Ingeniería de Software II

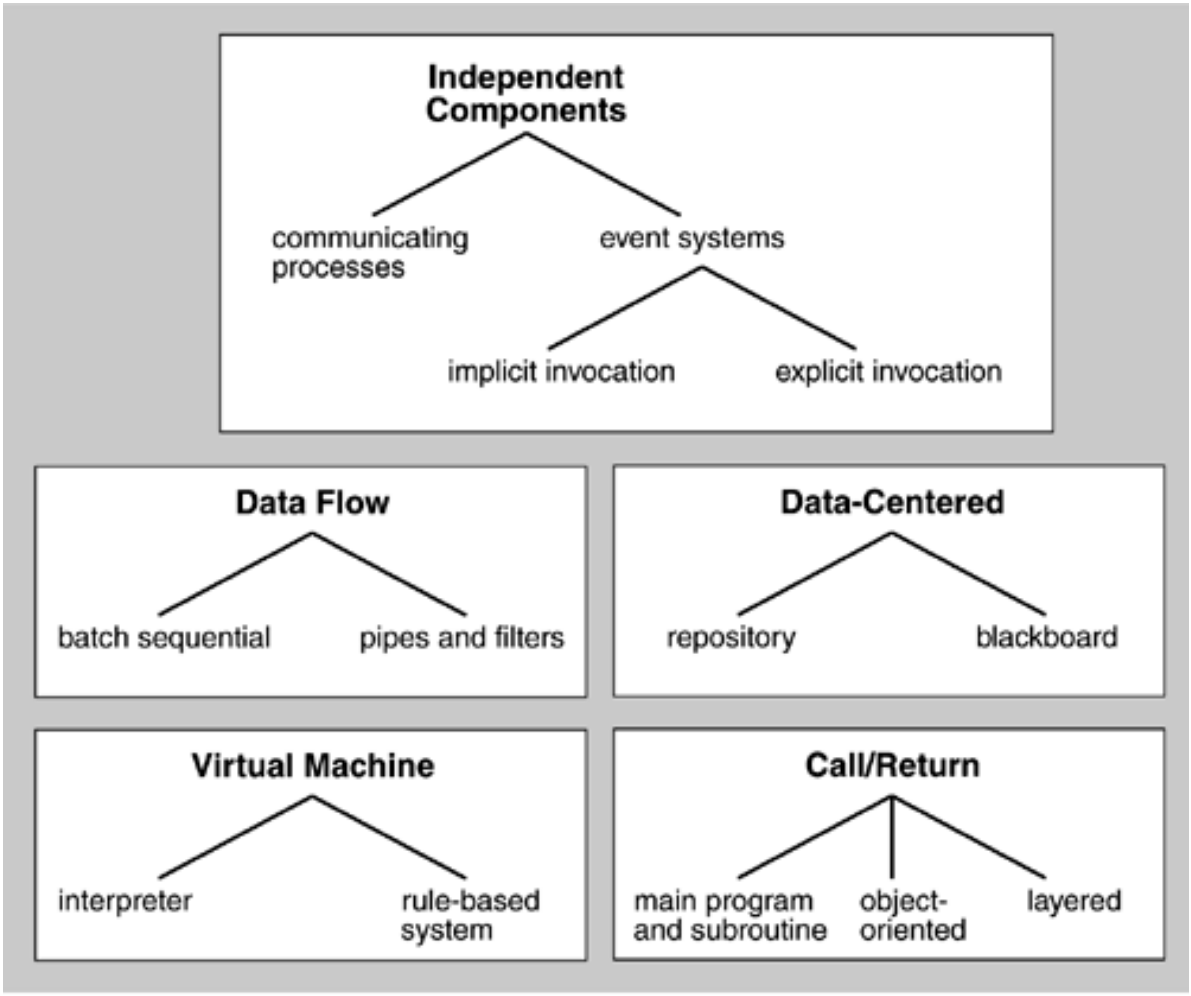
Ingeniería de Software II

Segundo Cuatrimestre de 2011

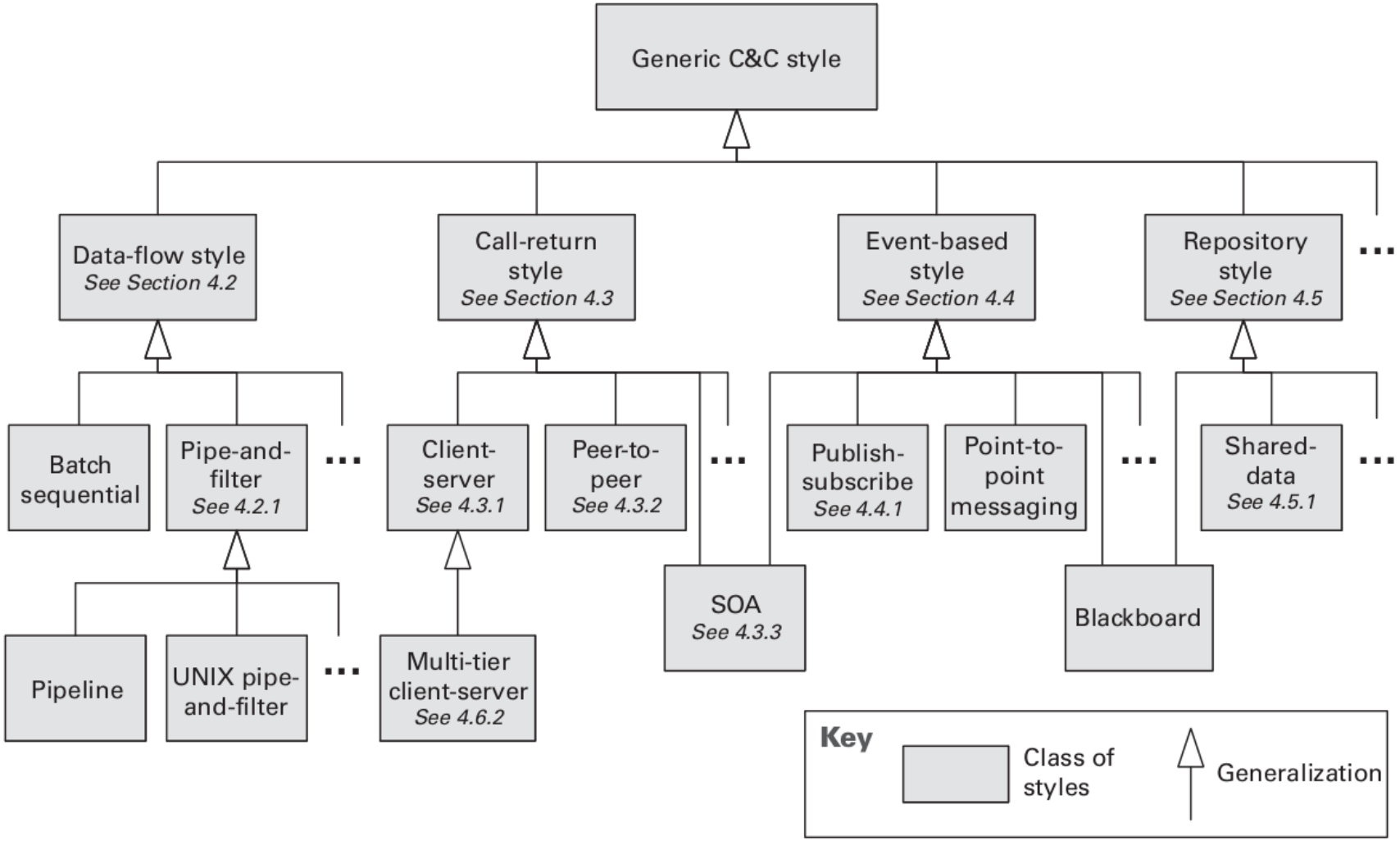
Clase 15: Estilos Arquitectónicos y Viewtypes

Buenos Aires, 17 de Octubre de 2011

Taxonomía de estilos arquitectónicos (Shaw, Garlan)



Refinamiento según estructura – C&C

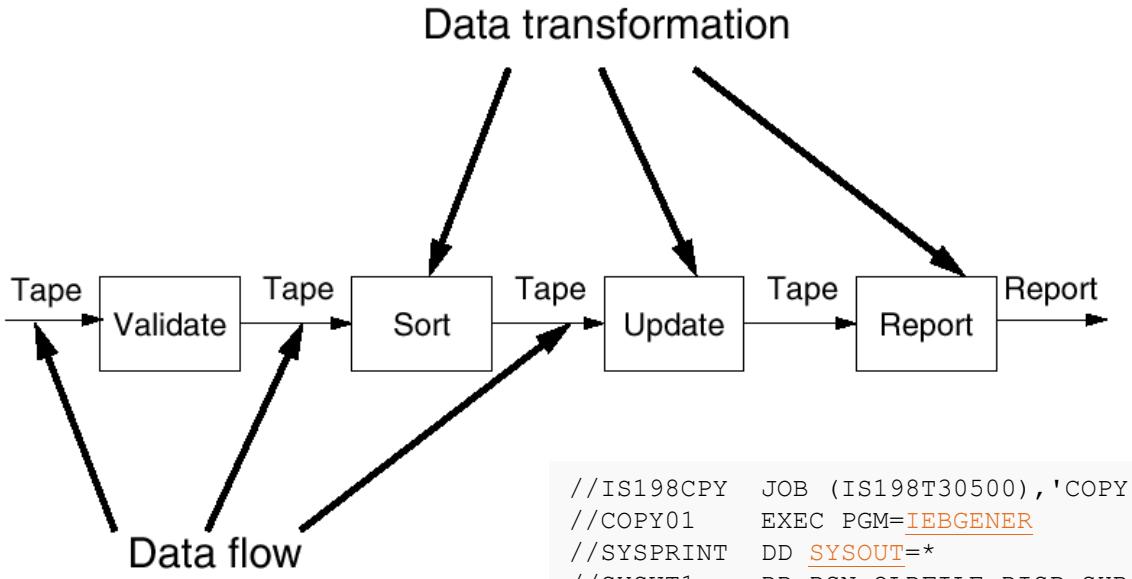


Arquitecturas tipo “Data – Flow”

- Estructura basada en transformaciones sucesivas de los datos de input
- Los datos entran al sistema y fluyen a través de los componentes hasta su destino final.
- Componentes de run-time
- Normalmente un programa controla la ejecución de los componentes (lenguaje de control)
- Dos tipos
  - Batch sequential
  - Pipe and filter

Batch Sequential

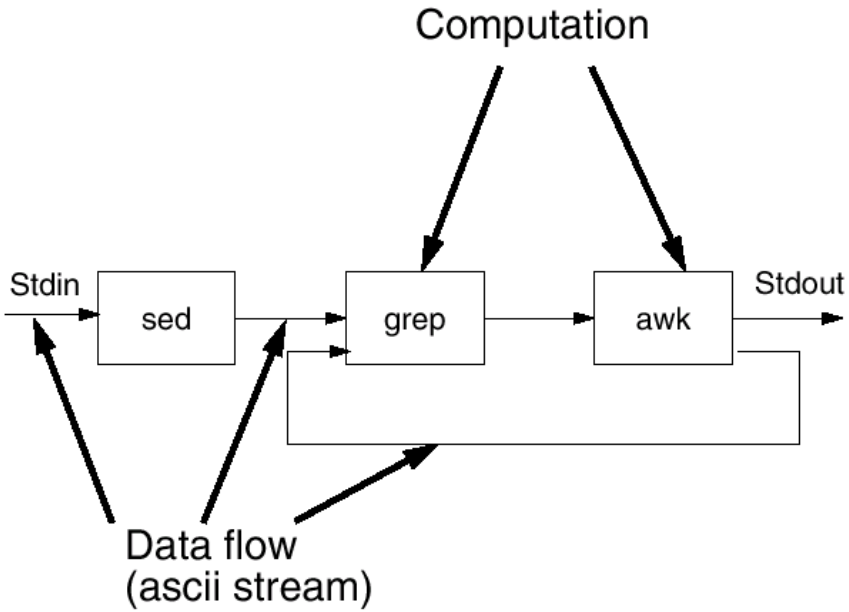
- Cada paso se ejecuta hasta ser completado, y recién después puede comenzar el siguiente paso.
- Usado en aplicaciones clásicas de procesamiento de datos

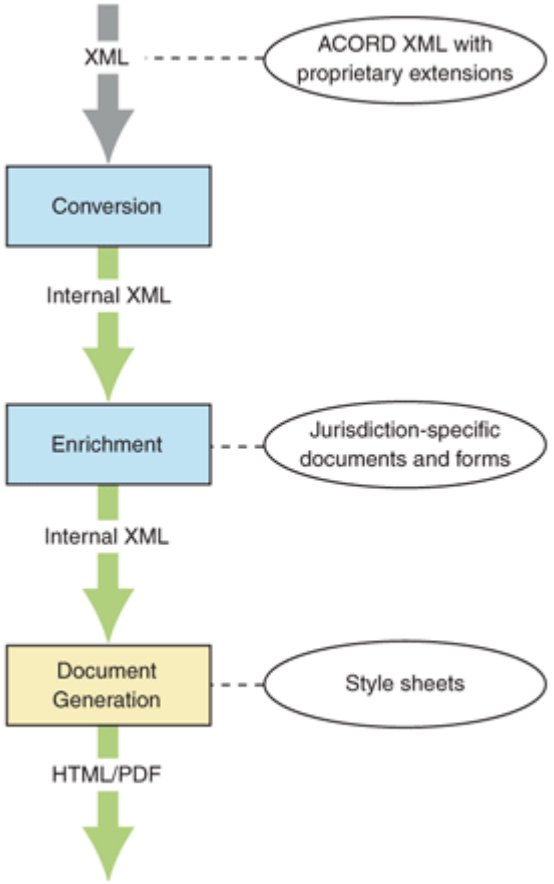



```
//IS198CPY  JOB (IS198T30500),'COPY JOB',CLASS=L,MSGCLASS=X
//COPY01    EXEC PGM=IEBGENER
//SYSPRINT  DD SYSOUT=*
//SYSUT1    DD DSN=OLDFILE,DISP=SHR
//SYSUT2    DD DSN=NEWFILE,
//             DISP=(NEW,CATLG,DELETE),
//             SPACE=(CYL,(40,5),RLSE),
//             DCB=(LRECL=115,BLKSIZE=1150) //SYSIN DD DUMMY
```

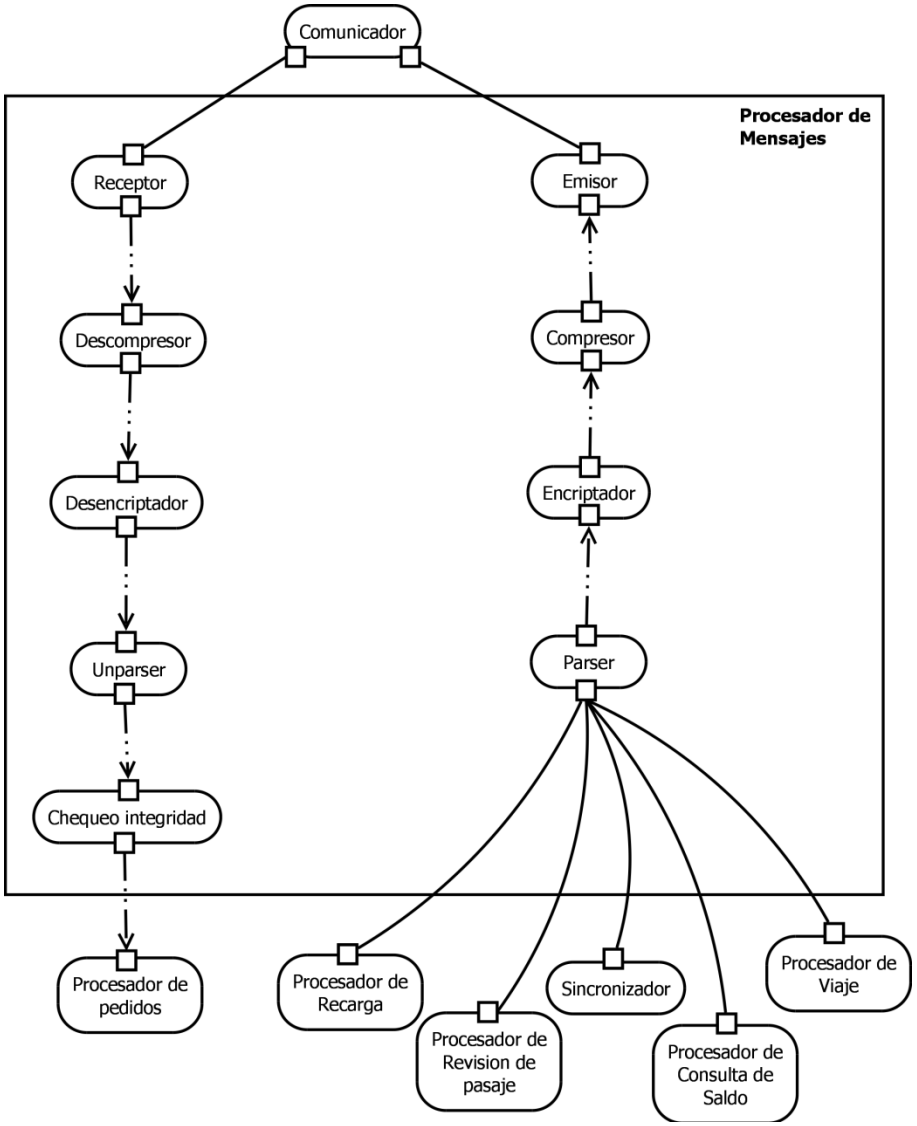
Pipes and Filters

- Cada componente tiene inputs y outputs. Los componentes leen "streams" de datos de su input y producen streams de datos en sus outputs de forma continua.
- Filters: ejecutan las transformaciones
- Pipes: conectores que pasan streams de un filtro a otro





Ejemplo SUBITE

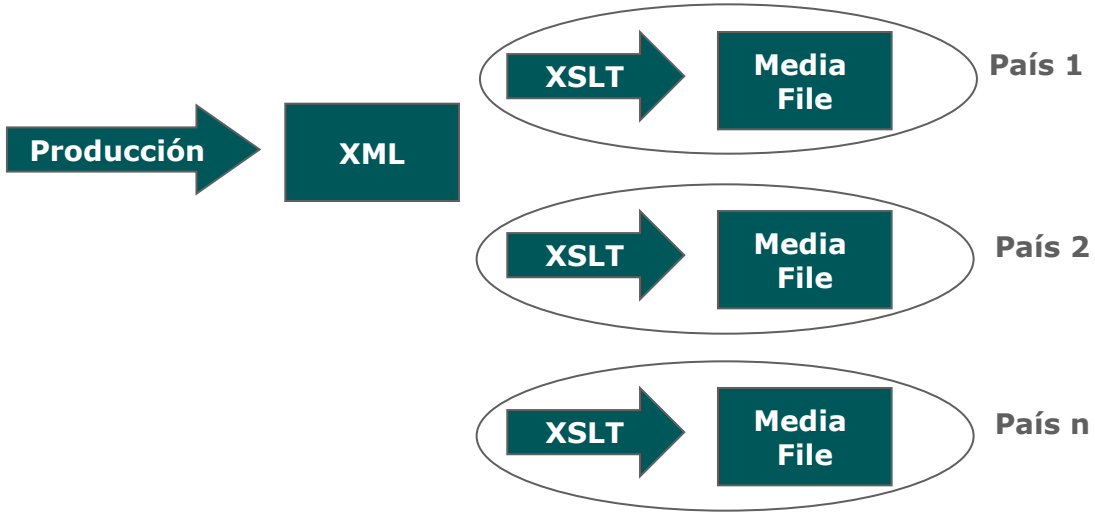


Pipes and filters – Ventajas y Desventajas

- Pros
  - Fácil de entender: la función del sistema es la composición de sus filtros
  - Facilidad de extensión – reuso – recambio de filtros
  - Posibilidad de ejecución concurrente
- Cons
  - "Mentalidad batch", difícil para aplicaciones interactivas
  - El orden de los filtros puede ser complejo
  - Overhead de parsing y unparsing
  - Problemas con tamaño de los buffers
- Extensiones: Bounded Pipes, Typed Pipes

Ejemplo de una metida de pata con este estilo

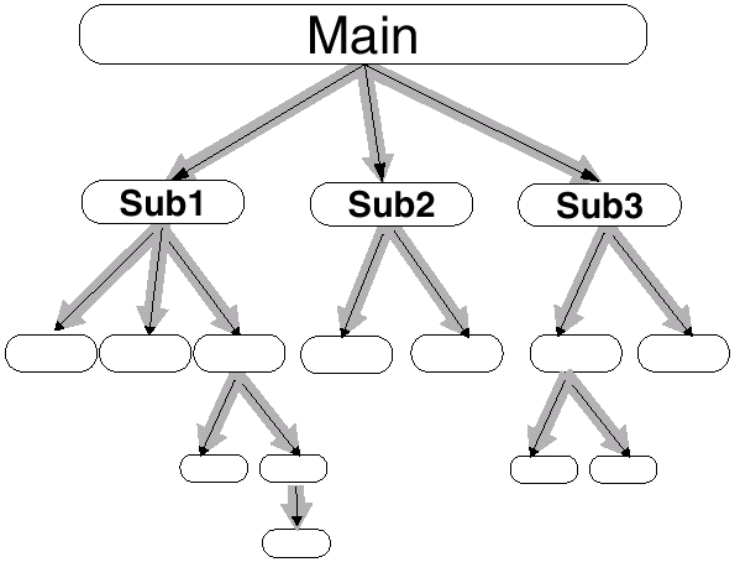
- Un sistema "Core" tiene un proceso productivo de generación "masiva" de cheques. Se va a implementar en varios países –> La flexibilidad es clave
- Proceso productivo complejo. A partir de pedidos de distintos clientes, estos se agrupan y generan un archivo para ser enviado a la imprenta
- Idea…
- ¡El "core" no se toca!



- Pero… hablando de pipes and filters… "Use this style whenever performance and scalability are not an issue"
- Al final… ¡Retrabajo!

"Call and Return"

- El estilo dominante en la mayoría de los sistemas
- Programa principal y subrutinas
  - Programación estructurada
  - Ventaja: concepto simple; desventajas: reuso limitado a funciones / procedimientos, limitada flexibilidad



Descomposición funcional
Invocación explícita y sincrónica
Jerarquía de módulos (coordinación vs. ejecución)

Call and Return (cont.)

- Arquitecturas orientadas a objetos
  - Information hiding
  - Encapsulamiento
  - Herencia, polimorfismo

- Ventajas: reuso a gran escala, todas las ventajas del encapsulamiento y el information hiding, correspondencia en los objetos del dominio con objetos del mundo real.
- Desventajas:…
- Information hiding: las decisiones de diseño que es probable que cambien son ocultadas en un módulo o un conjunto pequeño de módulos. Cada módulo tiene sus "secretos" (Parnas, 1972).

Layered o Multi - tier

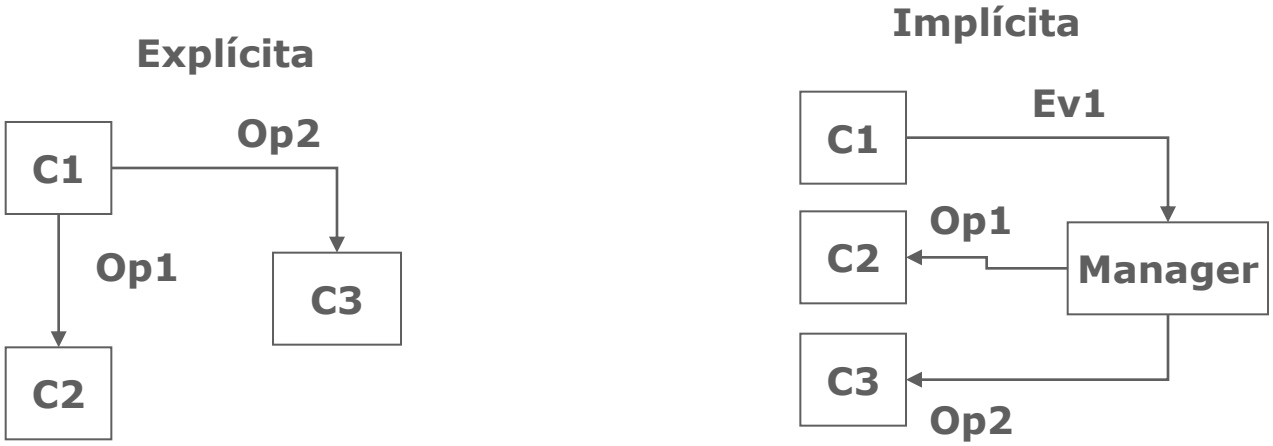
- Cada nivel provee servicios
  - Oculta en nivel siguiente
  - Provee servicios al nivel anterior
- En muchos casos el “bajar” de nivel implica acercarse al hardware o software de base
- Los niveles van formando “virtual machines”
- Ventajas: portabilidad, facilidad de cambios, reuso
- Desventajas: performance, difícil de encontrar la abstracción correcta, puede implicar salteo de niveles
- Ejemplos: arquitectura de 3 capas (presentación, reglas de negocio, datos)

Client server

- Una instancia de un estilo más genérico: sistemas distribuidos
- Los componentes son clientes (acceden a servicios) y servidores (proveen servicios)
- Los servers no conocen la cantidad o identidad de los clientes
- Los clientes saben la identidad de los servidores
- Los conectores son protocoles basados en RPC
- Distintos “flavors”

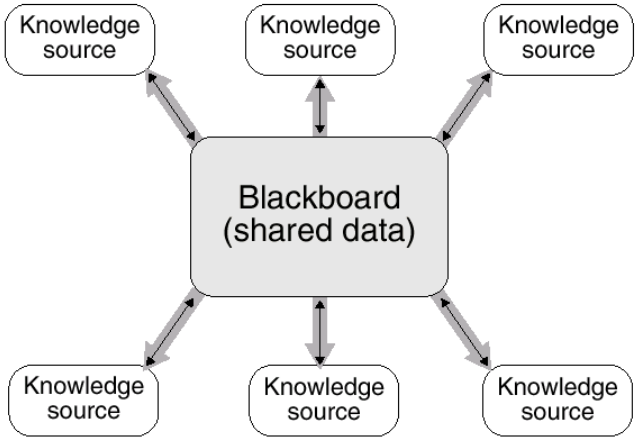
Componentes independientes

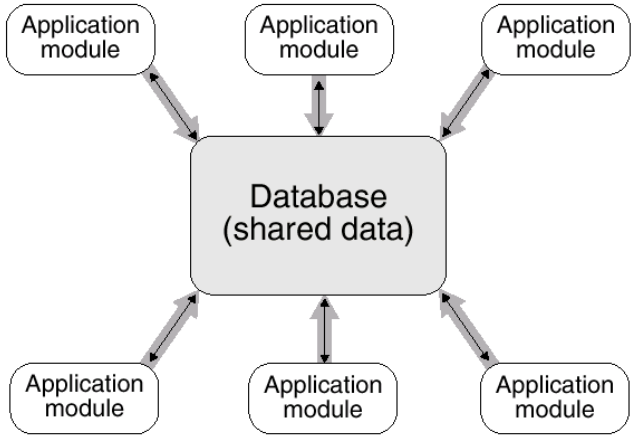
- Son arquitecturas de procesos que se comunican a través del envío de mensajes
- Componentes: procesos independientes
- Conectores: envío de mensajes
  - Sincrónico o asincrónico
- Sistemas de eventos - Invocación implícita



Arquitecturas centradas en datos

- Estructura de datos central (normalmente una base de datos) y componentes que acceden a ella. Gran parte de la comunicación está dada por esos datos compartidos.
- Normalmente presentes en todo sistema de información
- Dos tipos: blackboard y reposotorio (cada vez más "ocultos" en un esquema tipo "layered")





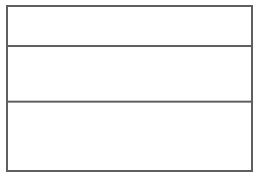
Viewtypes de una arquitectura

- Un sistema tiene varias estructuras
- Las estructuras pueden dividirse principalmente en tres grupos:
  - De Módulos: los módulos son unidades de implementación, una forma de ver al sistema basada en el código
  - De Componentes y Conectores: aquí los elementos son unidades de run-time y los conectores son los mecanismos de comunicación entre esos conectores
  - Estructuras de alocación o asignación ("allocation"): Muestra la relación entre elementos de software y los elementos en uno o más entornos externos en los que el software se crea y ejecuta
- Llamamos "Viewtypes" a las vistas de arquitecturas orientadas a estas tres estructuras

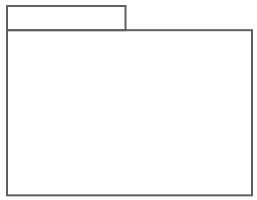
Estructuras del tipo módulo

- Un **módulo** es una unidad de código que implementa un conjunto de responsabilidades
  - Una clase, una colección de clases, una capa o cualquier descomposición de la unidad de código
  - Nombres, responsabilidades, visibilidad de las interfaces
- Relaciones:
  - Descomposición (“es parte de”): los módulos tiene relación del tipo “es un submódulo de”
  - Usos (“depende de”): los módulos tienen relación del tipo “usa a”. Se dice que un módulo A usa a B, si la correcta ejecución de B es necesaria para la correcta ejecución de A (no es lo mismo que invocación)
  - Clases (“es un”): los módulos en este caso son clases las relaciones son “hereda de” u otras asociaciones

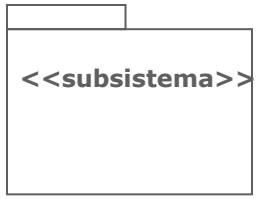
Vistas de Módulos con UML



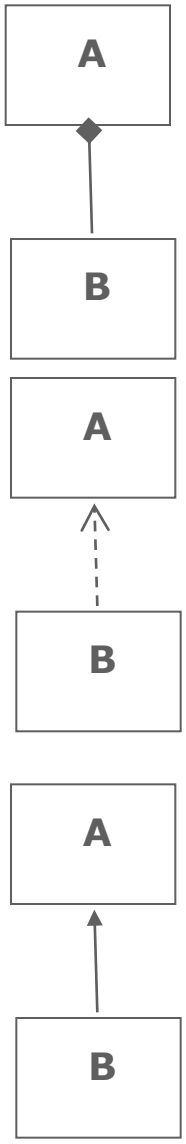
Clase



Paquete



Subsistema



Es parte de

Depende de

Es un

Ejemplo – Vista combinada de Módulos y C&C

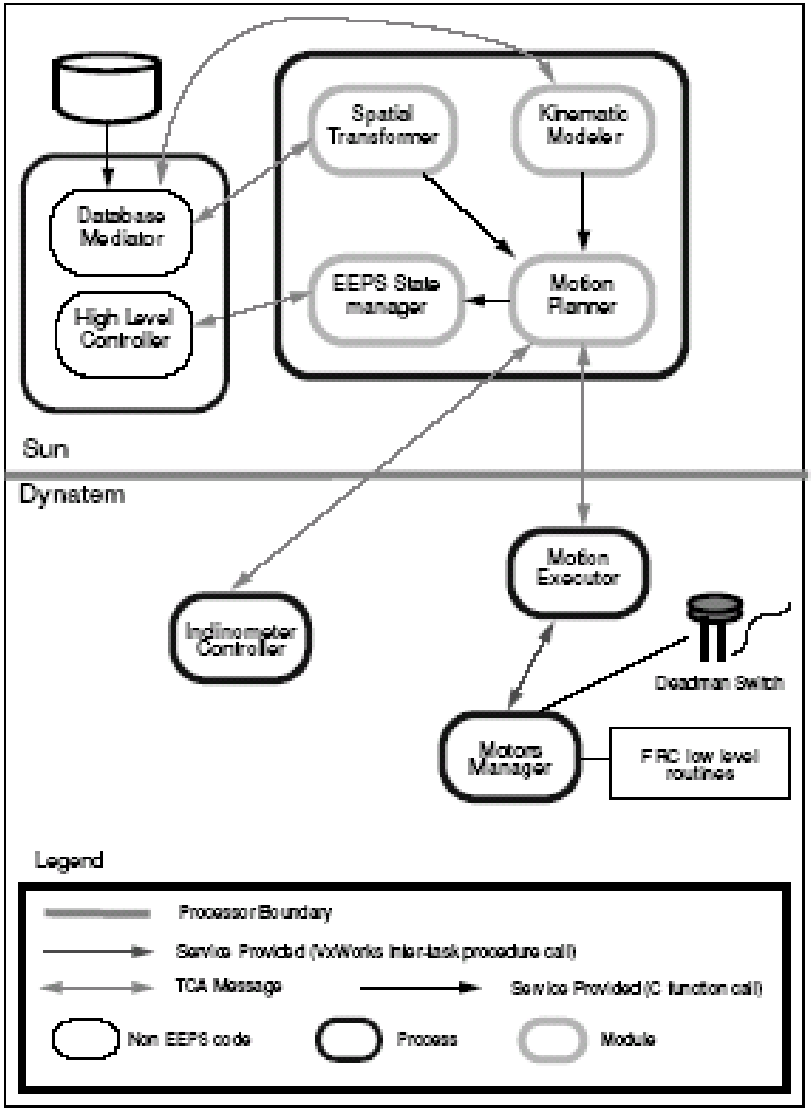


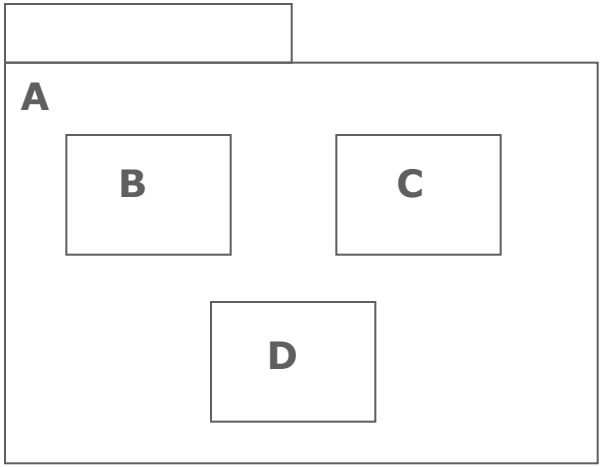
Figure 3-2: Allocation of Modules to Processes

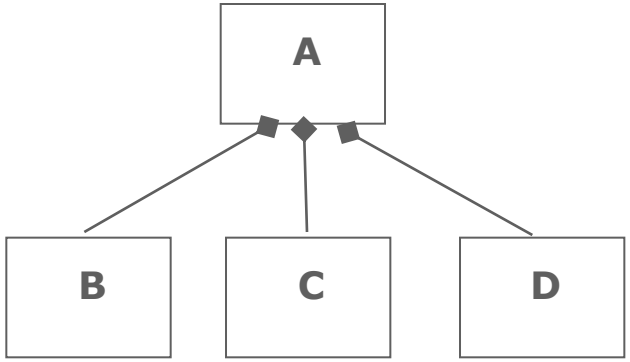
Relación de Descomposición

- Este estilo **está focalizado en la relación "es parte de"**
- Representa la descomposición del código en sistemas, subsistemas, etc
- Esta descomposición permite mostrar las responsabilidades del sistema y cómo éstas son fragmentadas entre distintos módulos y a su vez sub-módulos
- La relación de ***descomposición*** es una especialización de la relación ***es parte de***.
  - Se debe especificar el criterio utilizado para definir la descomposición

Vista de descomposición con UML

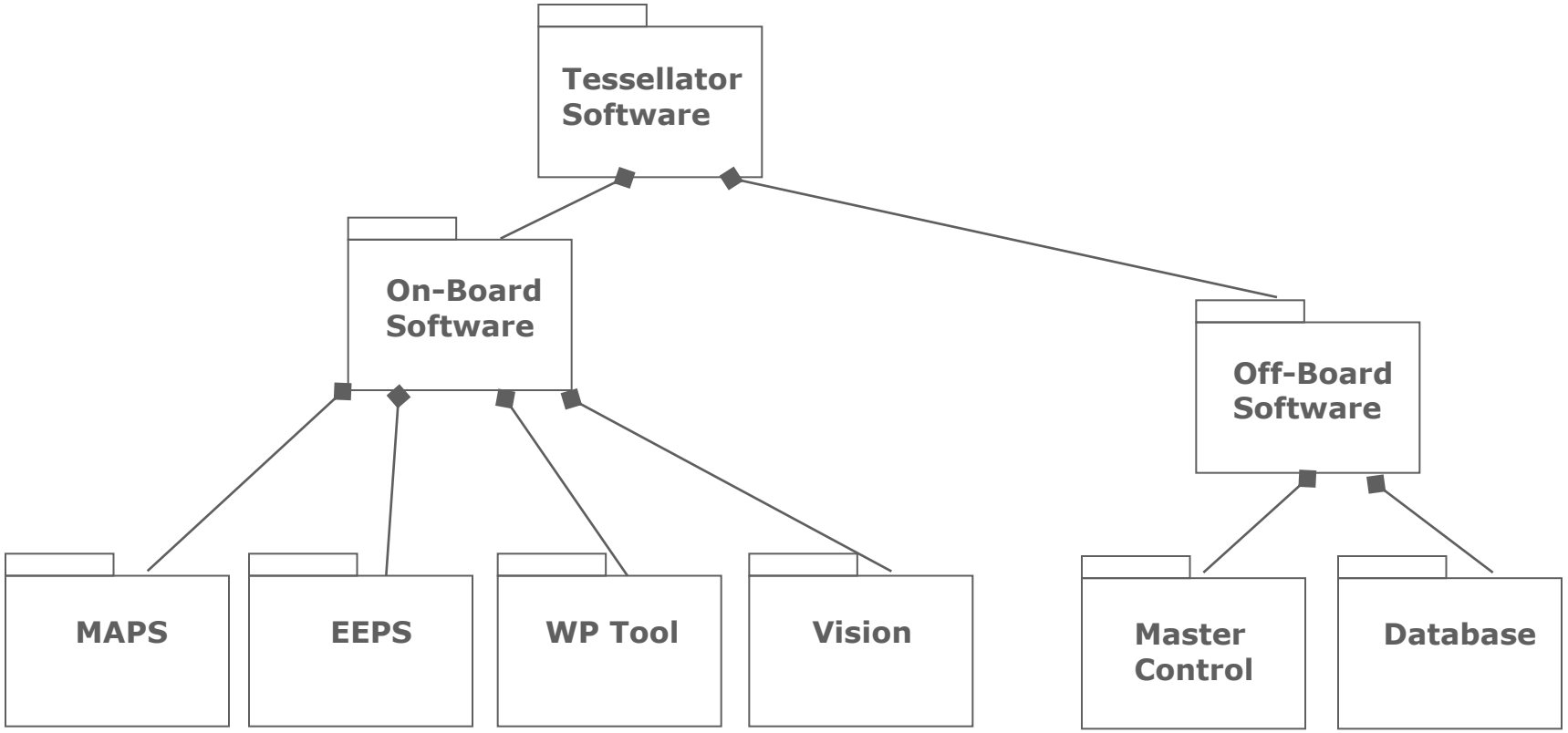
- Dos formas disponibles para la notación en UML





Ejemplo Tessellator

- El software del Robot Tessellator se divide inicialmente en dos grupos: "on board" y "off board". Los primeros incluyen el módulo MAPS para posicionamiento de la base móvil, EEPS para posicionamiento del brazo, VISION para el sistema de visión y RKW para el manejo de la herramienta de impermeabilización. Los subisistemas off-board incluyen el Control Maestro y de Acceso a la Base de Datos (DBA).



Descomposición: notación

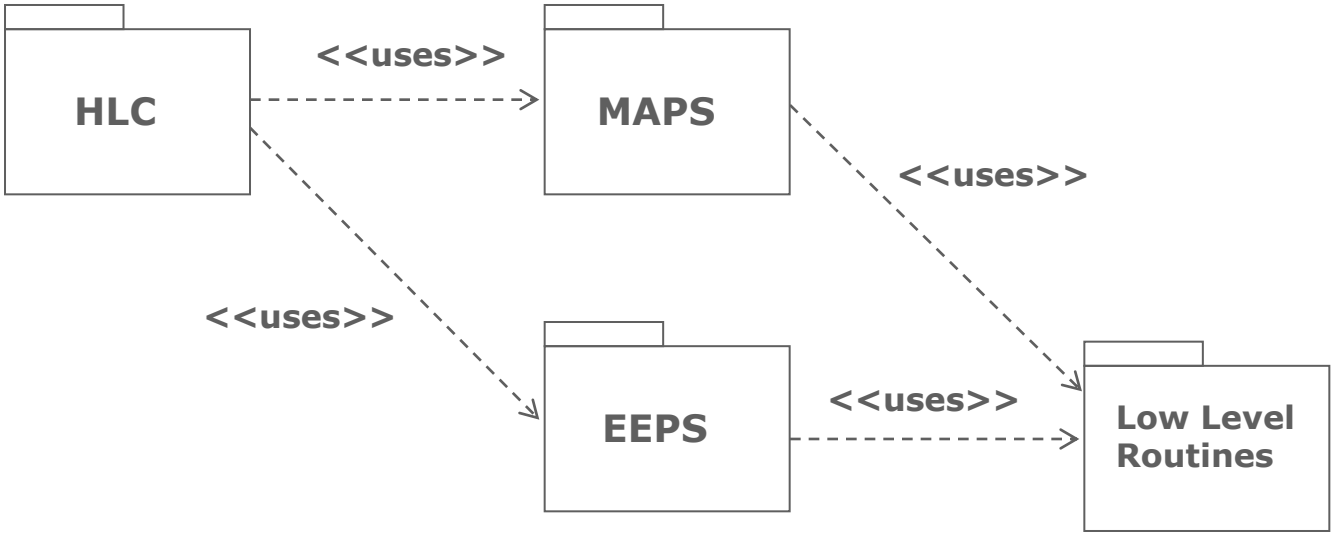
- Hardware-Hiding Module
  - Extended Computer Module
    - Data Module
    - Input/Output Module
    - Computer State Module
    - Parallelism Control Module
    - Program Module
    - Virtual Memory Module
    - Interrupt Handler Module
    - Timer Module
  - Device Interface Module
    - Air Data Computer Module
    - Angle of Attack Sensor Module
    - Audible Signal Device Module
    - Computer Fail Device Module
    - Doppler Radar Set Module
    - Flight Information Displays Module
    - Forward Looking Radar Module
    - Head-Up Display Module
    - Inertial Measurement Set Module

- Behavior-Hiding Module
  - Function Driver Module
    - Air Data Computer Module
    - Audible Signal Module
    - Computer Fail Signal Module
    - Doppler Radar Module
    - Flight Information Display Module
    - Forward Looking Radar Module
    - Head-Up Display Module
    - Inertial Measurement Set Module
    - Panel Module
    - Projected Map Display Set Module
    - Shipboard Inertial Nav. Sys. Mod.
    - Visual Indicator Module
    - Weapon Release Module
    - Ground Test Module
  - Shared Services Module
    - Mode Determination Module
    - Panel I/O Support Module
    - Shared Subroutine Module

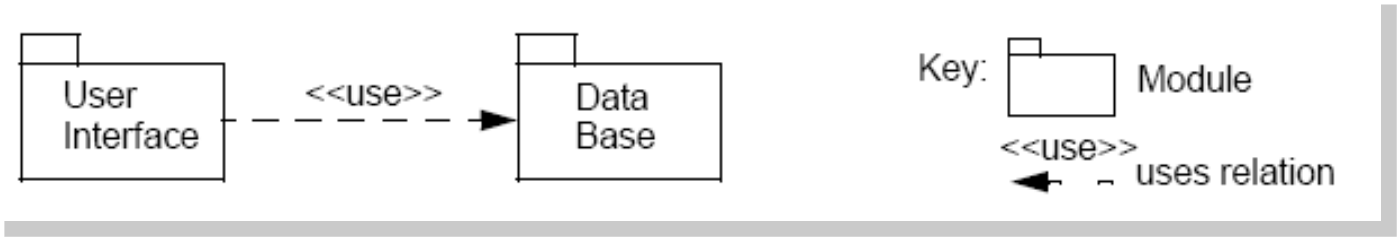
Relación de usos - Elementos y Relaciones

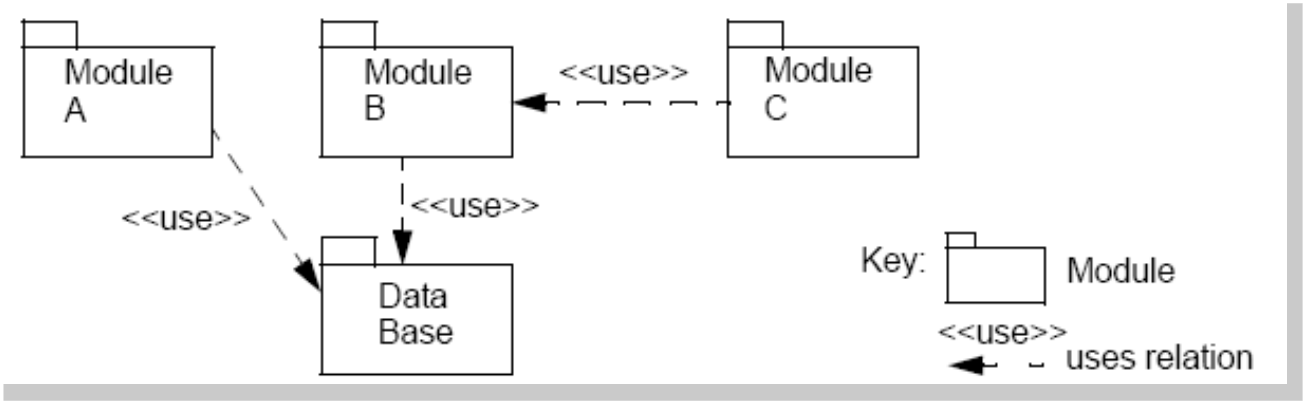
- Este estilo está focalizado en la relación ***depende de*** (especializada en la relación ***usa***)
- **Elementos**
  - Como en el viewtype en general, el elemento de este estilo es el ***módulo***
- **Relaciones**
  - La relación de ***usa*** es una especialización de la relación ***depende de***
  - Esta vista hace explícito como una funcionalidad se mapea con una implementación, mostrando la relaciones existentes entre las distintas partes de código que finalmente la implementan

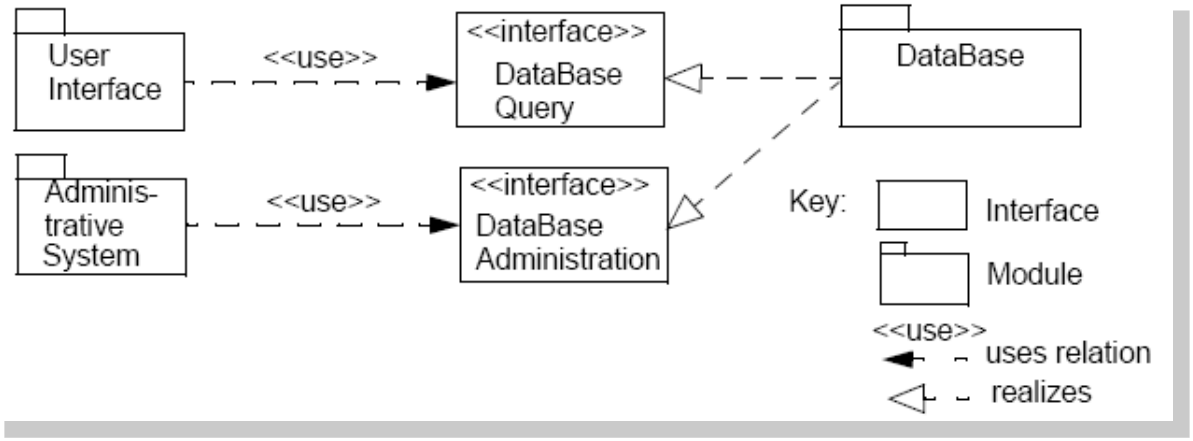
Ejemplo de vista de usos de alto nivel



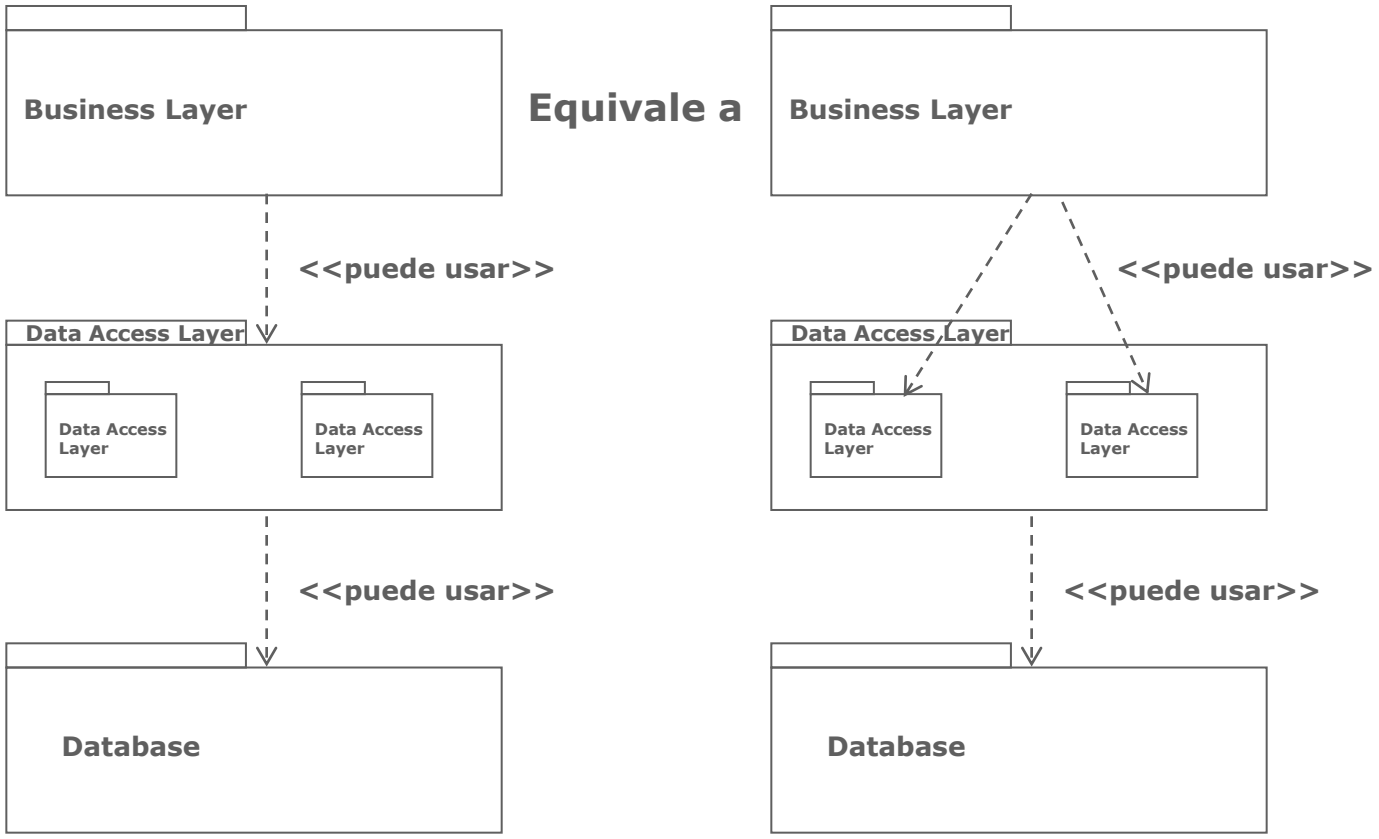
Notación







Caso particular de usos - Layered



Generalización

- Este estilo está focalizado en la relación ***es un*** (especializada en la relación ***generaliza***)
  - Este estilo es útil cuando el arquitecto desea soportar la posible extensión y evolución de la arquitectura

- Los módulos son organizados de modo de capturar concordancias y variaciones entre sí
  - él módulo ***padre*** es más general que los módulos ***hijos***
  - El módulo padre reúne todas las concordancias entre sus hijos, y cada hijo posee las variaciones correspondientes

Estructuras de componentes y conectores

- Estas estructuras están centradas en procesos que se comunican.
- Sus elementos son entidades con manifestación runtime que consumen recursos de ejecución y contribuyen al comportamiento en ejecución del sistema
- La configuración del sistema es un grafo conformado por la asociación entre componentes y conectores
- Las entidades runtime son instancias de *tipos* de **conector** o **componente**

Conectores

- Un conector representa un camino en la interacción en tiempo de ejecución entre dos o más componentes
- El tipo de conector indica la cardinalidad (cantidad de componentes en la interacción), las interfaces que soporta y las propiedades requeridas
- El tipo de conector se hereda generalmente del estilo
- El conector asume un conjunto de roles dentro de la arquitectura

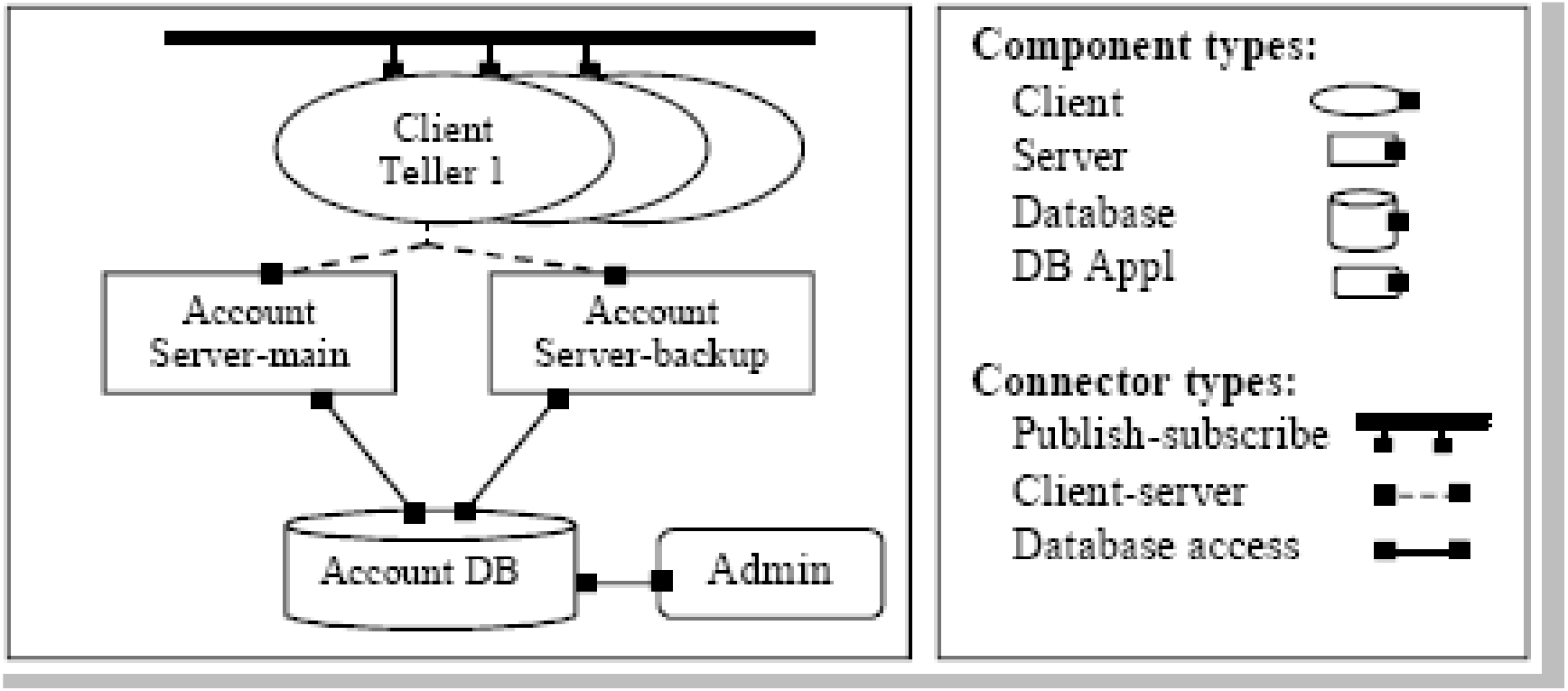
Relaciones

La relación es *attachment*: Indica qué componentes están vinculados con qué conectores

Formalmente siempre se asocian puertos de componentes con puertos de conectores

Un puerto de componente p_1, es vinculado con un rol de conector r, si el componente interactúa sobre el conector usando la interfaz descrita por p_1 y cumpliendo con la expectativas descritas por r

Ejemplo

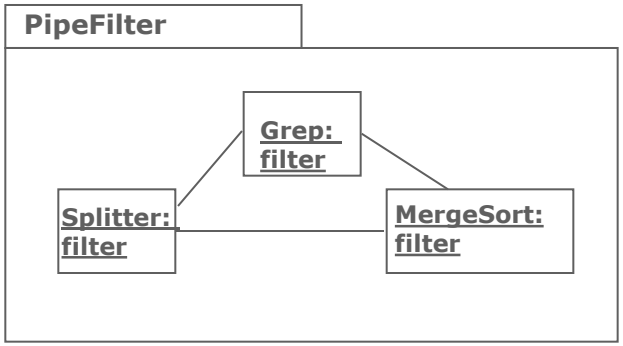


Utilidad

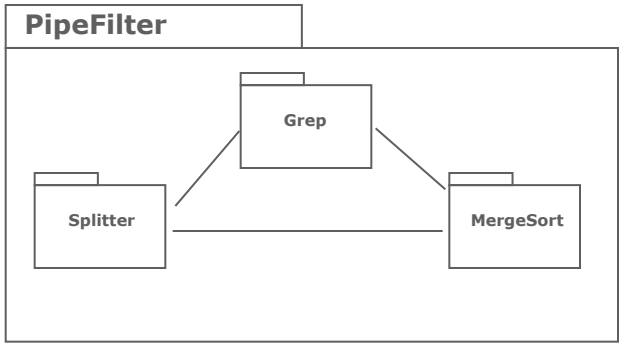
- ¿Cuales son los componentes ejecutables y como interactúan?
- ¿Cuáles son los repositorios y que componentes los acceden?
- ¿Qué partes del sistema son replicadas y cuantas veces?
- ¿Cómo progresan los datos a los largo del sistema a medida que éste se ejecuta?
- ¿Qué protocolos de interacción son usados por las entidades comunicantes?
- ¿Qué partes del sistema se ejecutan en paralelo?
- ¿Cómo la estructura del sistema puede cambiar a medida que se ejecuta?
- ¿Para qué no? Cuando lo que quiero analizar no depende del comportamiento en run time

Descripciones de componentes y conectores con UML

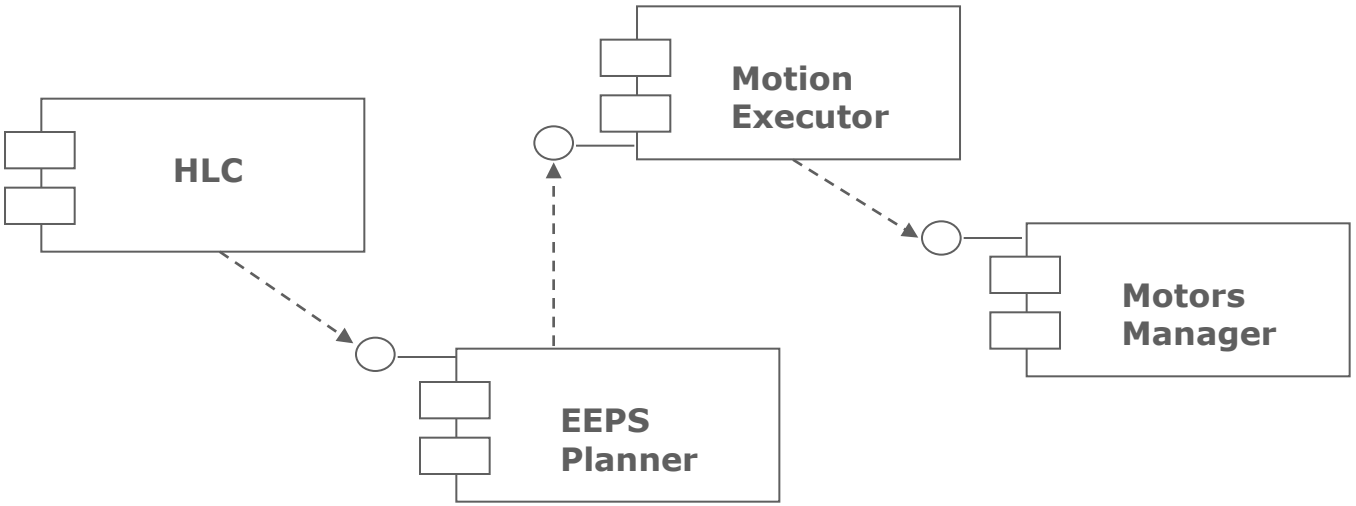
Filtros como objetos



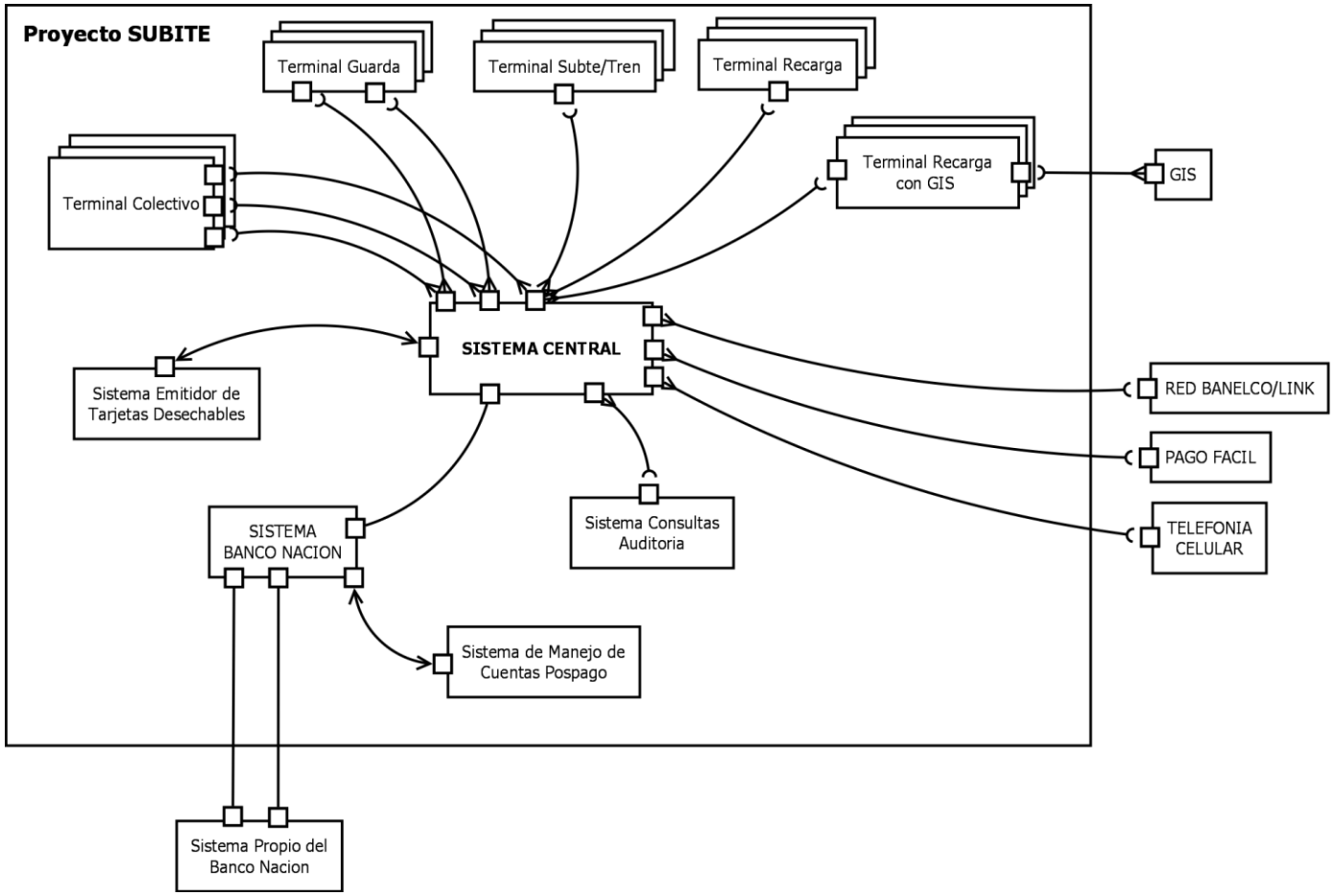
Filtros como packages



Procesos como Componentes UML



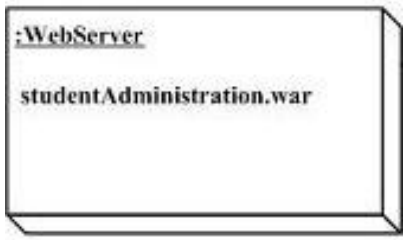
Ejemplo SUBITE



Estructuras de alocación o asignación

- Deployment: muestra cómo el software se asigna a hardware y elementos de comunicación
- Implementación: muestra cómo los elementos de software se mapean a estructuras de archivos en repositorios de control de la configuración o entornos de desarrollo
- Asignación de trabajo ("work assignment"): asigna la responsabilidad del desarrollo y la implementación a equipos de programadores

Nodo



Componente

UML 1.*

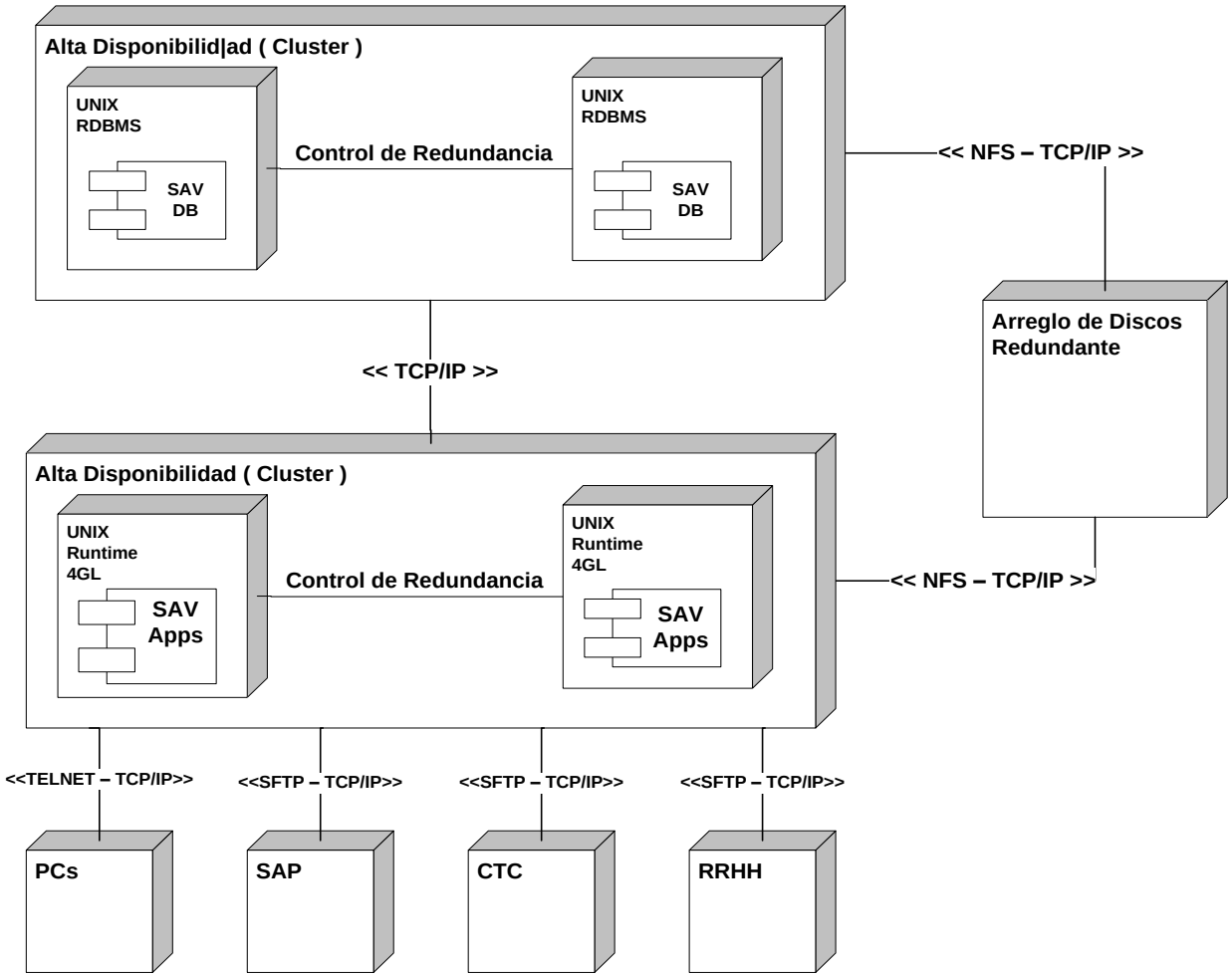
UML 2.0

Asociación

Dependencia

Vista de deployment

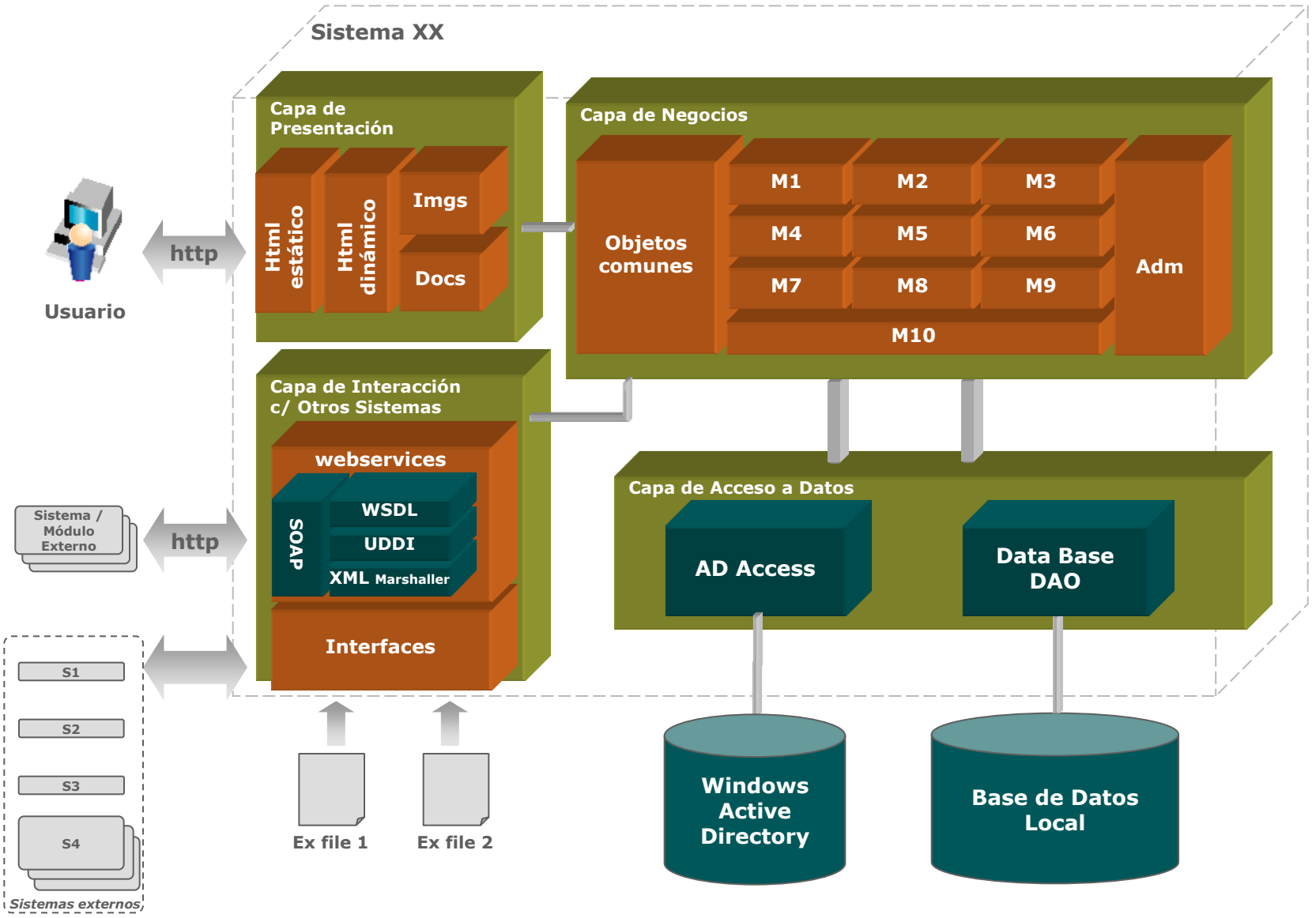
SAV – Esquema Despliegue – Ambiente de Producción



Al documentar, qué vistas elegir

- Cada estructura tiene una vista
- Los diferentes “stakeholders” necesitan distintas vistas
- Distintos tipos de sistemas pueden implicar distintas vistas
- Un proceso para definir vistas:
  1. Hacer una lista de vistas candidatas
  2. Combinar vistas. Ejemplos: implementación con módulos, una vista de niveles con la vista de módulos, vista de deploy con vistas de componentes y conectores.
  3. Definir prioridades

Ejemplo de vista combinada



Puntos clave para documentar una arquitectura

- El documento está escrito para el lector, no quien lo escribe
- Evitar ambigüedad
  - Explicar la notación – claves
  - Adjuntar información adicional, por ejemplo de interfaces
- Usar una organización estándar
  - Qué vistas, template a usar
- Registrar los motivos
  - Explicar el por qué de las decisiones tomadas
- Mantenerla actualizada, pero no demasiado
- Revisarla

Elementos esenciales

- Descripción de los requerimientos
  - Contexto del negocio, "rationale" para el producto, dominio
- Descripción del contexto
  - Sistemas con quienes interactúa, interfaces externas
- Uso de diagramas de arquitectura
  - Con prosa y descripción de cajas y líneas
- Consideración de restricciones de implementación
  - En la medida en que impactan la arquitectura
- Explicación del diseño arquitectónico
  - Como ataca los requerimientos y las restricciones de diseño
  - Alternativas

El template de RUP para documentar arquitecturas

1. Introduction
    - 1.1 Purpose
    - 1.2 Scope
    - 1.3 Definitions, Acronyms and Abbreviations
    - 1.4 References
    - 1.5 Overview
2. Architectural Representation (¿Cómo la representamos?)
3. Architectural Goals and Constraints
4. Use-Case View
    - 4.1 Use-Case Realizations
5. Logical View (descomposición en subsistemas o "packages")
    - 5.1 Overview
    - 5.2 Architecturally Significant Design Packages (para cada uno incluir clases más importantes)

El Template de RUP para documentar arquitecturas (cont.)

6. Process View (descomposición en “threads de control”)
7. Deployment View (hardware, redes)
8. Implementation View
9. Data View (optional)
10. Size and Performance
11. Quality

Un ejemplo de índice de un documento de arquitectura

- 1 Introduction
  - 1.1 Purpose and audience of this document
  - 1.2 Evolution of this document
  - 1.3 Reference Materials
  - 1.4 Terms and acronyms
- 2 Goals and Constraints of the architecture
  - 2.1 Overview of the XX System
  - 2.2 Main Characteristics of XX
- 3 Overview of the Application Architecture
  - 3.1 High level view of the application architecture
  - 3.2 Description of the application layers and modules
  - 3.3 WebServices
- 4 Application Architecture – Major Design Decisions
  - 4.1 Implementation of the Model-View-Controller Pattern
  - 4.2 Use of Patterns

Un índice de un documento de arquitectura (cont.)

- 4 Application Architecture – Major Design Decisions
  - 4.3 Implementation of the View (Usability)
  - 4.4 Error Handling Strategy
  - 4.5 Logging Strategy
  - 4.6 Object Persistence
  - 4.7 Implementation of the entity locking solution
  - 4.8 Managing Historical Data
  - 4.9 Database usage for batch processing
  - 4.10 Client State Administration
  - 4.11 Implementation of the Internationalization Features
  - 4.12 Implementation of the Security Features
  - 4.13 The use of a workflow system
  - 4.14 Integration Strategy
- 5 Significant Use Cases of the system
  - 5.1 Data manipulation processes
  - 5.2 Workflow processes
  - 5.3 Batch processes
  - 5.4 Application of XX's architectural definitions

Un índice de un documento de arquitectura (cont.)

- 6 Base Software
  - 6.1 EJB Version
  - 6.2 J2EE Version
  - 6.3 Version and edition of the Oracle Application Server
  - 6.4 Version of the Oracle Database
  - 6.5 Operating System

- 7 Servers Architecture
  - 7.1 Most Reliable alternative
  - 7.2 Reliable Alternative
  - 7.3 Basic Reliable alternative
  - 7.4 Basic Solution
  - 7.5 Alternatives for each country