FACULTAD DE INGENIERIA

Ingeniería de Software

Diseño de Software
Clase 2

Sommerville 10 (9*) — Capítulo 6
Learning UML 2.0 – Capítulos 12 y 15

Arquitectura de Software

Especificación de Requerimientos del sistema

En este camino hay mucho por hacer.

¿Comenzamos a programar para terminar lo antes posible?

- ¿Cuáles serían los riesgos?

Sistema instalado y funcionando

Arquitectura de Software

Especificación de Requerimientos del sistema

- Arquitectura de software
- Diseño detallado
- Implementación
- Verificación
- Liberación

No es un proceso en cascada.

Sistema instalado y funcionando

Arquitectura - ¿qué tan fácil es modificar… ?

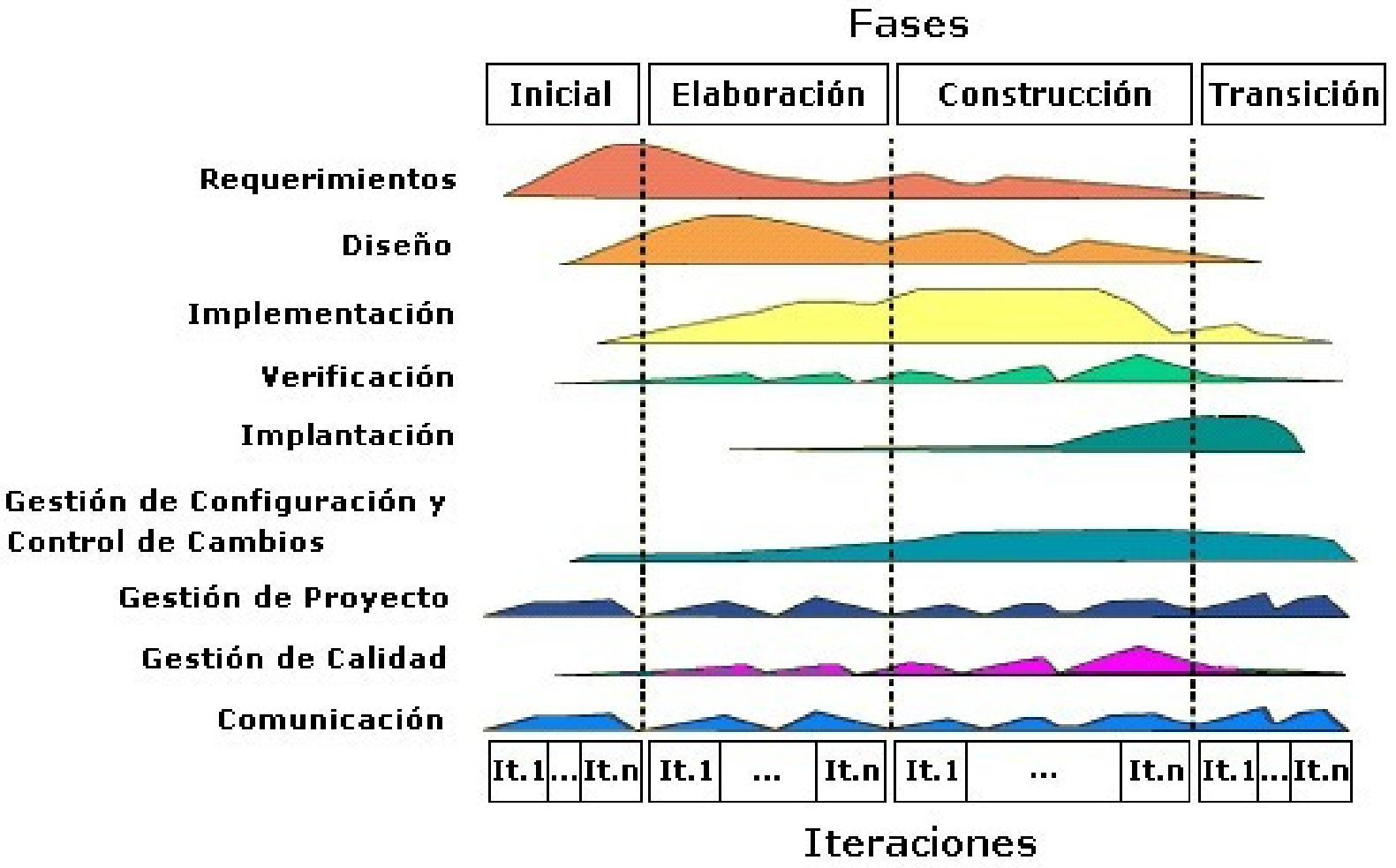
Diseño Arquitectónico

- Cómo debe organizarse un sistema y cómo tiene que diseñarse la estructura global de ese sistema.
- Salida → modelo arquitectónico que describe la forma en que se organiza el sistema como un conjunto de componentes en comunicación.
- Usualmente no resulta exitoso el desarrollo incremental de arquitecturas.

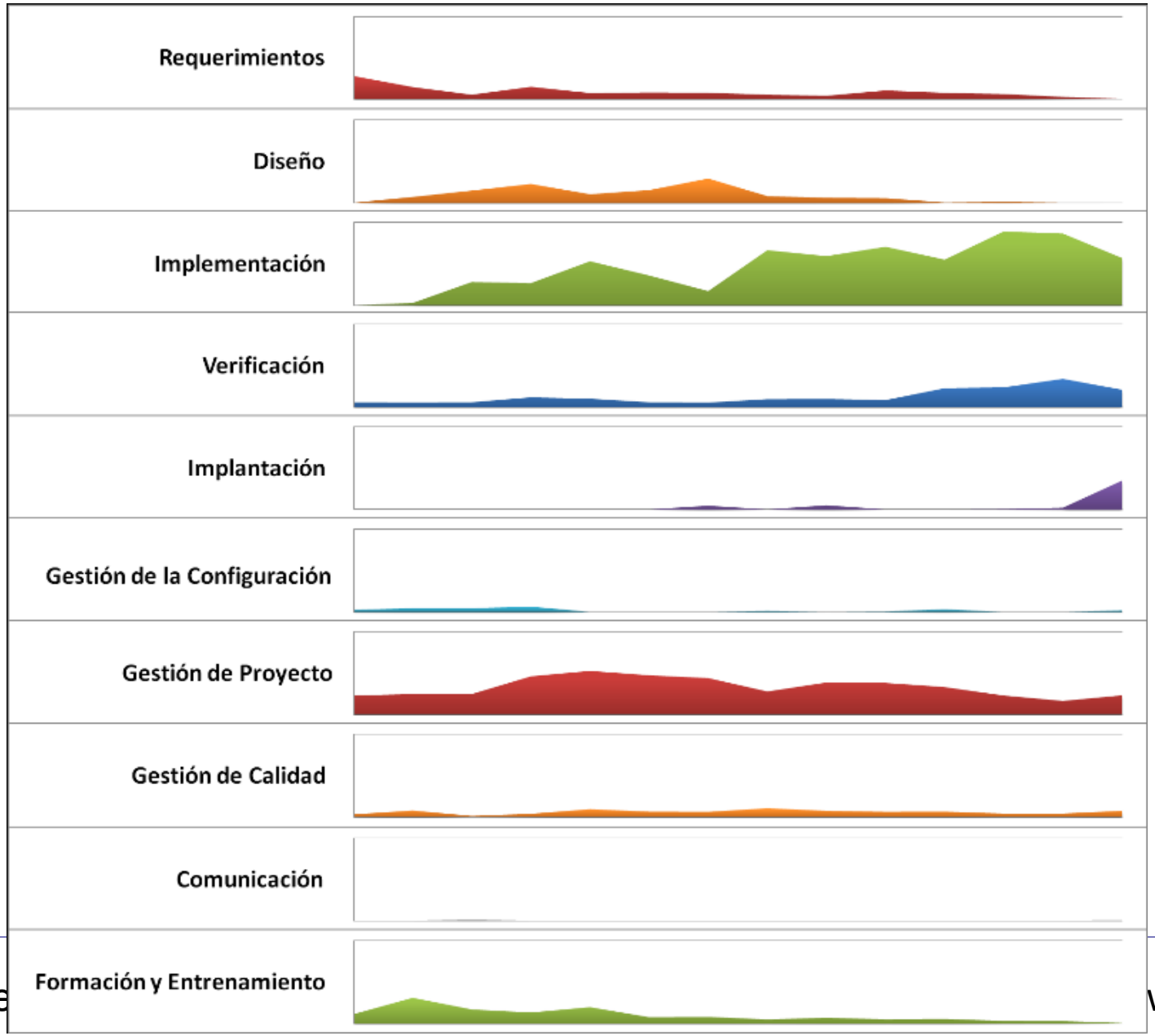
Diseño Arquitectónico

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

Diseño Arquitectónico



Ejemplo grupo de PIS 2015



Diseño de Software

- **Arquitectura en alto nivel** se refiere a la arquitectura de sistemas empresariales complejos que incluyen otros sistemas, programas y componentes de programas.
- **Arquitectura detallada** tiene que ver con la arquitectura de los programas individuales. En este nivel nos preocupamos de que los programas individuales sean descompuestos en componentes.

Ventajas de explicitar la arquitectura

- Comunicación con los participantes
  - Los modelos facilitan la discusión
- Análisis del sistema
  - Requisitos funcionales
  - Plan de diseño para negociar los requisitos
- Reutilización a gran escala

Decisiones en el diseño arq.

- Proceso creativo → decisiones en lugar de actividades

- ¿Existe una arquitectura genérica que se pueda usar?
- ¿Cómo va a ser distribuido el sistema?
- ¿Qué patrones o estilos son apropiados?
- ¿Se descompondrá en módulos?
- ¿Cómo va a ser evaluado el diseño de la arquitectura?
- ¿Cómo se documentará la arquitectura del sistema?

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 mas **patrones de arquitectura o "estilos"***

  * Capturan la esencia de una arquitectura y pueden ser instanciados de diferentes maneras.

¿Qué afecta y determina de la arquitectura?

- El estilo y la estructura de la arquitectura dependerán de:
  - Rendimiento
  - Seguridad
  - Protección
  - Disponibilidad
  - Mantenibilidad

  → Depende fuertemente de los requisitos no funcionales.

Conflictos entre soluciones

- Hay conflictos potenciales entre las soluciones. ¿Qué hacer?
  - Soluciones de compromiso
  - Diferentes estilos para distintas partes

- Evaluar un diseño arquitectónico es difícil porque la verdadera prueba es qué tan bien el sistema cubre sus requisitos funcionales y no funcionales cuando está en uso.

Patrones arquitectónicos

- Un patrón es una descripción abstracta de buena práctica, que se ensayó y puso a prueba en diferentes sistemas y entornos.
- Describe una organización de sistema que ha tenido éxito en sistemas previos.
- Incluye información sobre cuándo es y cuándo no es adecuado usar dicho patrón, así como las fortalezas y debilidades del patrón.

Patrones arquitectónicos

- 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)

Beneficios de usar patrones

- 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 patrones

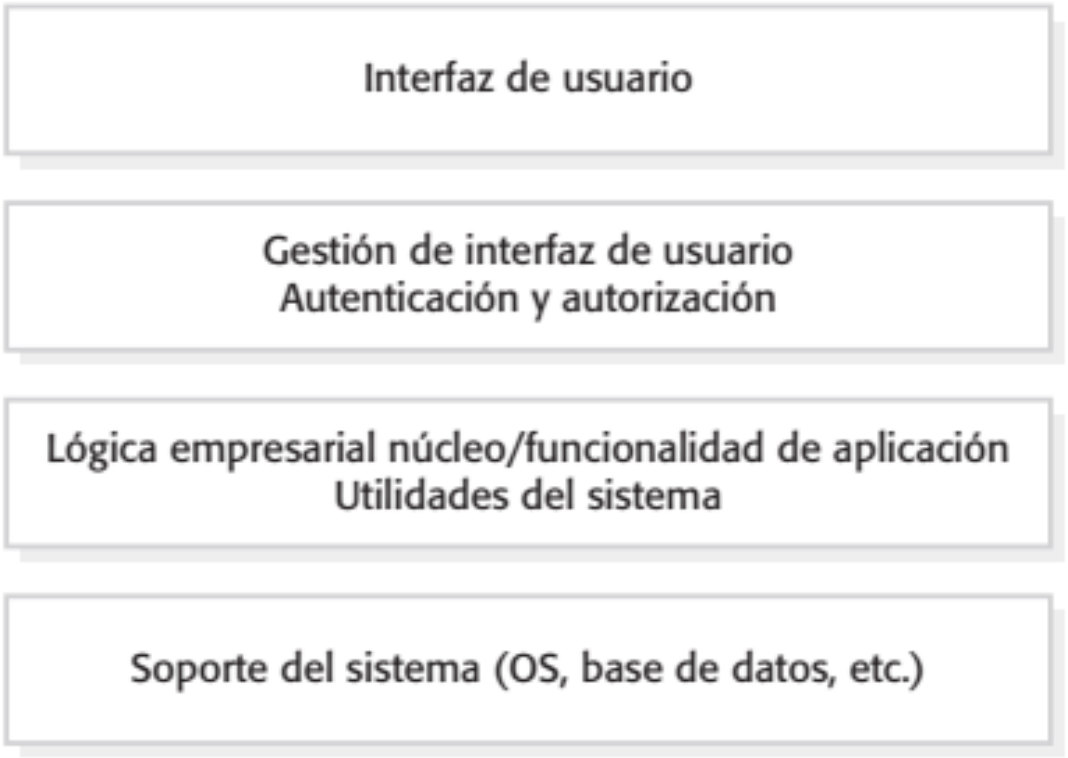
- Solución para el diseño del sistema
  - Algún estilo sirve
- Base para una adaptación
  - Algún estilo soluciona parcialmente los problemas
- Inspiración para una solución relacionada
  - Ningún estilo sirve, pero algunos ayudan a entender mejor los problemas
- Motivación para un nuevo patrón
  - Ningún estilo sirve, además encontramos una solución general al problema

→ Arquitectura en capas

- Desc → Organiza el sistema en capas con funcionalidad relacionada con cada capa. Una capa da servicios a la capa de encima.
- Cuándo → Nuevas facilidades encima de sistemas existentes; desarrollo disperso en varios equipos, y cada uno es responsable de una capa de funcionalidad; requisito de seguridad multinivel.
- Ventajas → Permite la sustitución de capas en tanto se conserve la interfaz. Aumentar la confiabilidad, en cada capa pueden incluirse facilidades redundantes (por ejemplo, autenticación).
- Desventajas → Es difícil ofrecer una separación limpia entre capas, El rendimiento suele ser un problema.

→ Arquitectura en capas

- Soporta el desarrollo incremental
- También permite implementaciones multiplataformas



→ Arquitectura de repositorio

- Desc → Todos los datos en un sistema se gestionan en un repositorio central, accesible a todos los componentes del. Los componentes no interactúan directamente..
- Cuándo → Grandes volúmenes de información o sistemas dirigidos por datos.
- Ventajas → Los componentes no necesitan conocer la existencia de otros componentes. Los cambios se propagan hacia todos los componentes. Gestión consistente y centralizada de los datos.
- Desventajas → Problemas en el repositorio afectan a todo el sistema. Posibles ineficiencias en la comunicación. Posibles dificultades al distribuir el repositorio.

→ Arquitectura de repositorio

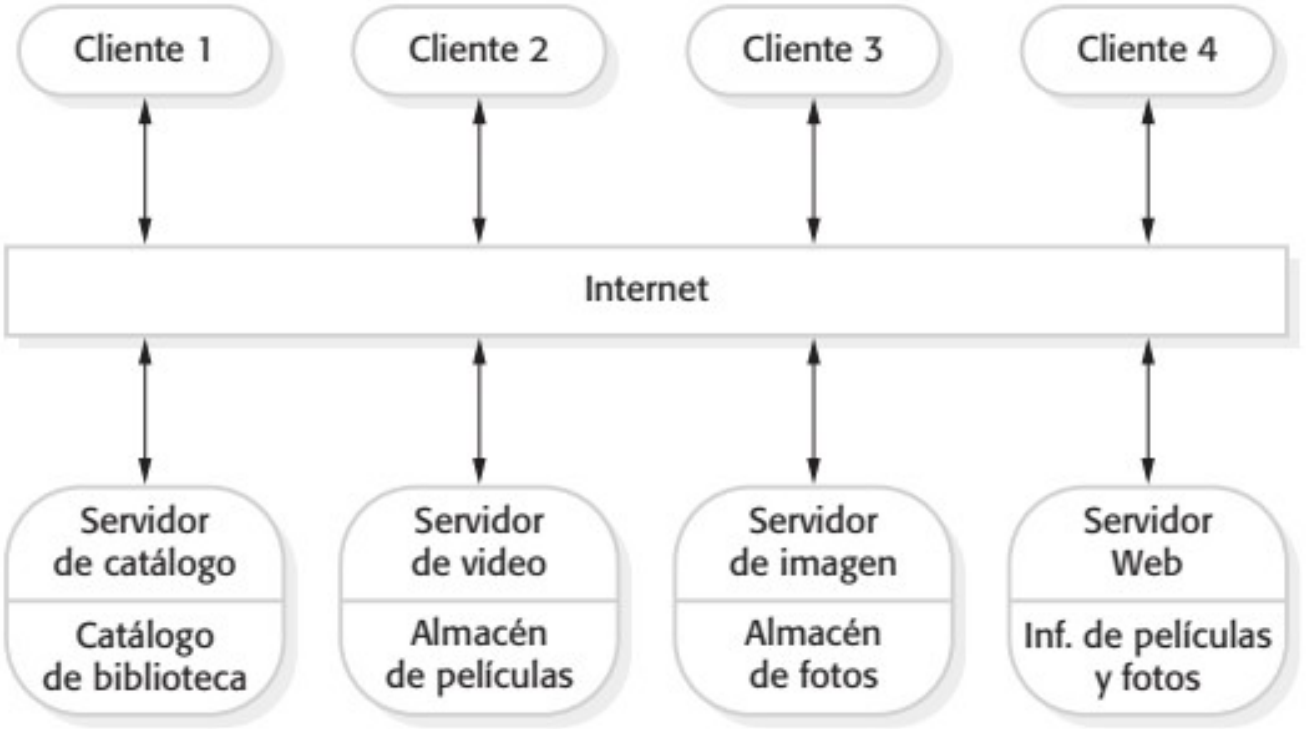
- Un enfoque alternativo es el pizarrón dónde se activan componentes cuando hay datos particulares disponibles. Adecuado para datos menos estructurados.

→ Arquitectura cliente servidor

- Desc → La funcionalidad del sistema se organiza en servicios, y cada servicio lo entrega un servidor independiente. Los clientes son usuarios de dichos servicios.
- Cuándo → Se usa cuando, desde varias ubicaciones, se tiene que ingresar a los datos en una base de datos compartida. También cuando la carga es variable (replicación de servidores).
- Ventajas → Los servidores se pueden distribuir a través de una red. La funcionalidad general estaría disponible a todos los clientes, no necesita implementarse en todos los servicios.
- Desventajas → Cada servicio son puntos de falla. El rendimiento depende de la red, así como del sistema. Posibles problemas administrativos cuando los servidores son propiedad de diferentes organizaciones.

→ Arquitectura cliente servidor

- Aunque a menudo se considera como arquitectura de sistemas distribuidos, el modelo lógico de servicios independientes que opera en servidores separados puede implementarse en una sola computadora.
- El beneficio importante es la separación e independencia.

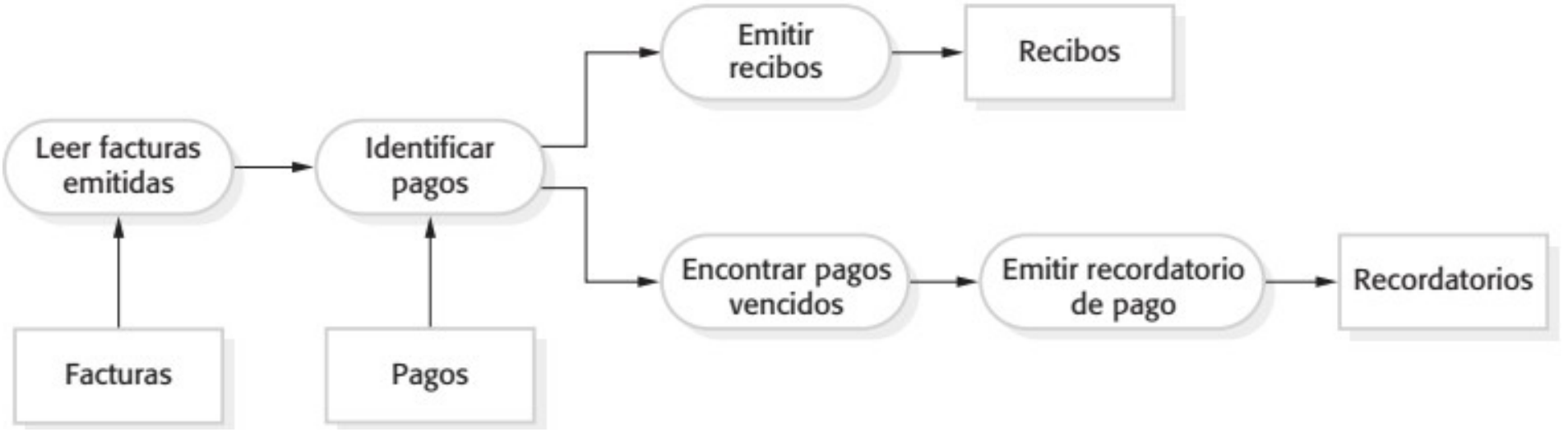


→ Arquitectura de tubería y filtro

- Desc → Cada componente de procesamiento (filtro) es discreto y realiza un tipo de transformación de datos. Los datos fluyen (como en una tubería) de un componente a otro para su procesamiento
- Cuándo → Se suele utilizar en aplicaciones de procesamiento de datos, donde las entradas se procesan en etapas separadas para generar salidas relacionadas.
- Ventajas → Fácil de entender y soporta reutilización. El estilo del flujo de trabajo coincide con la estructura de muchos procesos empresariales. La evolución al agregar transformaciones es directa. Permite concurrencia.
- Desventajas → El formato debe acordarse y respetarse. Esto aumenta la carga del sistema, y puede significar que sea imposible reutilizar transformaciones funcionales que usen otros formatos.

→ Arquitectura de tubería y filtro

- Usada para procesamiento automático de datos.
- Difícil de usar para sistemas interactivos.



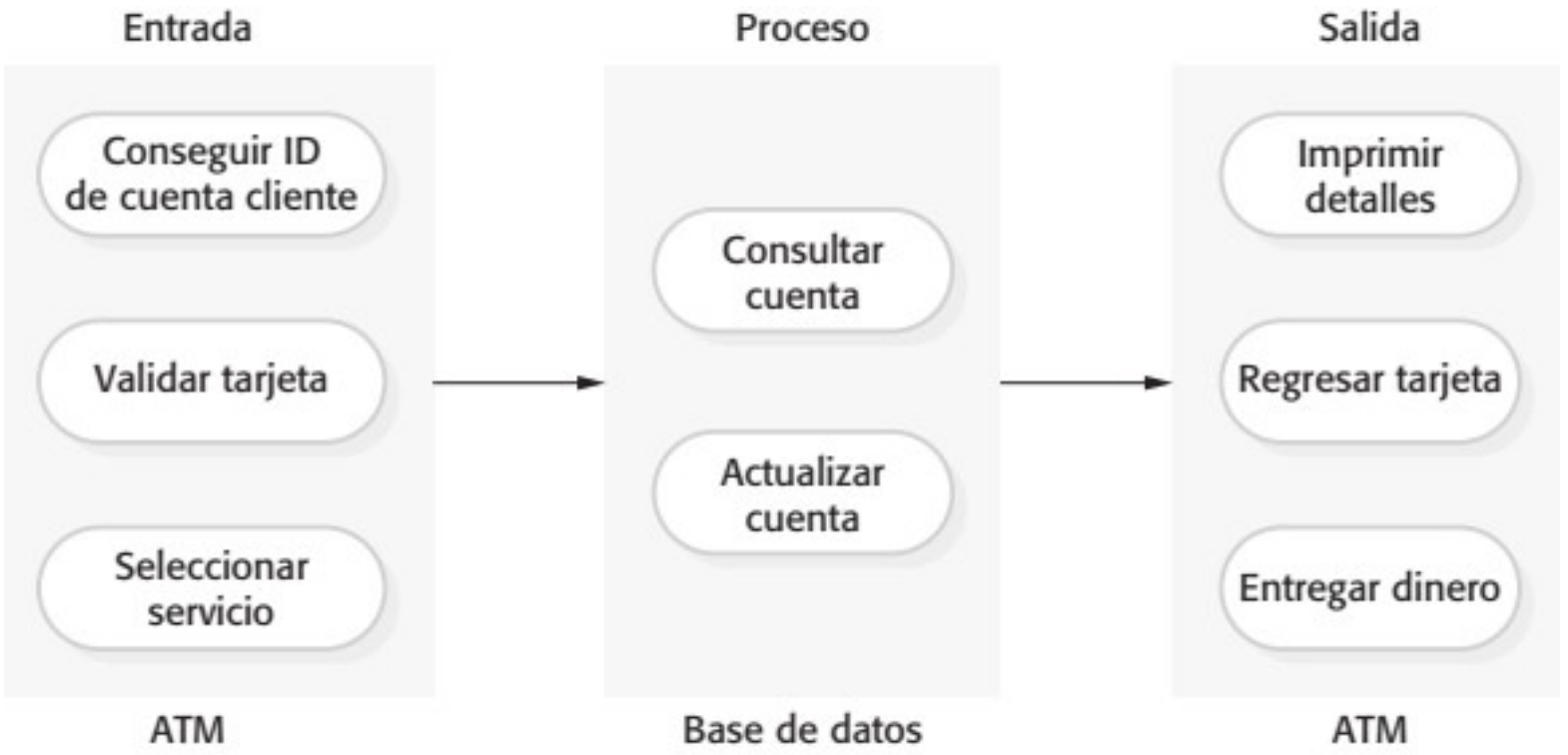
Arquitecturas de aplicación

- Encapsulan las principales características de una clase de sistemas.
- Uso de modelos de arquitecturas de aplicación:
  1) Como punto de partida para el diseño
  2) Como lista de verificación del diseño
  3) Como una forma de organizar el trabajo
  4) Como un medio para valorar componentes a reusar
  5) Como un vocabulario para hablar sobre tipos de aplicaciones

→ Sistemas de procesamiento de transacciones

- Procesan peticiones del usuario mediante la información o una base de datos. Una transacción de base de datos es una secuencia de operaciones que se trata como una sola unidad.
- En general, son sistemas interactivos donde los usuarios hacen peticiones asíncronas de servicios.
- Pueden organizarse como una arquitectura tubería y filtro.

→ Sistemas de procesamiento de transacciones

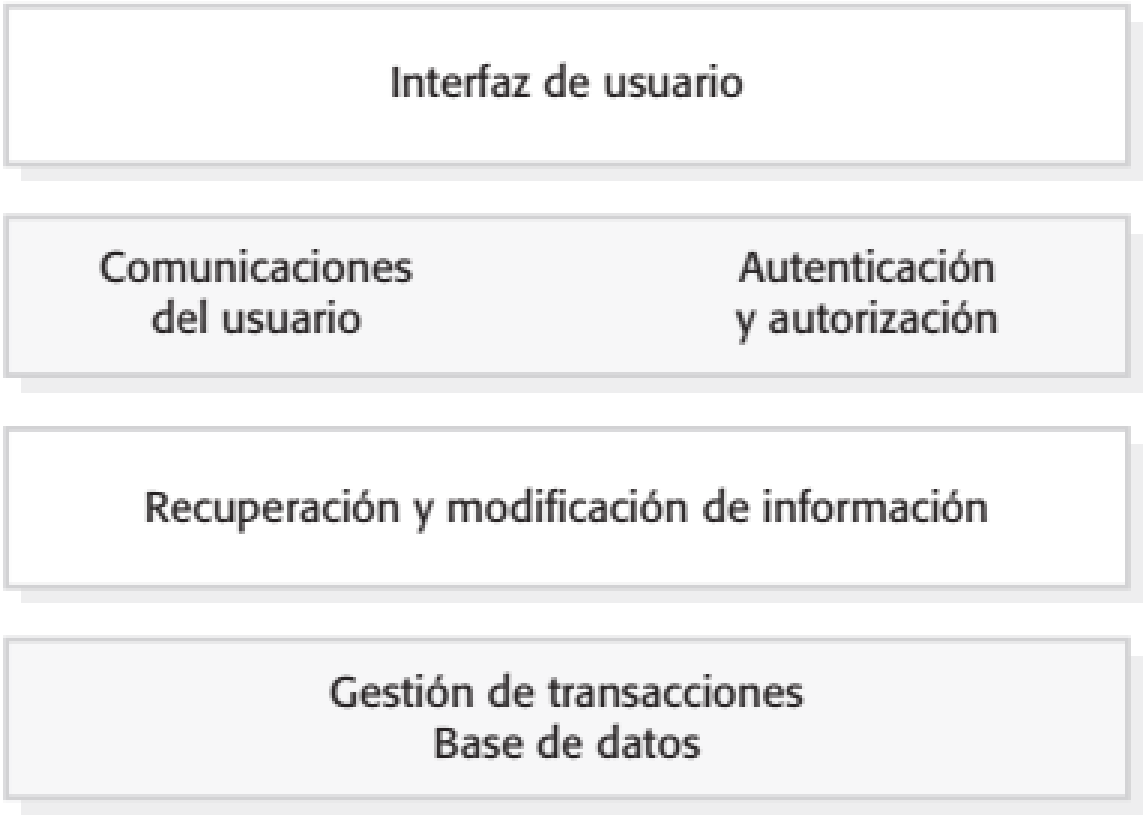


→ Sistemas de información

- Todos los sistemas que incluyen interacción con una base de datos compartida se consideran sistemas de información basados en transacciones.
- Un sistema de información permite acceso controlado a una gran base de información, tales como un catálogo de biblioteca, un horario de vuelos o los registros de pacientes en un hospital.
- Cada vez más, los sistemas de información son sistemas basados en la Web, cuyo acceso es mediante un navegador Web.

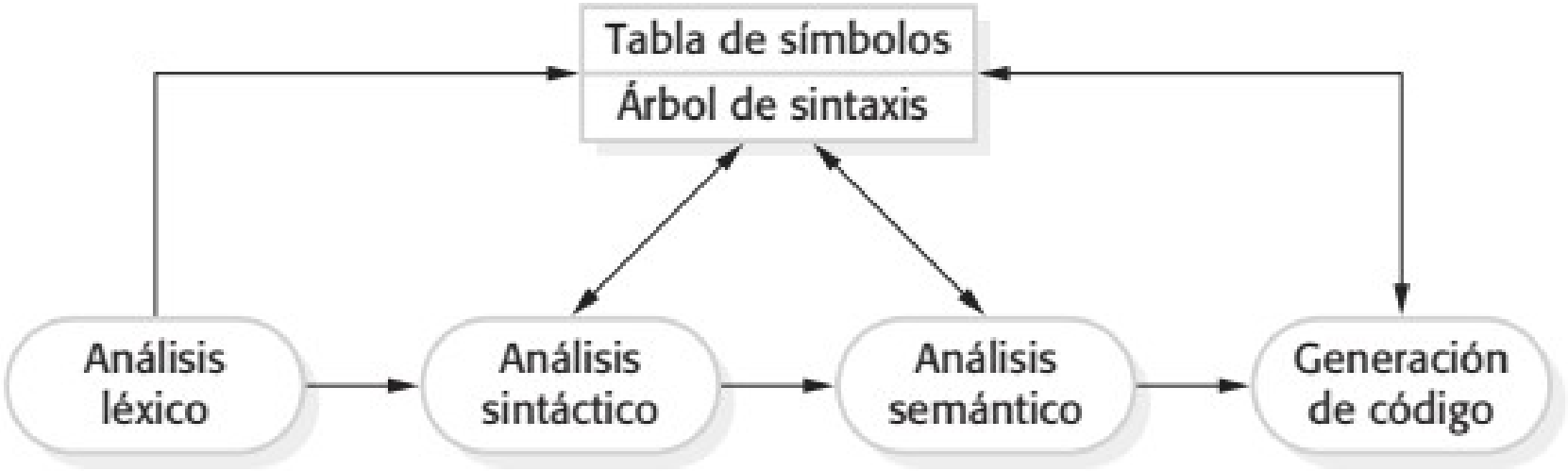
→ Sistemas de información

- Un modelo muy general para estos sistemas se basa en capas.



→ Sistemas de procesamiento de lenguaje

- Convierten un lenguaje natural o artificial en otra representación del lenguaje y, para lenguajes de programación, también pueden ejecutar el código resultante.
- Para compiladores (entornos más batch) puede utilizarse una composición de un repositorio con tubería y filtro.



→ Sistemas de procesamiento de lenguaje

- Cuando es más interactivo puede ser más efectivo utilizar un repositorio.

Sistemas de procesamiento de lenguaje

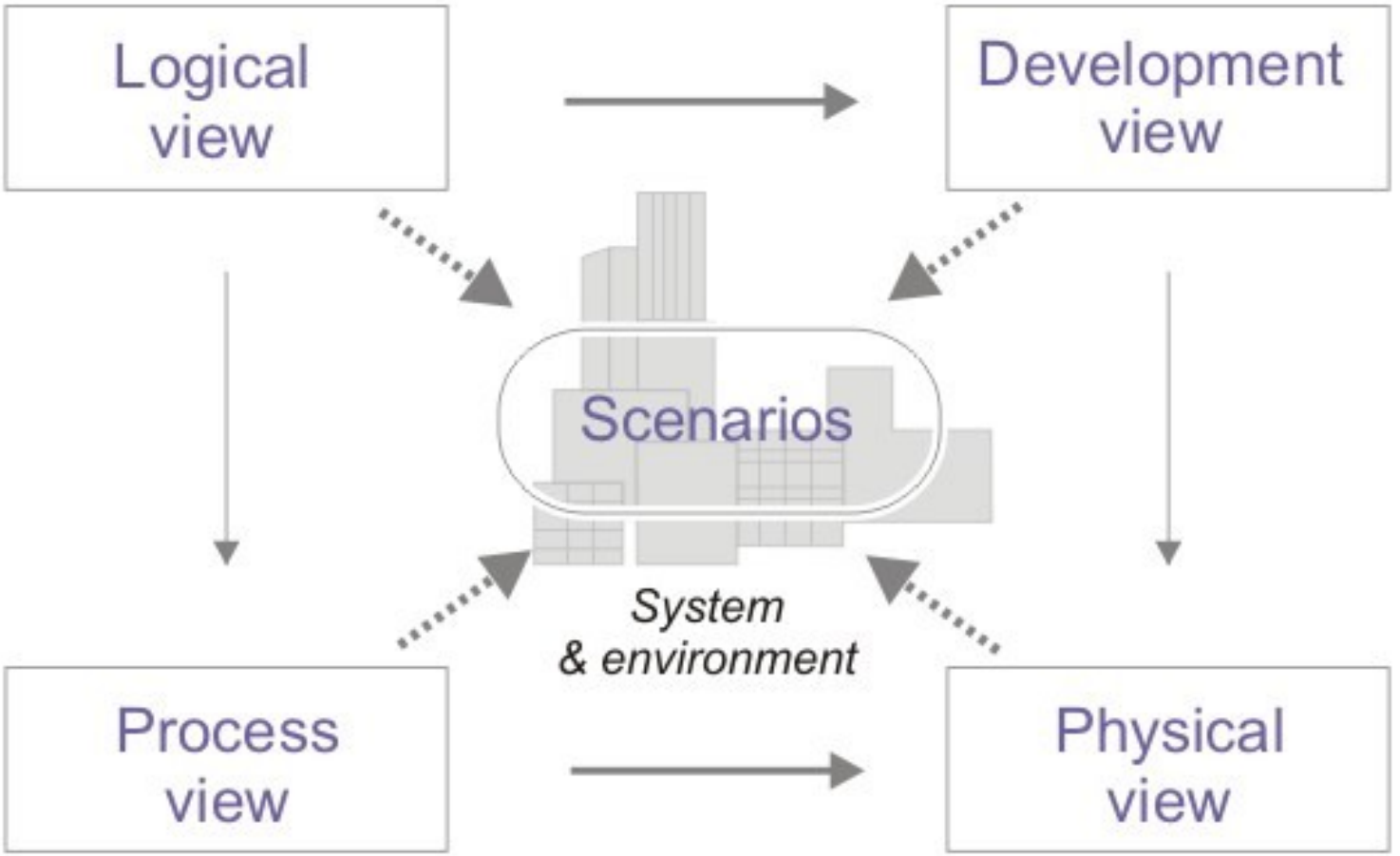
- Cuando es más interactivo puede ser más efectivo utilizar un repositorio.

Vistas arquitectónicas

- ¿Qué vistas o perspectivas son útiles para diseñar y documentar una arquitectura?
- ¿Qué notaciones deben usarse?

Vistas arquitectónicas

- Vistas – Diferentes opiniones
  - Modelo 4+1



Modelo 4+1

- **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)

Vistas arquitectónicas

- Notaciones - Diferentes opiniones
  - Cuando se usa UML se hace informalmente
  - Algunos autores proponen notaciones rápidas e informales
  - Otros lenguajes basados en componentes y conectores

→ Desarrollar las vistas que sean útiles para la comunicación sin preocuparse por completitud a no ser que sea un sistema crítico.

Vistas arquitectónicas

- Vista de casos de uso/escenarios → casos de uso
- Vista de procesos → andariveles, diag de actividad, bpmn
- Vista lógica → clases, secuencia, actividad
- Vista de desarrollo → diagrama de componentes
- Vista física → diagrama de despliegue (deploy)

Diagrama de componentes

- Un componente es una parte encapsulada, reusable y reemplazable del software – bloques de construcción.
- Pueden ser de tamaño chico (clase) hasta grande (subsistema).

Diagrama de componentes

- Se puede indicar interfaces requeridas y provistas.

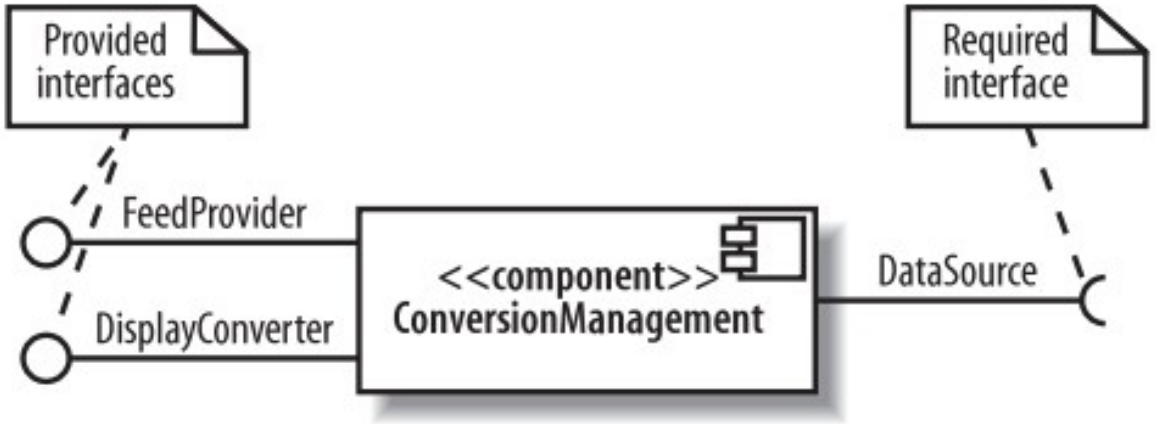


Diagrama de despliegue

- Muestra cómo el software se asigna al hardware y cómo se comunican las distintas piezas.
- Cada nodo representa una pieza de hardware.
- Dentro de cada nodo se pueden colocar artefactos de software los cuales ejecutarán en esa pieza de software
- Saliendo un poco de UML es frecuente colocar componentes (grandes) en los nodos para mostrar cómo será desplegado el software.

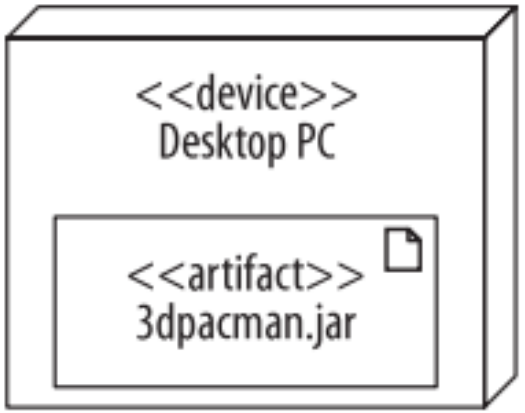


Diagrama de despliegue

- Se muestra la comunicación entre nodos con una línea sólida agregando un estereotipo indicando la forma de comunicación (ej, protocolo).