Orientación a objetos

Electrónica IV

Mg.Ing. Esteban Volentini (evolentini@herrera.unt.edu.ar)

https://facetvirtual.facet.unt.edu.ar/course/view.php?id=165

Manejo de la complejidad

- El software es intrínsecamente complejo y la solución es la estrategia "divide y vencerás”
- Durante el diseño se utiliza un enfoque Top-Down para dividir el problema en partes mas pequeñas
- Se utiliza la abstracción: de cada parte se especifican las entradas y salidas sin importar la implementación
- Este proceso se puede volver a aplicar nuevamente a cada parte hasta obtener componentes suficientemente simple para ser construidos

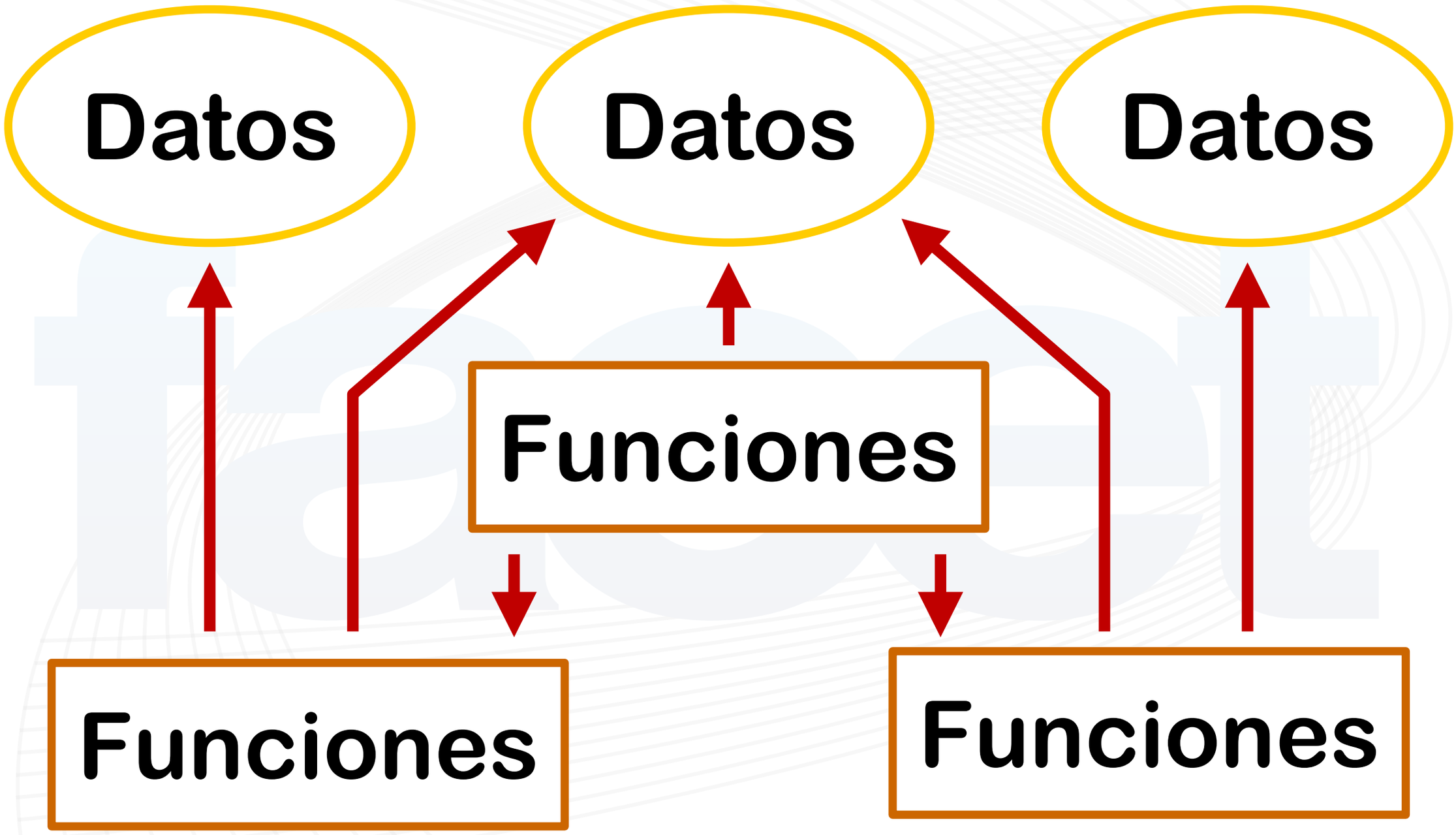
Manejo de la complejidad

- A partir de las entradas y salidas de cada componente se diseña el funcionamiento interno del mismo.
- Se construye cada componente y se prueba a partir de las entradas y salidas definidas durante la partición.
- Los componentes se integran para construir las piezas de orden superior del diseño.

Diseño estructurado

- La forma tradicional de dividir el problema parte de acciones que describen las tarea que se deben implementar en el software
- Estas tareas se subdividen en tareas mas simples con mas detalle
- El resultado es un conjunto de procedimientos que operan sobre un conjunto de datos

Diseño estructurado



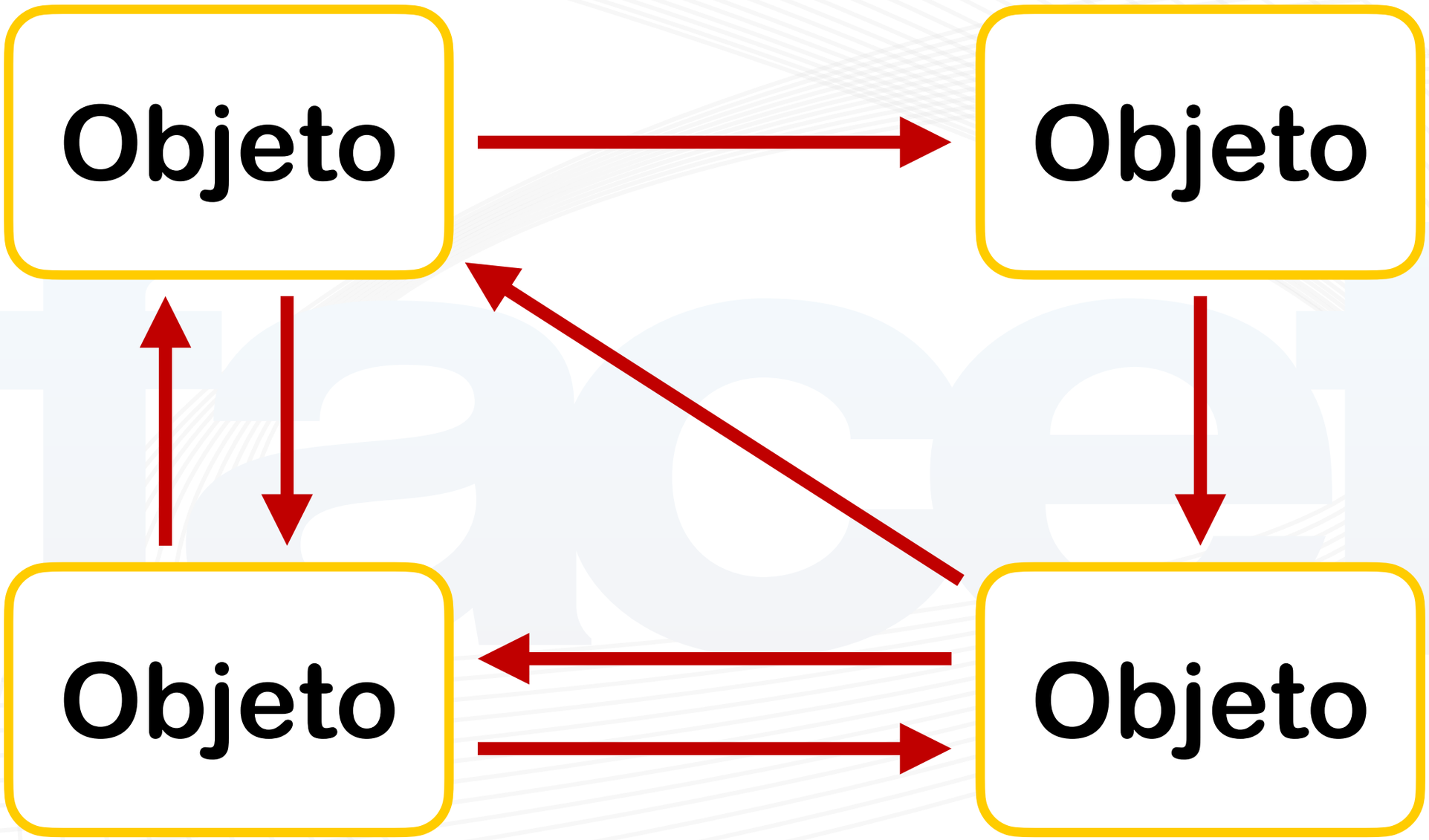
Ventajas y desventajas

- Es una forma mas natural de resolver el problema
- Un cambio en la estructura de los datos afecta a todos los procedimientos
- Un cambio en un procedimiento puede generar datos incoherentes para otro procedimiento
- Cuando el software es muy extenso y muchas personas participan del desarrollo el problema se agrava
- Se puede mitigar usando módulos aislados

Diseño Orientado a Objetos

- Presenta una forma alternativa de dividir el problema
- Se utilizan objetos como una representación mas cercana al mundo real
- Los datos y las funciones que los manipulan son internos de cada objeto e inaccesibles al resto
- Es equivalente a un enfoque estructurado con una división en módulos llevada al extremo

Diseño orientado a objetos



Ventajas

- Los objetos del diseño representen entidades de la realidad, lo que facilita la comprensión del programa y del problema
- Se obtiene una solución modular que aumenta la seguridad y facilita el mantenimiento
- Es más fácil reutilizar los objetos, lo que disminuye el esfuerzo de desarrollo

Características de DOO

- **Abstracción**: Permite separar la implementación de la interfaz de uso de un componente
- **Modalidad**: Permite modificar las características de algunos componentes sin afectar al resto
- **Encapsulamiento**: Permite ocultar la información interna de un componente al resto del sistema

Características de DOO

- **Relaciones**: Permite definir vinculaciones entre los componentes que colaboran para resolver una parte del problema
- **Herencia**: Permite crear un componente especializado a partir uno genérico
- **Polimorfismo**: Permite que dos componentes especializados que parten de un mismo ancestro realicen una misma operación en forma diferente

Características de los objetos

- **Identidad**: Permite diferenciar un objeto de otro, aun cuando sus atributos sean iguales
- **Estado**: El estado de un objeto está definido por el valor de sus atributos
- **Comportamiento**: Está definido por la acciones que se pueden realizar sobre el objeto

Atributos y Métodos

- Los atributos almacenan los datos del objeto. Pueden ser de cualquier tipo primitivo o incluso otro objeto
- Los métodos, son funciones o procedimientos que implementan las acciones definidas para objeto
- La única forma de manipular la información de un objeto es a travez de sus métodos

Interacción de los objetos

- Conceptualmente los objetos se comunican intercambiando mensajes
- Generalmente esto se traduce en un objeto que ejecuta un método de otro objeto
- Los distintos mensajes que puede recibir un objeto se denomina protocolo
- Es equivalente a una API del objeto

Clases de objetos

- Es habitual que en un software existan varios objetos que comparten estructura y comportamiento
- Por esta razón la estructura y el comportamiento se define para una clase de objetos y no para un objeto en particular
- Cada objeto es una instancia de una clase y comparte la estructura y comportamiento con los otros objetos de la mima clase

Implementación en C

- Cada archivo **.c** es equivalente a un paquete donde se define una o más clases
- El archivo **.h** declara los métodos públicos de las clases
- Para cada clase se declara un puntero a una estructura anónima para representar a un objeto en particular

Implementación en C

- La definición de los miembros de la estructura es privada al archivo **.c** y almacena los atributos de cada objeto.
- Los métodos son funciones que reciben como primer parámetro la referencia al objeto, es decir el puntero a la estructura con el estado del objeto.

Visibilidad de método y atributos

- Private: solo puede ser accedido internamente por el propio objeto
- Protected: solo puede ser accedido por el objeto o por un descendiente
- Package: solo puede ser accedido por objetos del mismo paquete
- Public: puede ser accedido desde cualquier parte del sistema