Ingeniería de Software II

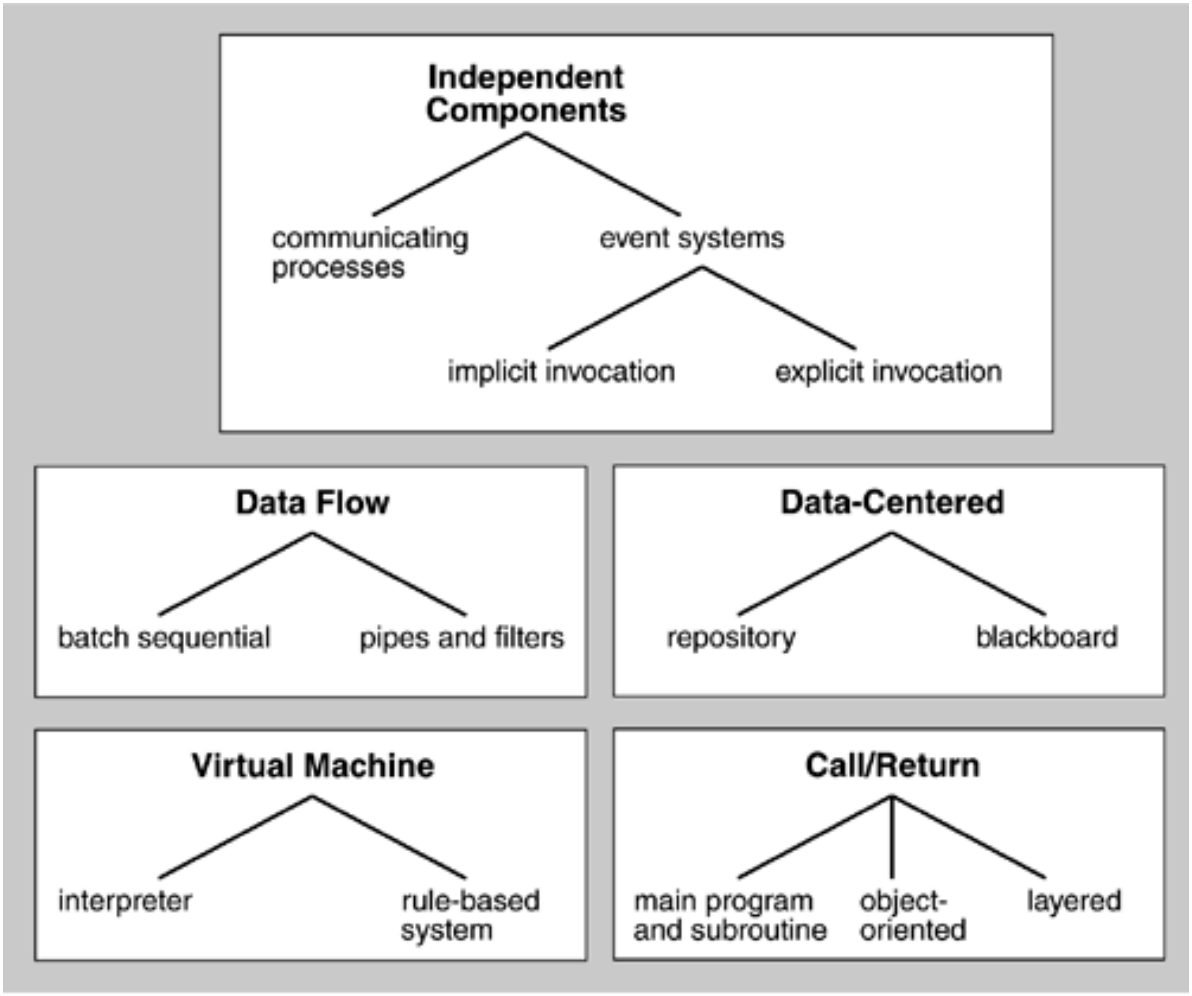
Ingeniería de Software II

Segundo Cuatrimestre de 2016

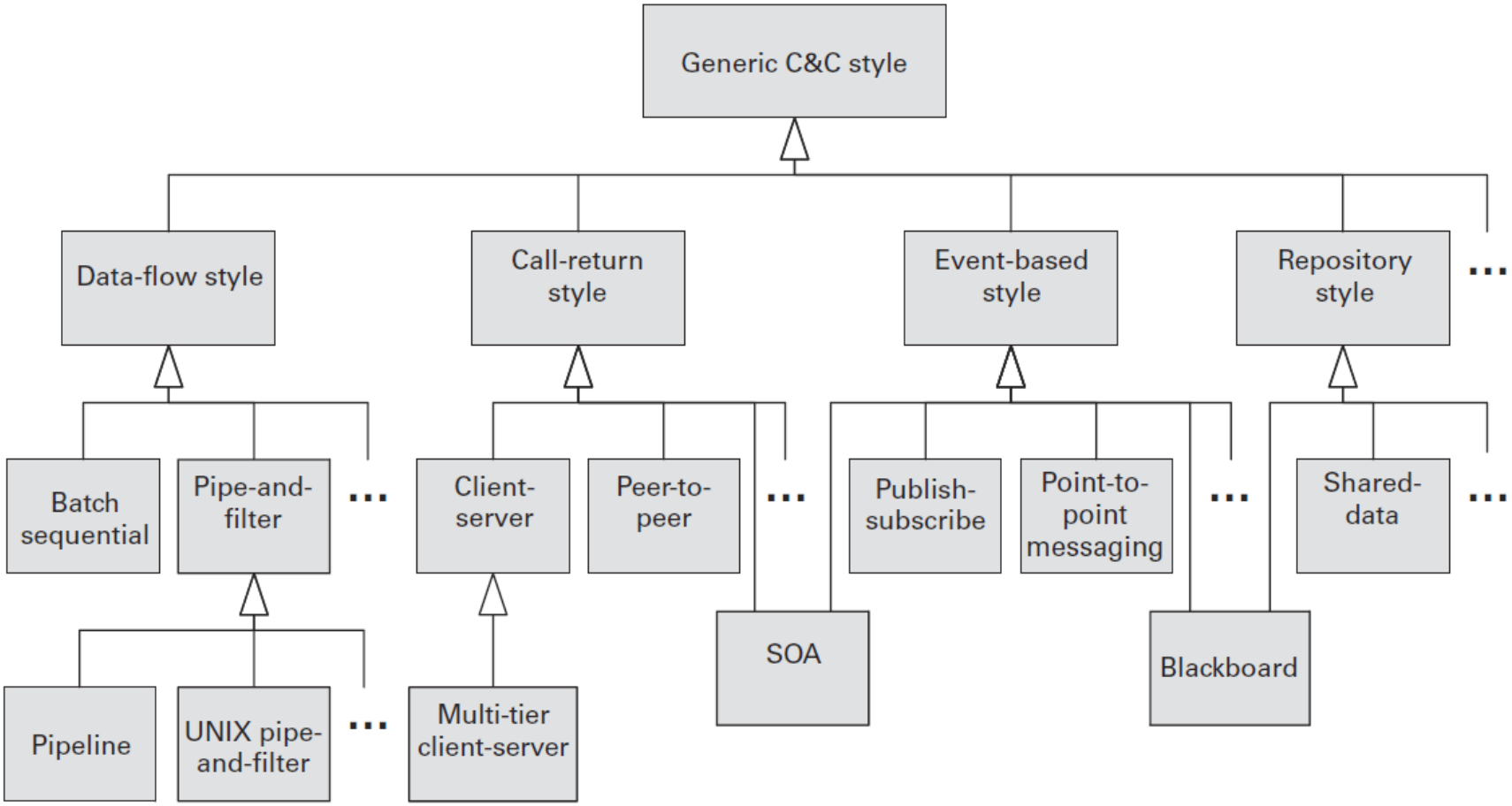
Clase 16: Estilos Arquitectónicos y Viewtypes

Buenos Aires, 24 de Octubre de 2016

Vista Componente y Conector
Taxonomía de estilos arquitectónicos



Vista Componente y Conector
Taxonomía de estilos arquitectónicos

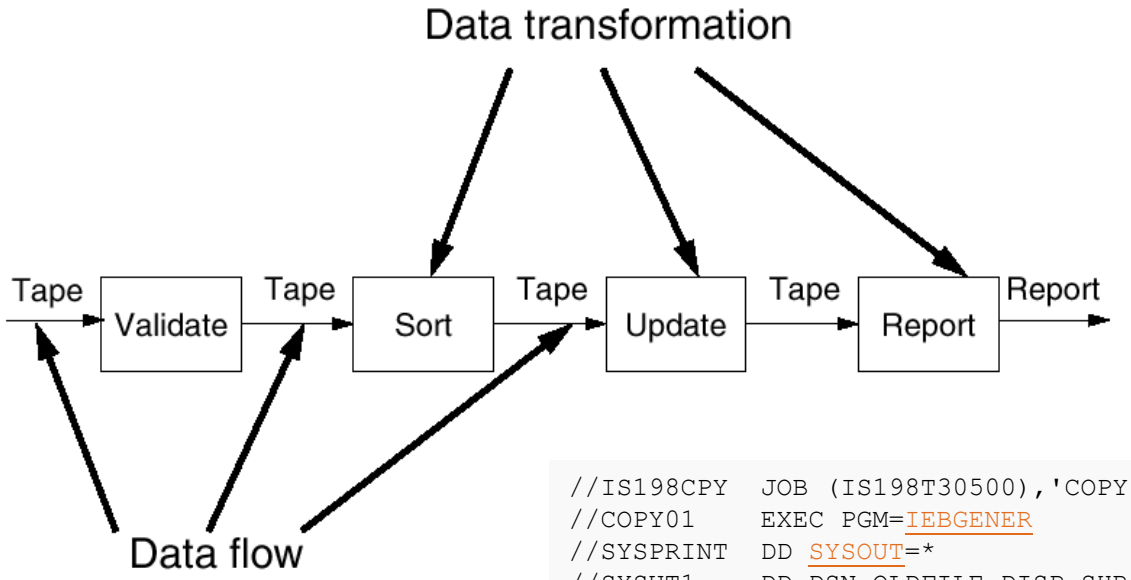


Arquitecturas tipo “Data – Flow”

- La estructura del sistema está basada en transformaciones sucesivas a datos de input
- Los datos entran al sistema y fluyen a través de los componentes hasta su destino final
- Los componentes son de run-time
- Normalmente un componente (de control) controla la ejecución del resto de los componentes
- Dos estilos
  - Batch sequential
  - Pipe and filter

Estilo Batch Sequential

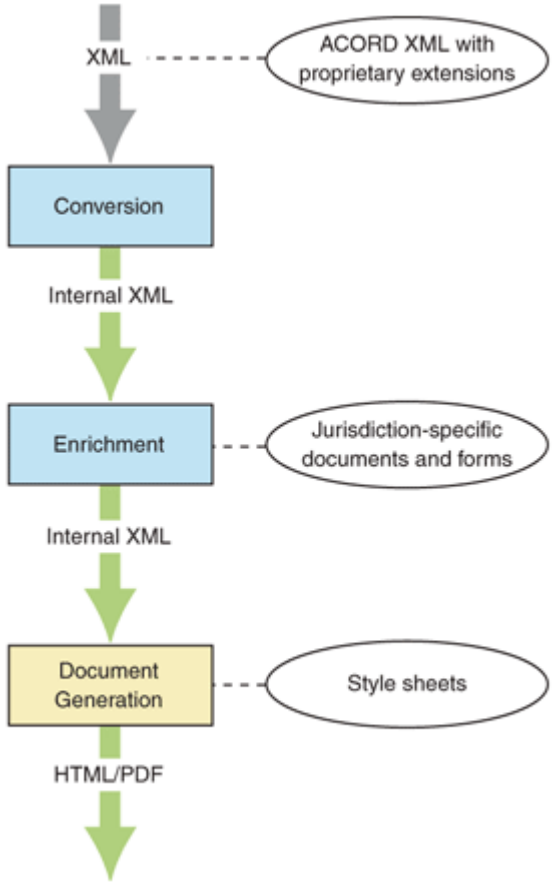
- Cada paso se ejecuta hasta ser completado, y recién después puede comenzar el siguiente paso
- Usado en aplicaciones clásicas de procesamiento de datos
- Muchas veces usadas a partir de un "proceso off line"

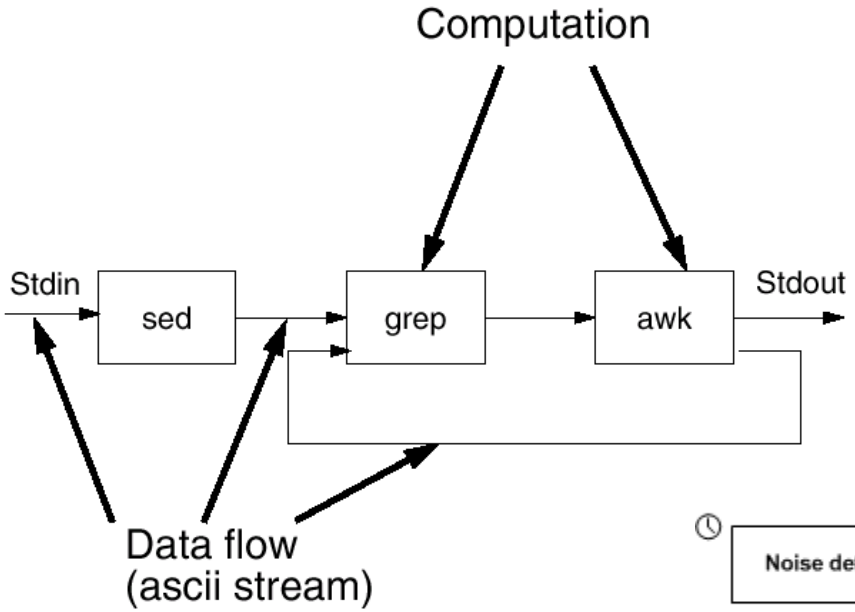


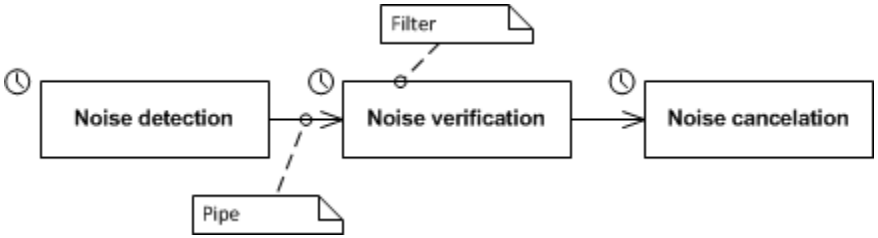

```
//IS198CPY  JOB (IS198T30500),'COPY JOB',CLASS=L,MSGCLASS=X
//COPY01    EXEC PGM=IEBGENER
//SYSPRINT  DD SYSOUT=*
//SYSUT1    DD DSN=OLDFILE,DISP=SHR
//SYSUT2    DD DSN=NEWFILE,
//             DISP=(NEW,CATLG,DELETE),
//             SPACE=(CYL,(40,5),RLSE),
//             DCB=(LRECL=115,BLKSIZE=1150)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

Estilo Pipes and Filters

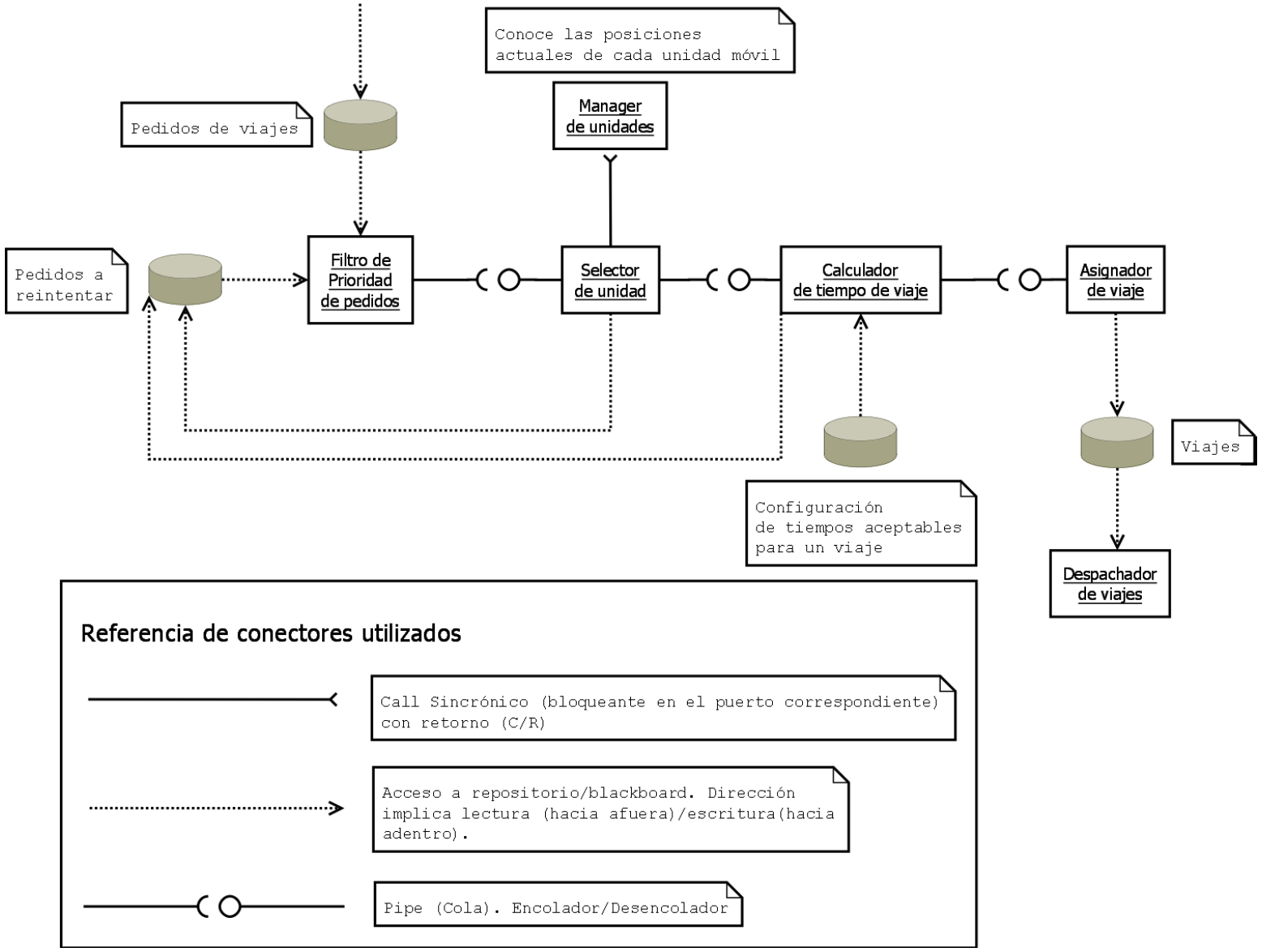
- Cada componente tiene inputs y outputs. Los componentes leen "streams" de datos de su input y producen streams de datos en sus outputs de forma continua
- Filters: ejecutan las transformaciones
- Pipes: conectores que pasan streams de un filtro a otro







Estilo Pipes and Filters (otro ejemplo)



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

Estilo Call and Return

- El estilo dominante en la mayoría de los sistemas
- Variaciones
  - Programa principal y subrutinas
  - Arquitecturas orientadas a objetos
  - Descomposición funcional o "modelar entes de la realidad"
- En call return tradicional
  - Invocación explícita y sincrónica
  - Jerarquía de módulos (coordinación vs. ejecución)

Estilo Layered o Multi - tier

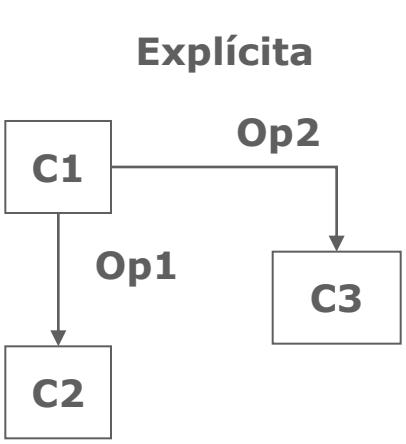
- Cada nivel provee servicios
  - Oculta en nivel siguiente
  - Provee servicios al nivel anterior
- En muchos casos el “bajar” de nivel implica acercarse al hardware o software de base
- Los niveles van formando “virtual machines”
- Ventajas: portabilidad, facilidad de cambios, reuso
- Desventajas: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

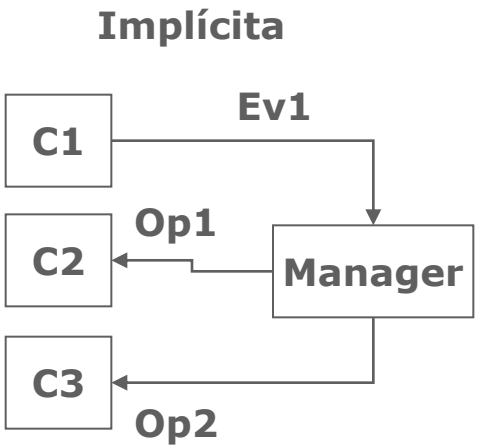
Estilo Client/Server

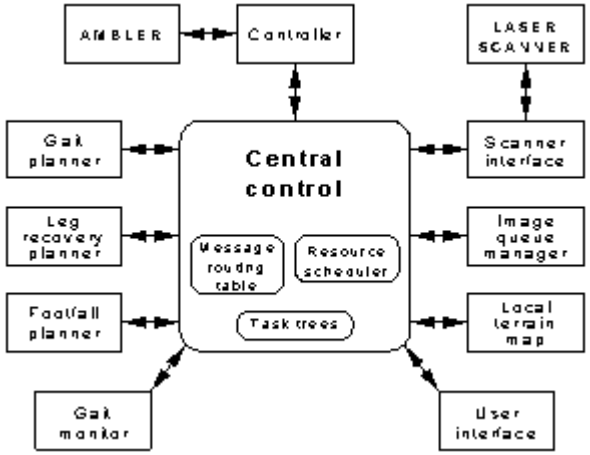
- Una instancia de un estilo más genérico: sistemas distribuidos
- Los componentes son clientes (acceden a servicios) y servidores (proveen servicios)
- Los servers no conocen la cantidad o identidad de los clientes
- Los clientes saben la identidad de los servidores
- Los conectores son protocoles basados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

Componentes independientes

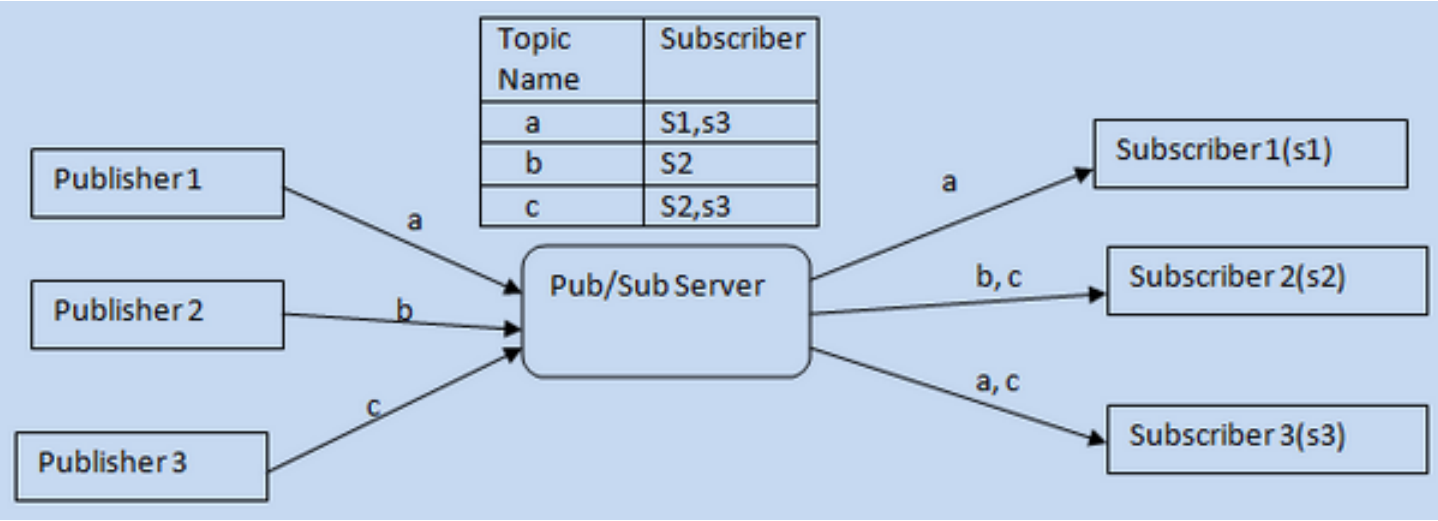
- Son arquitecturas de procesos que se comunican a través del envío de mensajes
- Componentes: procesos independientes
- Conectores: envío de mensajes
  - Sincrónico o asincrónico
- Sistemas de eventos - Invocación implícita





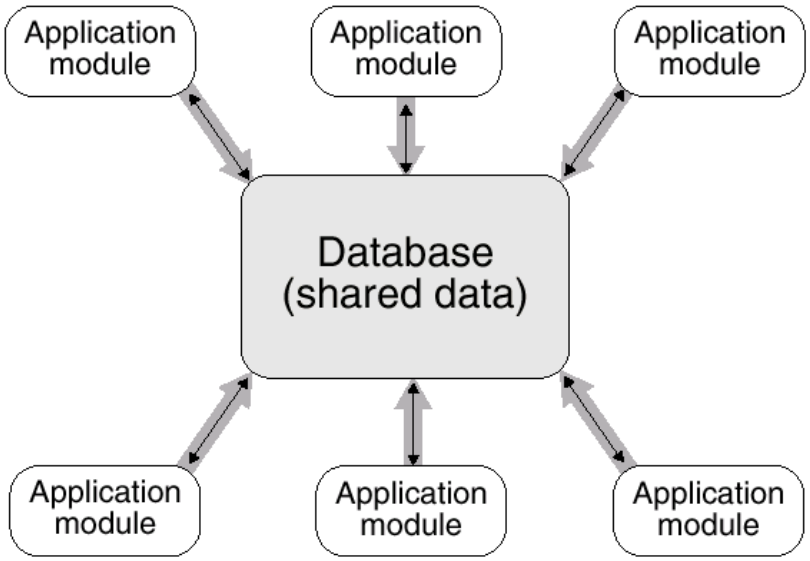


Estilo Publish/Subscribe

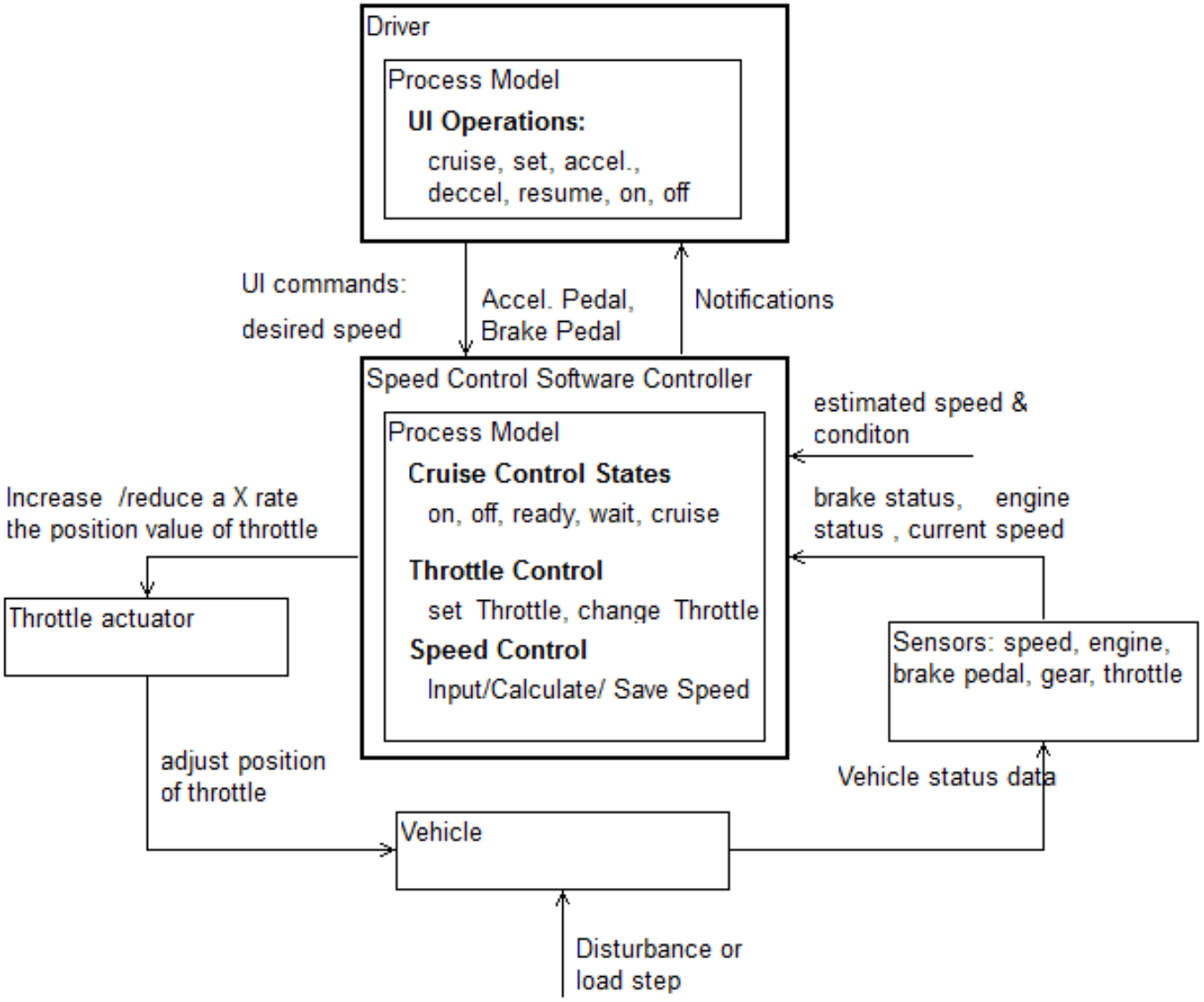


Arquitecturas centradas en datos

- 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



¿Qué estilo es este?



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 **independientes** y los conectores son los mecanismos de comunicación entre esos componentes
  - *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

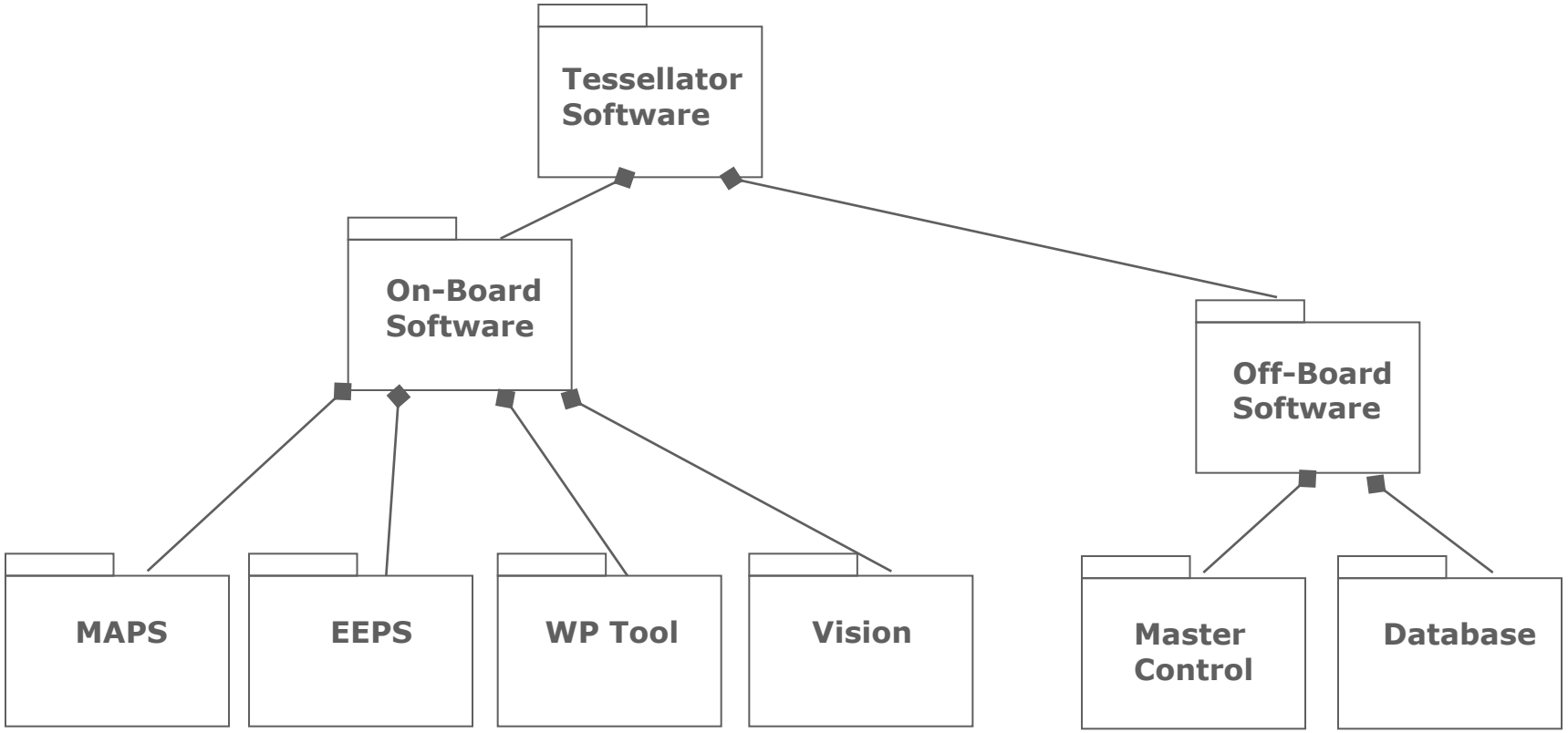
- Un **módulo** es una unidad de código que implementa un conjunto de responsabilidades
  - Una clase, una colección de clases, una capa o cualquier descomposición de la unidad de código
  - Nombres, responsabilidades, visibilidad de las interfaces
- Tipos de diagramas:
  - Descomposición (“es parte 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 (“se comporta como”): los módulos en este caso son clases y las relaciones son de herencia

Vistas de Módulos: utilidad

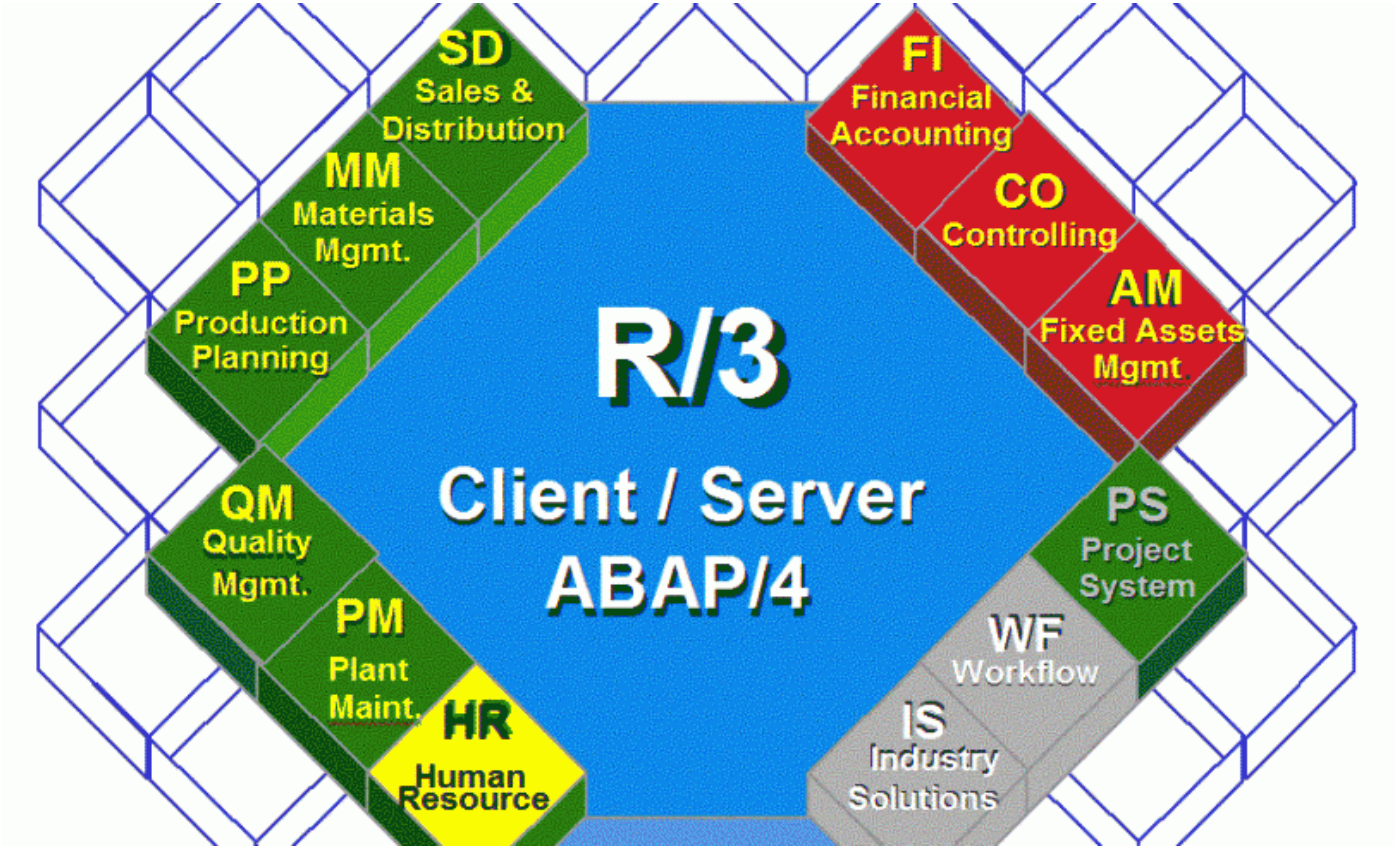
- Construcción
  - Para organizar el código fuente
- Análisis
  - A partir de estas vistas, es posible realizar distinto tipos de análisis Por ejemplo:
    - Trazabilidad de Requerimientos
      - Analiza como los requerimientos funcionales son soportados por las responsabilidades de los distintos módulos
    - Análisis de Impacto
      - Permite predecir el efecto de las modificación del sistema
- Comunicación
  - Pueden ser utilizadas para explicar las funcionalidades del sistema a alguien no familiarizado con el mismo

Ejemplo Tessellator - Descomposición

- 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).



Ejemplo SAP



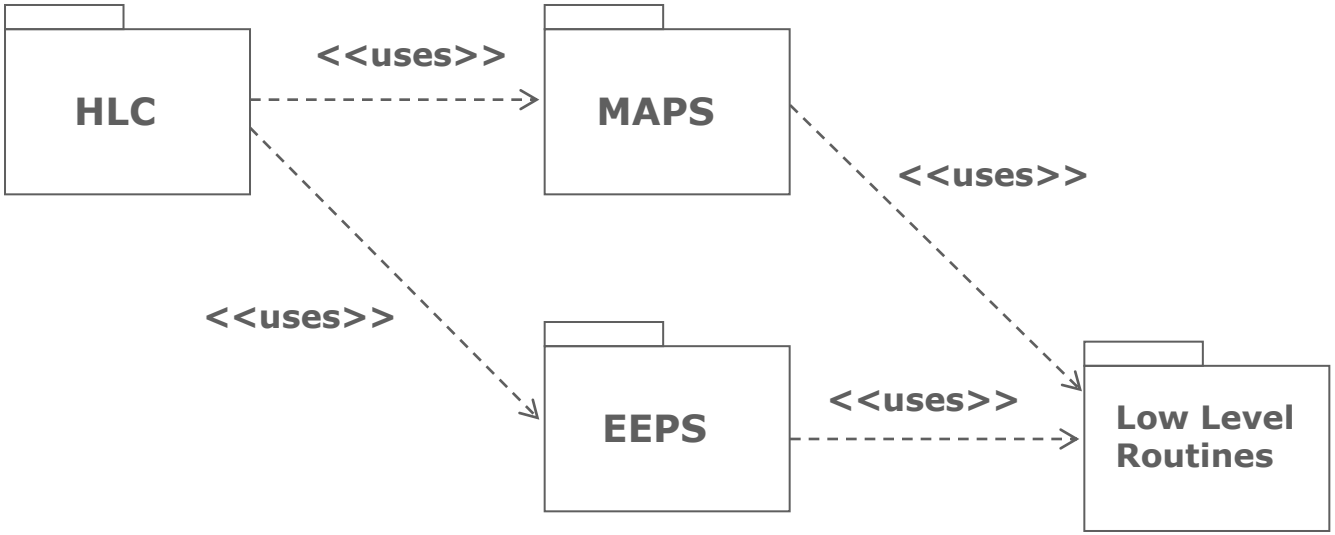
Fuente: sitio web SAP

Ejemplo A7E - Descomposición

- 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

Ejemplo Tessellator - Usos



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**
- Los componentes son entidades **independientes** y sólo se relacionan e interactúan a traves de conectores

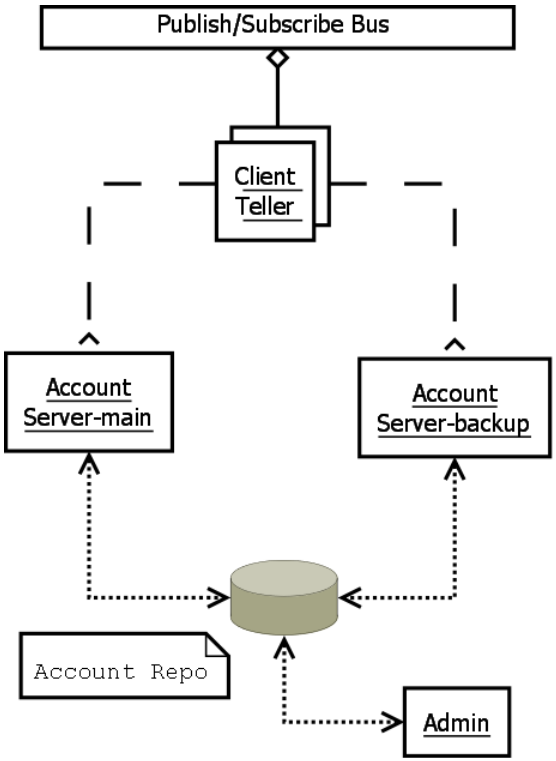
Conectores y Relaciones

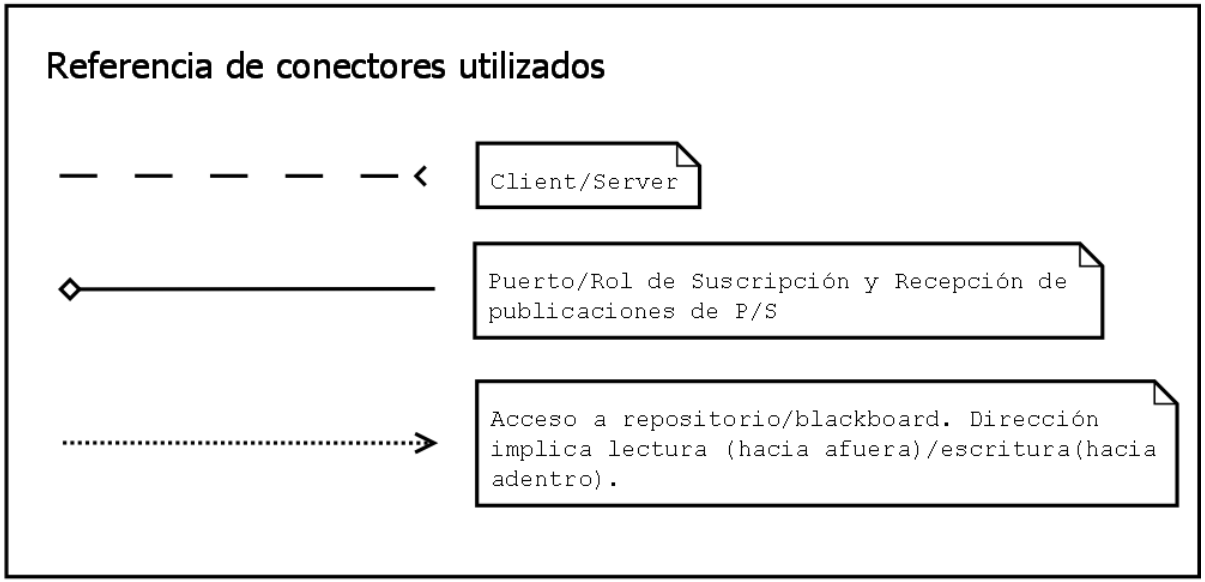
La relación es *attachment*: Indica qué componentes están vinculados con qué conectores

Formalmente siempre se asocian puertos de componentes con puertos de conectores (llamados roles)

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

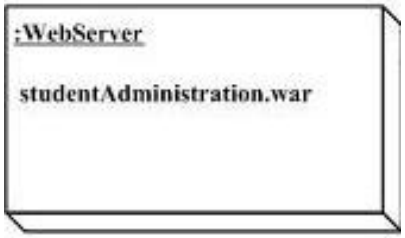




Estructuras de alocación o asignación

- 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



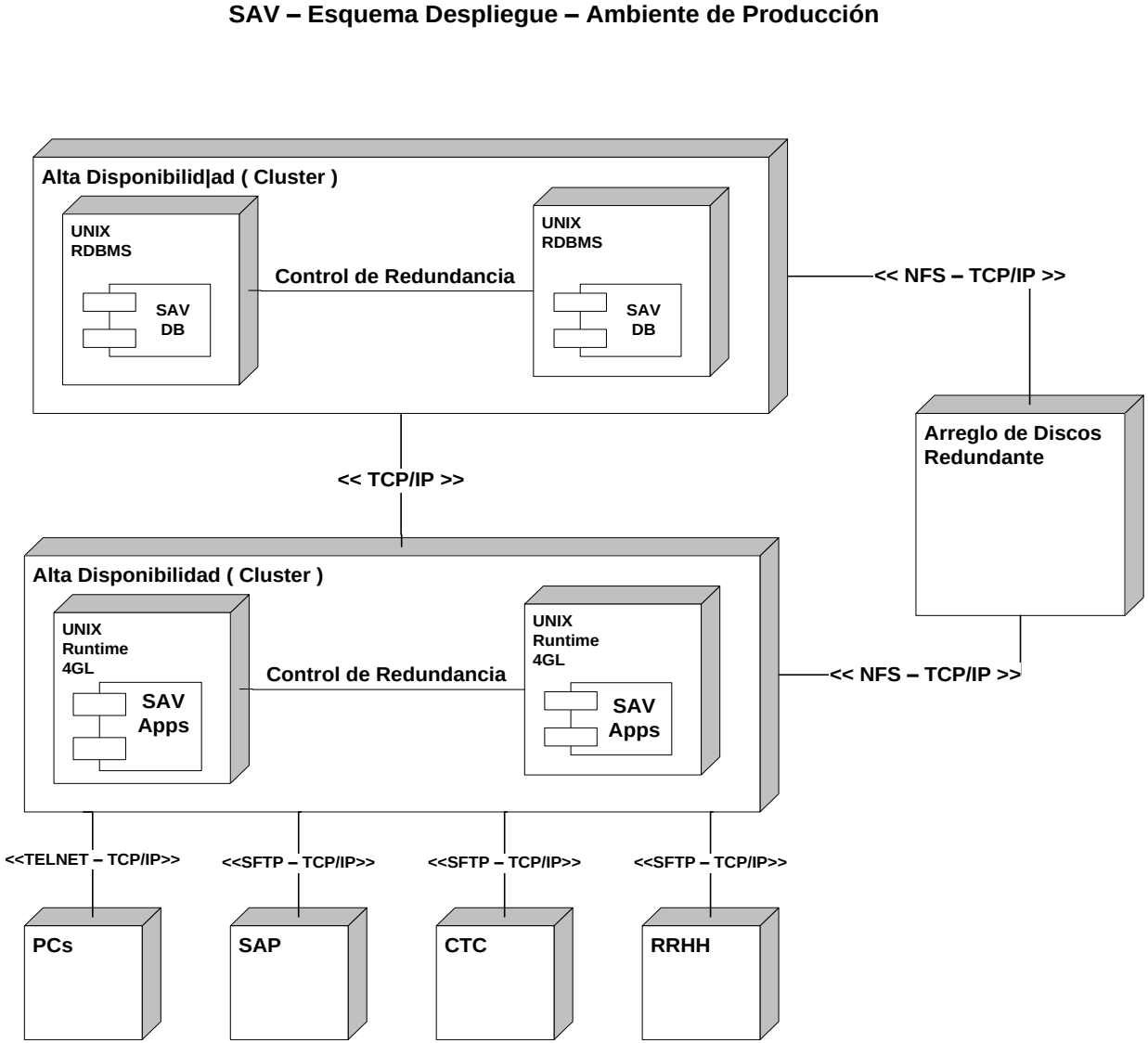
Componente

UML 2.0

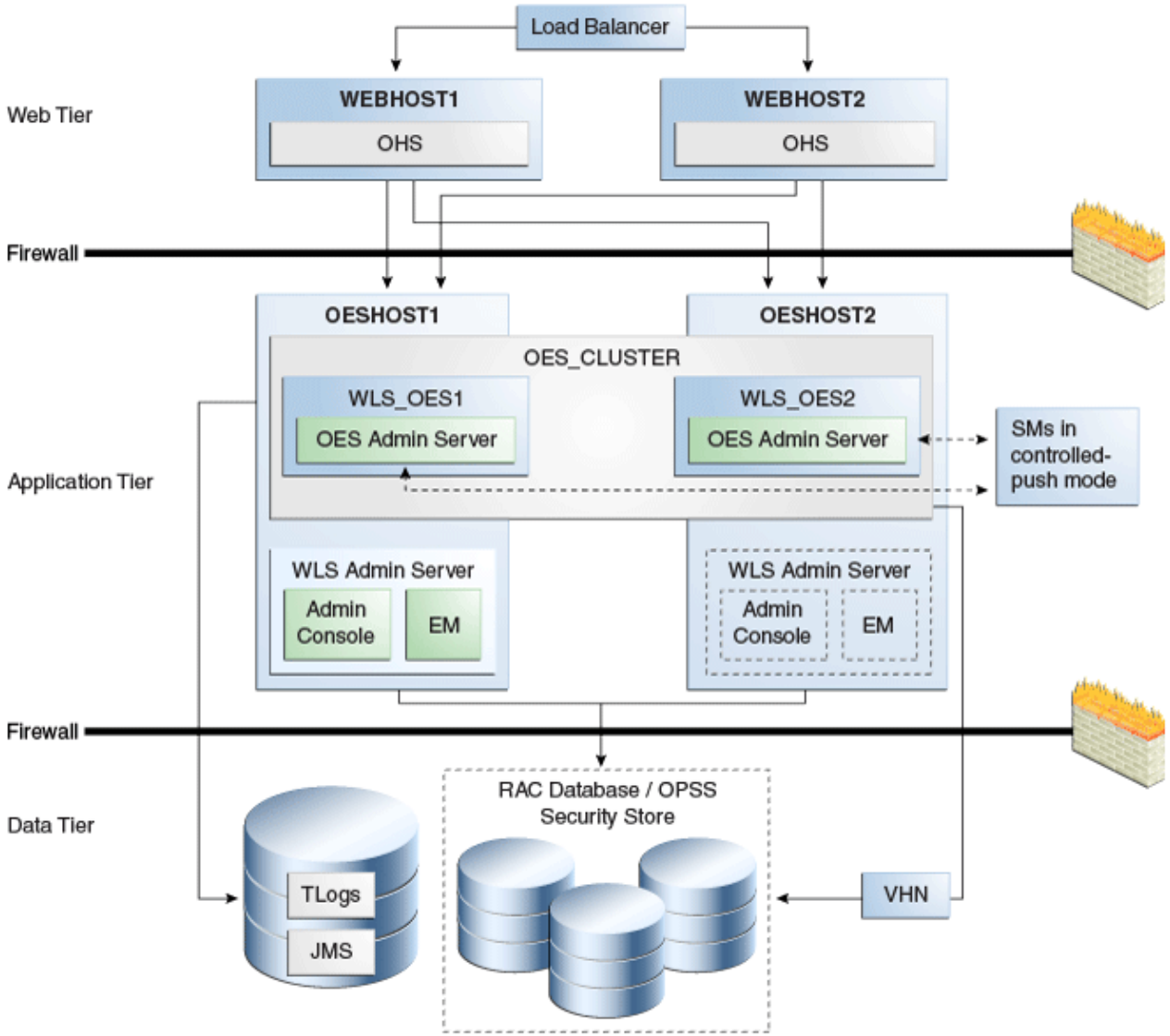
Asociación

Dependencia

Vista de deployment



Ejemplo



Fuente: sitio web Oracle

Documentación de Arquitecturas - 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