

APUNTES DE ASDM

Ingeniería del Software



Contenido

[**Tema 1: Arquitecturas Software** 2](#_Toc135670460)

[1. INTRODUCCIÓN 2](#_Toc135670461)

[2. ARQUITECTURA SOFTWARE EN SUBSISTEMAS 3](#_Toc135670462)

[3. ARQUITECTURA SOFTWARE ORIENTADA A OBJETOS 4](#_Toc135670463)

[4. ARQUITECTURA SOFTWARE CLIENTE/SERVIDOR 5](#_Toc135670464)

[5. ARQUITECTURA SOFTWARE ORIENTADA A SERVICIOS (SOA) 5](#_Toc135670465)

[6. ARQUITECTURA SOFTWARE ORIENTADA A COMPONENTES 6](#_Toc135670466)

[7. PATRONES DE COMUNICACIÓN 6](#_Toc135670467)

[**Tema 3: Desarrollo Dirigido por Modelos** 8](#_Toc135670468)

## **Tema 1: Arquitecturas Software**

## INTRODUCCIÓN

La **Arquitectura del Software** es la organización fundamental de un sistema en sus Componentes, las Relaciones entre sus componentes y el Entorno.

* **Componentes:** elementos del sistema que ofrecen servicios mediante una interfaz. La interfaz es la parte externamente visible.
* **Relaciones entre Componentes:** colaboración entre componentes para llevar a cabo una función.
* **Entorno:** espacio donde va a funcionar la aplicación.

La arquitectura tiene **vistas** para representar los aspectos del software, las cuales son:

* **Vista estática:** Componentes y Conectores.
* **Vista dinámica:** Objetos y Mensajes.
* **Vista funcional:** Nodos e Interconexiones.

**DEFINICONES**

**Patrón:** arquitectura recurrente usada en varios sistemas informáticos. Sus ventajas son: rendimiento, reutilización, bajo coste, …

**Diferencia entre Arquitectura Software y Diseño Software:** la arquitectura define las clases (estático) mientras que el diseño instancia la arquitectura.

**Para realizar el modelo arquitectónico se requiere:** dominio de la aplicación, modelo de requerimientos, estilos arquitectónicos y sus patrones.

Existen varios tipos de Arquitecturas del Software:

1. Arquitectura Software en Subsistemas.
2. Arquitectura Software Orientada a Objetos.
3. Arquitectura Software Cliente/Servidor.
4. Arquitectura Software Orientada a Servicios.
5. Arquitectura Software Orientada a Componentes.

## ARQUITECTURA SOFTWARE EN SUBSISTEMAS

Aplicable a sistemas de gran tamaño. Su **objetivo** consiste en que cada sistema realice una función concreta e independiente del resto.

* Los objetos que son parte del mismo objeto compuesto deben estar en el mismo subsistema.
* Si los objetos están gráficamente separados, se separan en distintos subsistemas.
* Los clientes y servicios deben de estar en subsistemas separados.
* Los objetos que interactúan con el usuario deben de estar en un subsistema separado del resto de objetos.
* Un objeto del mundo real externo debe conectarse a un único subsistema.
* Un objeto de control y todas las entidades y los objetos de E/S que controla deben formar parte de un subsistema.

**Tipos de Subsistemas:**

* Subsistema cliente.
* Subsistema de interacción con el usuario.
* Subsistema de servicio.
* Subsistema de control.
* Subsistema de E/S.

## ARQUITECTURA SOFTWARE ORIENTADA A OBJETOS

Se divide en tres pasos:

* **Paso 1: Diseñar las clases y su información oculta.**

Se determina a partir del modelo de análisis. Tipos de clases:

* + **Clase entidad <<entity>>:** encapsulan datos.
    - Clase de abstracción de datos <<data abstraction>>: encapsula la estructura de datos.
    - Clase Wrapper <<wrapper class>>: encapsula datos obtenidos de una fuente externa, como una base de datos.
  + **Clase límite <<boundary>>:** para comunicarse mediante una interfaz con el medio exterior a la aplicación.
    - Dispositivos E/S <<I/O>>: accede al dispositivo de E/S.
    - Interfaz de Usuario <<user interaction>>: el usuario interactúa con ella.
  + **Clase control <<control>>:** proporciona coordinación general.
    - Coordinador <<coordinator>>
    - Dependiente del estado <<state dependent control>>
    - Temporizador <<timer>>
  + **Clase lógica de aplicación <<business logic>>:** encapsula la lógica y algoritmos para una aplicación concreta.
    - Negocio <<business logic>>: define la toma de decisiones.
    - Servicio <<service>>: proporciona un servicio a los clientes.
    - Algoritmo <<algorithm>>: encapsula un algoritmo usado en el dominio del problema.
* **Paso 2: Diseñar las interfaces y las operaciones de clase.**

Se determina a partir del modelo dinámico. Cuando en los diagramas de secuencia llega un mensaje a un objeto, se agrega una operación a la clase con el nombre del mensaje y los parámetros (hay que identificar si los parámetros son de E/S). Las **operaciones** **crear**, **eliminar** y **modificar** se pueden sacar mediante modelo estático.

* **Paso 3: Diseñar la herencia.**

La **herencia** se usa en clases similares, pero **no** idénticas. La parte interna de las clases padres son visibles para los hijos. Los hijos no pueden modificar los atributos heredados del padre. Se utiliza como plantilla para la creación de subclases.

* **Polimorfismo:** diferentes clases pueden tener el mismo nombre de operación.
* **Vinculación dinámica:** una petición a un objeto puede cambiar entre una invocación y la siguiente. Junto con el polimorfismo se puede invocar una operación con el mismo nombre en diferentes objetos.

## ARQUITECTURA SOFTWARE CLIENTE/SERVIDOR

El servidor atiende y proporciona servicios a múltiples clientes (que son los solicitantes de servicios). Un servicio es un componente que cumple unas necesidades. Tipos de servicio:

* **Servicio secuencial:** un solo objeto responde a las peticiones.
* **Servicio concurrente:** varios objetos responden a las peticiones.

**Tipos de Cliente/Servidor:**

* **Múltiples clientes - Único servicio:** varios clientes acceden a un único servicio que los atiende.
* **Múltiples clientes - Múltiples servicios:** los servidores soportan múltiples servicios. Los clientes pueden comunicarse con más de un servicio a la vez. Los servicios se comunican entre ellos.
* **Cliente/Servicio Multicapa:** tiene una **capa intermedia** que proporciona ambos roles, el cliente y el servicio.

Las capas son clientes de la capa inferior y servicio de la capa superior. Hay como mínimo una capa intermedia que hace de cliente y servicio a la vez. El cliente está en lo más alto. Las capas deben de ser independientes y transparentes. Por ejemplo, un Navegador Web, servidor HTTP, …

## ARQUITECTURA SOFTWARE ORIENTADA A SERVICIOS (SOA)

**SOA**, múltiples servicios autónomos y distribuidos.

Los **servicios** no son fijos, son individuales, autónomos y reutilizables. Pueden ejecutarse en diferentes maquinas y estar implementados en diferentes lenguajes.

En SOA los servicios están débilmente acoplados y se **comunican** con los clientes mediante gerentes del servicio.

## ARQUITECTURA SOFTWARE ORIENTADA A COMPONENTES

Se utiliza en sistemas distribuidos. Consiste en:

* **Múltiples** componentes.
* Cada uno encapsula **cierta información**.
* Cada componente proporciona una **interfaz** a través de la cual se **comunica** con otros componentes.
* Los **componentes** se comunican entre sí de diferentes maneras.

**Pasos principales:**

1. Diseño de la arquitectura distribuida: mover los componentes a los nodos.
2. Diseño de componentes individuales: según la arquitectura orientada a objetos.
3. Implementar la aplicación: interconectar los diferentes componentes que están en lo nodos.

**Criterios para estructurar componentes:**

* Proximidad a los datos físicos.
* Rendimiento.
* Hardware especializado.

## PATRONES DE COMUNICACIÓN

* **Llamada con retorno:** paso de mensajes y diseño secuencial. Ej.: RPC
* **Mensaje asíncrono:** para sistemas distribuidos y concurrentes. El mensaje se envía sin esperar la respuesta. Los mensajes se encolan hasta que el consumidor los lea y envíe una respuesta.
* **Broker:** principalmente pasa SOA. Los servicios se registran en un bróker para que los solicitantes puedan encontrar el servicio. Hay varios tipos:
  + **White Pages:** proporciona los datos de un servicio concreto por su nombre.
  + **Yellow Pages:** proporciona servicios por categoría.

**Patrones del Broker:**

* **Registro de servicios:** los servicios tienen que solicitar registrarse una vez o cuando se recolocan.
* **Broker de reenvío:** recibe peticiones del cliente, lo envía al servicio, recibe la respuesta del servicio y la envía al cliente. Bajo rendimiento, pero seguro.
* **Broker handle:** el broker envía un identificador de servicio al cliente, para que este se comunique directamente con el servicio.
* **Descubrimiento de servicio:** el broker recibe una petición por criterios y responde con un catálogo de servicios, el cliente escoge uno y el broker le envía el ID del servicio para que se comuniquen directamente.

## **Tema 3: Desarrollo Dirigido por Modelos**

**DBM** 🡪 Desarrollo Basado en Modelos (Tom de Marco).

Tiene 2 capas: lenguaje y proceso.

**MDD** 🡪 Desarrollo Dirigido por Modelos.

**Mejora** el DBM, añadiendo modelos abstractos que llegan a ser concretos con una serie de transformaciones. Automatiza el proceso de análisis y diseño, reduciendo la intervención humana.

**Modelo** 🡪 Representación o esquema teórico de un sistema.

**Modelo Matemático** 🡪 técnicas formales rigurosas, permite la comprobación de especificaciones. Ej.: OCL, ALLOY.

**Modelo Diagramático** 🡪 modelo gráfico más sencillo, pero menos formal. Ej.: UML.

**Transformación** 🡪 generación de un modelo destino a partir de un modelo fuente.

**Definición de transformación** 🡪 conjunto de reglas de transformación.

**Regla de transformación** 🡪 descripción de cómo transformar un conjunto de construcciones de un lenguaje fuente a un lenguaje destino.

**MDS** 🡪 Modelado Especifico del Domino.

Se crea un modelo en un lenguaje especializado en el domino.

**MDA** 🡪 propuesta de MDD por la OMG (Object Managment Group). Hay varios tipos de modelos de MDA:

* **CIM (Modelo Independiente de la Computación):** se centra en el entorno del sistema y los requisitos para el mismo.
* **PIM (Modelo Independiente de la Plataforma):** se centra en las especificaciones del sistema.
* **PSM (Modelo Especifico de la Plataforma):** cambia el PIM con un enfoque especifico.
* **IM (Modelo de Implementación)**

**Capas de la arquitectura OMG (MDA):**

* **M0** 🡪 Instancias.
* **M1** 🡪 Modelo del sistema.
* **M2** 🡪 Metamodelo.
* **M3** 🡪 Meta-metamodelo.
  + MDF: lenguaje de la capa M3, se define a sí mismo. No existe otro meta nivel por encima de MDF.

**OCL** 🡪 lenguaje formal declarativo que agrega restricciones a un modelo.

**EMF** 🡪 permite la generación de código automática para construir herramientas y otras aplicaciones a partir de modelos de datos estructurados. Se especifican usando Ecore.