

## Diseño

- Diseño en el PUD
- Diseño de software
- Patrones arquitectónicos
- Diseño Orientado a Objetos en UML

## Iteración en PUD

- **Planificación de la Iteración**
  - **Captura de requerimientos:**
    - Modelo de casos de uso, Modelo de Dominio, ...
  - **Análisis:**
    - Diagrama de secuencia del sistema, Contratos, Modelo Conceptual...
  - **Diseño:**
    - Diagramas de interacción, Diagrama de Clases
  - **Implementación:**
    - codificación (Clases y métodos)
  - **Pruebas:**
    - verificación de la implementación
- **Evaluación de la iteración**

## Fases y entregas del Proceso Unificado de Desarrollo

- captura de requerimientos: qué SI debemos construir?
  - Modelo de casos de uso, Modelo de Dominio, ...
- análisis: qué debe hacer el SI?
  - Diagramas de secuencia del sistema, Contratos, ...
- diseño: cómo lo debe hacer el SI?
  - Diagramas de interacción, Diagrama de Clases, ...
- codificación:
  - Código Fuente (clases y métodos)
- pruebas:
  - Especificación de las pruebas de funcionamiento
- mantenimiento:
  - Documentación y revisión de todo lo anterior

Dependiente de  
la tecnología

## Construcción incremental e iterativa del SI

- **Modelo dinámico del sistema (comportamiento):**
  - Captura de requerimientos: Modelo de Casos de Uso
  - Análisis: Diagramas de secuencia del sistema, contratos
  - Diseño: Diagramas de interacción
- **Modelo estático del sistema (propiedades):**
  - Captura de requerimientos: Modelo de Dominio
  - Análisis: Modelo Conceptual
  - Diseño: Diagrama de clases
- **Implementación: codificación (clases y métodos)**

## Diseño de software

- Definir un SI con el suficiente detalle para permitir su implementación.
  
- Situación de partida (dominio del problema)
  - Resultado del análisis (qué debe hacer el SI?)
    - Especificación de los datos (Modelo de datos)
    - Especificación de procesos (Modelo de comportamiento)
    - Especificación de la interacción con el usuario (Modelo de la interfaz)
  
  - Recursos tecnológicos (hardware y software disponible)

## Diseño de software

- Situación final (dominio de la solución)
  - Resultado del diseño (cómo lo debe hacer el SI?)
    - Estructura interna del SI (Arquitectura del SI)
    - Diseño de los datos: diagrama de clases: atributos, BDs
    - Diseño de los programas: diagrama de clases: métodos
    - Diseño de la Interfaz: modelo de casos de uso reales
- Proceso de diseño
  - Metodologías de diseño
  - Patrones arquitectónicos y de diseño: adaptación de soluciones genéricas a problemas arquitectónicos y de diseño

## Diseño de software

- **Objetivos principales:**
  - Entender en profundidad los requisitos funcionales
  - Entender en profundidad los requisitos no funcionales y las restricciones relacionadas con las herramientas utilizadas: LPs, SOs, SGBDs, tecnologías de interfaz de usuario, etc.
  - Crear una abstracción de la implementación del SI (diseño)
  - Proporcionar un diseño apropiado para la fase de implementación
  - Descomponer el trabajo en partes que puedan ser implementadas en paralelo por equipos distintos

## Patrones arquitectónicos

- La arquitectura del SI es una descripción de los subsistemas y componentes, y las relaciones entre ellos.
- Determina:
  - La organización estructural del SI
  - La selección de los elementos estructurales
  - Las interfaces entre ellos
  - El comportamiento de los componentes
- Un componente es una parte física y reemplazable de un sistema



## Catálogo parcial de patrones arquitectónicos

- Llamada y retorno
  - estilo más usado en los grandes SI
    - Descomposición modular jerárquica
    - Orientado a objetos
    - En capas
- Centrado en los datos
  - permite la manipulación compartida de datos
    - Repositorio
    - Pizarra
- Flujo de datos
  - permite la transformación incremental de los datos
    - Batch secuencial
    - Tubos y filtros
- Modelo – vista – controlador
  - para sistemas interactivos

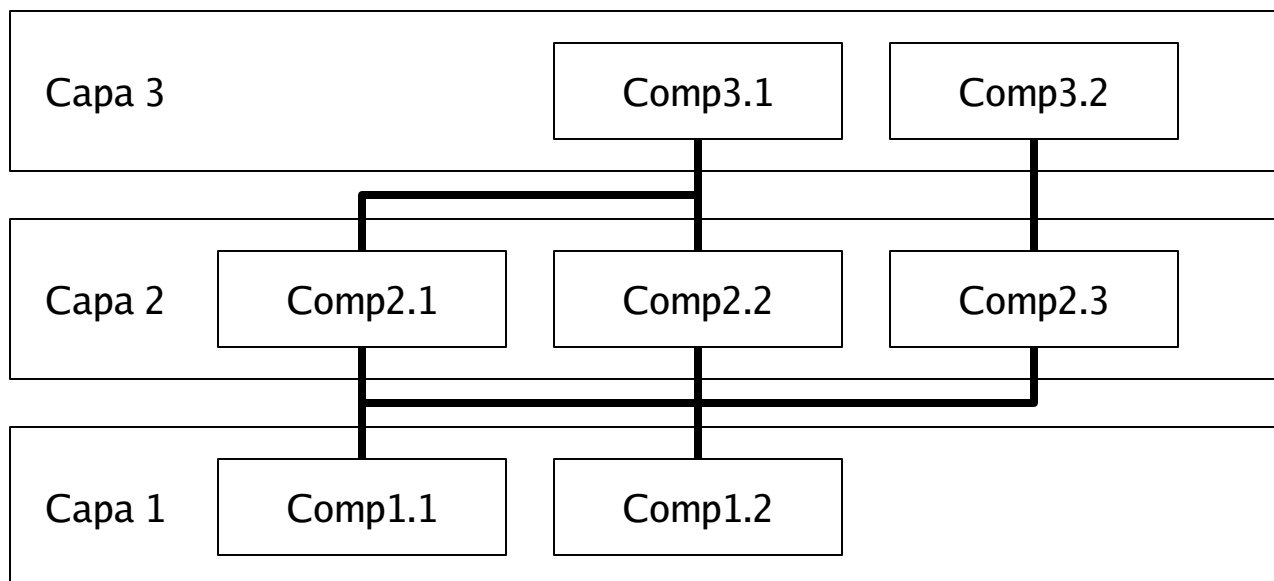
## Llamada y retorno (1)

- Descomposición modular jerárquica
  - Programa principal y subrutinas: programación estructurada.
  - Una subrutina (componente) recibe el control y los datos de uno de los módulos ascendientes, lo transforma y lo pasa a los módulos descendientes. El retorno se realiza en sentido contrario.
  - Facilita la modularidad.
- Orientación a objetos
  - Clases encapsulan datos y métodos.
  - La comunicación entre componentes se realiza mediante la invocación de servicios ofertados por ellos
  - Facilita la modularidad y la reusabilidad.

## Llamada y retorno (2)

### ▪ En capas

- Los componentes se agrupan en capas
- La comunicación sólo se establece entre componentes de la misma capa, o entre componentes de capas contiguas
- Facilita la modularidad y la portabilidad



## Centrado en los datos

- Los datos compartidos entre distintos clientes están en una estructura centralizada común
- Arquitectura originada en IA
- Repositorio: medio de almacenamiento pasivo.
  - Los clientes se comunican con el repositorio para actualizar los datos.
- Pizarra: medio de almacenamiento activo.
  - Los clientes se comunican con el repositorio para actualizar los datos y la pizarra notifica a los clientes los cambios que puedan ser de su interés

## Flujo de datos

- Batch secuencial
  - Secuencia de programas independientes
  - Un programa no empieza su ejecución hasta que los precedentes han terminado completamente
  - Facilita la modularidad y reusabilidad
  - Orientado a sistemas de ejecución diferida (batch)
- Flujo de datos
  - Secuencia de procesos que realizan modificaciones incrementales de los datos
  - Cualquier proceso puede empezar su ejecución cuando empieza a recibir datos
  - Arquitectura típica de los sistemas UNIX

## Modelo – vista – controlador

- Descomposición del SI en:
  - Modelo: implementa las funcionalidades y datos del SI
  - Interfaz de usuario:
    - Vista: gestiona cómo se muestra la información al usuario
    - Controlador: gestiona la interacción con el usuario

(Repasar diagrama de Jacobson)

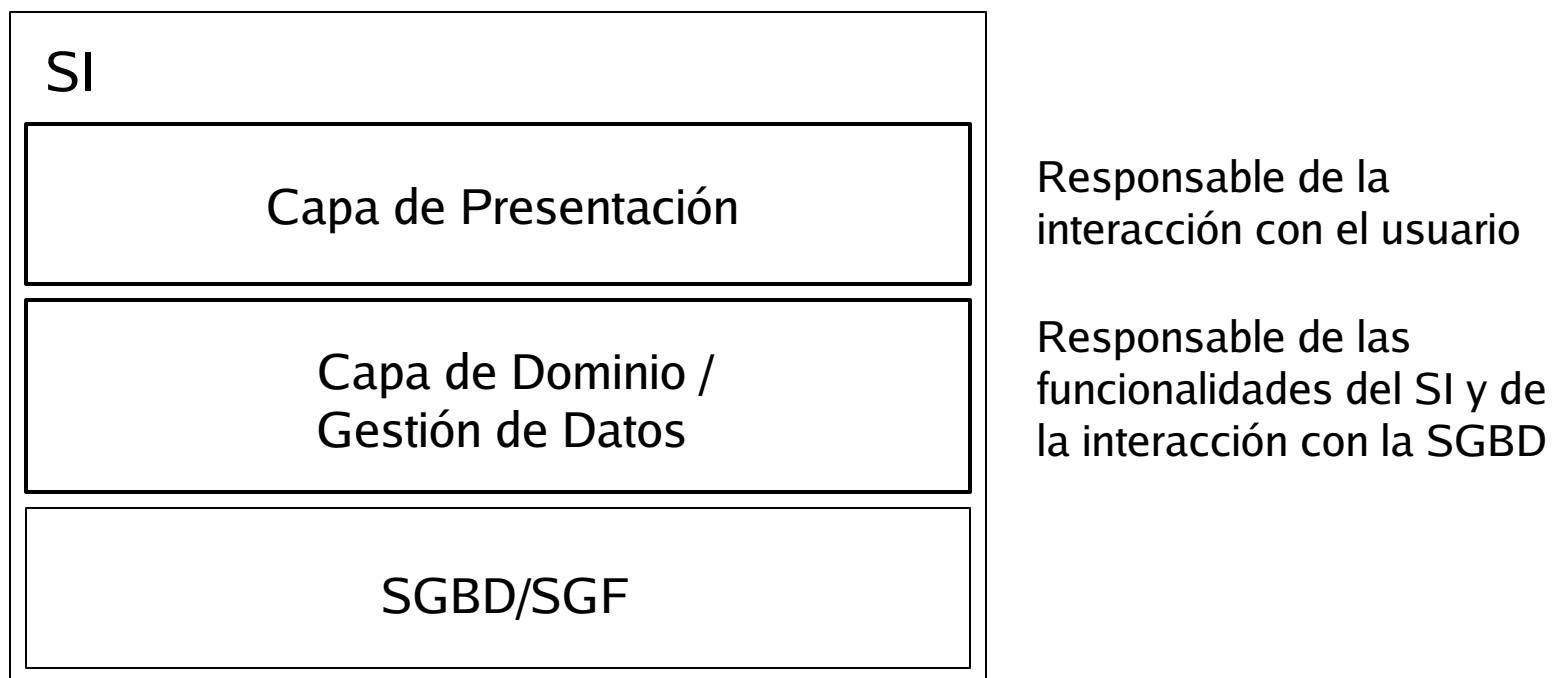
## Arquitectura lógica de un SI en una capa



Responsable de la interacción con el usuario, funcionalidades del SI y la interacción con el SGBD

Arquitectura poco modificable, reusable, portable

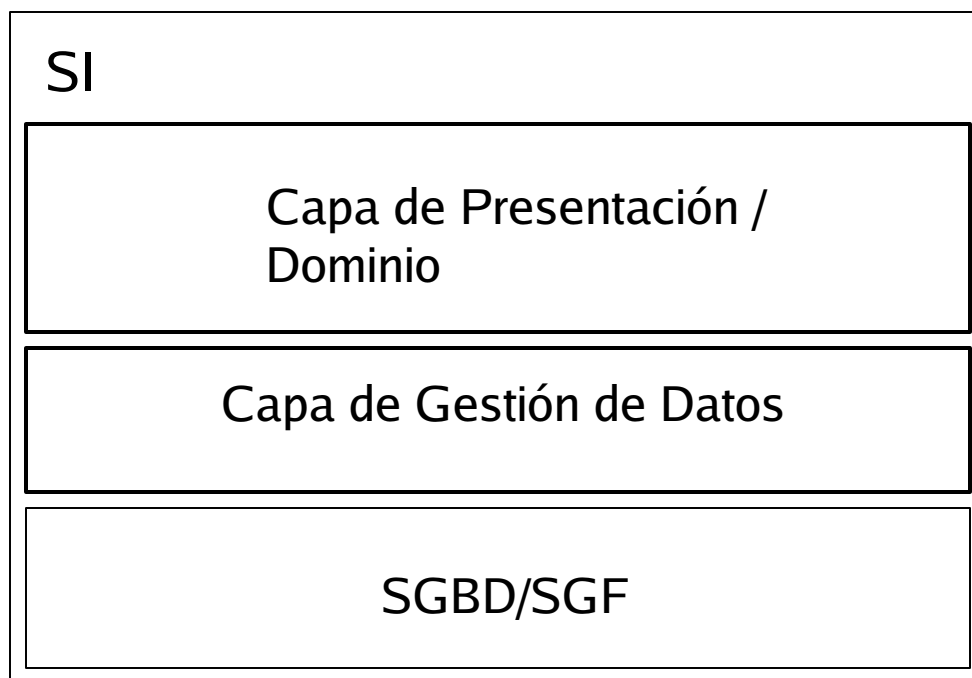
## Arquitectura lógica de un SI en dos capas (1)



- Interfaz portable, cambiable, reusable (cambios locales)
- Funcionalidades poco portables, cambiables y reusables (dependencia de la SGBD)



## Arquitectura lógica de un SI en dos capas (2)

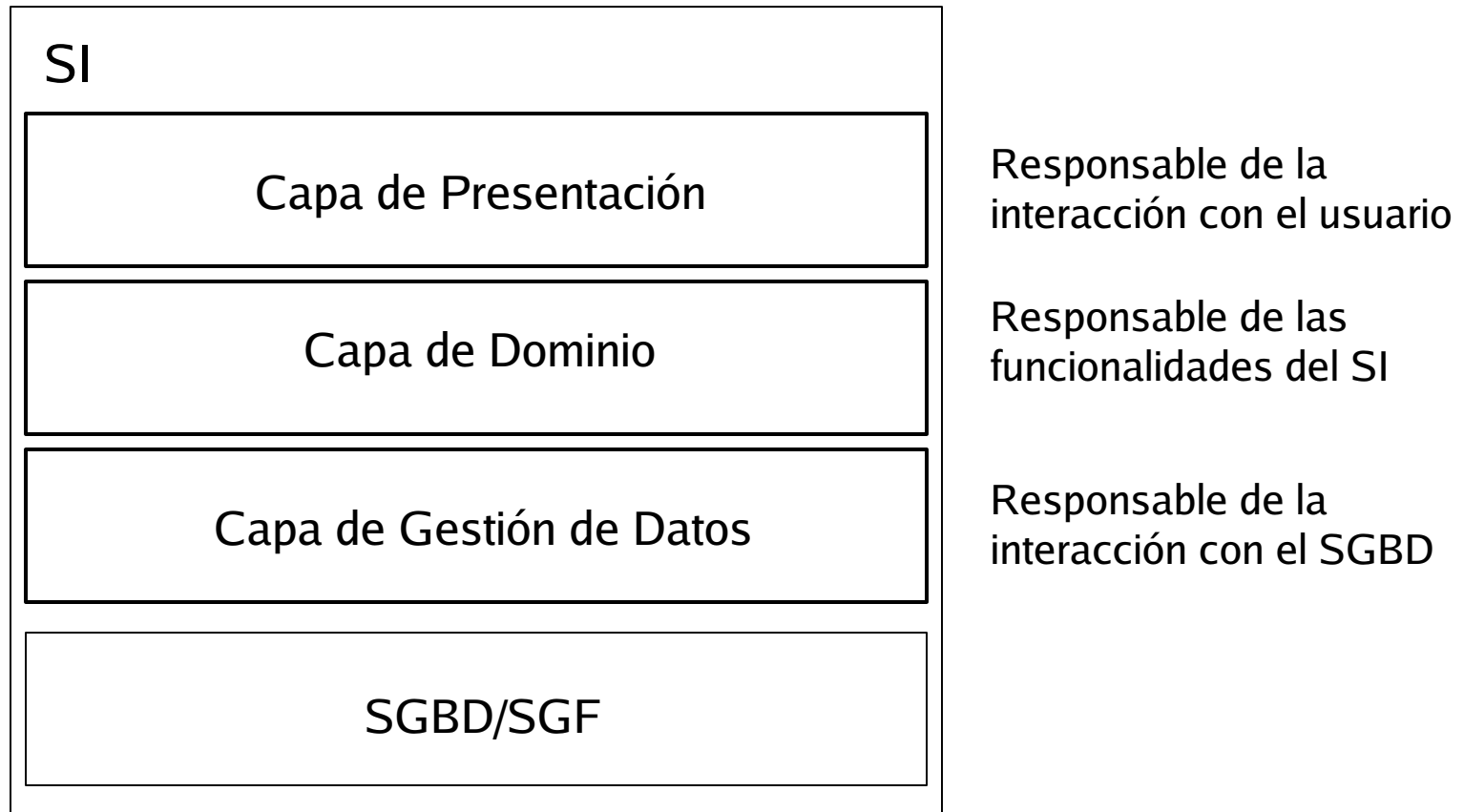


Responsable de la interacción con el usuario y de las funcionalidades del SI

Responsable de la interacción con la SGBD

- Funcionalidades poco portables, cambiables y reusables (dependencia del GUI y el diseño de las pantallas)
- Acceso a los datos portables, cambiables y reusables

## Arquitectura lógica de un SI en tres capas (1)



- Arquitectura cambiabile, reusable, portable

## Arquitectura lógica de un SI en tres capas (2)

- La capa de presentación conoce cómo presentar los datos a los usuarios, pero ignora que transformaciones debe hacer para dar respuesta a los usuarios
  - Tramita las peticiones de los usuarios
  - Tratamiento de: ficheros, diálogos, menús, botones, listados, ...
- La capa de dominio conoce cómo satisfacer al usuario, pero ignora cómo se almacenan los datos y cómo deben presentarse al usuario
  - Recibe las peticiones, consulta los datos, los trata y devuelve los resultados
- La capa de gestión de los datos conoce dónde y cómo se almacenan los datos pero no sabe cómo tratarlos, ni presentarlos a los usuarios
  - Interacciona con los SGBDs: representación persistente del estado del dominio

## Patrones arquitectónicos

- A menudo, los SI combinan dos o más estilos arquitectónicos
  
- Actualmente los SI siguen:
  - Patrón arquitectónico en 3 capas
    - presentación, dominio y gestión de datos
  - Patrón OO
    - dentro de cada capa
  - Patrón modelo – vista – controlador
    - en la capa de Presentación