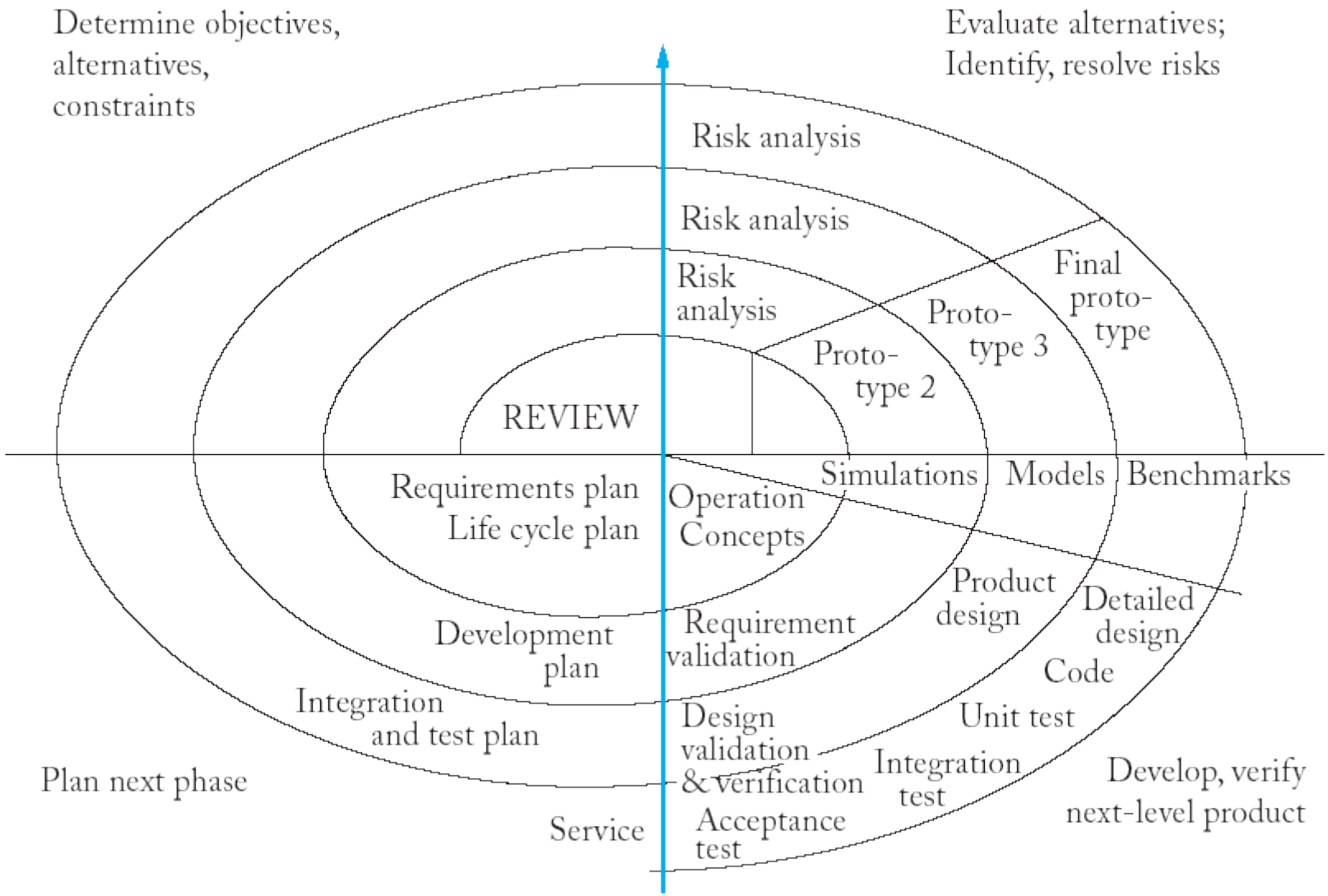
Ciclo de Vida Incremental (I)

- Se va creando el sistema añadiendo componentes funcionales (incrementos).
- En cada iteración se actualiza el sistema con nuevas funcionalidades o requisitos (se parte de una versión previa y se añaden nuevas funciones).
- Se ajusta a entornos de alta incertidumbre (no es necesario identificar todos los requisitos desde el principio).

Ciclo de Vida Incremental (II)



Ciclo de Vida en Espiral



Concepto de Metodología de Desarrollo

- Conjunto de procedimientos, técnicas, herramientas y un soporte documental que ayuda a los desarrolladores a realizar nuevo software.
- Una metodología se elabora a partir del marco definido por un ciclo de vida.
- Guía a los desarrolladores en la elección de las técnicas adecuadas para cada estado del proyecto.
- Facilita la planificación, gestión, control y evaluación de los proyectos.

Las metodologías especifican...

- Cómo se debe dividir un proyecto en etapas
- Qué tareas se llevan a cabo en cada etapa
- Qué salidas se producen y cuándo se producen
- Qué restricciones se aplican
- Qué herramientas se van a utilizar
- Cómo se gestiona y controla un proyecto

Ejercicios

- Averigua qué son los siguientes conceptos y qué relación o diferencias existen entre ellos: Java, JVM, JDK, JRE, Bytecode, Javadoc
- ¿Qué es Metrica v3? Haz un resumen de sus principales características y de su estructura.
- Investiga acerca de tres lenguajes de programación de alto nivel: ¿son compilados? ¿interpretados? ¿qué herramientas (compiladores, etc...) libres o comerciales existen sobre ellos?
- Descubre tres herramientas de seguimiento de bugs y comenta brevemente sus características principales.