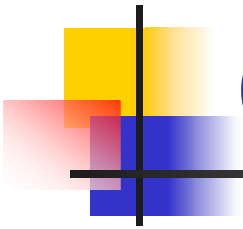
Servicios Web y MDA

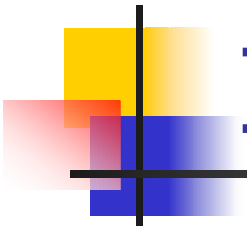
Antonio Vallecillo
Universidad de Málaga
Depto. Lenguajes y Ciencias de la Computación
av@lcc.uma.es
http://www.lcc.uma.es/~av



Contenido

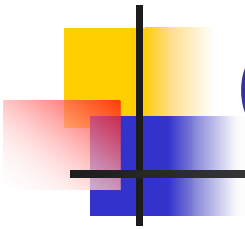
- Objetos, componentes y Servicios Web
  - ¿Qué son y cómo se relacionan?
  - Nuevos acrónimos y estándares (SOAP, WSDL, UDDI)
- Implementación de Web Services
  - Arquitectura de desarrollo
  - CORBA, EJB y .NET
  - Otros aspectos: seguridad, transacciones,...
- Model Driven Architecture (MDA)
  - ¿Qué es y qué aporta?
- Conclusiones

Parte I: Introducción



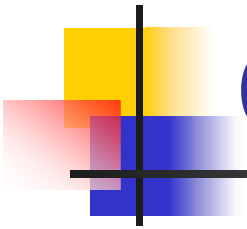
Introducción

- Al principio (?) eran los objetos
  - Un objeto es un modelo de una entidad
  - Un objeto está caracterizado por su estado o bien por su comportamiento [RMODP-2]
  - Propiedades
    - Encapsulación
    - Herencia como mecanismo de composición
    - Polimorfismo
    - Vinculación dinámica



Objetos: ventajas

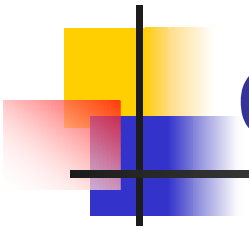
- Mecanismos muy apropiados para diseñar aplicaciones (abstracción, encapsulación,...)
- Amplia difusión y aceptación
- Multitud de lenguajes
- Multitud de herramientas
- Notaciones de modelado (UML)
- Metodologías de desarrollo (RUP, ...)
- Estándares internacionales (OMG, ISO)



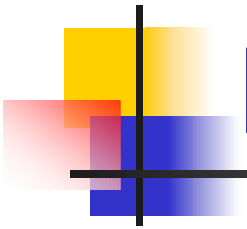
Objetos: desventajas

- ¿Distribución y concurrencia?
- ¿Reutilización por terceros?
- ¿Gestión del ciclo de vida de los objetos?
- ¿Servicios comunes?
- ¿Aplicaciones “abiertas”?
- ¿Interoperabilidad?
- Mecanismos rudimentarios de comunicación
- Mecanismos rudimentarios de composición (básicamente la herencia)

Plataformas de objetos distribuidos (middleware)



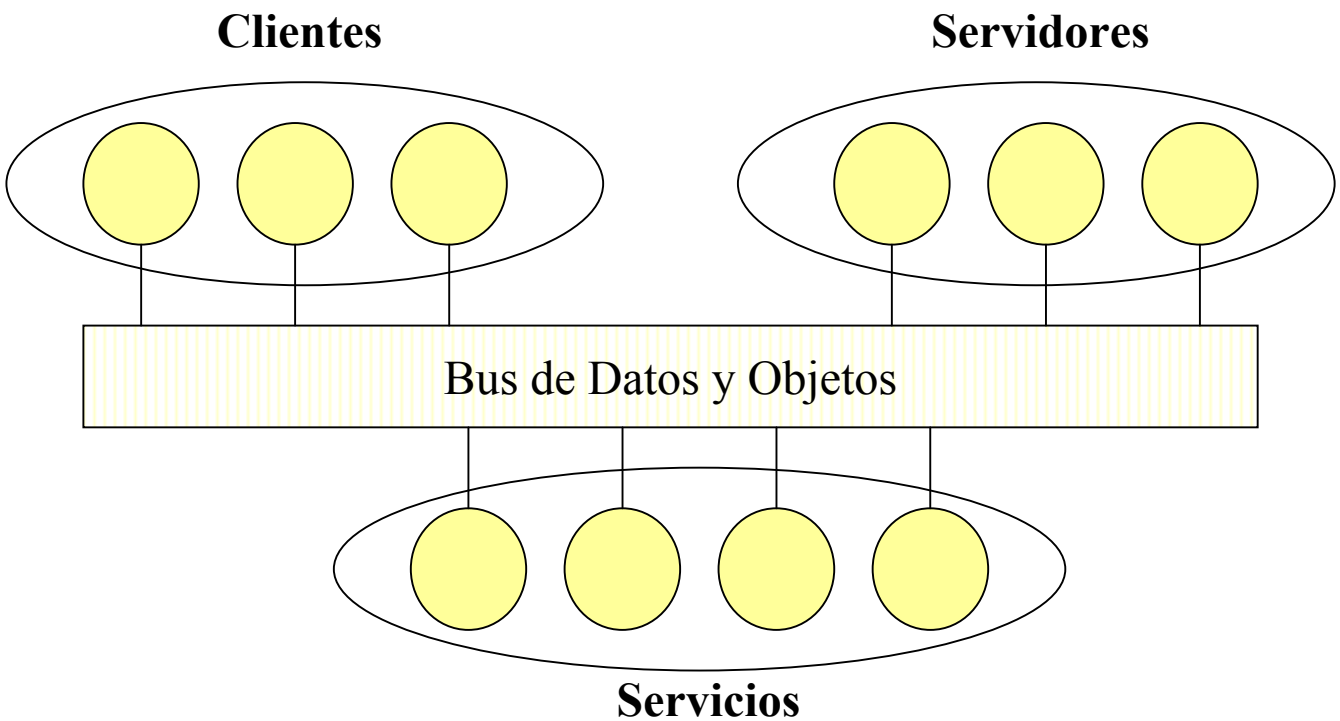
- CORBA, DCOM, Java/RMI
- Resuelven la distribución e interoperabilidad
- Basadas en "modelos de objetos"
- Introducen [Krieger & Adler, 1998]
  - Interfaces
  - Contenedores
  - Metadatos y capacidades reflexivas
  - Entornos de desarrollo integrados (IDEs)
  - Servicios comunes (naming, trading, transacciones, directorios, etc.)



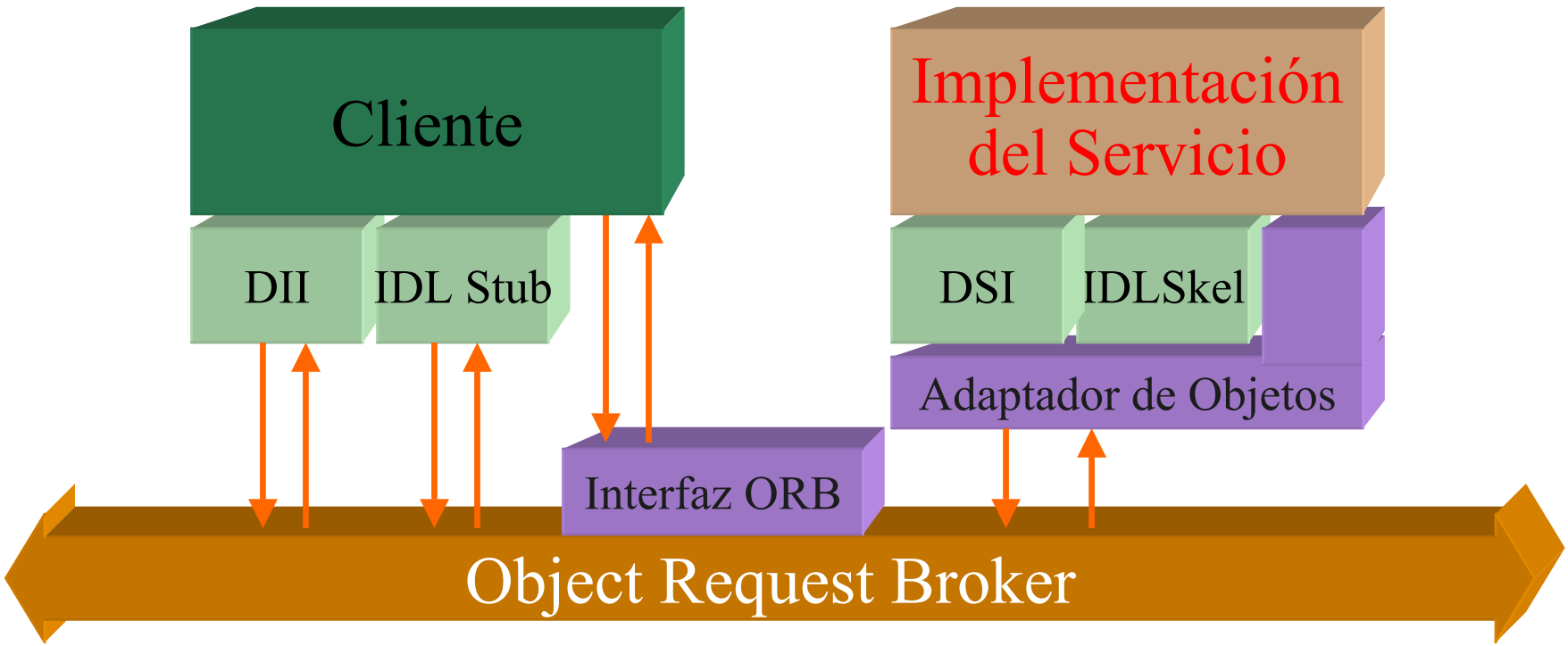
Middleware: conceptos

- Bus de datos para la comunicación entre objetos ("ORB")
- Transparencia de la heterogeneidad, la dispersión y la activación de objetos
- Lenguajes de Descripción de Interfaces (IDL)
- Repositorios de interfaces
- Comunicación síncrona basada en RPCs
- Servicios (seguridad, transacciones, ...)
- Ej: CORBA (1.1, 2.0), JavaBeans y COM

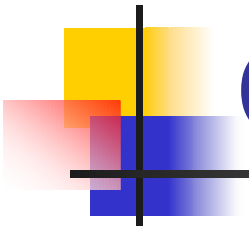
Middleware



Estructura básica de un ORB



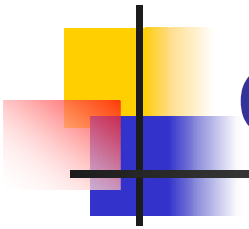
Programación Orientada a Componentes



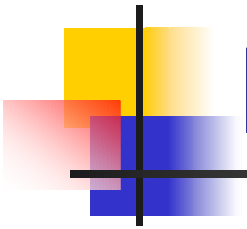
- "Extensión" de la POO
- Sistemas abiertos y distribuidos
- Basada en la noción de *COMPONENTE*

> "Unidad de composición de aplicaciones software que posee un conjunto de requisitos, y que ha de poder ser desarrollado, adquirido, incorporado al sistema y compuesto con otros componentes, de forma independiente en tiempo y espacio"
>
> [Szyperski, 1998]

Características de un componente software

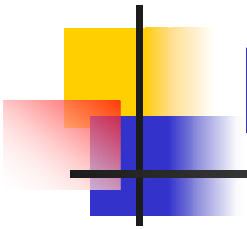


- Introspección
- Eventos y comunicaciones asíncronas
- Enlazado dinámico y composición tardía
- Binario (caja negra)
- Interfaces y *contratos*
- Servicios ofrecidos y requeridos
- Desarrollo independiente del contexto
- Reutilización por composición
- Granularidad (¿?)



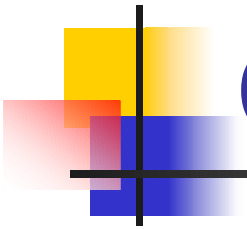
Mercado de componentes

- Reutilización de componentes externos (Commercial-Off-The-Shelf, COTS)
- COTS es una clase especial de componente software, normalmente de grano grueso, que presenta estas características [Meyer 2001]:
  - son vendidos o licenciados al público en general
  - los mantiene y actualiza el propio vendedor, quien conserva los derechos de la propiedad intelectual
  - están disponibles en forma de múltiples copias, todas idénticas entre sí
  - su código no puede ser modificado por el usuario



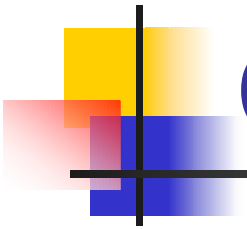
Mercado de componentes

- Actores
  - Proveedores de Software Independientes, intermediarios, ...
  - Usuarios
- Catálogos y repositorios de componentes
- Extensión de IDLs para incorporar información de marketing
  - Autor, Calidad de Servicio, precio/licencia, ...
- Ej: componentsource, flashline, wrldcomp,...



Componentes: ventajas

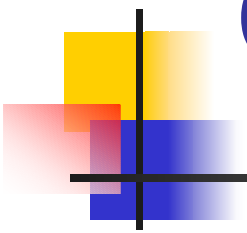
- Solucionan muchos de los problemas de los objetos y la programación distribuida
- Posibilitan la reutilización (pero...¿eso realmente reduce costes?)
- Posibilitan el mercado global de software – a través de componentes COTS (*commercial off-the-shelf*) –
- Interoperabilidad (¿de verdad?)



Componentes: desventajas

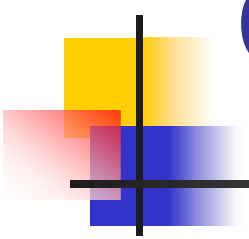
- El mercado global de componentes puede no ser viable
  - Tanto técnicamente....
  - ...como desde el punto de vista económico
- La interoperabilidad no funciona tan bien como la pintan
- Los componentes comerciales son complejos de desarrollar, adaptar y usar
- Los modelos de comunicación entre ellos no se adaptan demasiado bien a Internet

COTS: Problemas técnicos (comprador)



- ¿Cómo reconozco los componentes que necesito? (COTS *trading*)
- ¿Cómo adapto los componentes COTS de acuerdo a mis requisitos?
  - Si hay que adaptar más del 20% de la funcionalidad de un componente, es mejor desarrollarlo a medida!
- El problema de las "dependencias"
- ¿Cómo gestiono las incompatibilidades, los conflictos, y las lagunas al componer COTS?
- ¿Cómo se gestionan los requisitos "extra-funcionales"?

COTS: Problemas técnicos (vendedor)



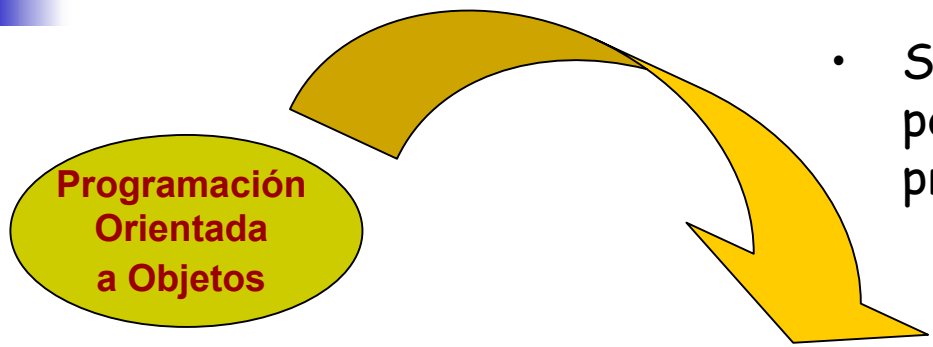
- ¿Cómo diseño los componentes para maximizar su reutilización?

  "Maximizing reuse minimizes use"
- ¿Cómo gestiono las versiones de los componentes? ¿Y el mantenimiento?
- El problema de las "dependencias"
- ¿Para cuántas plataformas desarrollo?
- ¿Cómo se tratan los requisitos extra-funcionales?

COTS: Problemas de mercado

- **El Software: ¿producto o servicio?**
- ¿Me interesa vender mis componentes?
  - ¿Si tengo un componente bueno, quiero vendérselo a la competencia?
  - ¿Lo vendo o lo licencio?
  - ¿Me interesa mantenerlo? ¿Y la formación?
- ¿Qué vende mi compañía, productos o servicios?
- ¿Qué pasa con la calidad?
- ¿Legalmente a qué me comprometo?

La evolución

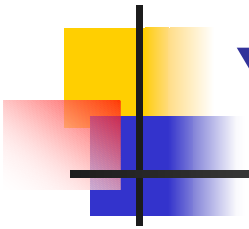


- Sin embargo, la cosa no acaba aquí, porque todavía quedan muchos problemas por resolver....

- Arquitectura Software?
- Requisitos extra-funcionales? [¿aspectos?]
- Interoperabilidad?
- Pervasiveness?
- ....

Otro ingrediente: Internet

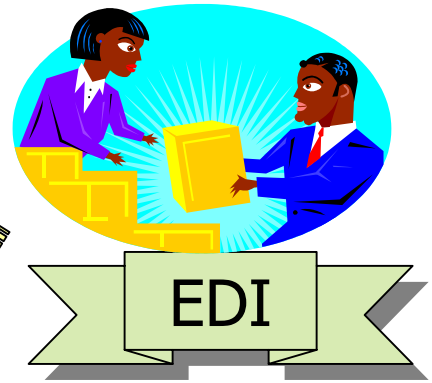
- Nuevas posibilidades
  - ☺ Mejor conectividad y acceso
  - ☺ Gran infraestructura de comunicaciones
  - ☺ Datos al alcance de todos
  - ☺ Simplicidad de protocolos y estándares
- Algunas deficiencias
  - ☹ Seguridad y tolerancia a fallos
  - ☹ Tiempos de respuesta
  - ☹ Distribución de datos, no de computación



Y finalmente: EDI y B2B

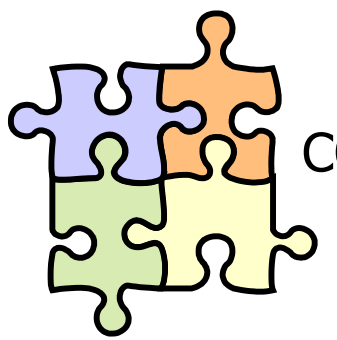
- EDI (Electronic Data Interchange)
  - Estándar para comercio electónico (>10años)
  - Posibilita el B2B
  - Difícil y caro de implementar
  - Alejado de los middleware convencionales
    - Necesita redes y protocolos propietarios (VAN: value-added networks) entre participantes
  - ...pero funciona!

¡Web Services!

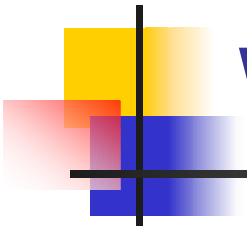


WebServices

Middlewares



Parte II: Definición y caracterización de Web Services

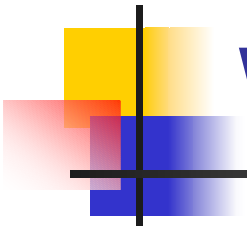


Web Services (1)

- "Modular, self-describing applications that can be published, located and invoked from anywhere on the Web or a local network.

  The provider and the consumer of the Web service do not have to worry about the operating system, language environment, or component model used to create or access the service, as they are based on ubiquitous and open Internet standards, such as XML, HTTP, and SMTP"

[Claudwell et al, 2001]



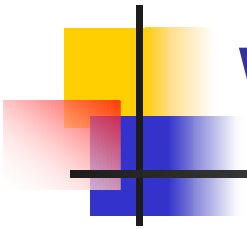
Web Services (2)

- "Internet-based modular applications that perform a specific business task and conform to a specific technical format"

[Mark Colan, IBM]

- "An abstraction of a service provided by some organization as visible from a Web-enabled client, utilizing the www as transport medium, and www transport protocols and formats"

[M. Koethe, MediaOne]

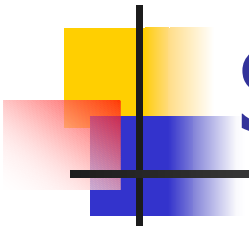


Web Services (y 3)

- "A software application identified by an URI, whose interfaces and binding are capable of being identified, described and discovered by XML artifacts and supports direct interactions with other software applications using XML based messages via Internet-based protocols"

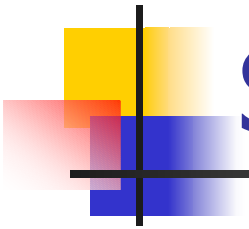
[W3C, 2002]

Qué ofrecen de nuevo los Web Services? (técnicamente)



- *Pervasiveness*: acceso a servicios desde cualquier sitio en la red
- Mejor interoperabilidad
- Simplicidad (..¿qué precio pagamos por eso?)
- Permiten pasar de las típicas aplicaciones Web (2-tier), a aplicaciones más complejas (n-tier)
- Menor acoplamiento y mayor granularidad que la que se consigue con los componentes

Qué ofrecen de nuevo los Web Services? (marketing)



- Alquiler de servicios externos frente a desarrollo (¿reducción de costes?)
- Alquiler de servicios externos frente a compra de software ¿mejor negocio?
  (eg. Adobe's Distiller)
- Alquiler de servicios propios frente a venta de software ¿mejor negocio?
  (eg. Adobe's Distiller)

Ejemplos de Web Services

- Conversores (moneda, unidades, ...)
- Servicios de cotización en bolsa
- Calculadoras
- Asignación de IDs y GUIDs
- Comprobación del tiempo, el estado del tráfico, precios de subastas, ...

Ojo: todos tienen características comunes

www.xmethods.net
www.salcentral.com

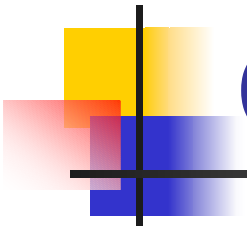
Componentes y Web Services: ¿no son lo mismo?

Web Services	CORBA
SOAP, HTTP	IIOP
URLs	IORs
WSDL	IDL
UDDI	Naming Service, Interface Repository

Componentes vs.Web Services

	Componentes	**Web Services**
Perspectiva arquitectónica	Elementos **internos** de un sistema	Elementos que se ven **desde fuera** del sistema
Modelo de despliegue	Despliegue "físico" (install-and-use)	El sw "existe" en algún lado (connect-and-use)
Niveles de intercambio de información	Mayoritariamente **dentro** de la empresa	Mayoritariamente **entre varias** empresas
Niveles de acoplamiento	Débil	Muy débil
Comunicación	**Middleware** (eg. IIOP)	Web-based (SOAP/XML sobre http)

Fuente: Cutter Consortium



Componentes "y" Web Services

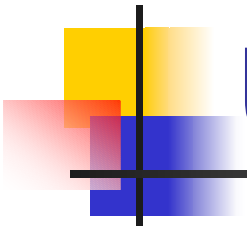
- "Note, however, that all this does not mean that your CORBA objects and EJBs have suddenly become superfluous. On the contrary: they supply the implementations for your web services. Without them, you have no web services"

  [Steve Vinoski]

- "A robust public marketplace for components hasn't emerged. Web Services represent a new revenue stream essential for the future health of the [hardware and software] business"

  [Grady Booch]

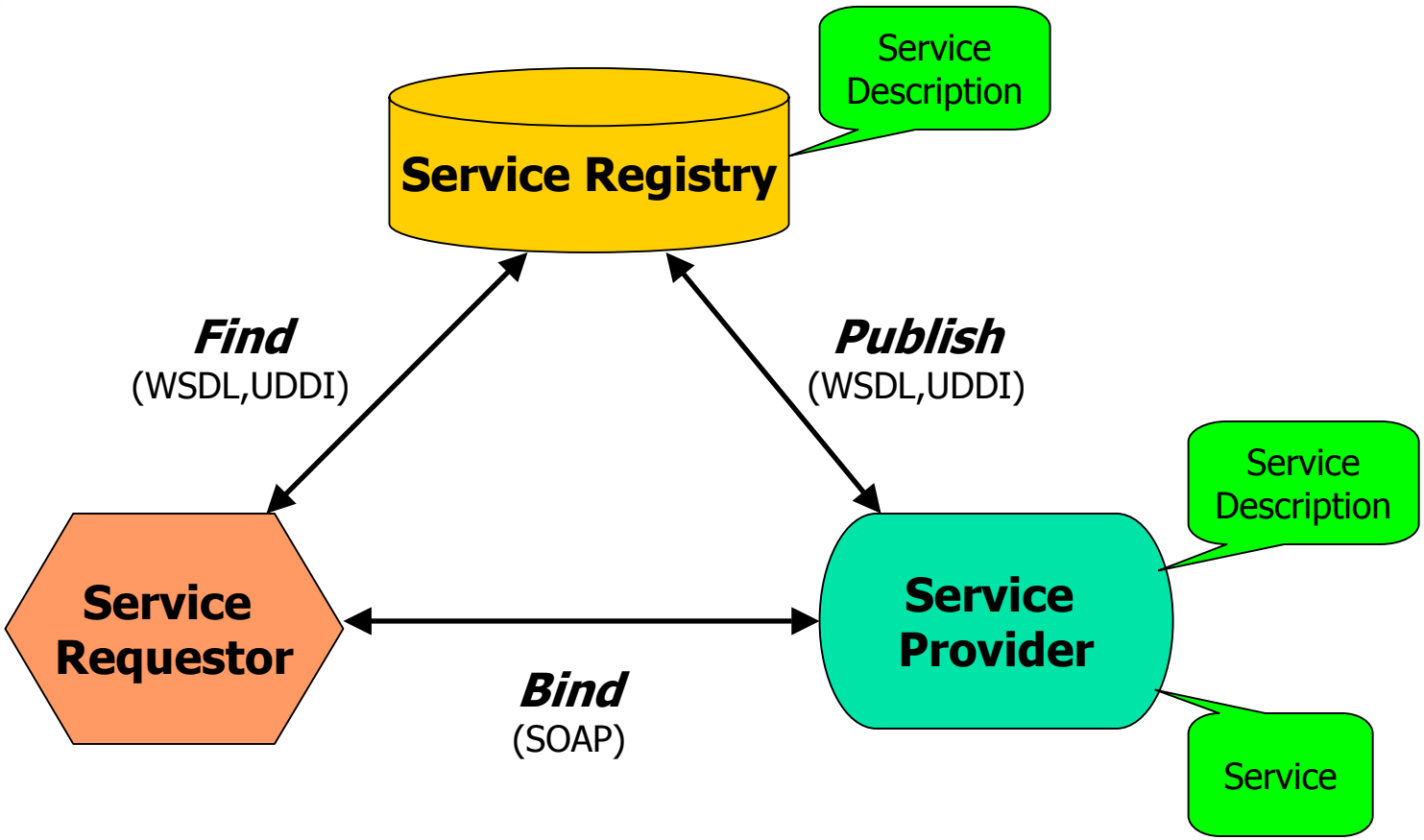
Parte III: Arquitectura y uso de los Web Services



Utilización de Web Services

- **Descubrirlos** (**Discovery**)
  - UDDI – Universal Description, Discovery & Integration
- Conocer **qué hacen** exactamente, y **cómo se usan** (**Description**)
  - WSDL – Web Services Description Languaje
- **Invocarlos** (**Invocation**)
  - SOAP – Simple Object Access Protocol

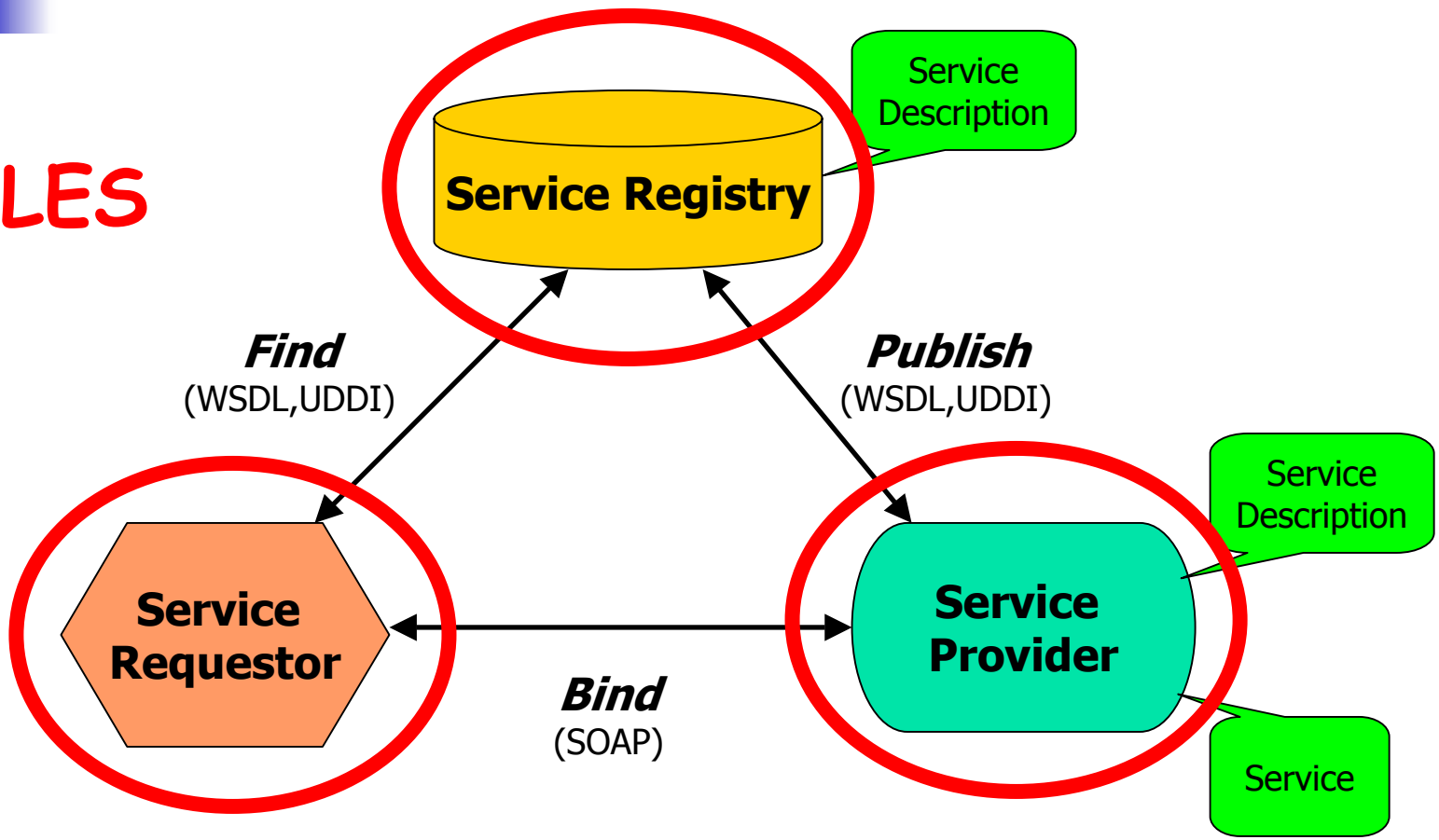
Web Services: Arquitectura



Fuente: IBM Web Services Conceptual Architecture 1.0

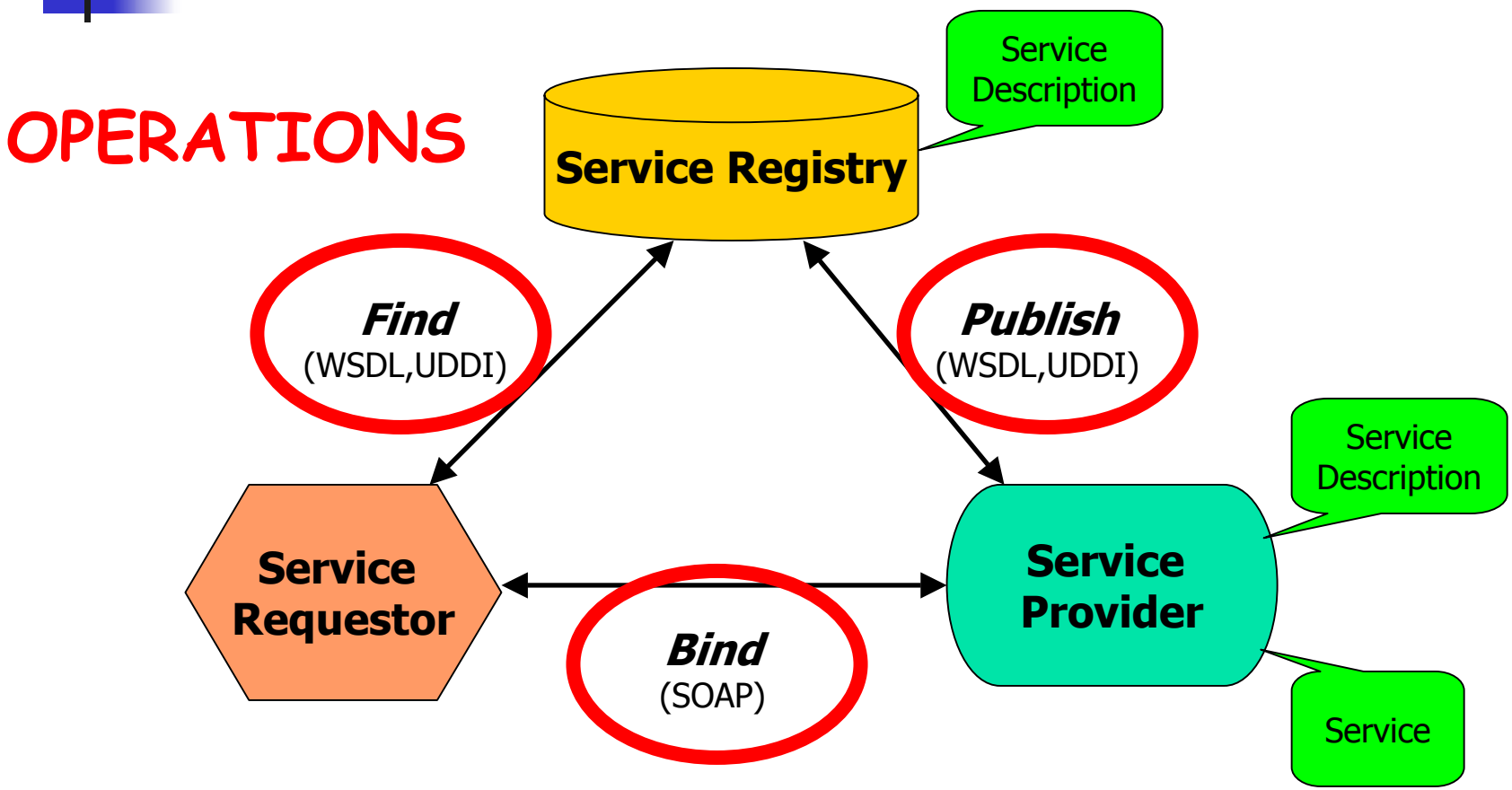
Web Services: Arquitectura

ROLES



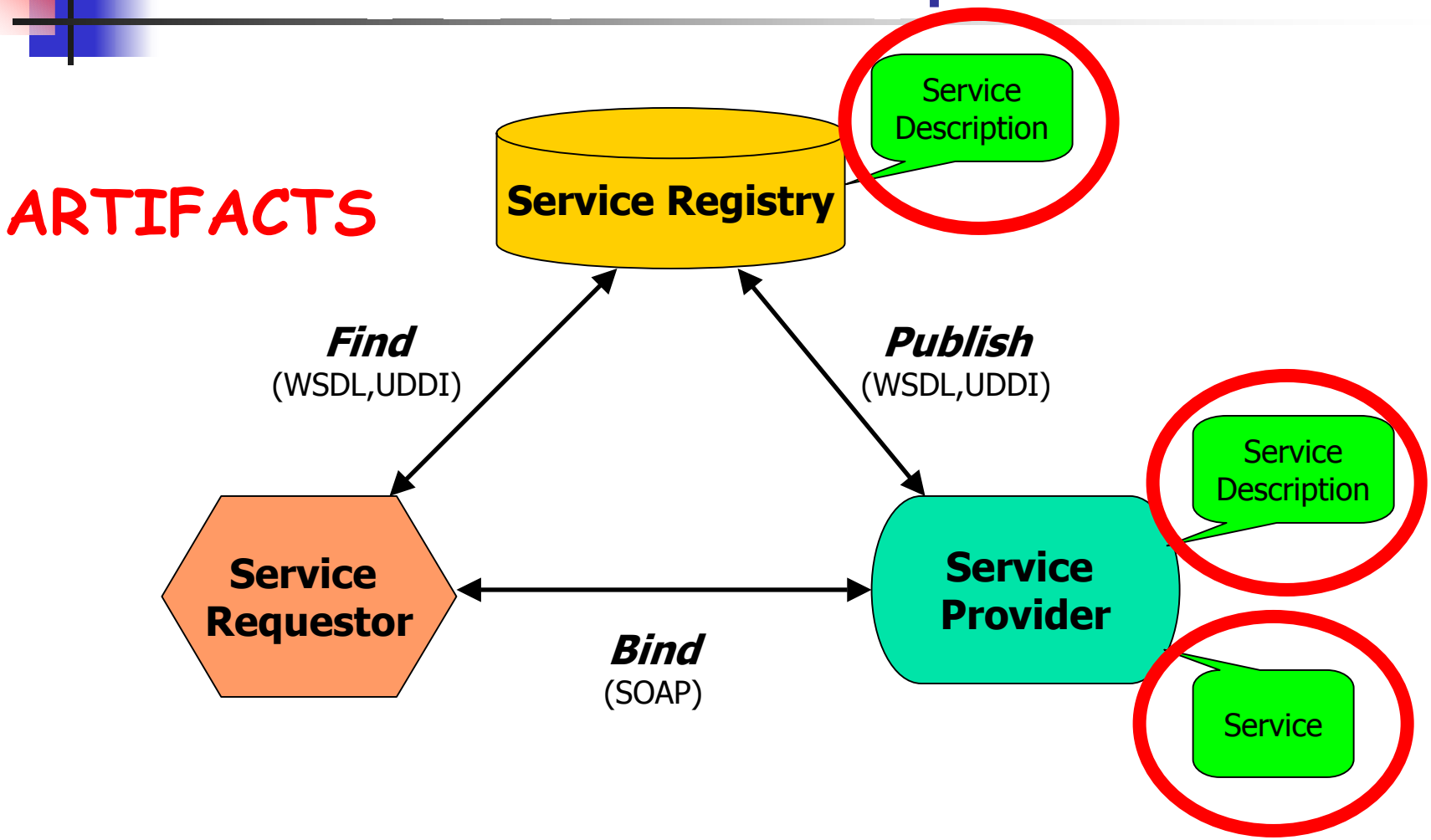
Fuente: IBM Web Services Conceptual Architecture 1.0

Web Services: Arquitectura



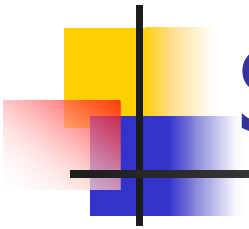
Fuente: IBM Web Services Conceptual Architecture 1.0

Web Services: Arquitectura



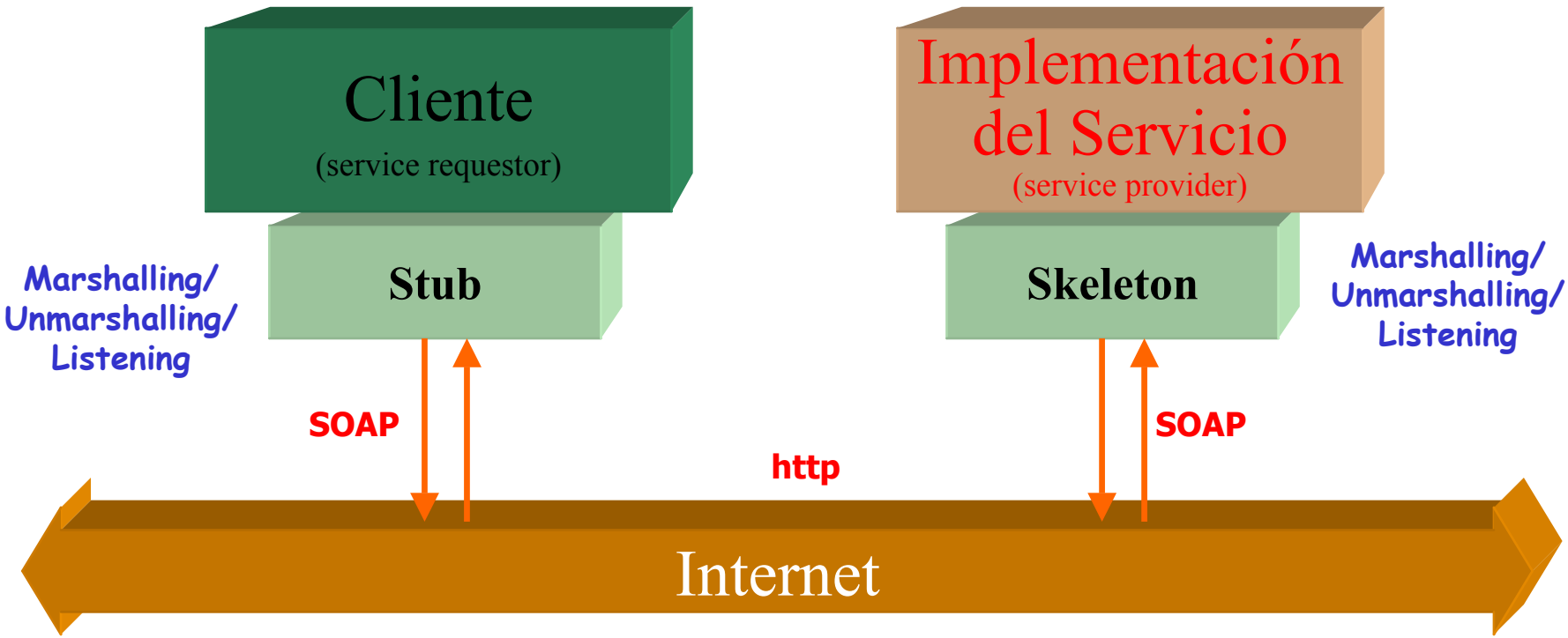
Fuente: IBM Web Services Conceptual Architecture 1.0

Ciclo de vida de los Web Services



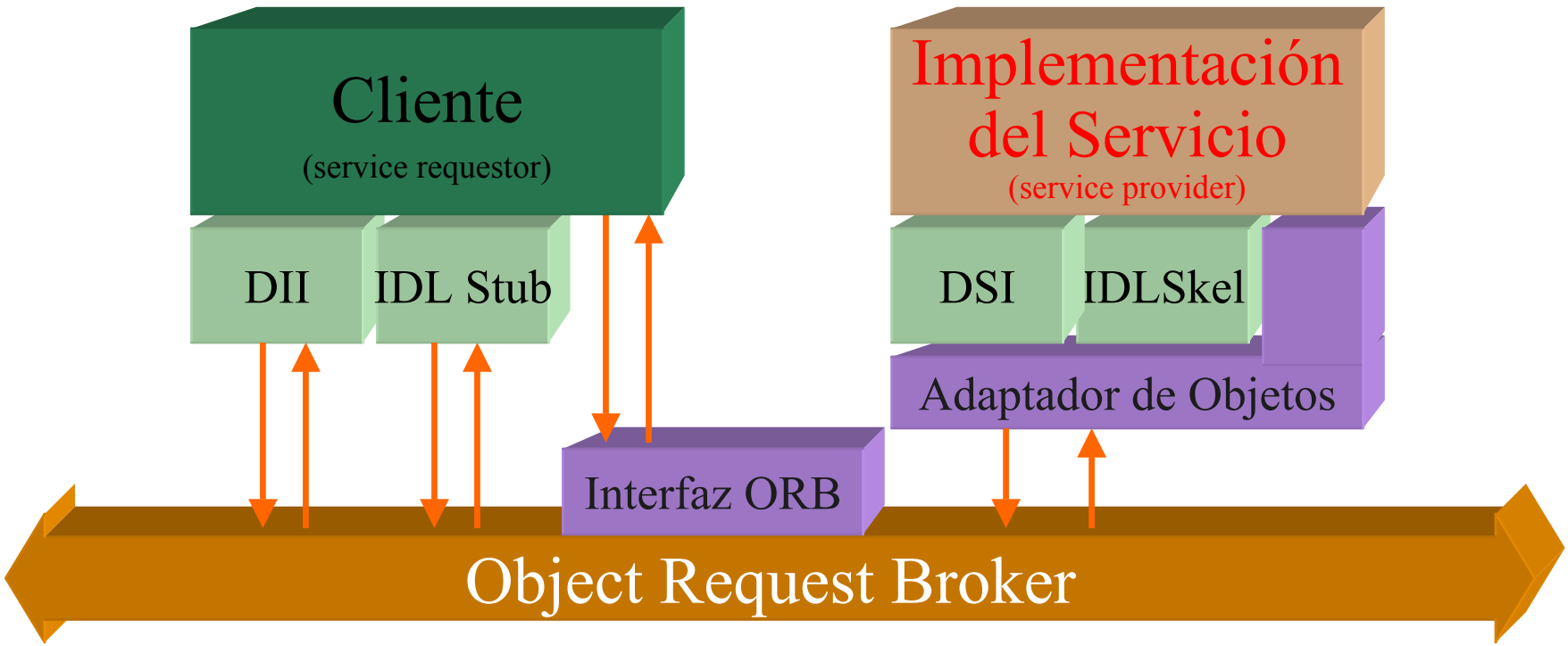
- **Construcción (Build)**
  - Diseño, desarrollo y test del servicio
  - Definición de la descripción de la interfaz
- **Despliegue (Deploy)**
  - Publicación y registro del servicio
  - Despliegue de los ejecutables en la Web
- **Ejecución (Run)**
  - El servicio está operativo y accesible
- **Gestión (Manage)**
  - Gestión y administración del servicio

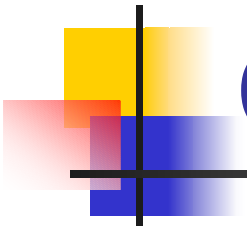
¿Cómo funciona la invocación?



Invocación usando SOAP sobre http

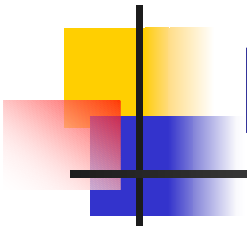
Comparémoslo con el ORB





Comparación

- La arquitectura de los Web Services
  - ☺ Es más simple
  - ☺ Usa Internet y sus tecnologías asociadas
  - ☹ Está menos optimizada
  - ☹ No dispone de POA, servicios comunes,etc.
- En ambos casos existen herramientas para automatizar la creación de los listeners, proxies, stubs, skeletons, etc.

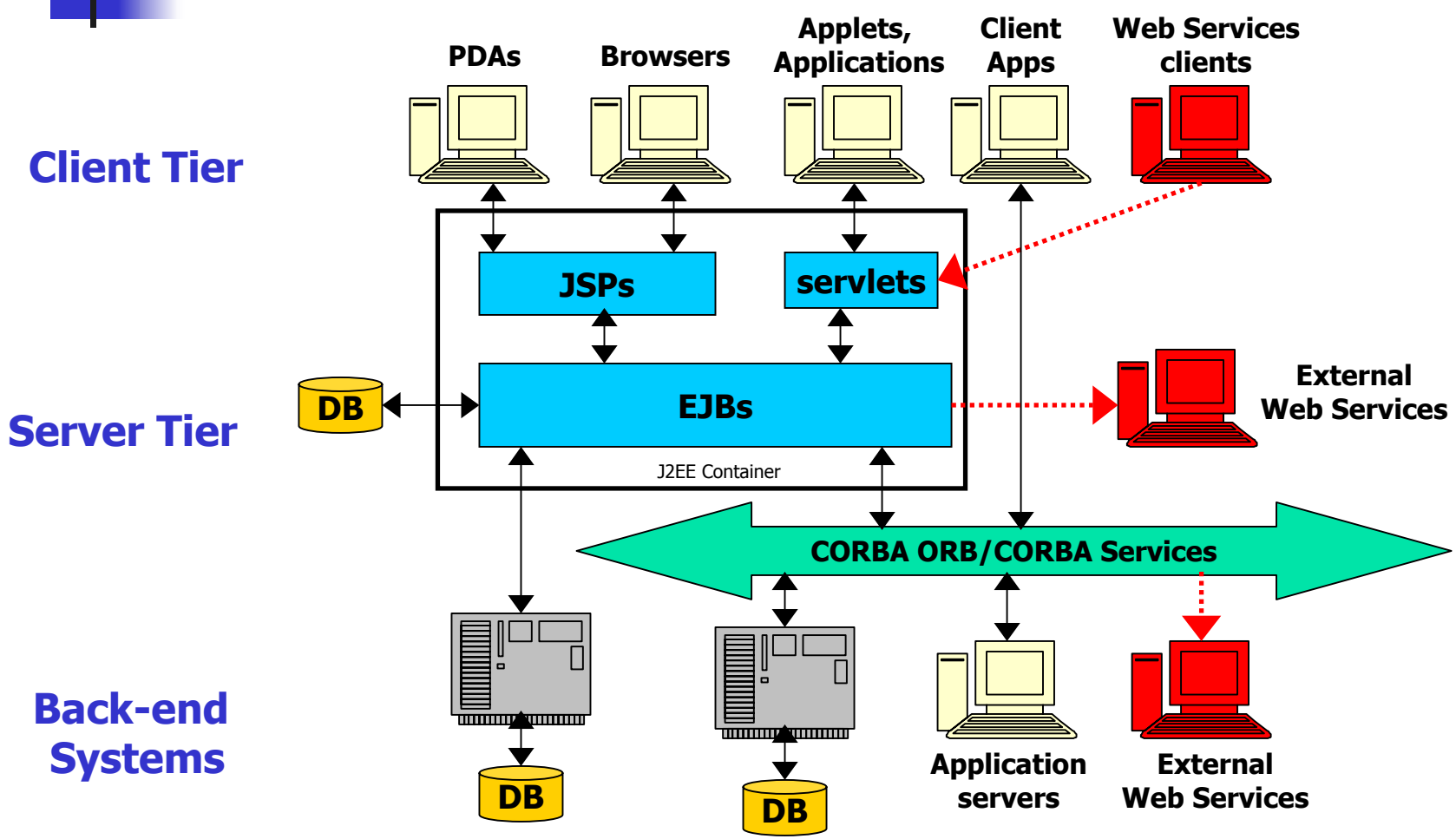


Desarrollo de aplicaciones

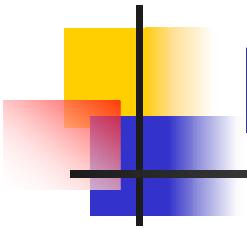
"A well-architected system is typically formed from a set of parts that embody a clear separation of concerns and a balanced distribution of responsibilities"

Grady Booch

Desarrollo de aplicaciones con Web Services: ejemplo

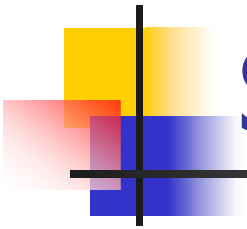


Parte IV: Tecnología y estándares



Estándares técnicos

- SOAP
- WSDL
- UDDI
- *XLANG* (Microsoft) o *WSFL* (IBM)
- *DAML* (DARPA Agent Markup Language)
- *RDF* y la "*Semantic Web*"



SOAP

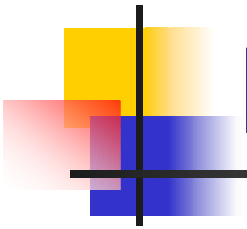
- Estándar de-facto para interconexión
- Permite el intercambio de información estructurada y con tipos entre entidades (peers) descentralizados
- Codificación y empaquetamiento basado en XML para intercambiar datos, mensajes, RPCs
- SOAP proporciona principalmente:
  - La construcción “envelope”,
  - Un conjunto de reglas de codificación,
  - La representación de RPCs (convenciones)

SOAP example

POST /StockQuote HTTP/1.1
Host: www.calculator.com
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn
<SOAP-ENV**:Envelope**
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
SOAP-ENV:**encodingStyle**="http://.../soap/encoding/">
<SOAP-ENV:**Body**>
<m:**plus** xmlns:m="http://www.calculator.com/interface">
<**a**>17</**a**> <**b**>2</**b**>
</m:**plus**>
</SOAP-ENV:**Body**>
</SOAP-ENV:**Envelope**>

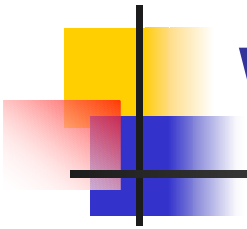
SOAP example (response)

HTTP/1.1 200 OK
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn
<SOAP-ENV:**Envelope**
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
SOAP-ENV:**encodingStyle**="http://.../soap/encoding/"/>
<SOAP-ENV:**Body**>
<m:**plusResponse** xmlns:m="http://www.calculator.com/interface">
<**result**>19</**result**>
</m:**plusResponse**>
</SOAP-ENV:**Body**>
</SOAP-ENV:**Envelope**>



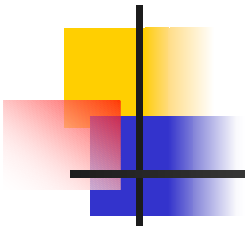
Extensiones de SOAP

- Attachments
- Routing e intermediarios
- Mensajería fiable
- Seguridad (XML Signatures, XML Encryption)
- Calidad de servicio (QoS)
- Context sensitivity (Intelligent Web Services)
- Transacciones



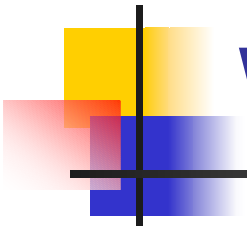
WSDL

- SOAP permite expresar invocaciones y respuestas sueltas
- Pero también es necesario describir los servicios, como colecciones de operaciones y respuestas
- **W**eb **S**ervice **D**escription **L**anguage fue desarrollado por Microsoft e IBM para describir servicios Web
- Independiente del método de transporte o método de codificación final



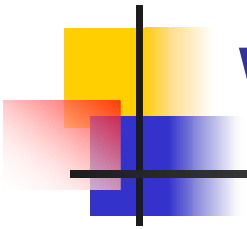
Documentos WSDL

- Un documento WSDL contiene definiciones:
  - **Types** – a container for data type definitions using some type system (such as XSD).
  - **Message** – an abstract, typed definition of the data being communicated.
  - **Operation** – an abstract description of an action supported by the service.
  - **Port Type** –an abstract set of operations supported by one or more endpoints.
  - **Binding** – a concrete protocol and data format specification for a particular port type.
  - **Port** – a [concrete] single endpoint defined as a combination of a binding and a network address
  - **Service** – a [concrete] collection of related endpoints



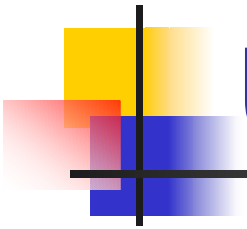
WSDL

- Primitivas de transmisión
  - One-way
  - Request/Response
  - Solicit/Response
  - Notification
- Protocol bindings
  - SOAP
  - HTTP GET/POST
  - MIME



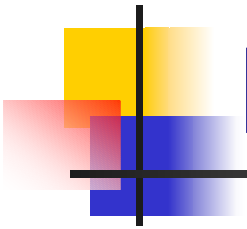
WSDL

- Las descripciones WSDL son complejas y difíciles de construir manualmente
- Los fabricantes ofrecen generadores automáticos de documentos WSDL:
  - Microsoft SOAP Toolkit for COM
  - IBM Web Services Toolkit: Java, EJBs, COM
  - .NET (fue diseñado para trabajar con Web Services)
    - [Web Service] y [WebMethod] attributes.
    - Genera automáticamente los skeletons, listeners, etc.
    - Generan hasta páginas de prueba y acceso a los servicios



UDDI

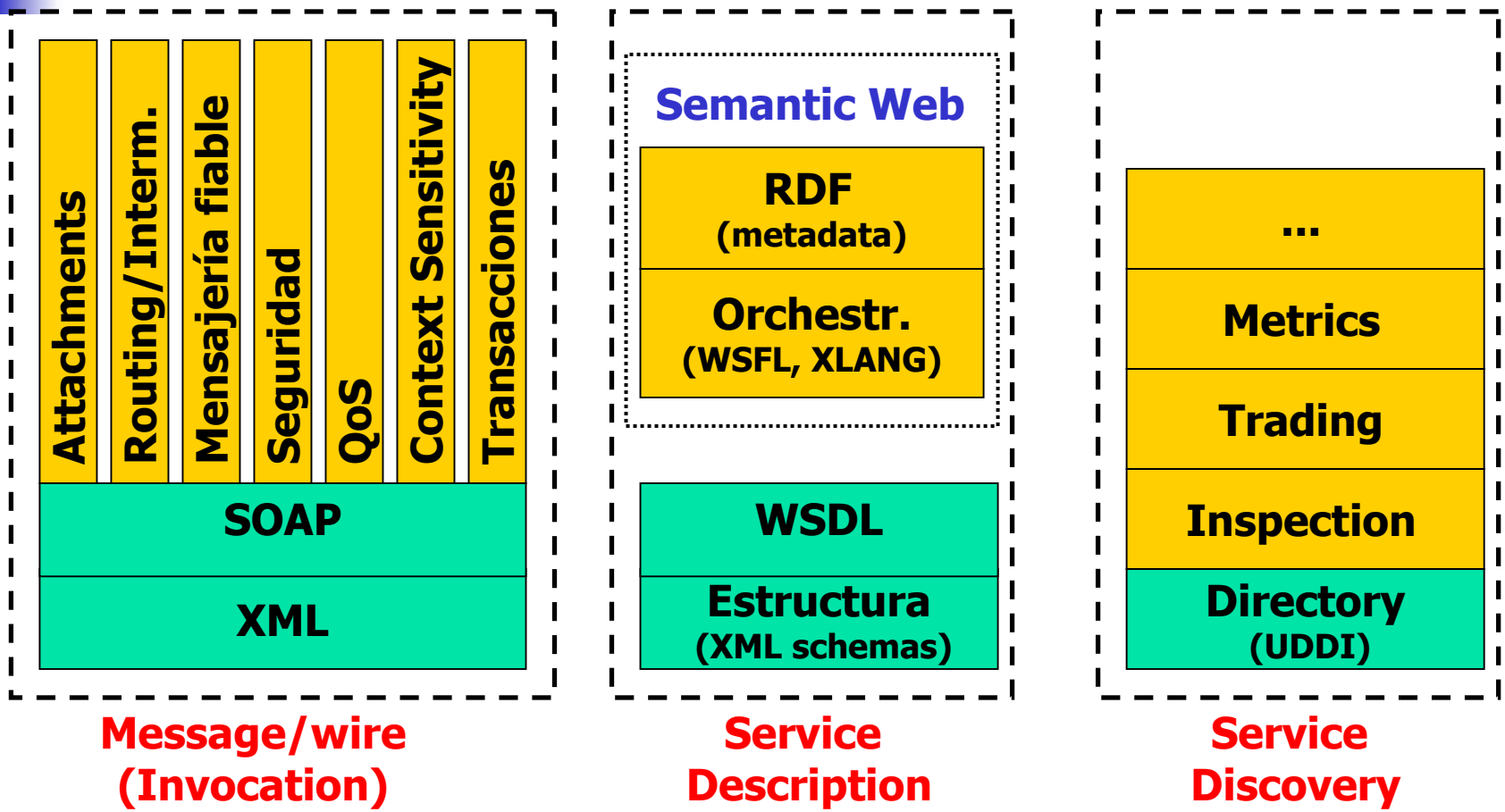
- **U**niversal **D**escription, **D**iscovery and **I**ntegration
  - Desarrollado por IBM, Microsoft y Ariba
- Permite mantener un ***registro global*** de Web Services
  - Con operaciones para: publicar (publish), ojear (browse) y retirar (un-publish) Web Services
  - Registro replicado (y consistente!)
  - Operado por IBM, Microsoft y HP
  - Cualquier aplicación (incluidos Search engines) pueden consultarlo para descubrir servicios

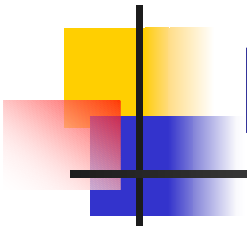


El registro de UDDI

- Utiliza taxonomías estándares para clasificar servicios (NAICS, UNSPSC,...)
- Almacena tres tipos de información
  - White pages
    - Nombre del negocio, informaciones de contacto, etc.
  - Yellow Pages
    - Clasificación de la compañía y el servicio (taxonomías)
  - Green Pages
    - Información técnica sobre los servicios, su descripción, y cómo invocarlos

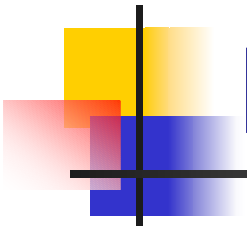
Tecnologías utilizadas





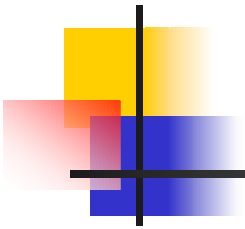
Productos comerciales

- HP's Web Services Platform
- IBM's Web Service Architecture (Web Sphere Application Server)
- Iona's E-Business Platform
- Microsoft's .NET Framework
- Oracle's Dynamic Services
- Sun's Open ONE
- ....



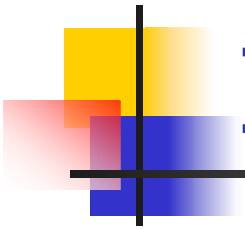
Productos comerciales

- Todos ofrecen herramientas para el desarrollo e implementación de aplicaciones con Web Services *de forma transparente*
  - Parsers de XML, SOAP, WSDL, etc.
  - Invocación de Web Services
  - Búsqueda en repositorios y registros
  - Conexión con otras herramientas (pe. mail)
  - Pruebas y acceso manual desde páginas Web
  - Integración de forma natural en sus entornos de desarrollo convencionales



Muchos cabos sueltos (todavía)

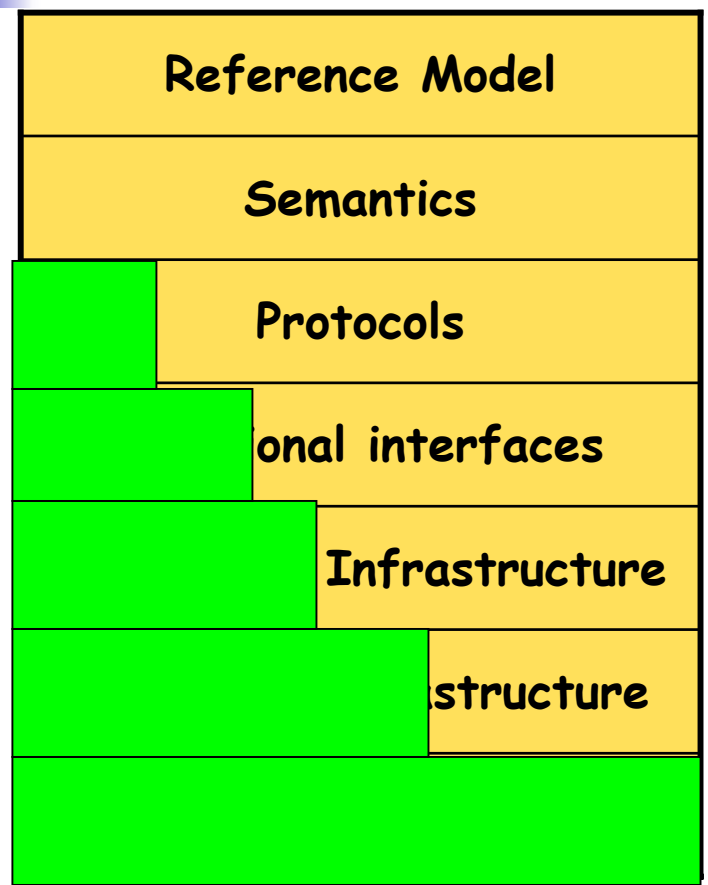
- Asincronía, latencia, fragmentación, fallos en nodos y comunicaciones,....(propios de los sistemas distribuidos)
- Polling: degradación de prestaciones y poca escalabilidad de la solución
- La identidad de los objetos se pierde en los servicios Web
- En general, los Web Services no mantienen el estado
- ¿Interoperabilidad semántica?
- ¿Contratos?
- ¿Calidad de servicio? (definición, monitorización,...)
- ¿Negociación?
- ¿Tarificación? (licencias, acuerdos de uso, re-venta, etc.)
- ¿Legislación?
- Marketing: ¿Es el *software* un producto, o un servicio?



Interoperabilidad

"The only way you can have reliable communication between two parties at the semantic level is if they agree on a common complete type system"

Niveles de interoperabilidad



Extra-functional properties and requirements (QoS, "ilities", "nesses")

Ontologies. "Meaning" and behavior of operations

Service access protocols (choreographies)

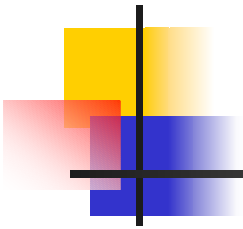
Agreements on data types, operation names and parameter types.

Application specific protocols and conventions (entities identification, comms semantics, exceptions mngt)

Services used (location, naming,transactions, security) and Lifecycle management (POAs)

Access and Interconnection mechanisms used: CORBA, RMI, SOAP, XML, SQL, or combinations of them

Sólo esto cubren los Web Services

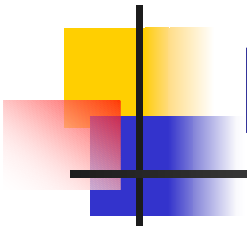


Resumen

Web Services: A new buzzword in town

- ☺ Proporcionan una sólida base sobre la que construir aplicaciones distribuidas en Internet (esp. EDI y B2B)
- ☺ Complementan a las tecnologías actuales de desarrollo de aplicaciones: componentes, middlewares y frameworks
- ☺ Permiten hacer un mejor uso de las facilidades que ofrece Internet en cuanto a programación distribuida
- ☺ Ofrecen no sólo soluciones técnicas a la ingeniería del software, sino también de marketing y comerciales
- ☺ Existen ya herramientas comerciales para su desarrollo y uso industrial
- ☹ Pero [por sí solos] no son una panacea:
  - No son un nuevo paradigma: ofrecen poco más que un "wiring standard"
  - No solucionan todos los problemas de los sistemas abiertos y distribuidos
  - Están en una fase inicial

Parte V: MDA

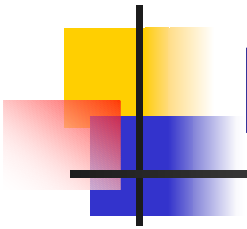


Más allá de la tecnología...

Demasiados modelos y plataformas

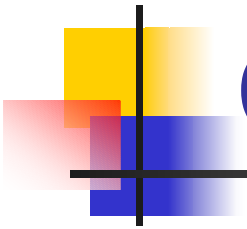
- de componentes, de servicios Web...
- no interoperables realmente!
- ¿Cual es la mejor?

- Evolucionan demasiado deprisa
  - Obsoletas...¿que pasa con mi inversión?
  - ¿Cual es la que saldrá mañana?



MDA

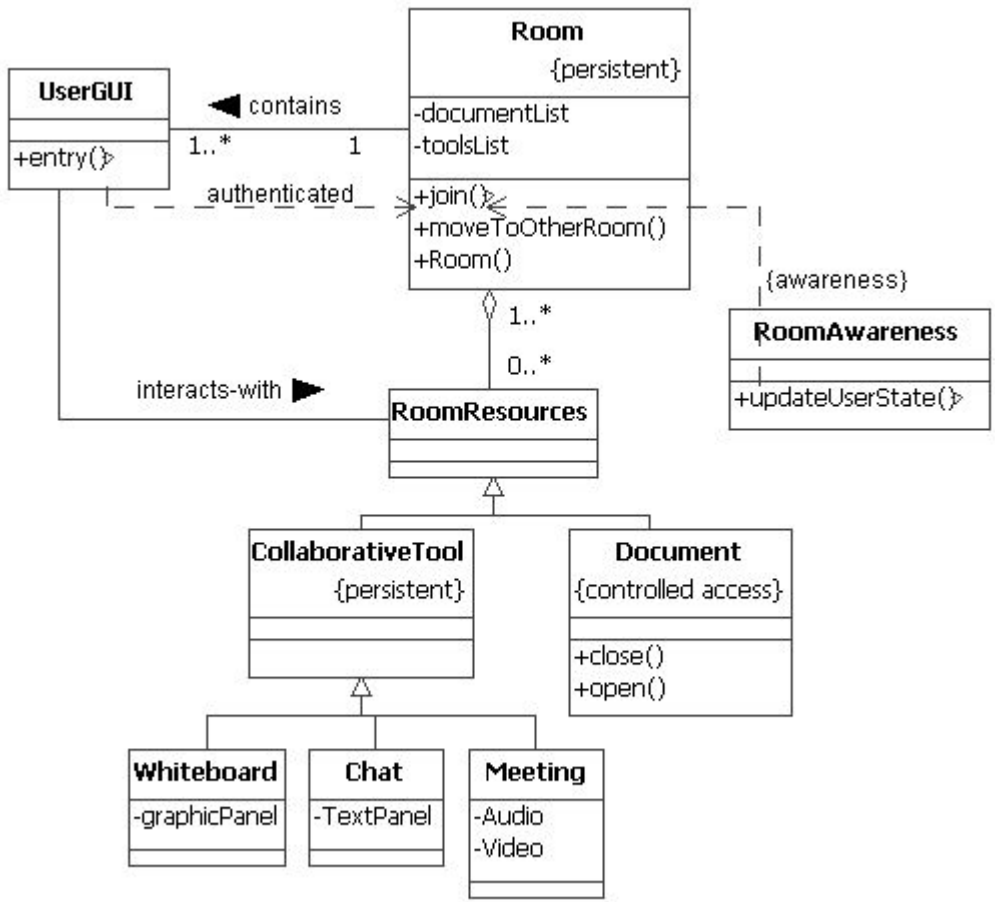
- Nueva orientación de las actividades de la OMG para los siguientes 10 años
- Basada en modelos, no en plataformas tecnológicas
- Define:
  - Modelos independientes de la plataforma
  - Modelos abstractos dependientes de la plataforma
  - Transformaciones entre PIMs y PSMs
- Permite preservar los PIM cuando aparecen nuevas plataformas!!!



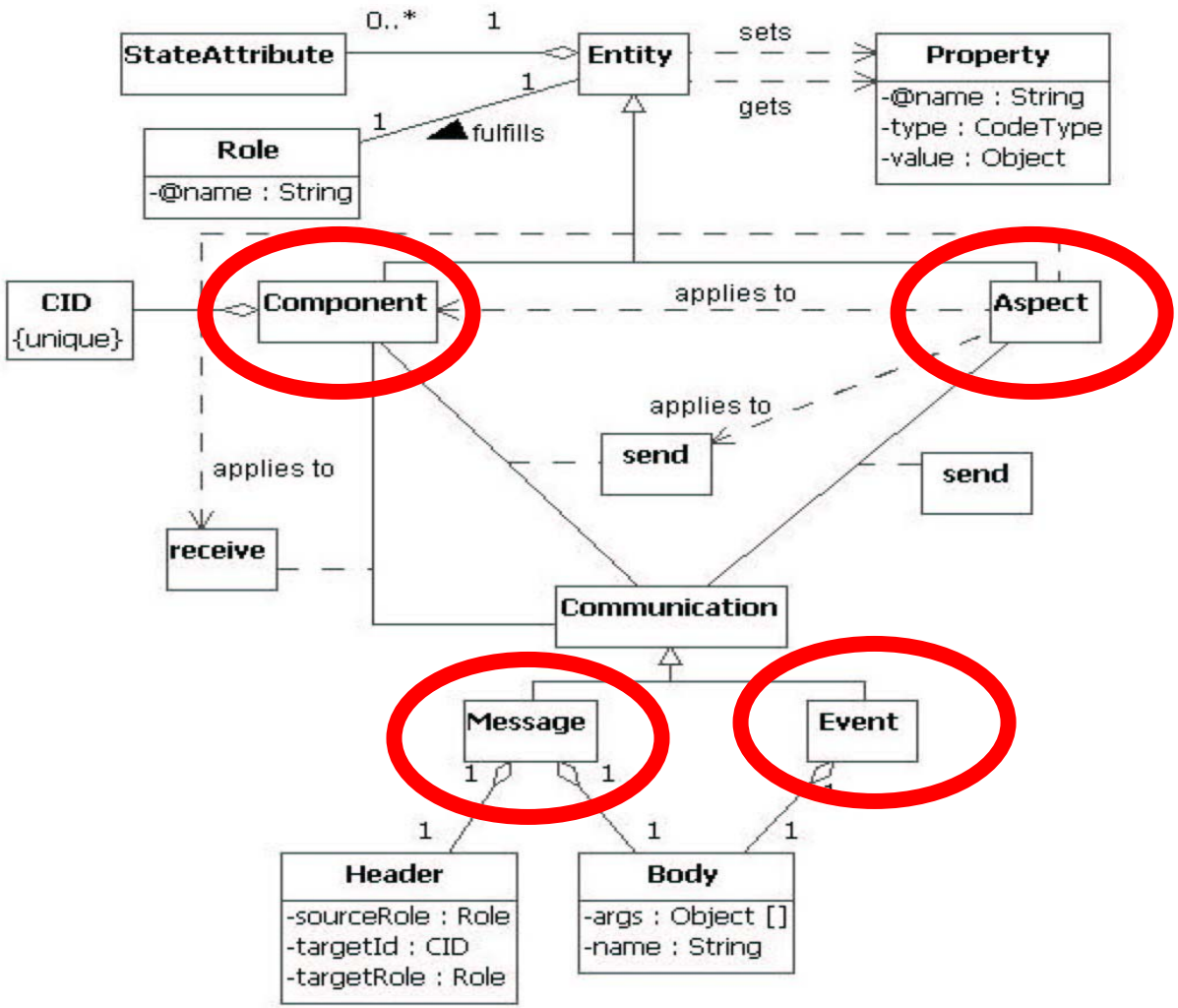
Conceptos claves en MDA

- Modelo
  - Representación de un sistema
- Plataforma
  - Conjunto de subsistemas y tecnologías que proporcionan una funcionalidad clara.
- CIM, PIMs y PSMs
  - Computation Independent, Platform Independent, and Platform Specific Models
- Mappings
  - Mecanismos para transformar elementos de un modelo en elementos de otro modelo.

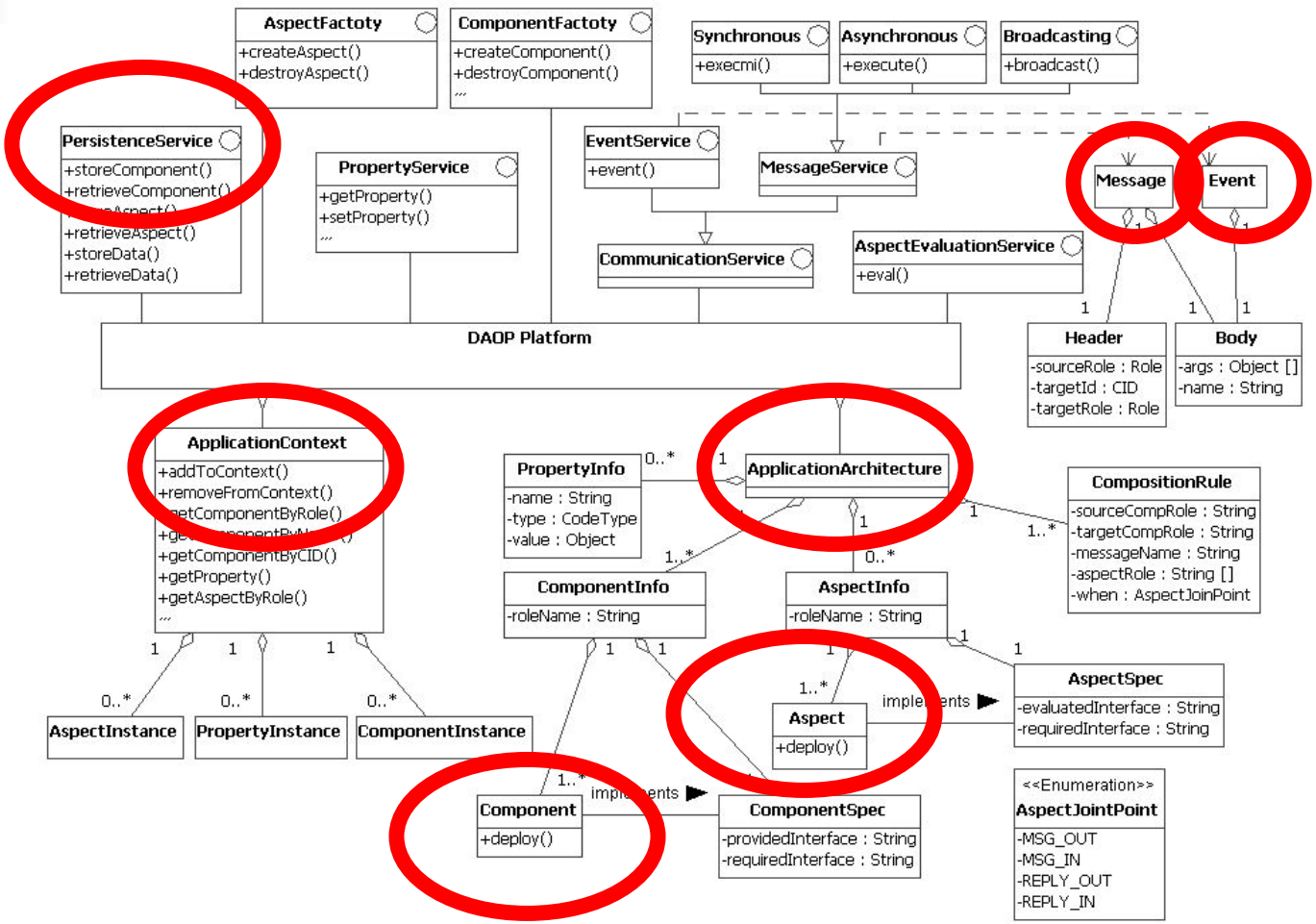
1) CIM de una Oficina Virtual



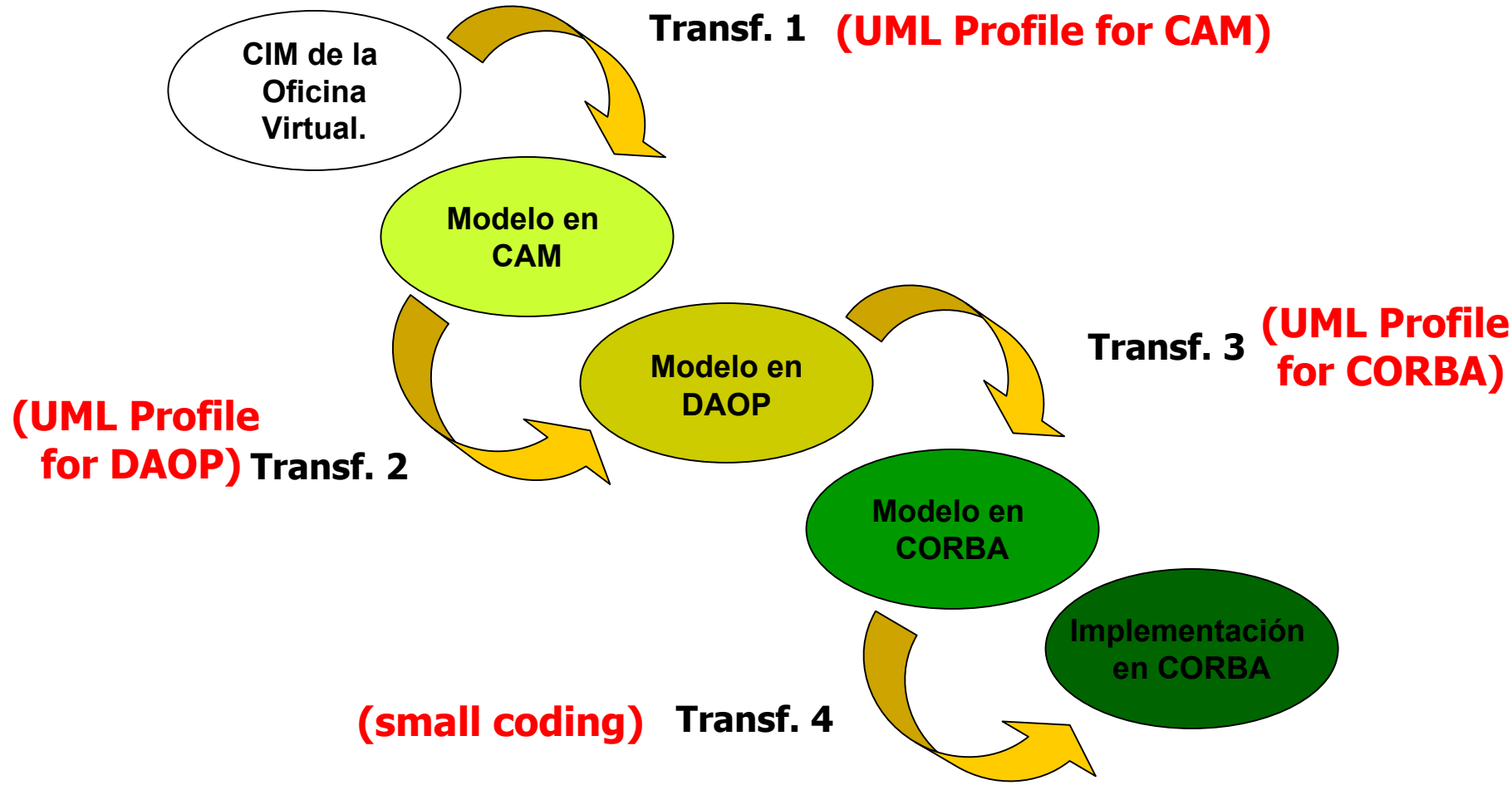
2) Component-Aspect Model

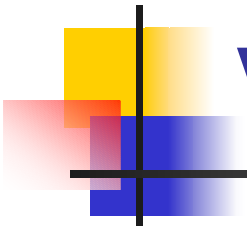


3) Plataforma "DAOP"



Los distintos "Modelos"

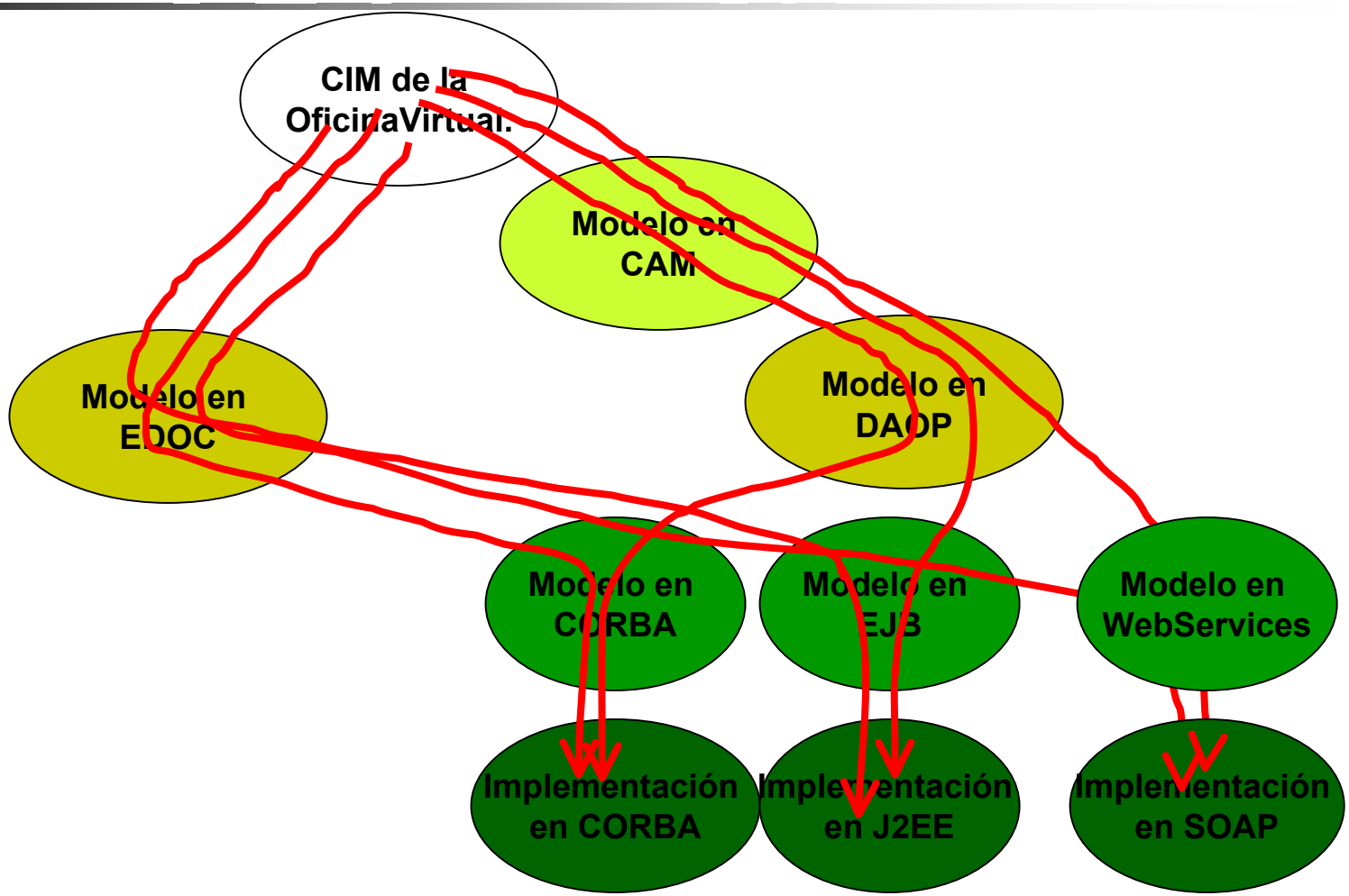




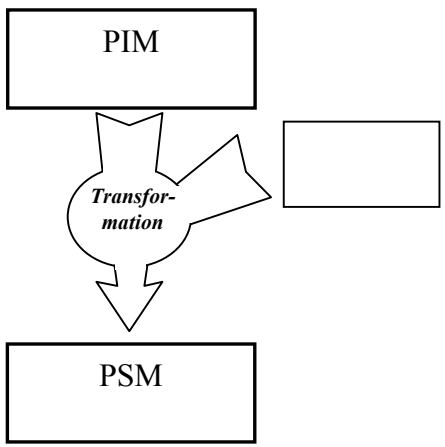
Ventajas

- Cada Modelo es independiente del resto, y define sus propias “entidades”
- El desarrollo de software se convierte en una sucesión de transformaciones entre los distintos modelos
- Cada transformación es entre un PIM y un PSM
- Las transformaciones podrían automatizarse
- Flexibilidad y facilidad de evolución

Otras posibilidades

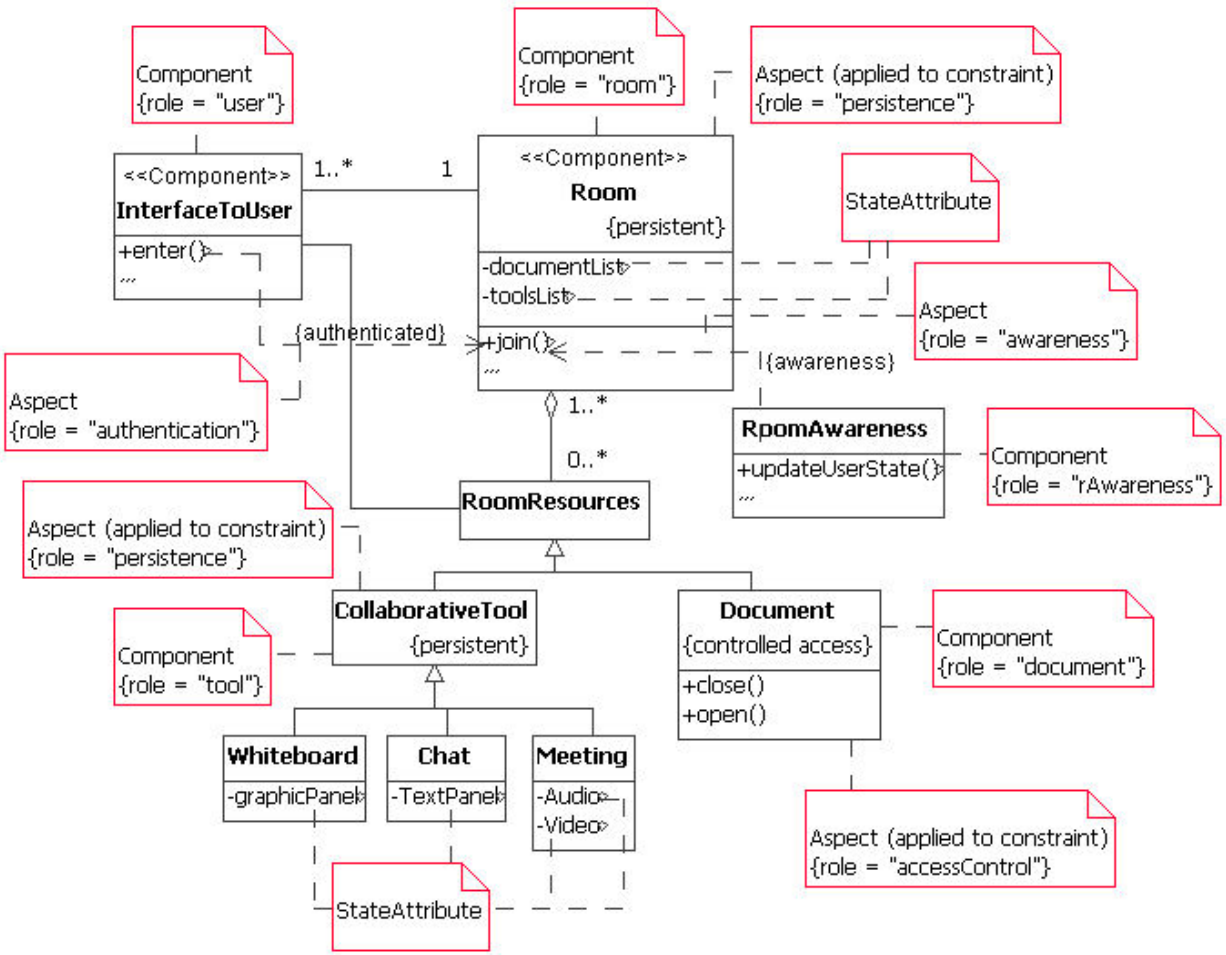


Transformaciones

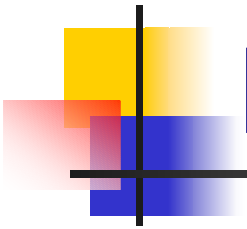


- En este caso hemos usado UML Profiles y “marcas”
  - Los UML profiles definen las entidades de cada modelo
  - Se marcan las entidades del PIM con el PSM
  - Se definen reglas de transformación

Ejemplo de CIM Marcado

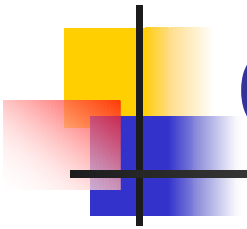


Resumen: MDA y Servicios Web



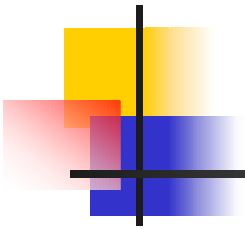
- EDOC proporciona un UML profile (ECA) que permite la integración con WS.
- ECA – Enterprise Collaboration Archit.
- Complementan perfectamente:
  - WSDL: vinculación y "endpoints"
  - ECA: coreografías, sesiones, componentes y conversaciones anidadas
- **No more manual coding!** (?)

Parte VI: Conclusiones



Conclusiones

- Los Servicios Web proporcionan toda una serie de ventajas muy importantes para el desarrollo de aplicaciones distribuidas en Internet
  - No proporcionan soluciones a todos los problemas, pero sí a muchos de ellos
  - Su objetivo no es sustituir a las tecnologías existentes, sino complementarlas
- Las tecnologías evolucionan demasiado rápido
  - Quien protege la inversión?
  - Como estar preparado para evolucionar con las tecnologías?
- MDA ofrece una solución excepcional y a largo plazo



Referencias

- P. Cauldwell et al. ***Professional XML Web Services***. Wrox Press, 2001.
- C. Szyperski. ***Component Software. Beyond Object-Oriented Programming***. Addison-Wesley. 1998.
- Mark Birbeck et al. ***Professional XML***. 2ª ed. Wrox Press, 2001.
- Heather Kreger. "Web Services Conceptual Architecture 1.0". IBM Technical Report, 2001.
- Peter Herzum. "Web Services and Service-oriented architectures". Distributed Enterprise Architecture Advisory Service, Cutter Consortium, 2001.
- James Kao. "Developer's Guide to Building XML-based Web Services". Sun Microsystems, June 2001.
- David Krieger y Richard Adler. "The Emergence of Distributed Component Platforms". Computer 41(3):43-53, 1998.
- *Software Development* Magazine (www.sdmagazine.com). Columna "Beyond Objects"
- IONASphere Magazine (www.iona.com/sphere)

Sitios de interés (org)

- www.w3.org -- WWW Consortium (XML,SOAP..)
- www.uddi.org -- UDDI
- www.ebXML.org -- Electronic Business XML std.
- www.rosettanet.org -- RosettaNet
- www.xcbl.com -- Common Business Library (xCBL)
- www.cXML.org -- Commerce XML (cXML)
- www.microsoft.com/biztalk -- BizTalk Framework
- www.BizTalk.org -- BizTalk library and repository
- www.XML.org -- OASIS's XML registro y repositorio
- Web Services@omg.org -- OMG dist. list for Web Services

Sitios de interés (comerciales)

- www.XMLBus.com -- IONA's site for Web Services
- www.ibm.com/websphere -- IBM's site for Web Services
- www.hp.com/go/webservices -- HP's site for Web Services
- www.microsoft.com/net -- Microsoft's site for Web Services
- otn.oracle.com/products -- Oracle's site for Web Services
- www.sun.com/sunone -- Sun's site for Web Services
- www.talkingblocks.com -- Talking Blocks 2.0
- www.componentsource.com -- Repositiorio comercial de COTS
- www.flashline.com -- Repositiorio comercial de COTS
- www.wrldcomp.com -- Repositiorio comercial de COTS
- www.salcentral.com -- Repositorio de Web Services
- www.xmethods.net -- Repositorio de Web Services

Fin de la presentación