FACULTAD DE INGENIERIA

Fundamentos de Ingeniería de Software

Arquitectura de software

María de las Nieves Freira

Universidad de la República

Temario

- Definición
- Diseño de la arquitectura
- Ventajas de explicitar la arquitectura
- Evaluación de arquitecturas
- Modelos de arquitectura
- Decisiones de diseño
- Reutilización
- Patrones y estilos

Arquitectura de software (I)

Especificación de requisitos de software (SRS)

En este camino hay mucho por hacer.

¿Comenzamos a programar para termina

¿Cuáles serían los riesgos?

Sistema instalado y funcionando

Arquitectura de software (2)

Especificación de requisitos del software (SRS)

No es un proceso en cascada

- Arquitectura de software
- Diseño detallado
- Implementación
- Verificación

Sistema instalado y funcionando

Arquitectura de software (3)

Los sistemas complejos están compuestos de sub-sistemas que interactúan bajo el control de un diseño de sistema.

Arquitectura de software

- Los subsistemas que componen el sistema,
- las interfaces y
- las reglas de interacción entre ellos.

Diseño de la arquitectura

- Es una etapa temprana del proceso de diseño del sistema.
- Representa la asociación entre la especificación y los procesos de diseño.
- A menudo se lleva a cabo en paralelo con algunas actividades de especificación.
- Comprende la identificación de los principales componentes del sistema y sus relaciones.

¿Qué tan fácil es modificarla? (I)

¿Qué tan fácil es modificarla? (2)

- Me gustaría que el ascensor quedara del otro lado.
- Estaría bárbaro que el puente estuviera 23 pisos más arriba, la vista sería mejor.

Ventajas de explicitar la arquitectura

- Comunicación con *stakeholders*
- La arquitectura puede ser usada como foco de discusión de los *stakeholders*.
- Análisis del sistema
  - Sirve para ver si se pueden satisfacer los requisitos no funcionales del sistema.
- Reutilización a gran escala
  - La arquitectura puede ser reutilizable a través de una gama de sistemas.
  - Se pueden desarrollar arquitecturas de líneas de productos.

Evaluación de arquitecturas

- Cambiar la arquitectura de un producto ya construido requiere mucho esfuerzo.
- Entonces, es importante evaluar la arquitectura antes de implementarla completamente.
- Verificar los requisitos de calidad establecidos .
- Evaluaciones a posteriori resultan útiles como forma de aprendizaje y estudio de posibilidades de mejora, p. ej. para una nueva versión del producto.

Modelos de arquitectura

- Facilitar la discusión sobre el diseño del sistema.
  - Una visión de alto nivel de la arquitectura de un sistema es útil para la comunicación con los *stakeholders* del sistema y la planificación de proyectos.
- Documentar el diseño de una arquitectura
  - El objetivo es producir un modelo completo del sistema que muestre los diferentes componentes en del sistema, sus interfaces y sus dependencias.

Modelo 4+1 (I)

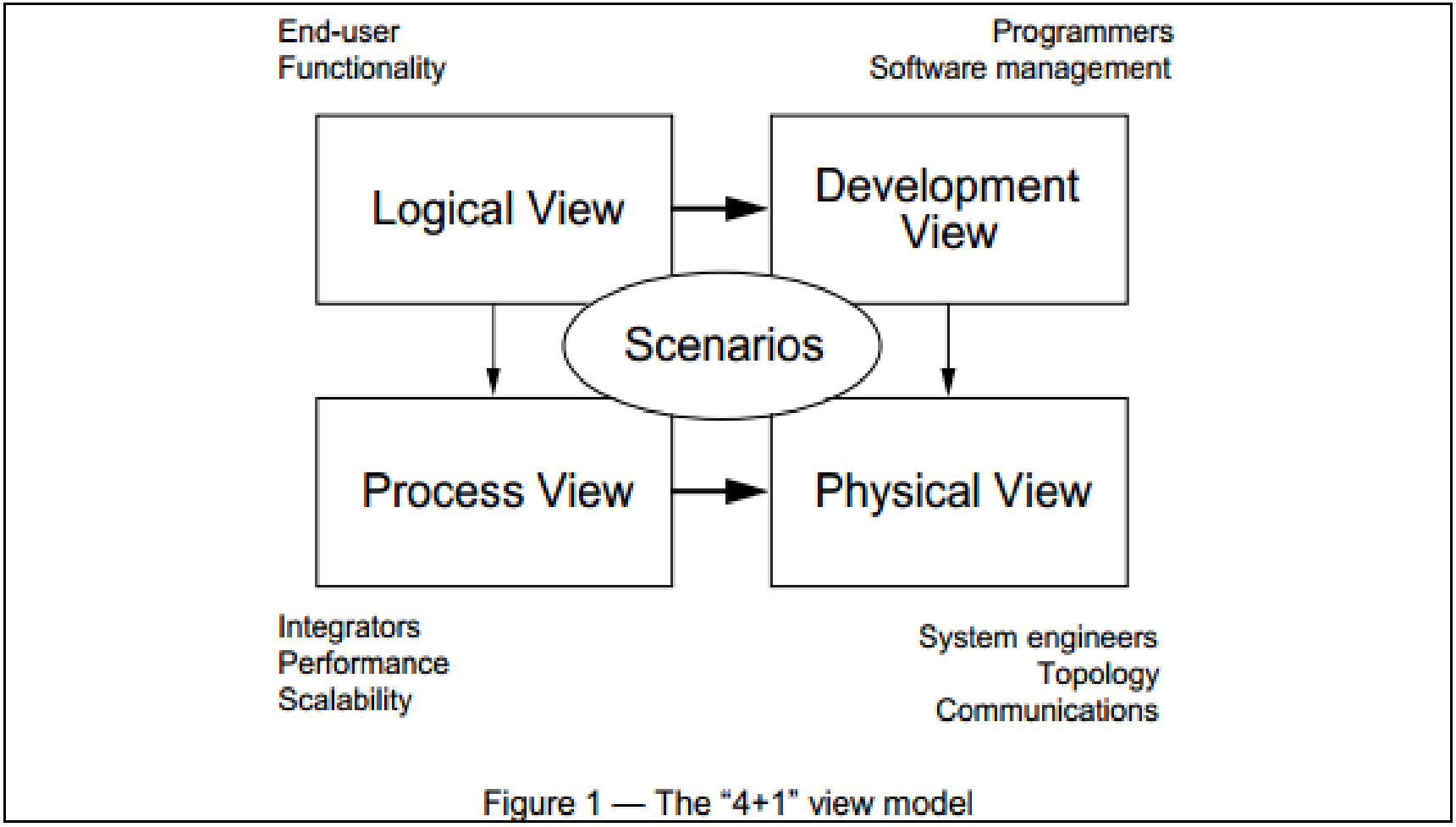


Figure 1 — The "4+1" view model

Modelo 4+1 (2)

- **Vista lógica**: abstracciones claves en el sistema como objetos o clases de objetos.
- **Vista del proceso**: interacción de procesos en tiempo de ejecución.
- **Vista de desarrollo**: como el software se descompone para el desarrollo.
- **Vista física**: hardware del sistema y como los componentes de software son distribuidos.
- Relacionados con los **casos de uso o escenarios** (+1)

Decisiones de diseño de arquitectura (I)

- El diseño de la arquitectura es un **proceso creativo,** por lo que el proceso varía según el sistema que va a ser desarrollado.

- Sin embargo, hay una **serie de decisiones comunes** a todos los procesos de diseño, y estas decisiones afectan las características no funcionales del sistema.

Decisiones de diseño de arquitectura (2)

- ¿Existe una **arquitectura genérica** que se pueda utilizar?
- ¿Cómo va a ser **distribuido** el sistema?
- ¿Qué **estilos** de arquitectura son **apropiados**?
- ¿Se **descompondrá** en módulos?
- ¿Cómo va a ser **evaluado** el diseño de la arquitectura?
- ¿Cómo debe ser **documentada** la arquitectura?

Reutilización de la arquitectura

- Generalmente los sistemas que pertenecen al **mismo dominio** tienen **arquitecturas similares** que reflejan los conceptos del dominio.
- La arquitectura de un sistema puede ser diseñada en torno a uno o más **patrones de arquitectura o «estilos»**
- Capturan la esencia de una arquitectura y pueden ser instanciados de diferentes maneras.

¿Qué afecta y qué la determina?

- La arquitectura de software afecta
  - el desempeño
  - la seguridad (*security* y *safety*)
  - la disponibilidad
  - la mantenibilidad
  - …
- Depende fuertemente de los requisitos no funcionales.

Conflictos entre soluciones

- El sistema debe tener «muy» buen desempeño y ser «muy» mantenible.
- ¿Cuál es el conflicto al momento de elegir el estilo arquitectónico?
- ¿Cómo se puede solucionar?
- Solución de compromiso.
- Diferentes estilos para distintas partes del sistema.

Patrones de software

- Propósito
  - Compartir una solución probada, ampliamente aplicable a un problema particular de diseño.
  - El patrón se presenta en una forma estándar que permite que sea fácilmente reutilizado.
- Cinco piezas importantes de un patrón
  - Nombre
  - Contexto
  - Problema
  - Solución
  - Consecuencias (positivas y negativas)

Estilos, patrones e *idioms*

- Los patrones de diseño se agrupan en tres tipos:
  - **Estilos arquitectónicos:** Soluciones de organización a nivel del sistema.
  - **Patrones de diseño:** Soluciones a problemas detallados de diseño de software.
  - ***Idioms*:** Soluciones útiles para problemas específicos en algún lenguaje de programación.

Estilos arquitectónicos

- Un **estilo arquitectónico**
  - expresa un esquema de **organización estructural** para sistemas de software.
  - provee un conjunto de tipos de **elementos predefinidos**, especifica sus **responsabilidades** e incluye reglas y guías para organizar las **relaciones entre ellos.**

Patrones de diseño

- **Un patrón de diseño**
  - provee un esquema para **refinar** los elementos de un sistema de software o las relaciones entre ellos.
  - describe una **estructura recurrente** de elementos de diseño interconectados que soluciona un **problema general** de diseño **dentro de un contexto particular.**

Idioms

- **Un *idiom***
  - es un **patrón de bajo nivel**, específico para un **lenguaje** de programación.
  - describe **como implementar** aspectos particulares de elementos o de las relaciones entre ellos usando las características de un lenguaje particular.

Beneficios de usar estilos

- Permite seleccionar una **solución entendible y probada** a ciertos problemas, definiendo los principios organizativos del sistema.
- Al basar la arquitectura en estilos que son conocidos se **facilita comunicar** las características importantes de la misma.

Formas de usar estilos (I)

- Solución para el diseño del sistema
  - Algún estilo sirve.
  - Se adopta y se usa como una de las estructuras centrales de la arquitectura.

- Base para una adaptación
  - Algún estilo soluciona parcialmente los problemas.
  - Se puede buscar adaptar el estilo para las restricciones particulares que se tienen.

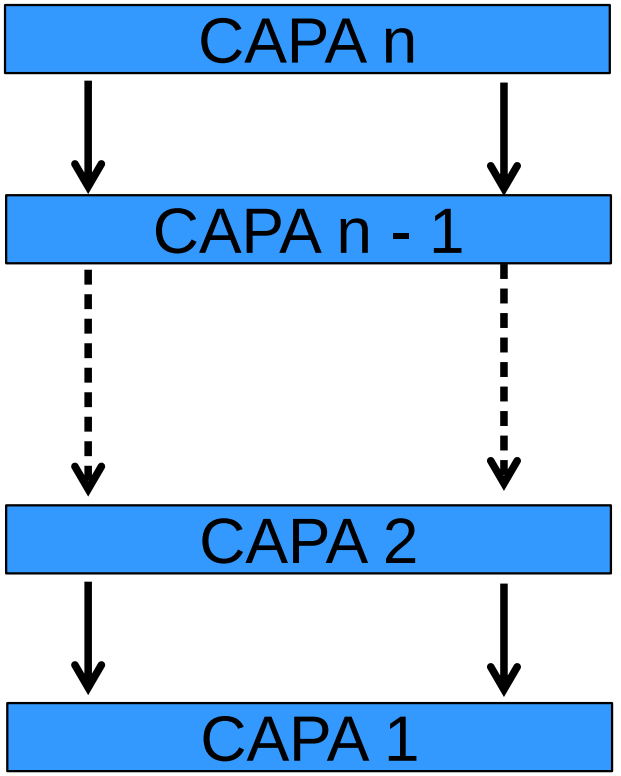
Formas de usar estilos (2)

- Inspiración para una solución relacionada
  - Ningún estilo sirve.
  - Sin embargo, entender problemas que solucionan ciertos estilos puede llevar a entender mejor el problema actual.
  - Entonces, se puede encontrar una solución que de alguna forma está relacionada con estilos existentes.

- Motivación para un nuevo estilo
  - El problema no es atacado por ningún estilo encontrado.
  - Es una buena oportunidad para encontrar una solución general al problema y crear un nuevo estilo.

Estilo: capas jerárquicas

- El sistema se organiza en capas.
- Cada una **provee un conjunto de servicios** a las capas superiores y **requiere servicios** de las inferiores.



Estilo: cliente-servidor (I)

- El sistema se descompone en **servicios** y sus servidores asociados, y en **clientes** que acceden y usan dichos servicios.
- Estilo compuesto por:
  - Servidores que ofrecen servicios.
  - Clientes que usan servicios ofrecidos por los servidores.
  - Una red que permite que los clientes accedan a los servidores (si clientes y servidores no están en una misma máquina).
- Los clientes conocen a los servidores pero no a otros clientes, y los servidores no tienen porqué conocer a los clientes.

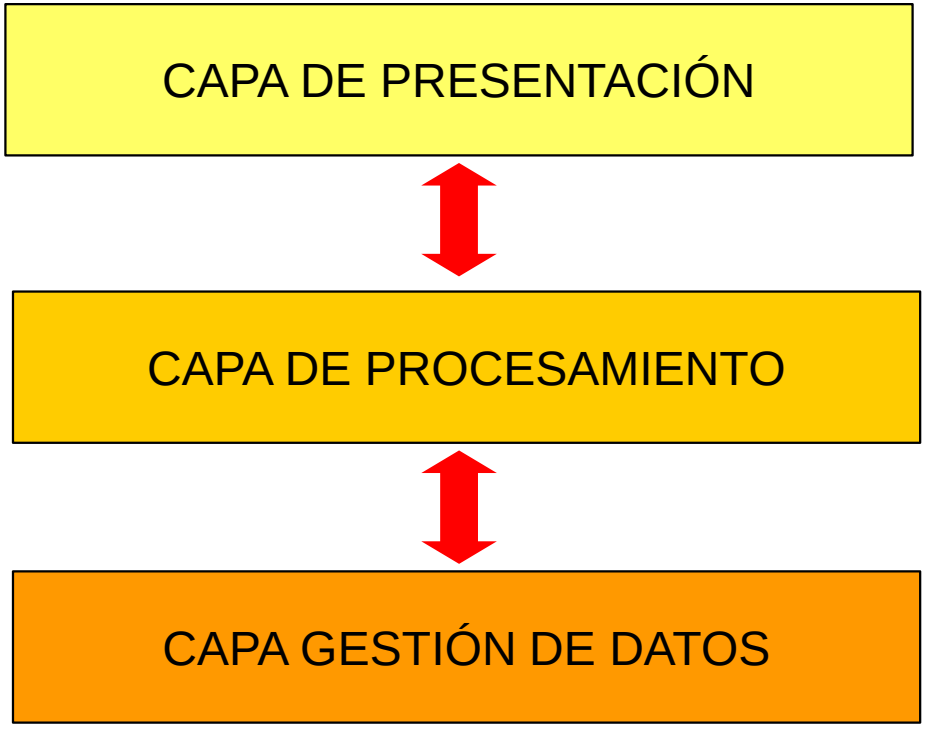
Estilo: cliente-servidor (2)

C_i = clientes

S_i = servidores

Estilo: cliente-servidor (3)

- El diseño de sistemas cliente-servidor debería reflejar la estructura lógica de la aplicación.
- Una forma de mirar a una aplicación es la siguiente:



Estilo: cliente-servidor (4)

- Capa de presentación
  - Presentación de información al usuario e interacción con este.
- Capa de procesamiento
  - Implementación de la lógica de la aplicación
- Capa gestión de datos
  - Operaciones de bases de datos y archivos
- En sistemas distribuidos es importante distinguir claramente esta separación ya que es posible distribuir cada capa en computadoras diferentes

Estilo: cliente-servidor en 2 niveles

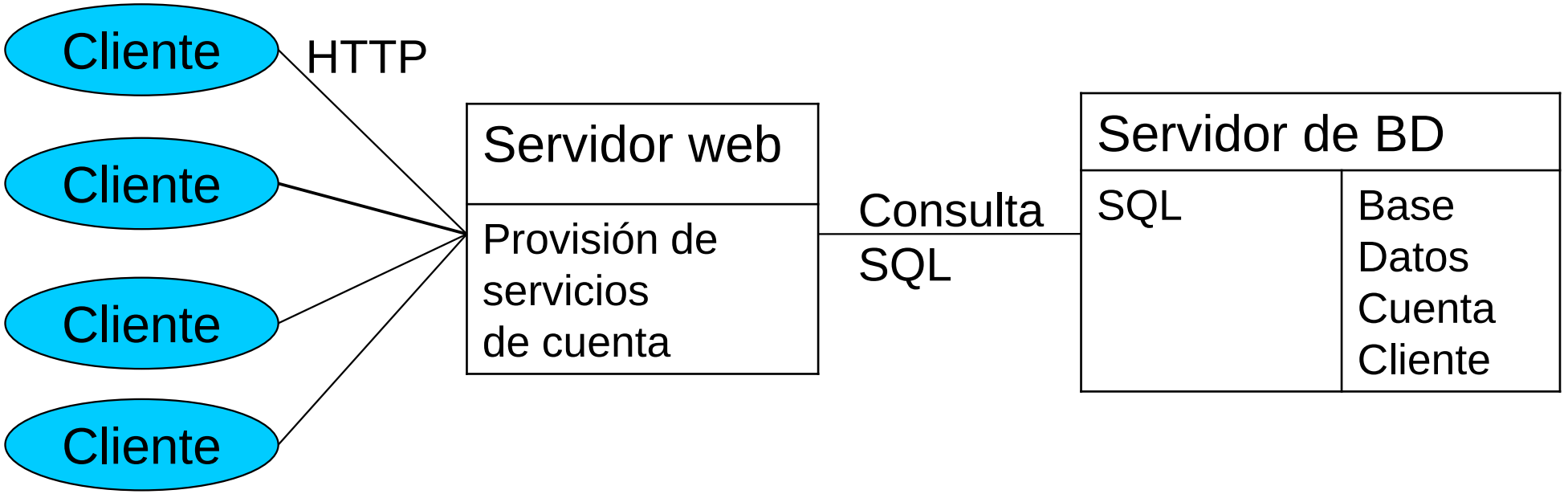
- Es la arquitectura más simple cliente-servidor.
- Varios clientes y un servidor (o varios servidores idénticos).
- 2 formas diferentes:
  - Cliente fino

    El procesamiento de la información y la gestión de los datos ocurre en el servidor. El cliente sólo es responsable de ejecutar el software de presentación.
  - Cliente grueso

    El servidor es responsable sólo de la gestión de los datos. El cliente ejecuta el procesamiento de la aplicación (la lógica de la aplicación) y la presentación.

Estilo: cliente-servidor en 3 niveles

- La presentación, la lógica y los datos se separan como procesos lógicos diferentes distribuidos en distintas máquinas
- Ejemplo: sistema bancario por internet



Estilo: peer-to-peer

- Sistemas descentralizados donde las computaciones pueden ser realizadas en cualquier nodo de la red
  - En principio no hay distinción entre clientes y servidores
  - El sistema se diseña para usar el poder de almacenamiento y procesamiento de múltiples computadoras en una red
- Ejemplos:
  - Sistemas para compartir archivos
  - Mensajería instantánea

Estilo: service oriented architectura (SOA)

- Organizaciones que quieren hacer su información accesible a otros programas pueden hacerlo definiendo y publicando una interface de servicio web
- Un **servicio web** es una representación estándar para algún recurso computacional o de información que puede ser usado por otros sistemas
- El servicio es independiente de las aplicaciones que lo usan
- Se pueden construir aplicaciones mediante el uso de distintos servicios

Próxima clase:

Diseño de software