

Arquitectura de Software

Vistas de Componente y Conector (CyC)

Contenidos

- Vistas de CyC
- Componentes, Puertos, Conectores, Roles, Relaciones, Restricciones y Propiedades
- Tipos e instancias
- Representación en UML
- Relación con otras vistas
- Estilos de CyC

Vistas de CyC

- Muestran elementos con presencia en tiempo de ejecución
 - Consumen recursos y contribuyen a ejecutar el comportamiento del sistema
- Componentes
 - Ejemplos: Procesos, objetos, clientes, servidores, almacenes de datos, servlet de java, ...
- Conectores
 - Caminos de interacción entre los componentes
 - Ejemplos: Llamada a procedimiento, invocación de servicio, RPC, socket, RMI, CORBA, protocolos de comunicación, flujos de información, acceso a almacenamiento, ODBC, JDBC, ...

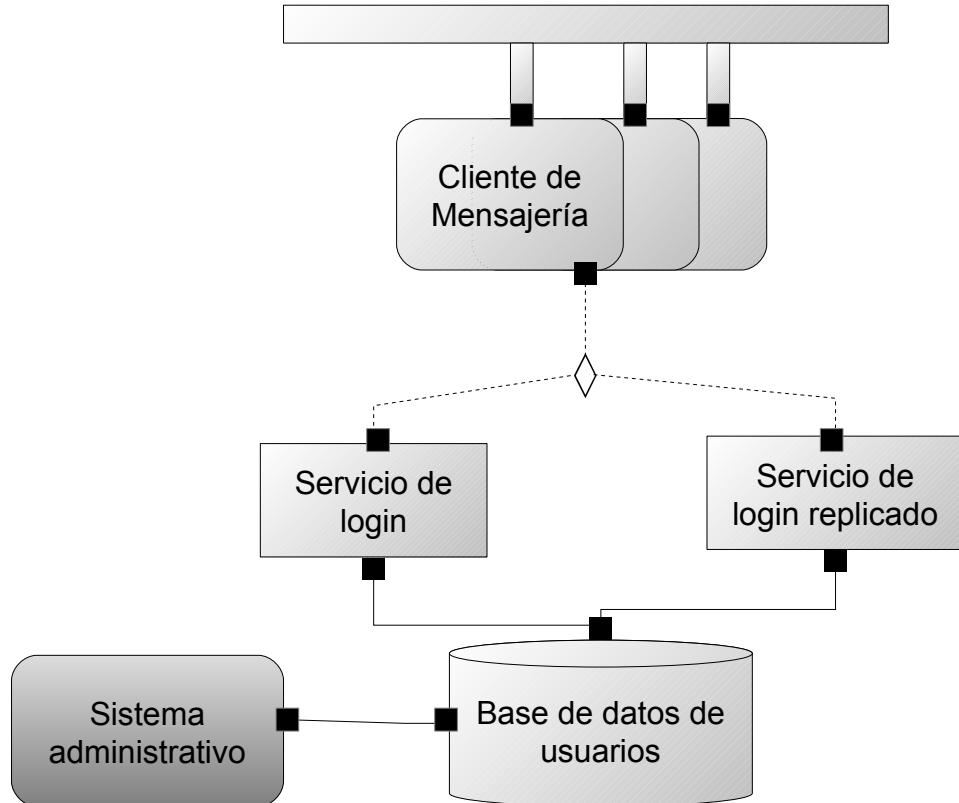
Vistas de CyC

- Ubicuas en la práctica
- Casi todos los diagramas de cajas y flechas que se autodenominan “de arquitectura” son vistas de CyC
 - Aunque generalmente son demasiado informales, están mal explicados y van mezclados con elementos propios de otras vistas

¿Para qué se usan?

- Muestran cómo funciona el sistema
- Guían el desarrollo especificando estructura y comportamiento de los elementos en tiempo de ejecución
- Permiten razonar sobre la calidad del sistema en tiempo de ejecución
 - Prestaciones, fiabilidad, disponibilidad

Representación informal



Leyenda

■	Interfaz
□	Cliente
■	Aplicación
□	Servicio
○	Almacén de datos
—	Canal de comunicación por eventos
—	Acceso a base de datos
◇	Petición-respuesta en cliente-servidor

Representación informal ...

- ¿Qué representa el diagrama?
 - Una “Vista primaria” de la arquitectura del sistema en tiempo de ejecución
- Componentes
 - Tenemos un repositorio compartido con cuentas de clientes accedido por dos servidores y un componente administrativo
 - Los clientes se comunican entre sí por eventos P/S
 - Los clientes interaccionan con los servidores vía C/S
 - Los dos servidores mejoran la disponibilidad
 - El componente de administración mantiene la BBDD

Representación informal ...

- Cada tipo de conector muestra una forma diferente de interacción entre los componentes
 - C/S
 - Permite recuperar información de manera concurrente y síncrona (vía peticiones de servicio). Soporta comutación por error de manera transparente
 - Acceso a BBDD
 - Soporta acceso transaccional y autenticado (lectura, escritura y monitorización de la BBDD)
 - Publicación/suscripción
 - Soporta eventos asíncronos (publicación y notificación)

Representación informal ...

- Cada conector representa una forma compleja de interacción y su implementación no será trivial
 - C/S
 - Cómo inician la sesión los clientes
 - Cómo se gestiona el error
 - Cómo terminan las sesiones
- Los conectores no son necesariamente binarios
- Podemos descomponer un componente (incluso un conector) en otra vista q “subarquitectura”
- Sólo hemos nombrado los tipos de componentes y conectores
 - Documentar sus detalles/propiedades q Documentación de soporte

¿Qué preguntas contestan?

- ¿Cuáles son los principales componentes del sistema y cómo interaccionan?
- ¿Qué partes del sistema están replicadas, y cuántas veces?
- ¿Cómo progresan los datos por el sistema conforme este se ejecuta?

¿Qué preguntas contestan?

- ¿Qué protocolos de comunicación se usan?
- ¿Qué partes del sistema se ejecutan en paralelo?
- ¿Cuáles serán las prestaciones, fiabilidad o disponibilidad (estimadas) del sistema?

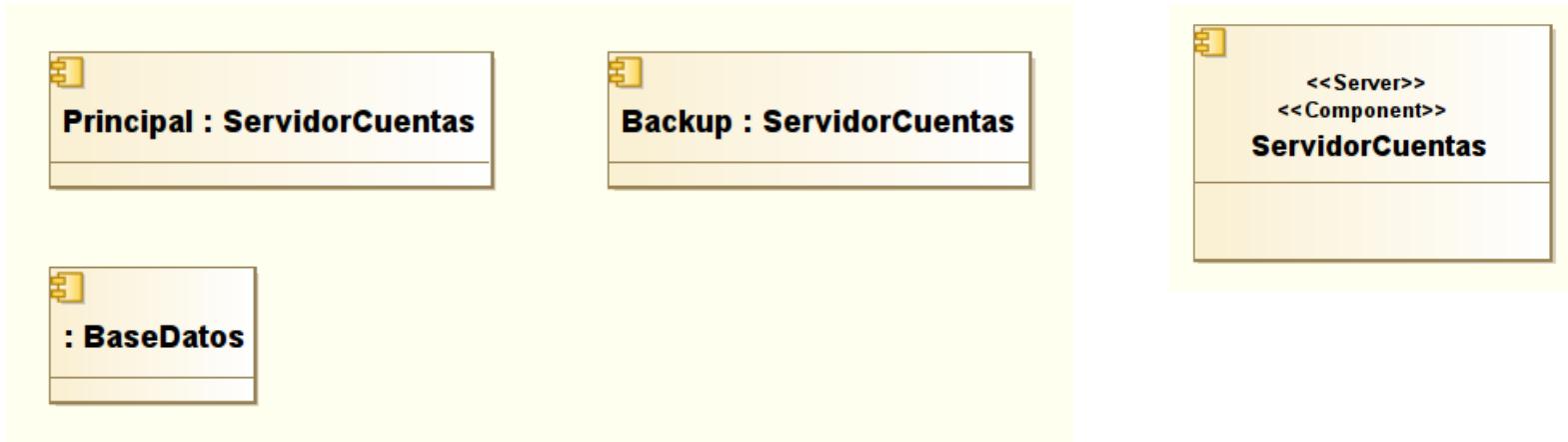
Conclusión

- La combinación de diagramas de CyC con su documentación asociada proporciona un vehículo fundamental para:
 - Comunicar las intenciones de diseño del arquitecto
 - Razonar sobre el comportamiento del sistema
 - Justificar decisiones de diseño en términos de atributos de calidad

Componentes

- Unidades de procesamiento principal y almacenes de datos
- Tienen puertos para interactuar con otros componentes mediante conectores
 - Un puerto es una interfaz de un componente, un punto de interacción con su entorno

CyC en UML: Componentes



- Instancias de componentes en UML (izquierda) y un tipo de componente (derecha)

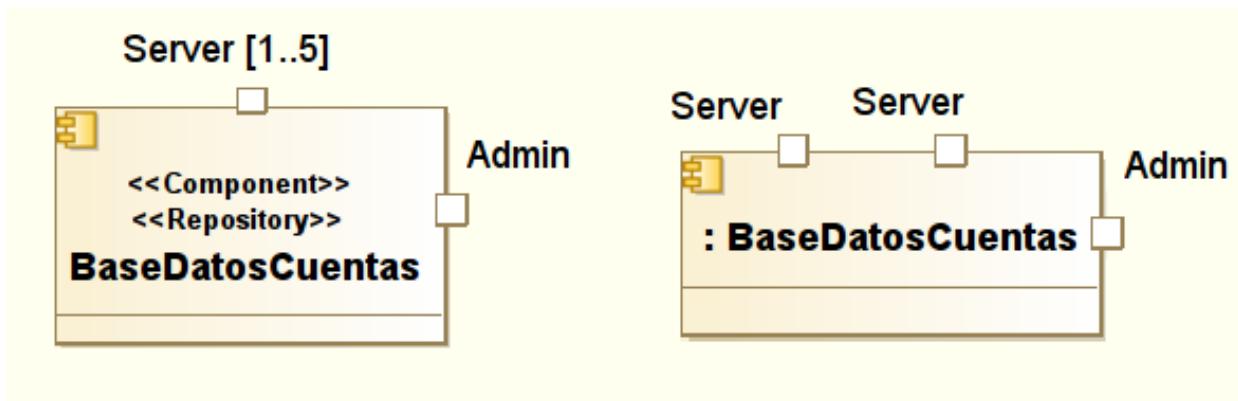
Puertos

- Los puertos suelen tener tipos explícitos
 - Definen el comportamiento que puede dar en la interacción
 - Es necesario documentar los puertos explícitamente en la Documentación de soporte
- Los puertos son diferentes a las “interfaces” de los módulos, ya que se pueden replicar

Puertos ...

- Un componente puede tener varios puertos (mismo o distintos tipos)
 - P. ej. un servidor puede atender concurrentemente a N clientes
- Si tiene “subarquitectura”, hay que documentar la relación entre puertos “internos” y “externos”
 - Mediante delegación de interfaces

CyC en UML: Puertos



- Puertos en un tipo de repositorio (izquierda) y puertos en una instancia de este tipo

Pregunta

- Si tenemos un fichero de texto o XML (configuración, log, datos...) en nuestro sistema, ¿lo podemos/debemos representar como componente en una vista de CyC?



Conectores

- Caminos de interacción entre los componentes
- Pueden ser sencillos ...
 - Invocación de un servicio, una tubería (pipe), cola de mensajes asíncronos, broadcast de eventos
- ...o complejos
 - Canal de comunicación transaccional, bus de comunicaciones entre servicios

Roles

- Tienen roles, que indican cómo los componentes pueden usarlos en sus interacciones
 - Un rol es la interfaz de un conector, un punto de interacción del mismo

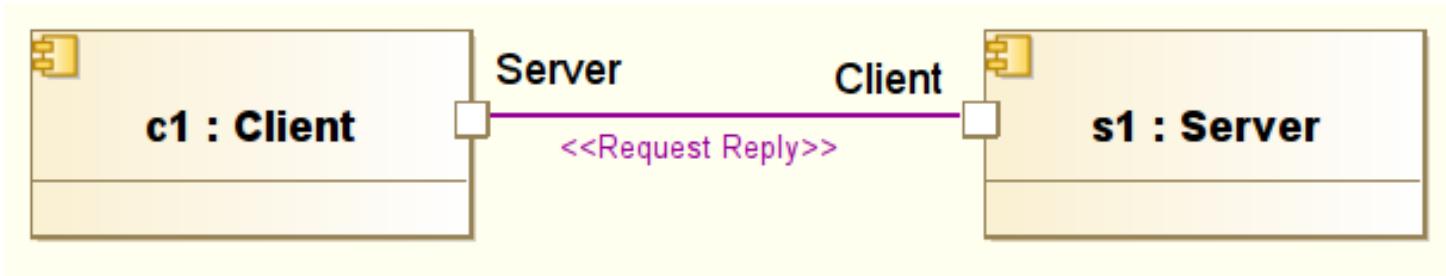
Roles

- Un rol define las expectativas sobre un participante en una interacción
 - P.ej. El rol invoca-servicio puede requerir que el componente invocador (cliente) inicialice primero la conexión
- Esto se puede documentar como una especificación de protocolo

Roles

- No todos los conectores son binarios (tener exactamente 2 roles)
- Algunos sí lo son
 - Un conector C/S puede tener dos roles: invoca-servicio, proporciona-servicio
 - Una tubería puede tener dos roles: lector y escritor
- Algunos tienen varios roles, de los mismos o distintos tipos
 - Un conector P/S puede tener varios roles para publicadores y varios roles para subscriptores

CyC en UML: Conectores



- Los conectores sencillos de CyC se pueden representar como conectores UML (rectas), de puerto a puerto, y con un estereotipo indicando el tipo de conector

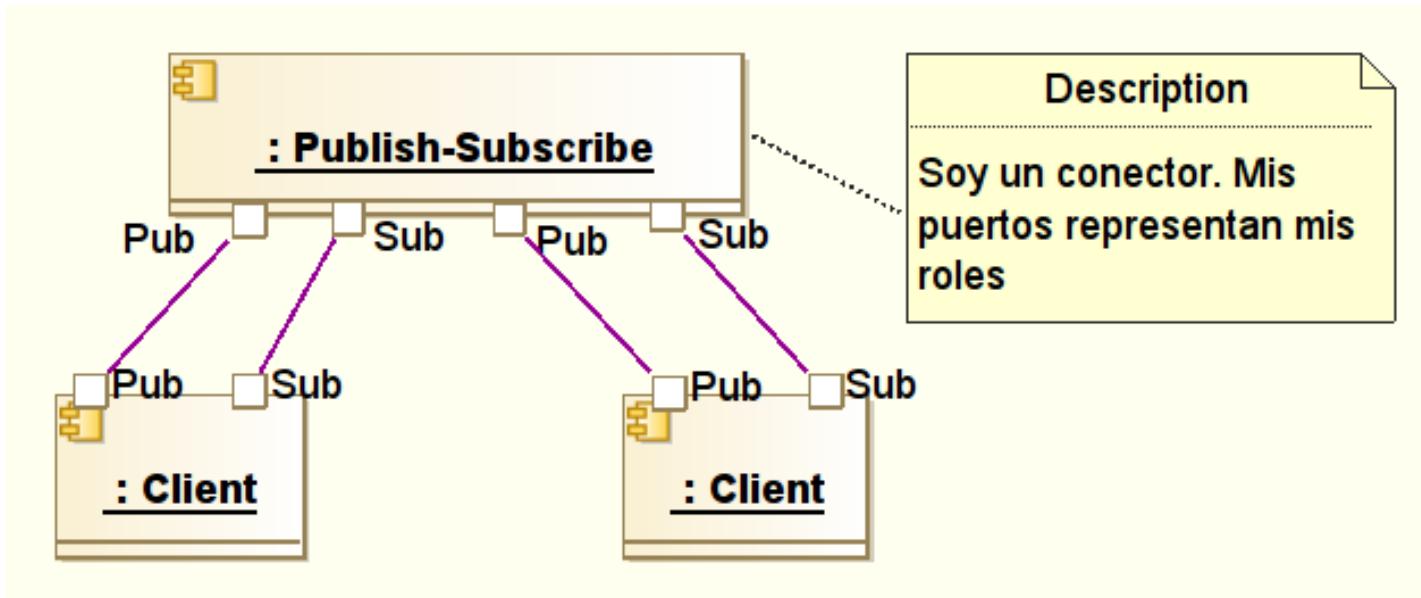
CyC en UML: Conectores

- Para poner roles en el conector se pueden usar las etiquetas de los extremos del conector
 - Generalmente lo que etiquetaremos serán los puertos
- Información adicional se puede incluir en una nota o en una etiqueta (*tag*) del conector

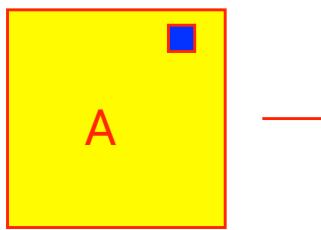
Conectores complejos

- Si fabrico un componente que es un mediador en una interacción entre otros, puede representarse mejor como un componente

CyC en UML: Conectores



- Los conectores complejos de CyC se pueden representar como componentes UML



gunta



- ¿Que diferencias hay entre modelar dos procesos que se comunican por sockets, o dos procesos que se comunican por RPC (p.ej. Sun RPC) en una vista de CyC?

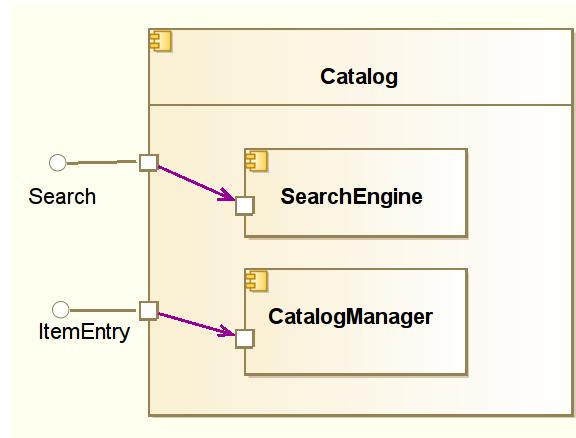


Relaciones

- Acoplamiento (*attachment*)
 - Los puertos de los componentes se asocian con los roles de los conectores formando así grafos de componentes y conectores
 - Será válido cuando puertos y roles sean compatibles, de acuerdo a las restricciones que establezcamos en la documentación

Relaciones

- Delegación de interfaces
 - Los puertos [externos] del componente se asocian con otros internos del mismo (de su “subarquitectura interna”)
 - También posible para roles



Restricciones

- Los componentes solo pueden acoplarse a conectores
 - Y los conectores solo a componentes
- Los acoplos solo pueden hacerse entre puertos y roles compatibles
- La delegación de interfaces solo es válida entre puertos compatibles
 - O entre roles compatibles
- Los conectores no pueden estar aislados

Propiedades

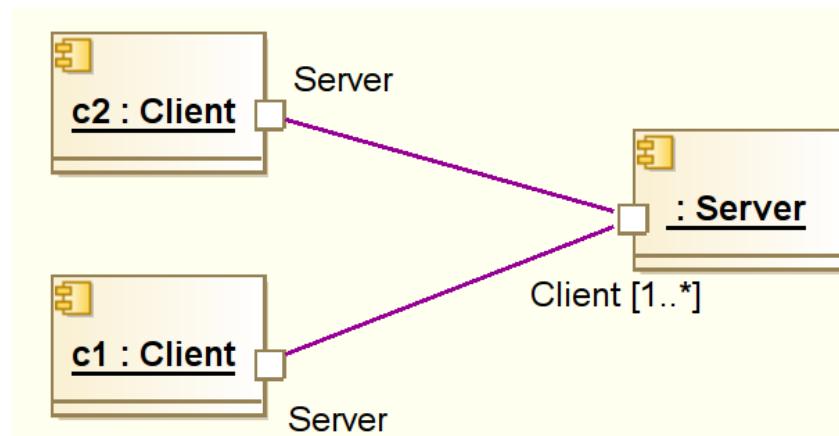
- Los componentes y conectores tienen propiedades
 - Al menos nombre y tipo
 - Y otras dependiendo del tipo: algunas necesarias para guiar la implementación, y otras para realizar análisis sobre la vista
- Puertos y roles también pueden tener propiedades
 - Tasa máxima de peticiones de un puerto servidor

Ejemplos de propiedades

- **Fiabilidad** (% fallo), **prestaciones** (tiempo de respuesta, latencia), **requisitos** (memoria necesaria), **seguridad** (¿requiere autenticación?), **conurrencia** (¿usa su propio thread?), **tasa de peticiones máxima soportada** (propiedad del puerto de un servidor)

Pregunta

- De este diagrama de una vista de CyC, ¿qué podemos deducir sobre la concurrencia en las relaciones entre clientes y servidor?



Tipos e instancias

- En una vista CyC suele haber instancias de componentes y conectores de varios tipos
- Un tipo es un componente o conector no definido completamente
- Una instancia es el resultado de completar esa definición
 - Pero siendo conforme al tipo

Tipos e instancias

- Las guías de estilos arquitecturales proporcionan tipos
 - Suficientes para ilustrar lo esencial de los elementos de ese estilo...
 - ...pero abstractos: hay que especializarlos para concretarlos
 - La especialización puede ser de dominio de problema y/o tecnológica

Pregunta

- ¿Cuál es el tipo de este componente?
¿Server, ServidorCuentas,
Component?

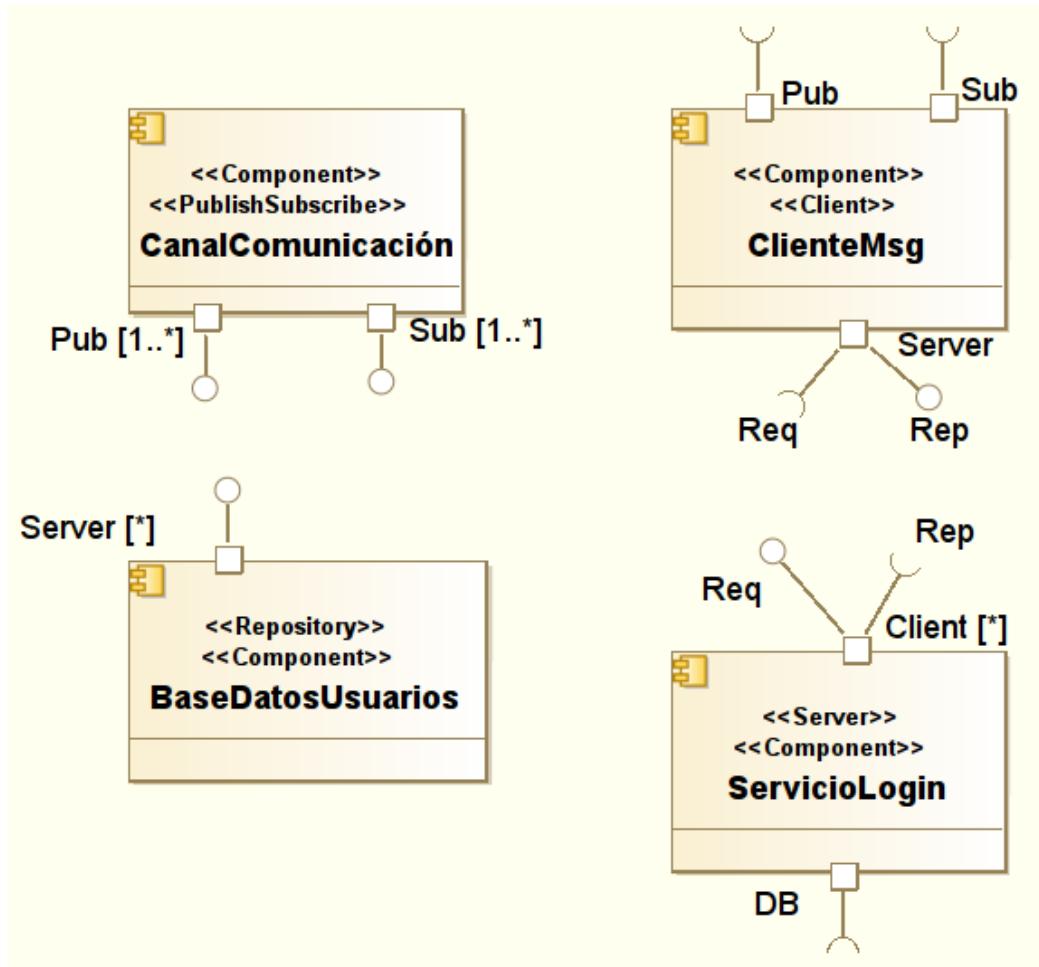


CyC en UML: Interfaces

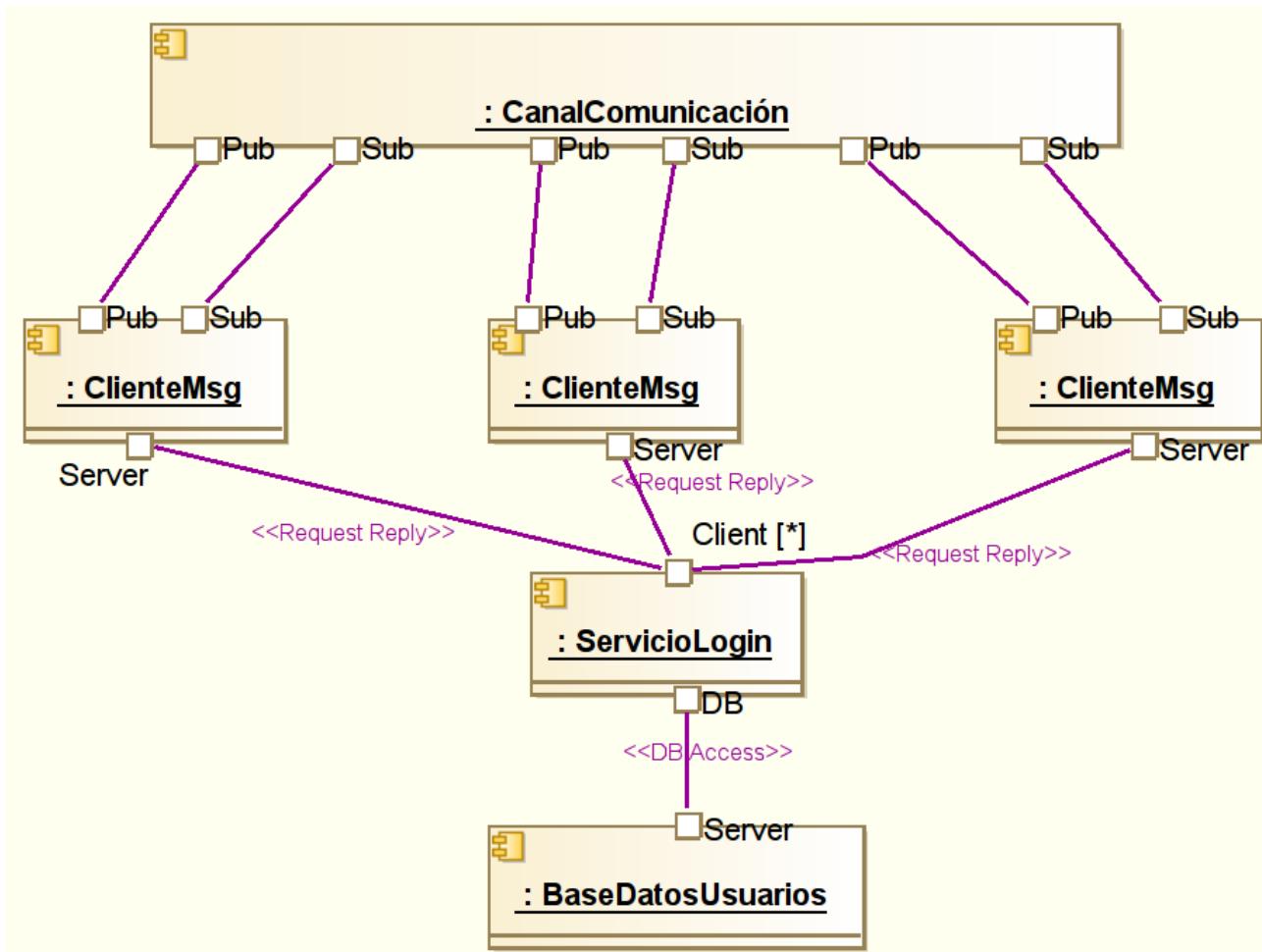


- En UML un componente puede proporcionar o requerir interfaces
 - Van asociadas a un puerto; un puerto puede proporcionar y requerir varias
 - Evitar las interfaces en las instancias de los componentes. Usarlas en la definición de los tipos

Representación en UML



Representación en UML



Pregunta

- Una vista de CyC ilustra un sistema en “tiempo de ejecución”. ¿Es una foto fija, una traza de una ejecución, la unión de todas las trazas posibles, alguna combinación de esto, otra cosa...?



Relación con otras vistas

- La relación con la vista de módulos (código) puede ser compleja
 - Mismo módulo ejecutado por muchos elementos de CyC, un componente de CyC ejecutando código definido en varios módulos, un componente CyC con varios puertos cuyas interfaces están todas definidas por el mismo módulo etc.

Relación con otras vistas

- La relación con las vistas de distribución (despliegue, instalación) suele ser más sencilla
 - Las vistas de CyC representan elementos de tiempo de ejecución, y suele ser útil, y bastante inmediato, relacionarlos con las plataformas físicas y canales de comunicación donde se ejecutan

Algunos estilos de CyC

- Filtro y Tubería (*Pipe-and-Filter*)
- Cliente-Servidor (*Client-Server*)
- Peer-to-Peer (*P2P*)
- Arquitectura Orientada a Servicios (*Service-Oriented Architecture*)
- Publicador-Subscriptor (*Publish-Subscribe*)
- Datos compartidos (*Shared-Data*)