Tema 1: El desarrollo del software

Programación Modular y Orientada a Objetos

Felipe Ibañez, Mikel Larrañaga,
Juanan Pereira, Begoña Ferrero
Dpto. de LSI

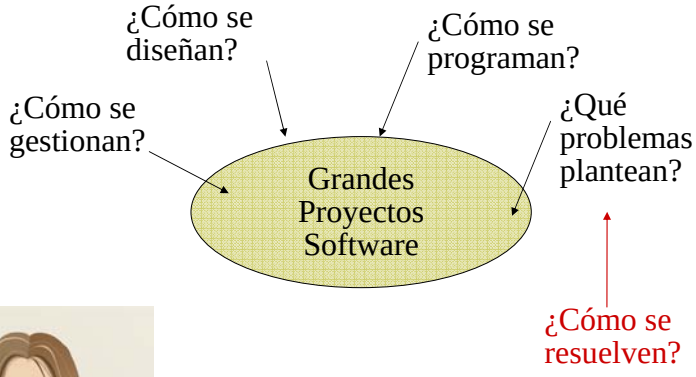
Contenidos

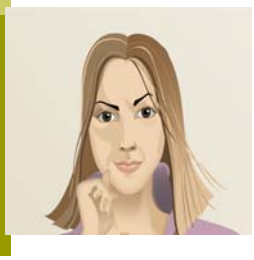
- Objetivos
- Crisis del software
- Evolución del Software
  - Programación NO estructurada, estructurada, Programación Orientada a Objetos(POO)
- Problemas de la programación estructurada
- Modularidad
- La POO como solución
- Principios del desarrollo de software

Objetivos

- *Justificar la asignatura como un paso necesario en el aprendizaje de la programación*
- *Problemas que se plantean en la elaboración de programas grandes, (crisis del Software)*
- *Características requeridas en el software para resolverlos y, como consecuencia, principios de diseño*

Problemas en el desarrollo del Sw





4

Problemas en el desarrollo del Sw

Programas Pequeños

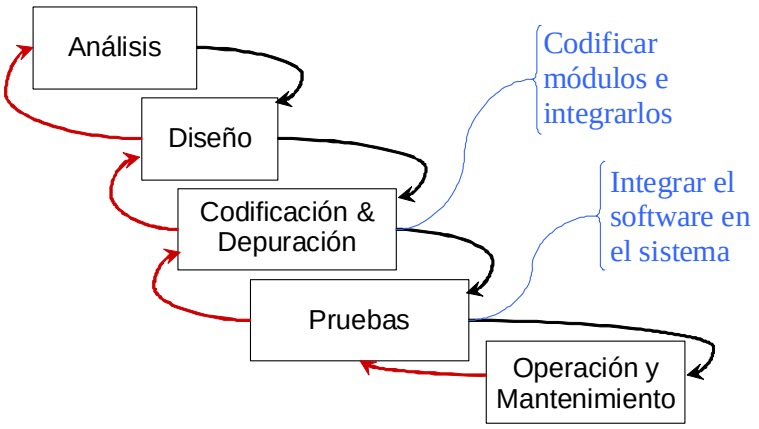
- El problema se entiende fácilmente. Sus objetivos están bien definidos.
- A menudo no se necesita documentación detallada
- Se realizan en poco tiempo
- El código es suficiente para entender el programa
- Facilidad de verificación y modificación

Programas Grandes

- Problemas muy complejos
- Código muy largo
- Se necesita un equipo y buena comunicación
- Documentación compleja: identificación de requisitos, especificación, diseño,...
- Difíciles de cambiar

Problemática similar a la de los grandes proyectos de ingeniería

Ciclo de vida del desarrollo del sw



Síntomas de la Crisis del Software

Lo pide directamente el cliente

- **Respuesta**: los sistemas a menudo no corresponden con las necesidades del usuario
- **Fiabilidad**: fallos
- **Modificabilidad**: mantenimiento costoso, complejo y propenso a errores

Deseable

- **Portabilidad**: software desarrollado para una plataforma es difícil de usar sobre otra diferente
- **Eficiencia**: los esfuerzos de desarrollo de software no optimizan los recursos involucrados

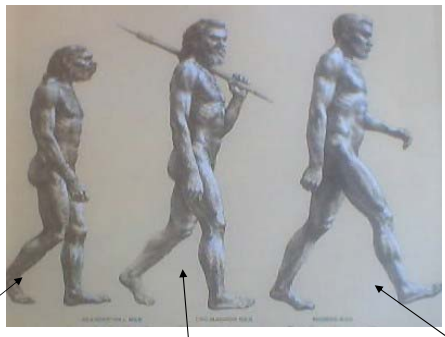
Consecuencias de la Crisis del Sw

- Los proyectos grandes terminan **fuera de plazo** y exhiben **capacidades menores** que las prometidas
- El Software comparado con el hardware es **caro**

Soluciones para abordar la crisis

- Aplicación de técnicas de ingeniería: métodos, lenguajes y herramientas apropiados, utilizados de forma efectiva
- Los lenguajes de programación deben soportar los nuevos métodos de desarrollo de software

Evolución del software



Programación
NO estructurada

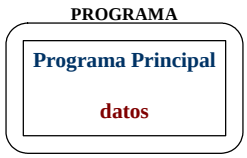
Programación
Estructurada
(o prrocedimental)

Programación
Orientada a objetos

MODULARIDAD

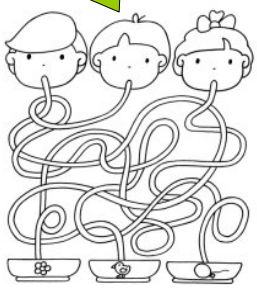
Programación NO estructurada

- Un sólo bloque principal
- Típica de los lenguajes de bajo nivel
- Variables globales a todo el programa
- Problemas:
  - Suelen contener secuencias de código repetidas
  - Cualquier modificación en el programa → revisar todo el software



Programación NO estructurada

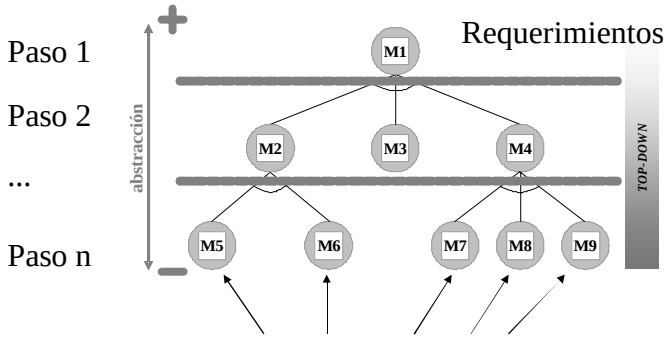
¿Alguna duda sobre por qué se le llama spaghetti code?



```
Función noEstructurada1
        escribir("Teclea dos números: ");
        leer(a, b);
        si (b = 0) entonces saltarA(eti3)
        c2 ← 1
        d ← a
eti2:   si (c2 < b) entonces c1 ← 0
        sino saltarA(eti4)
        f ← d
        d ← 0
eti1:   d ← d + f
        c1 ← c1 + 1
        si (c1 < a) entonces saltarA(eti1)
        c2 ← c2 + 1
        saltarA(eti2)
eti3:   d← 1
eti4:   escribir(d)
```

Programación estructurada

Diseño TOP-DOWN



Procedimientos y Funciones

Programación estructurada (II)

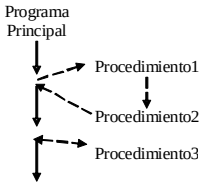
```
Programa principal
    entero a, b, d;
    pedir_numero(a);
    pedir_numero(b);
    d=potencia(a, b);
    escribir(d);
```

Variables globales

parámetros

```
Función entero potencia (entero a, entero b)
        entero c;
        c←1
        mientras(b>0) hacer
          principio
                c←multiplicar(a, c);
                b←b-1;
        fin
        devolver(c);
```

Variables locales

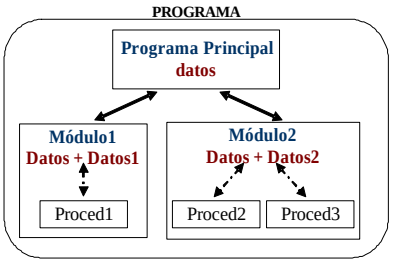


Programación estructurada (III)

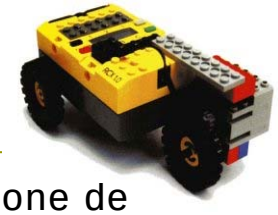
- Problemas:
  - Si se cambia el tipo de dato de las variables globales, hay que cambiar todas las funciones que acceden a ellos
  - Dificiles de diseñar y no modelan bien el mundo real
- Desventajas:
  - Dificultad para la definición de nuevos tipos de datos

Modularidad

- Agrupar en módulos los procedimientos con una funcionalidad común
- Cada módulo puede contener sus propios datos
- Ámbito de las variables



Modularidad



- Una aplicación grande se compone de varios módulos
  - Los módulos deben ser **independientes**
  - **En las aplicaciones se integrarán distintos módulos que se comunican entre sí**
  - Buen diseño → Separación de módulos

¿Por qué modular las aplicaciones?

Descomponer el problema en partes más simples
Descomposición Modular

El sistema está compuesto de una serie de módulos independientes comunicados entre sí
Composición Modular

Facilitar la comprensión del sistema y de cada una de sus partes
Inteligibilidad

Las modificaciones debidas a cambios en la especificación afectan a un número reducido de módulos
Continuidad Modular

Si se produce un error en un módulo, éste sólo afecta a dicho módulo
Protección Modular

Programación Orientado a Objetos

- Paradigma de programación que define los programas en términos de objetos

Programación Orientado a Objetos

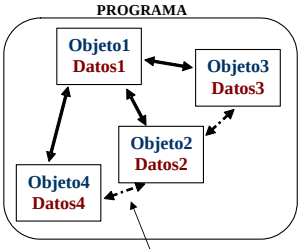
Objetos

Encapsulamiento

- **Estado** (datos ó atributos)
- **Comportamiento** (métodos)
- **Identidad** (propiedad que identifica a un objeto)

Clase

Familia de objetos con caracteristicas similares



Los objetos interactuan entre ellos enviandose mensajes

Programación Orientado a Objetos

objeto1
color rojo
tipo deportivo
matricula 1234 ABC
velocidadMax 300

objeto2
color azul
tipo policiaco
matricula 0000 PM
velocidadMax 250

objeto3
color amarillo
tipo elegante
matricula 9876 DEF
velocidadMax 150

objeto4
color rojo
tipo utilitario
matricula 9235 FCB
velocidadMax 200

Principios de desarrollo del Sw

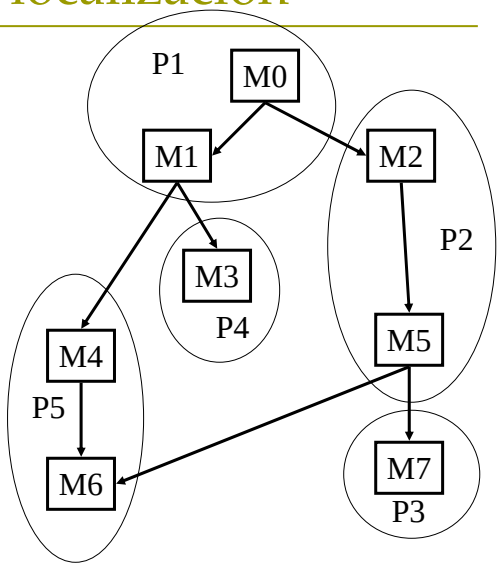
1. Abstracción
2. Modularidad
3. Ocultación de la información

Abstracción

- La esencia de la **abstracción** es extraer las **propiedades esenciales** mientras omitimos las que no lo son; ponerse al nivel del usuario final y enfocar los datos y las operaciones desde su punto de vista
- Se aplica tanto a datos como a algoritmos
- Ejemplo: descripción de un coche a distintos niveles de abstracción:
  - Desde el punto de vista de la secretaria de la DGT
  - Desde el punto de vista de un conductor
  - Desde el punto de vista del mecánico
  - Desde el punto de vista de un diseñador de piezas

Modularidad y localización

- Las aplicaciones grandes se deben estructurar en módulos con sentido que puedan reutilizarse.
- Estos módulos pueden estar interrelacionados
- Para garantizar la fiabilidad es necesario definir y limitar la información accesible desde otros módulos



Ocultación de la información

- El objetivo de la ocultación de la información es que cada módulo sólo muestre lo estrictamente necesario a otros módulos, ocultando detalles innecesarios
- Por ejemplo, las variables locales que utiliza un procedimiento no deben ser visibles fuera de él
- **Beneficios**:
  - Mejorar la inteligibilidad
  - Mejorar la modificabilidad
  - Aumentar la fiabilidad

Bibliografía

- [Meyer, 99] "Construcción de Software Orientado a Objetos" (2ª edición). B. Meyer. Prentice-Hall, 1999
- [Joyanes, 98] "Programación Orientada a Objetos" (2ª edición). L. Joyanes, McGraw-Hill, 1998