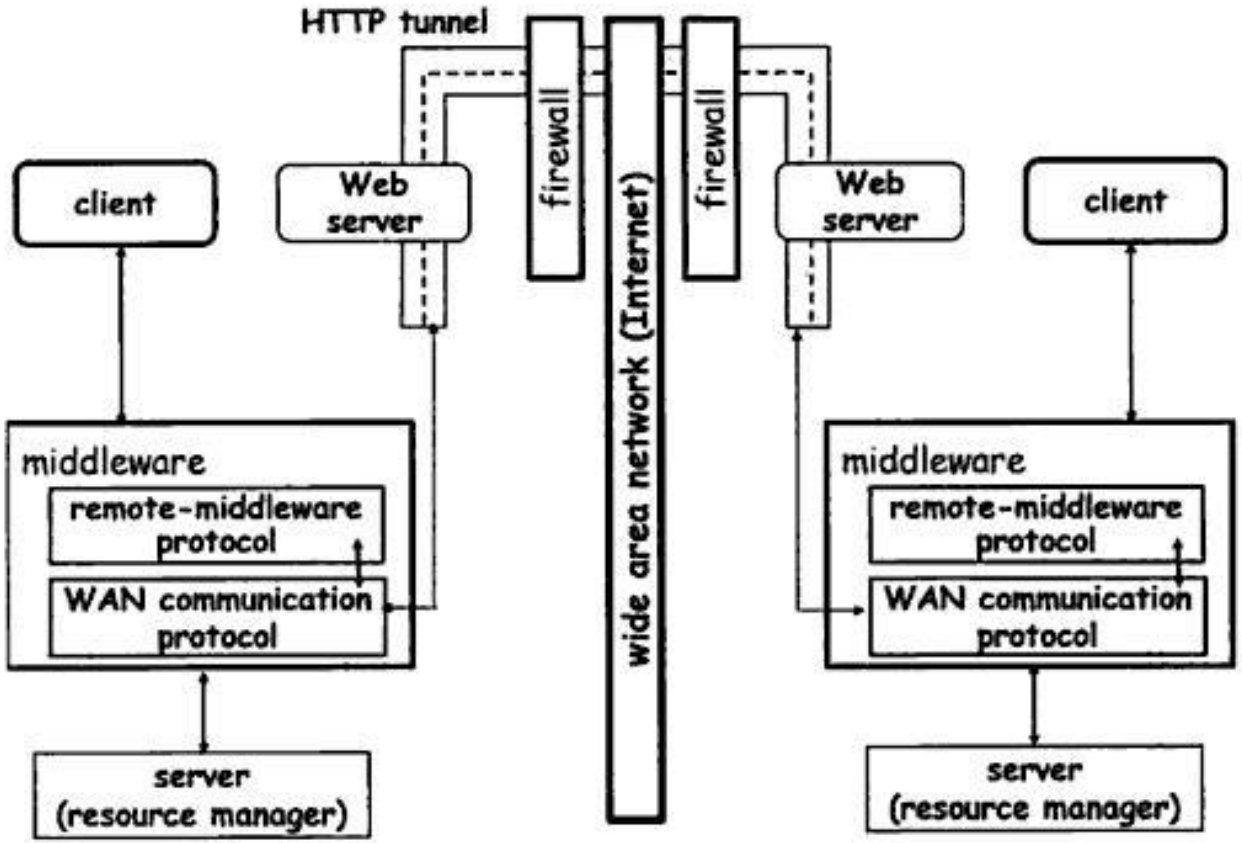
Web Services

Agenda

- Introducción
  - Motivación, surgimiento y definición
- Web Services SOAP
  - Características
  - Tecnologías
    - SOAP, WSDL, UDDI
- Seguridad en Web Services SOAP
  - Transporte
  - Mensaje

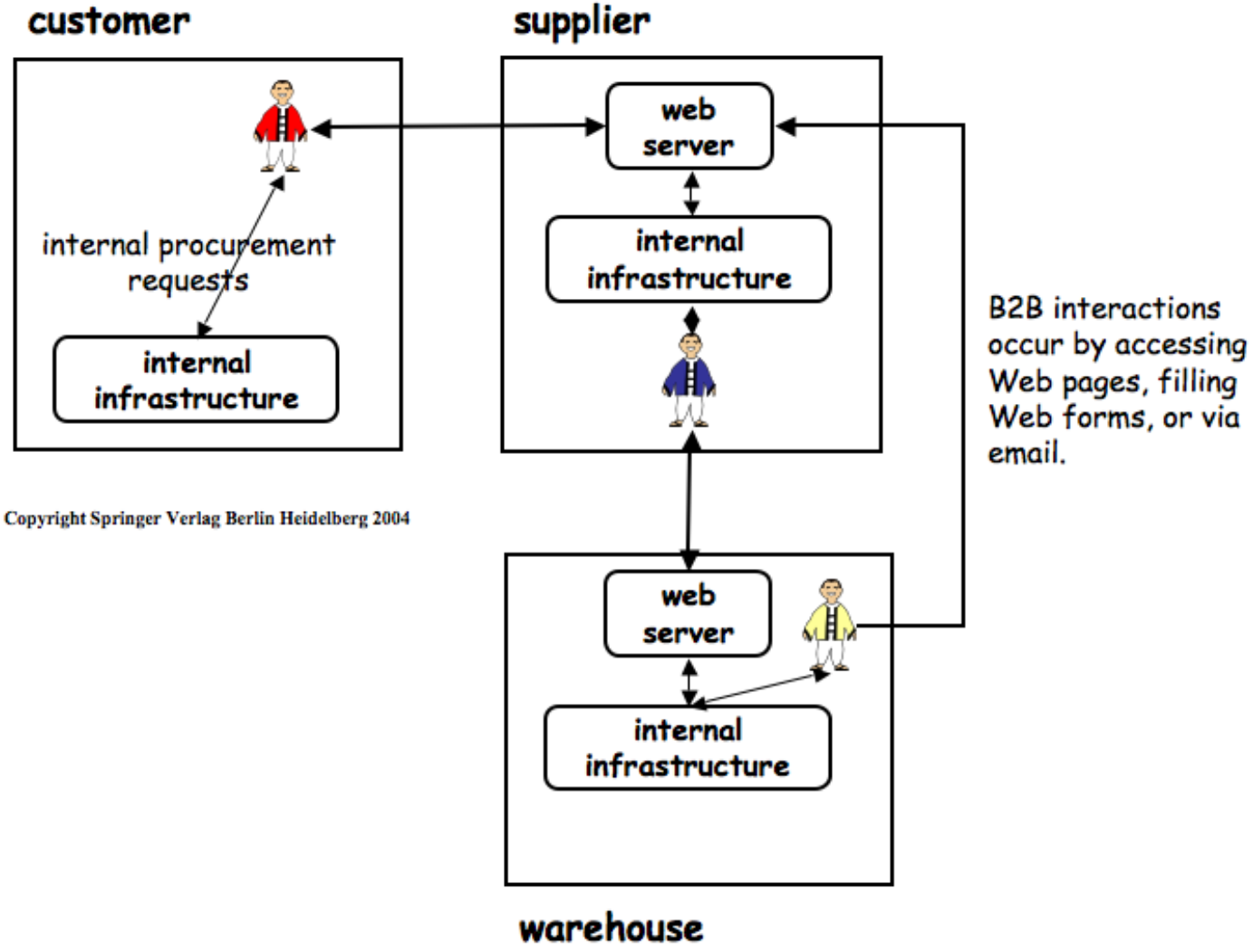
Motivación (1)



Motivación (1)

- Interés de atravesar los firewalls
- Middlewares existentes no proveían tales características

Motivación (2)

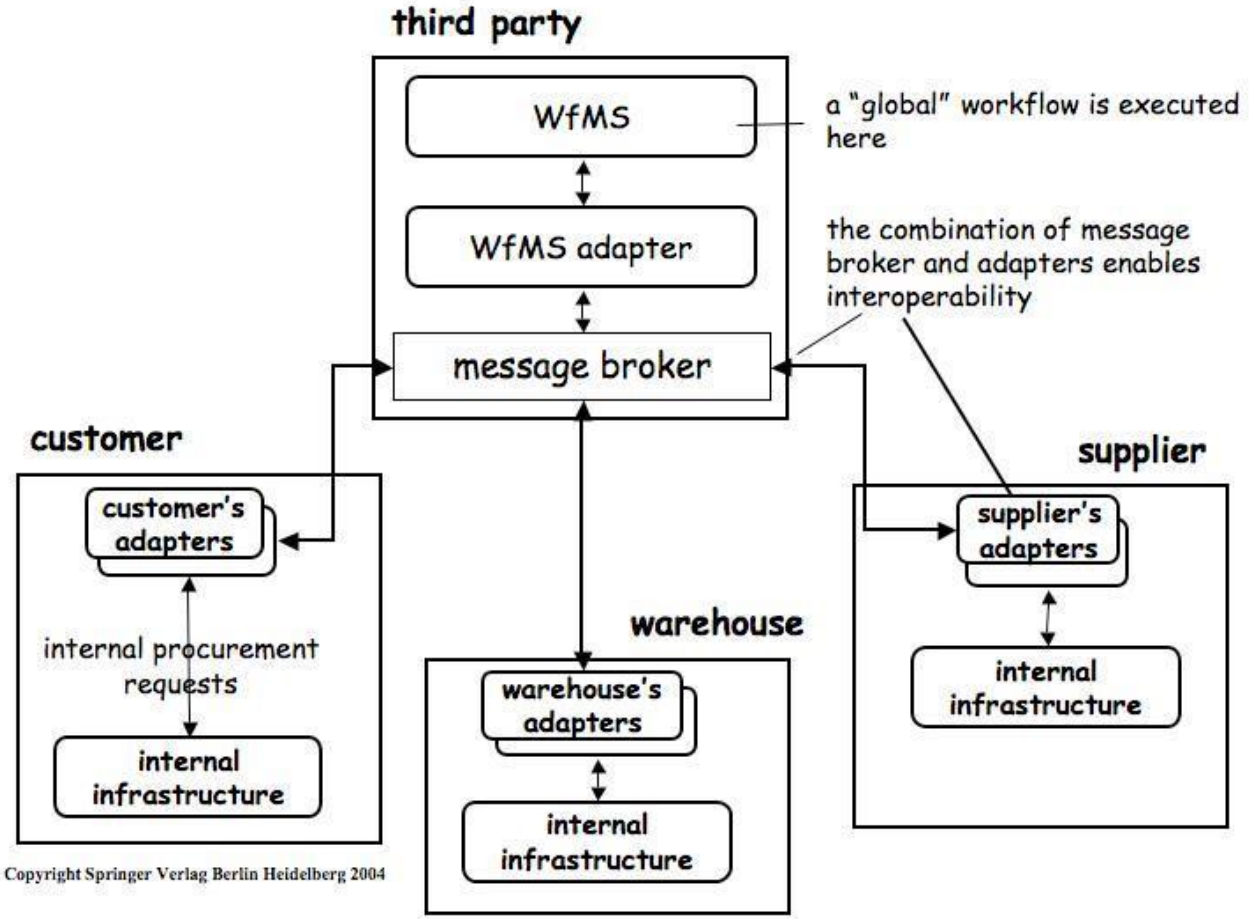


Motivación (2)

- Los procesos B2B se llevaban a cabo de forma manual vía Web, fax o e-mail

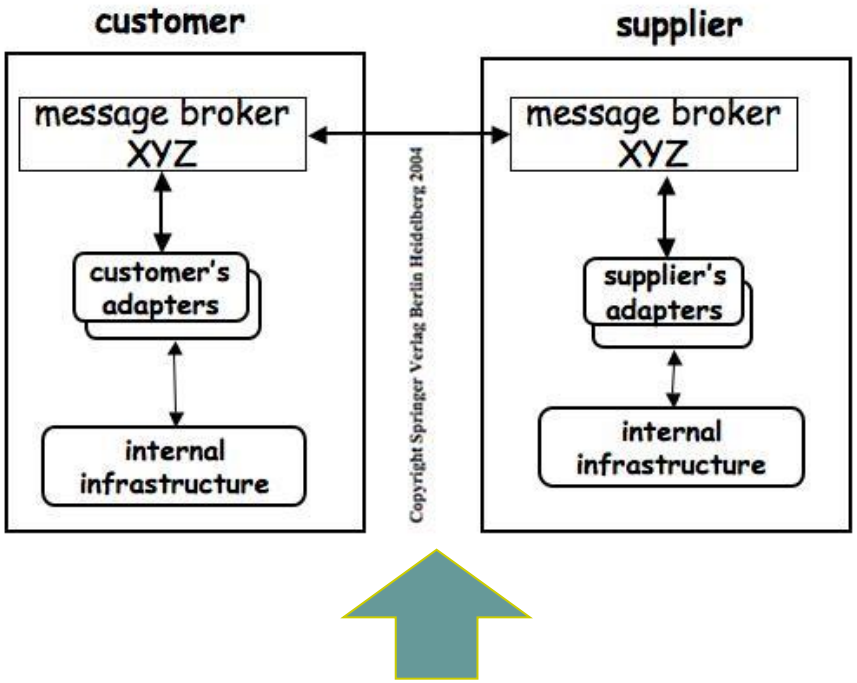
- Algunos problemas de automatización en transacciones B2B incluían:
  - Integración entre organizaciones, dónde alojar al middleware?
    - Un tercero debía hostear el middleware
    - Todos los involucrados deben acordar usar el middleware

Motivación (2)



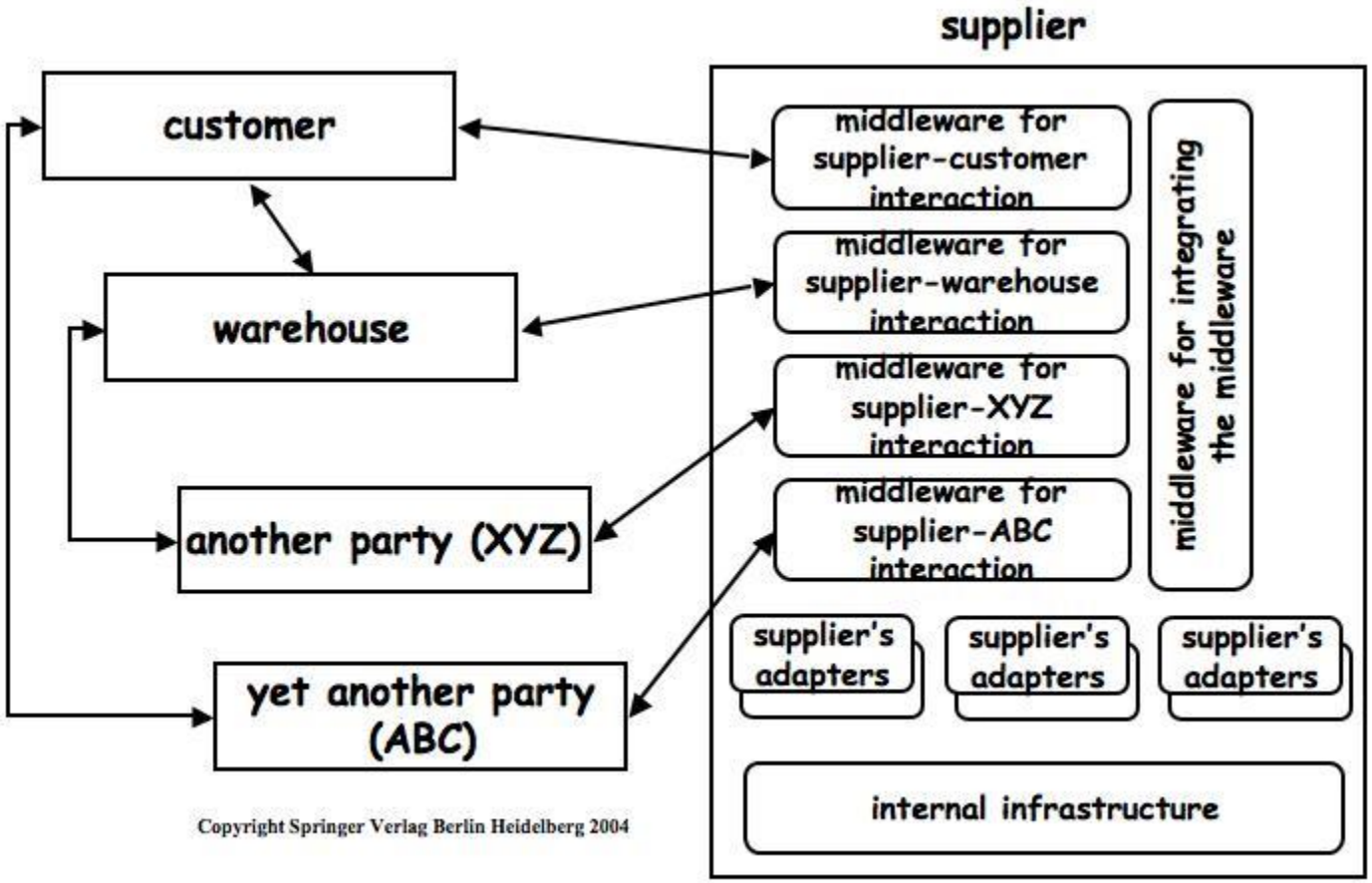
Motivación (2)

- Confianza entre empresas
  - es baja y a veces, inexistente.
- Autonomía
  - Las empresas desean ser lo más autónomas posibles unas de otras
- Confidencialidad
  - Las empresas no quieren guardar sus datos y transacciones en middleware de terceros



Resultado final: integración punto a punto mediante message brokers específicos

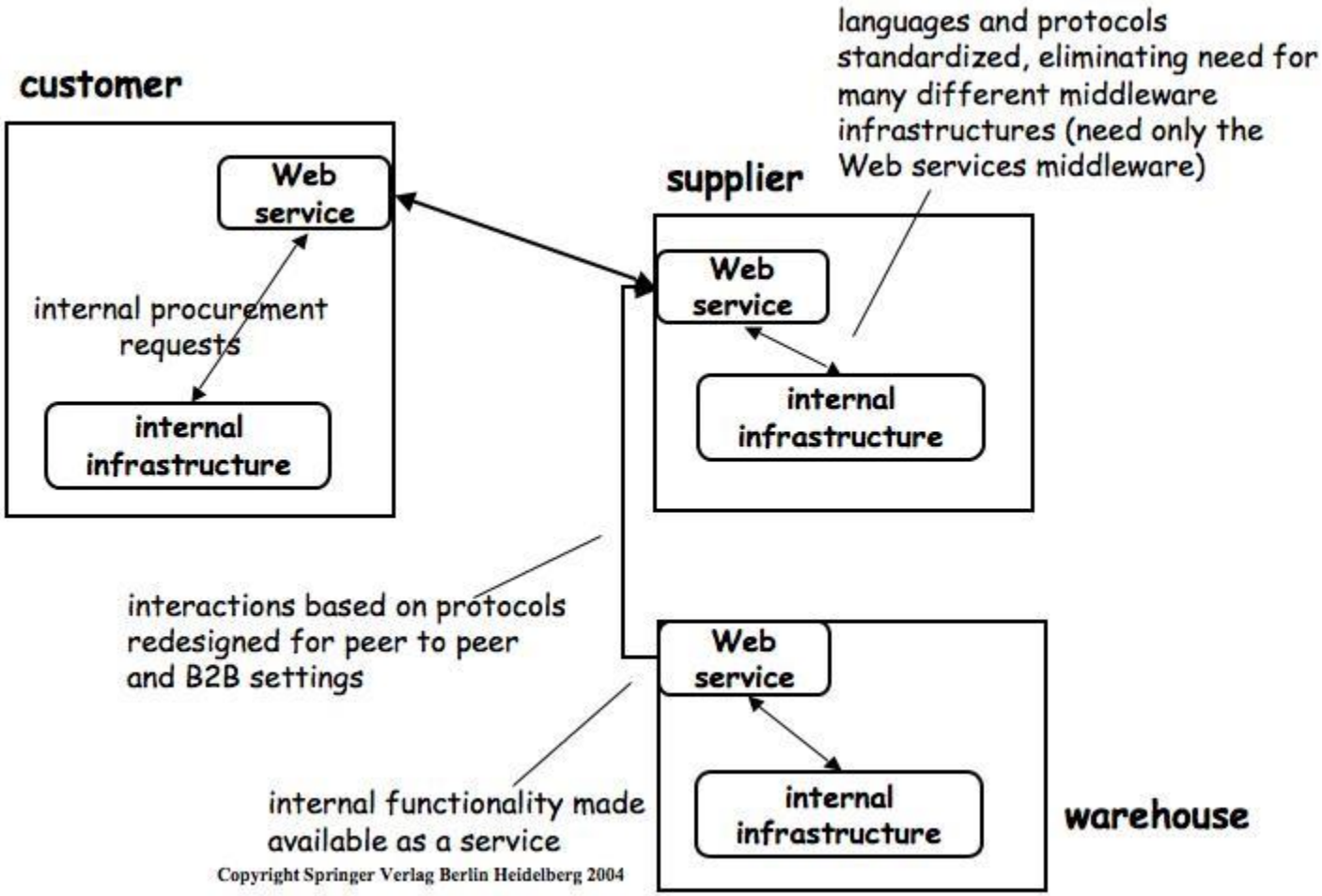
Sin embargo…


supplier
customer
warehouse
another party (XYZ)
yet another party (ABC)
middleware for supplier-customer interaction
middleware for supplier-warehouse interaction
middleware for supplier-XYZ interaction
middleware for supplier-ABC interaction
middleware for integrating the middleware
supplier's adapters
supplier's adapters
supplier's adapters
internal infrastructure
Copyright Springer Verlag Berlin Heidelberg 2004

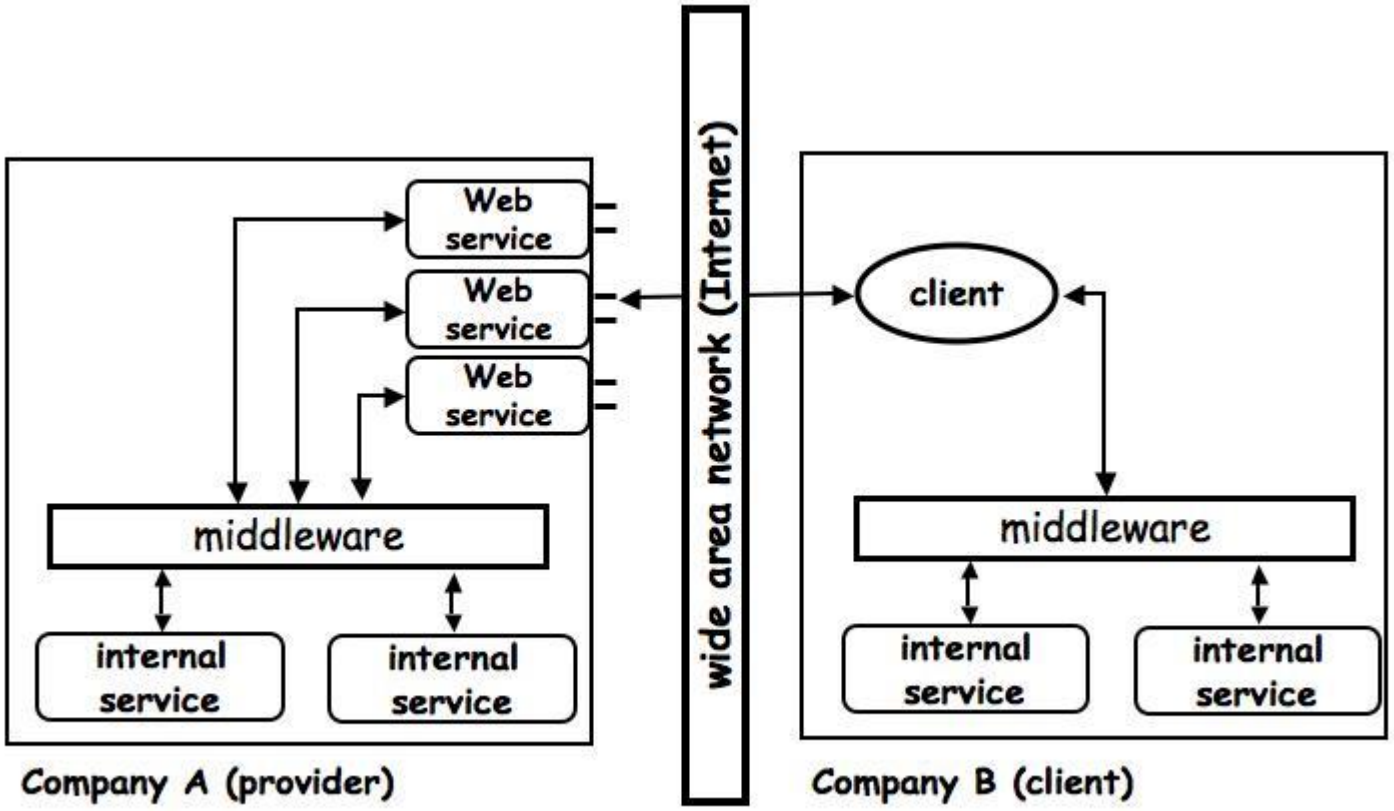
Problema

- No es escalable
  - Se debía tener un broker específico por cada integración
- Suposición falsa
  - Una plataforma de integración basada en middleware puede ser centralizada, en donde todos los componentes (de diferentes empresas) que integra confían en ella.
- El surgimiento de las tecnologías Web y de protocolos de comunicación (HTTP) y formatos (primero HTML y luego XML) estándares, posibilitaron la creación de un middleware convencional denominado Web Services, que posibilita la integración de aplicaciones en escenarios B2B

B2B integration via WS



WS & EAI



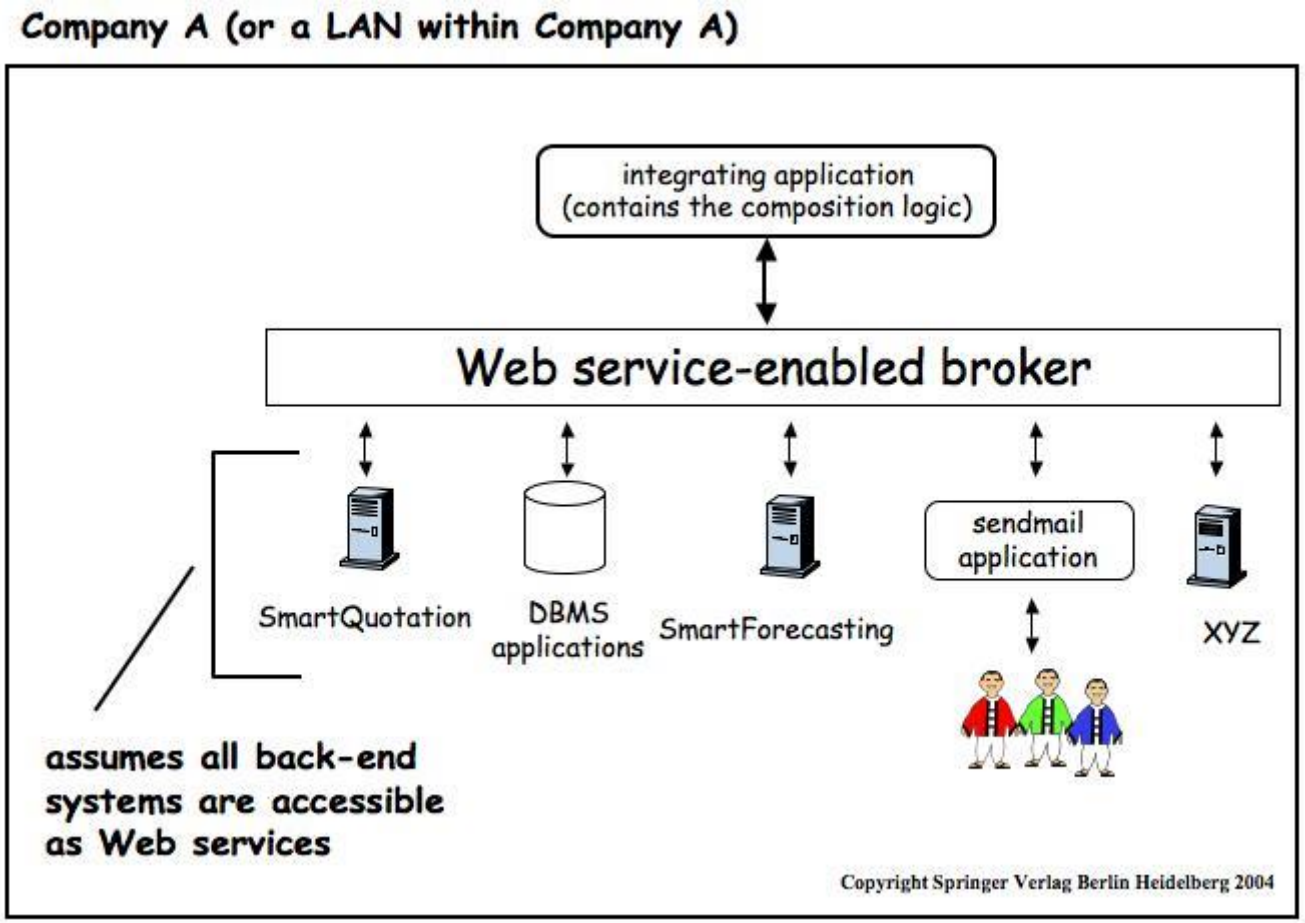
WS & EAI

- El objetivo principal de los Web Services es proveer una forma sencilla de exponer los sistemas de información de la empresa de una forma controlada y estándar

- De alcanzar este objetivo, los Web Services pueden ser utilizados para posibilitar EAI entre empresas sin la necesidad del uso de adaptadores específicos.

- De esta forma, los Web Services permiten a los clientes acceder a los sistemas de información internos de la empresa de una forma estándar

WS & EAI dentro de la compañía



WS & EAI dentro de la compañía

Laboratorio de Integracion de Sistemas

- Dentro de una empresa, los Web Services permiten una integración EAI sin la necesidad de costosos adaptadores
  - Los Web Services toman el lugar de los adaptadores
  - Ejemplo
    - Conectar aplicaciones .NET con JEE
    - Conectar aplicaciones .NET con Cobol
    - …

Web Services: ¿qué es?

- Representa una funcionalidad (de negocio)
- Se encuentra disponible a través de Internet o una Intranet
- Utiliza una forma estandarizada de mensajería (generalmente XML)
- No se encuentra atado a ningún sistema operativo ni ningún lenguaje de programación
- Se puede auto describir (generalmente vía XML)
- Puede ser descubierto a través de un mecanismo de búsqueda

Web Services: ¿Por qué?

- Business to Business integration (B2B)
  - Acuerdos comerciales entre organizaciones
    - Ej: Posibilidad de pagar facturas a través de redes de cobranza
- Enterprise Application Integration (EAI)
  - Sistemas desarrollados por diferentes empresas
    - Diferentes tecnologías
  - Silos de información
    - Contabilidad, inventario, logística
  - Fusión de empresas

Web Services: ¿Cómo?

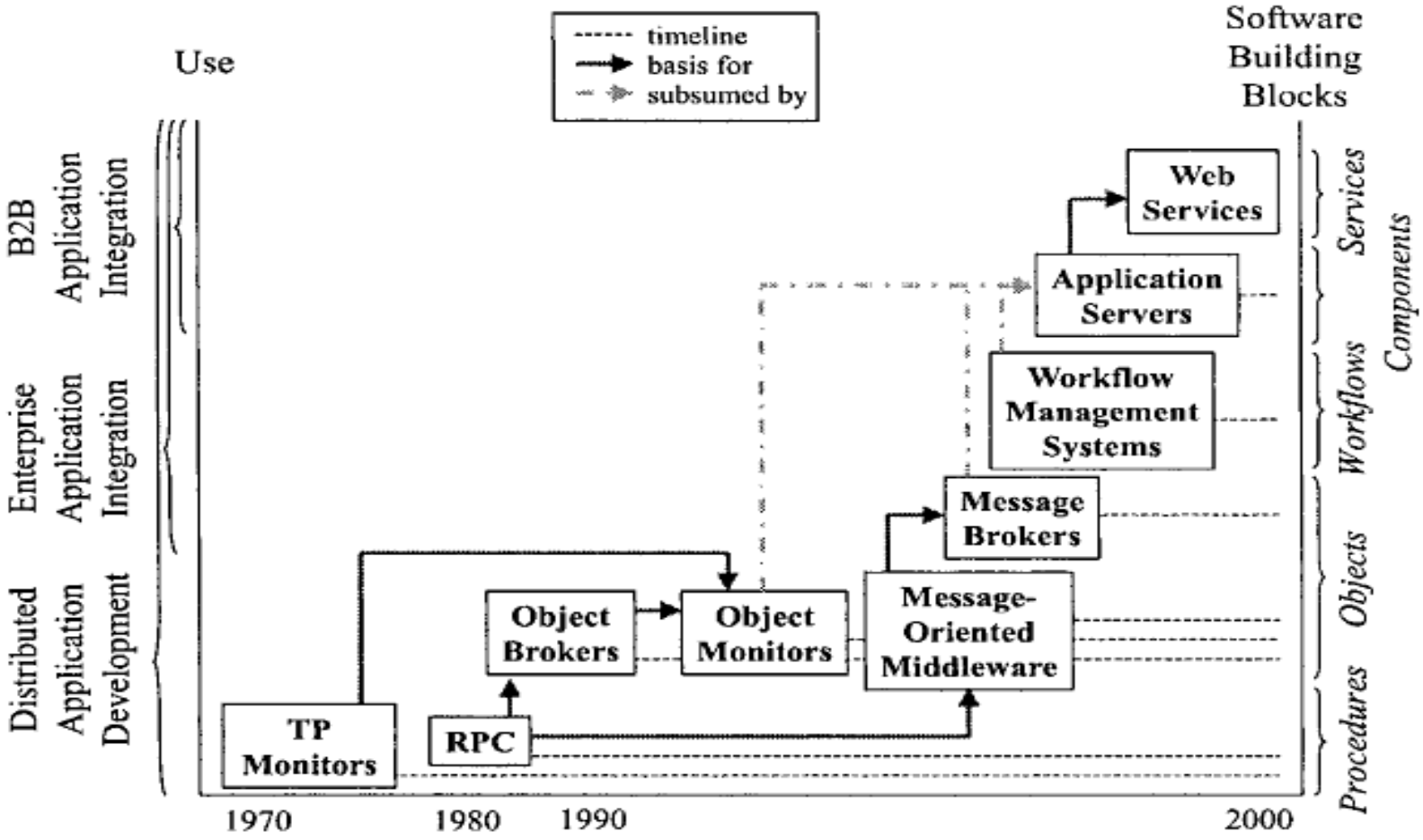
- El término Web Service nace aproximadamente en el año 2000 por iniciativa de MS e IBM

- Producen una gran cantidad de estándares
  - Primera generación: WSDL, SOAP, UDDI
  - Segunda generación: WS-*

Definición

- "A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically **WSDL**). Other systems interact with the Web service in a manner prescribed by it's description using **SOAP** messages, typically conveyed using **HTTP** with an **XML** serialization in conjunction with other Web-related standards."
  - *World Wide Web Consortium (W3C), 2006*

Evolución Middleware



Semantic Management of Middleware. Ramesh Jain. Amit Sheth. Springer 2006.

Tipos de Web Services

- Simples o de información
  - Operaciones de corta duración
  - Patrón de comunicación Request/Response
    - Pedido-Espera-Respuesta
- Compuestos o de procesos de negocios
  - Operaciones de larga duración
  - Coordinación de operaciones I/O

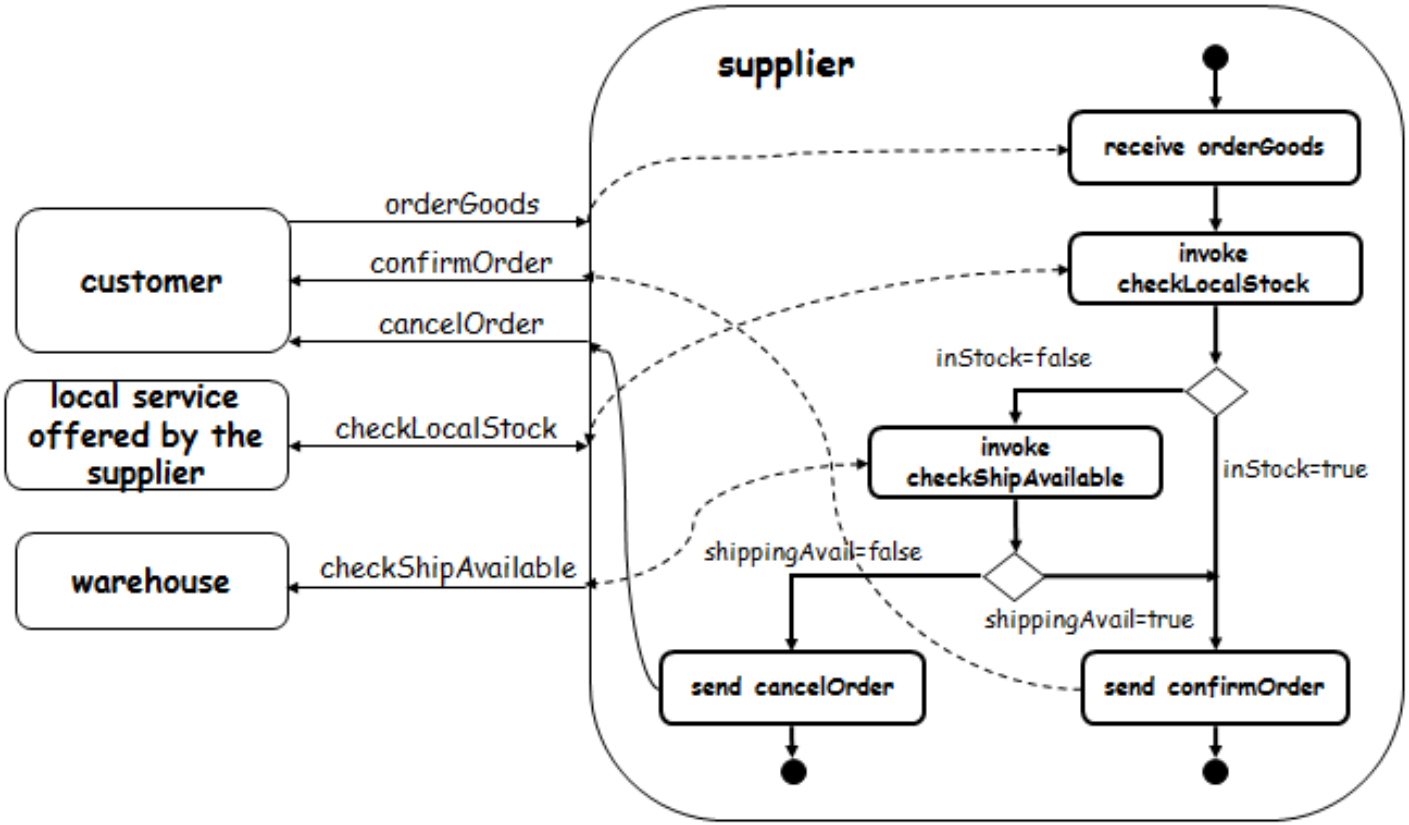
Simples o de Información

- Exponen las funcionalidades de negocios de una aplicación de forma estándar
  - *Back End escritos en Java/EJB, VB, C#, C++, …*
- Mecanismos de comunicación sincrónicos
- Operaciones atómicas
  - Una única operación por request
  - No son transaccionales
    - Por más que su backend si lo sea
- Son stateless
  - No mantienen estado entre pedidos

Compuestos o procesos de negocios

- Composición de servicios simples para la implementación de procesos de negocios
  - P. ej: Procesos de compra
- Coordinación de operaciones
- Transacciones de larga duración
- Mecanismos de comunicación asincrónicos
- Servicios de alta granularidad
- Stateful
  - Mantienen el estado entre pedidos

Ejemplo



Granularidad de los servicios

- La granularidad de un servicio refiere al alcance que tiene la funcionalidad expuesta por el servicio
- Los servicios pueden tener dos niveles de granularidad:
  - Granularidad fina: son servicios que proveen un acceso básico a datos u operaciones rudimentarias.
    - “CheckLocalStock”, “confirmOrder”, “cancelOrder”
  - Granularidad gruesa: son servicios compuestos por servicios de granularidad fina.
    - “OrderGoods”

Tecnologías para el desarrollo de Web Services

SOAP, WSDL, UDDI

Web Services

- Iniciativa de Microsoft e IBM en el año 2000 para resolver necesidades de las áreas EAI y B2B

- Resultado:
  - Primera generación de Web Services
  - Un conjunto de estándares de comunicación basado en XML y tecnologías Web (http, tcp, smtp, etc)
    - WSDL, SOAP, UDDI

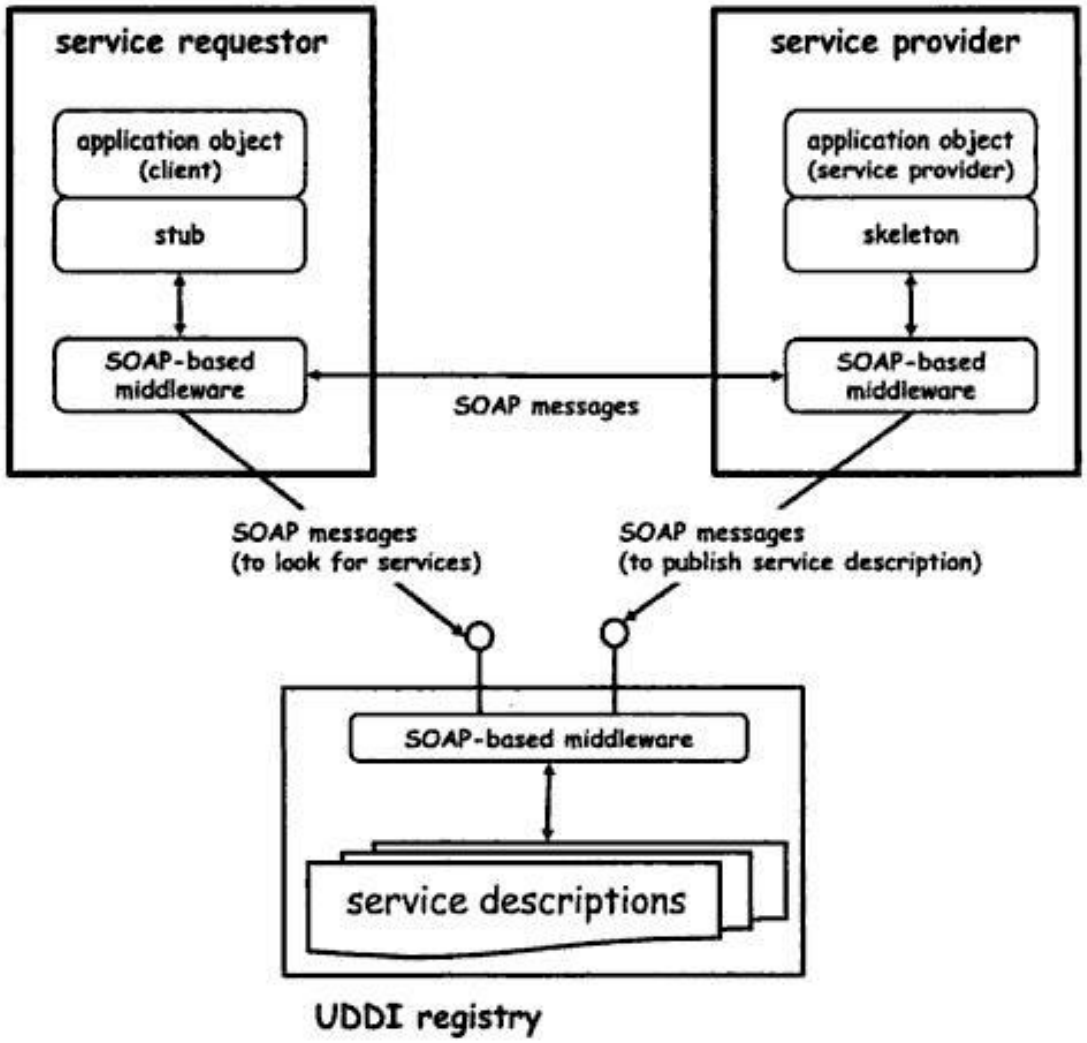
Requerimientos básicos para llevar a cabo una comunicación

- Sintaxis
- Formato
- Transporte
- Interfaces
  - Definición de operaciones y parametros I/O
- Descubirimiento/Búsqueda

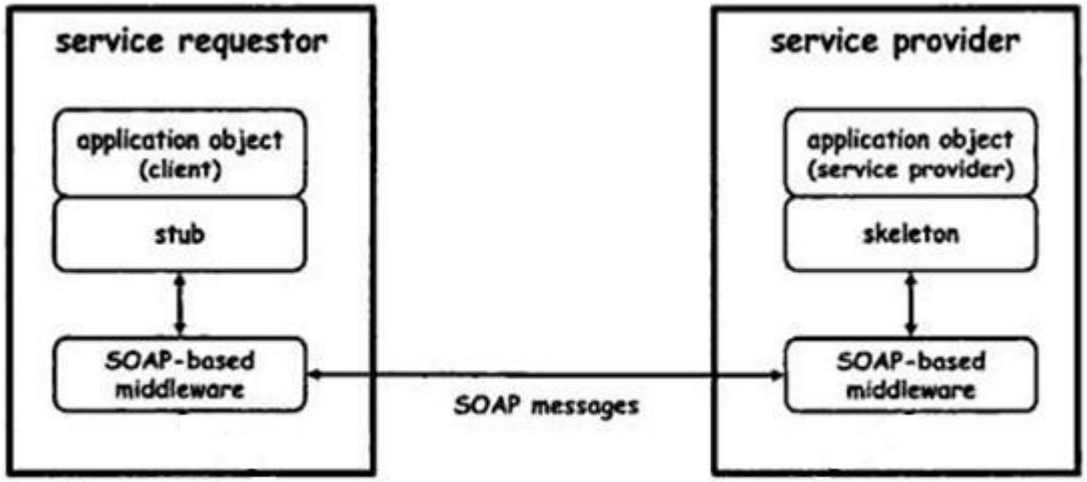
Requerimientos básicos para llevar a cabo una comunicación

- Sintaxis
  - XML
- Formato
  - SOAP
- Transporte
  - Tecnologías Web (http, smtp, etc)
- Interfaces
  - WSDL
- Descubrimiento/Búsqueda
  - UDDI

¿Cómo funciona (teoría)?



¿Cómo funciona (práctica)?



- Los registros públicos no fueron exitosos
- Los estándares existentes (UDDI y ebXML) son demasiado “pesados”
- “Buenas prácticas”: Intercambiar los WSDLs y endpoints por e-mail

Simple Object Access Protocol (SOAP)

SOAP

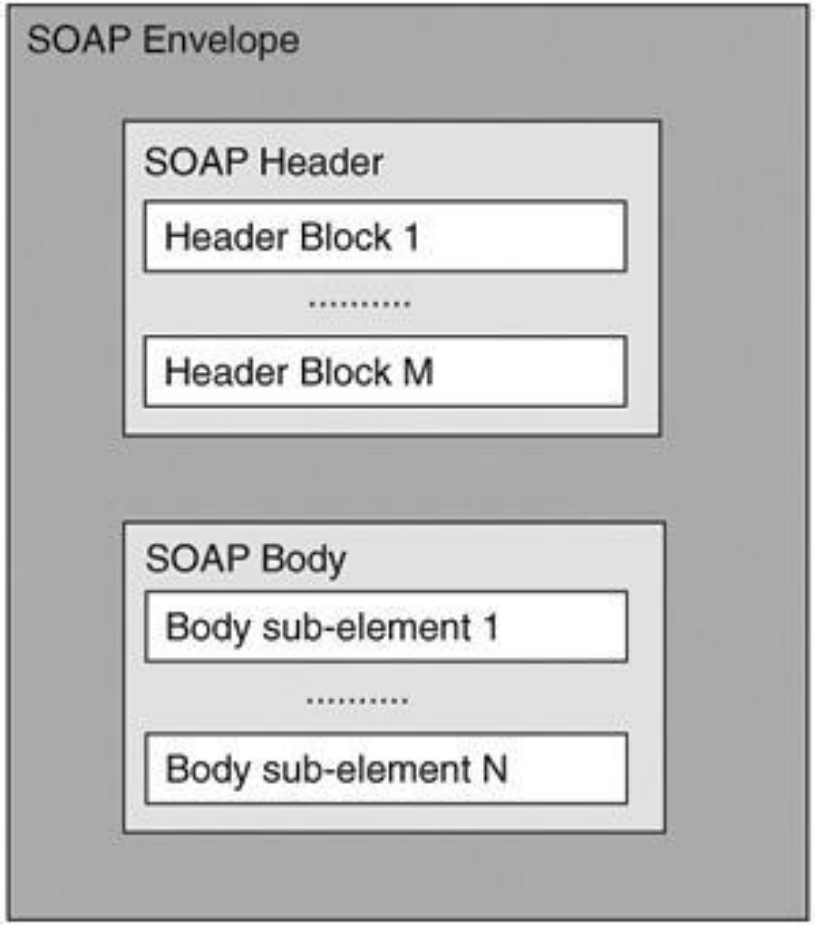
- Estándar de la W3C que describe un formato de mensaje (basado en XML) y mecanismos para intercambiar información entre aplicaciones, en un ambiente distribuido y descentralizado.

- Actualmente, se encuentra en la versión 1.2.

SOAP

- Provee una forma estándar de estructurar mensajes utilizando XML
- Define mecanismos para utilizar distintos protocolos de transporte para el envío de mensajes
- Especifica un modelo de procesamiento que indica cómo se deben procesar los mensajes
- Una forma de adjuntar datos no-XML a los mensajes.

Mensaje SOAP



Un ejemplo

```
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-
   envelope">
   <env:Header>
    <pns:qualityOfService xmlns:pns="http://example.org/qos">
       <pns:priority>3</pns:priority>
       <pns:timestamp>2004-02-25T01:00:00-03:00</pns:timestamp>
       <pns:persist>true</pns:persist>
    </pns:qualityOfService>
   </env:Header>                                          Header
   <env:Body>
    <bmns:businessPO xmlns:env="http://example.org/po">
       <bmns:description>Widgets</bmns:description>
       <bmns:quantity>100</bmns:quantity>
       <bmns:price>20.5</bmns:price>
    </bmns:businessPO>
   </env:Body>                                            Body
</env:Envelope>

                                                          Envelope
```

Mensaje SOAP

- La estructura básica de un mensaje SOAP consiste de un elemento "**Envelope**" el cual contiene
  - un elemento opcional "**Header**"
  - un elemento requerido "**Body**"
    - que puede incluir un elemento "**Fault**"

- El "Envelope" es el elemento raíz de todo mensaje SOAP e identifica un documento XML como un mensaje SOAP

Mensaje SOAP – Header

- El “Header”, de estar presente, debe ser el primer elemento del “Envelope”
- Provee un mecanismo de extensión que permite incluir información extra en mensajes SOAP (seguridad, transacciones, etc)
- Puede contener varios “header blocks” que son una forma de agrupar lógicamente la información

Mensaje SOAP – Header

```
... ... ...
<soap:Header>
  <t:trans xmlns:t="http://......./transaction/"
  soap:mustUnderstand="1">
       <t:id>12345</t:id>
  </t:trans>
  <s:sec xmlns:m="http://......./security/"
  soap:mustUnderstand="1">
       <s:user>john</s:user>
       <s:pass>1234</s:pass>
  </s:sec>
</soap:Header>
... ... ...
```

Mensaje SOAP – Body

- Contiene la información a ser intercambiada entre el cliente y servicio

- En el “Body” típicamente se especifica:
  - una solicitud para efectuar cierta operación
  - la respuesta a cierta solicitud que puede ser:
    - un resultado o
    - un error (fault)

Mensaje SOAP – Body

```
<soap:Body>
   <t:Translate xmlns:t="http://...../translations">
        <t:Word>red</t:Word>
   </t:Translate>
</soap:Body>
```

solicitud

```
<soap:Body>
  <t:TranslateResponse
   xmlns:t="http://...../translations">
     <t:Translation>rojo</t:Translation>
  </t:TranslateResponse>
</soap:Body>
```

resultado

Mensaje SOAP – Fault

- El elemento “Fault” indica la ocurrencia de un error en el procesamiento del mensaje
  - Comprensión del mensaje
    - Error de sintaxis
  - Infraestructura
    - Error de red: HTTP 500 Internal Server Error
  - Negocio
    - No existe la cuenta con número X

Mensaje SOAP – Fault

- Tiene 5 sub-elementos:
  - Code
  - Reason
  - Detail
  - Node (opcional)
  - Role (opcional)

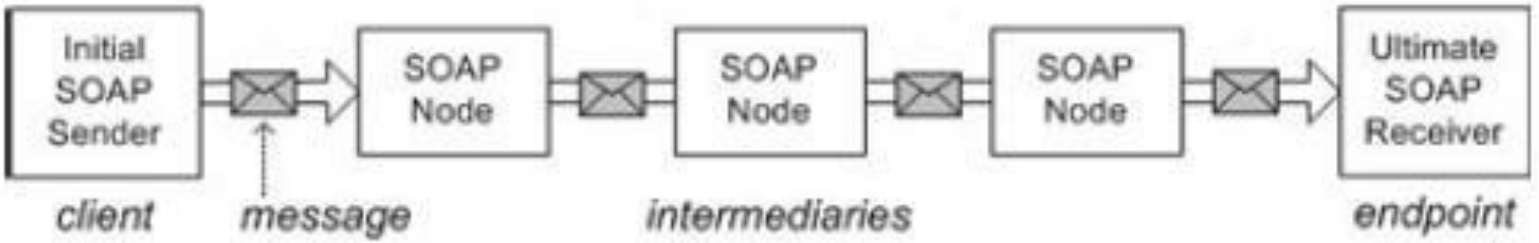
Ejemplo

```
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
   m="http://www.plastics-suply.com/product-prices">
<env:Header/>
<env:Body>
  <env:Fault>
    <env:Code>
      <env:Value>env:Sender</env:Value>
      <env:Subcode>
        <env:Value>m:InvalidPurchaseOrder</env:Value>
      </env:Subcode>
    </env:Code>
    <env:Reason>
      <env:Text xml:lang="en-UK">
        Specified product did not exist
      </env:Text>
      <env:Text xml:lang="es">
        El producto solicitado no existe
      </env:Text>
    </env:Reason>
  </env:Fault>
</env:Body>
</env:Envelope>
```

Procesamiento de Mensajes

- En el modelo SOAP pueden existir varios nodos intermediarios que procesan los mensajes

- Usos comunes: auditoría, compresión, seguridad, etc.



http://www14.informatik.tu-muenchen.de/konferenzen/Jass05/courses/6/Papers/10.pdf

Procesamiento de Mensajes

- Los “header blocks” pueden incluir información que especifique para qué rol está destinado el bloque (atributo role)
- SOAP define tres posibles valores para dicho atributo: none, ultimateReceiver y next
- SOAP no incluye un mecanismo para especificar la ruta que debe seguir un mensaje SOAP hasta llegar al WS destino

Ejemplo

```
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    <s:sec xmlns:m="http://......./security/"
         soap:actor="https://services.com/esb/">
         <s:user>john</s:user>
         <s:pass>1234</s:pass>
    </s:sec>
  </env:Header>
  <soap:Body>
   <t:Translate xmlns:t="http://...../translations">
         <t:Word>red</t:Word>
   </t:Translate>
</soap:Body>
</env:Envelope>
```

SOAP MEP

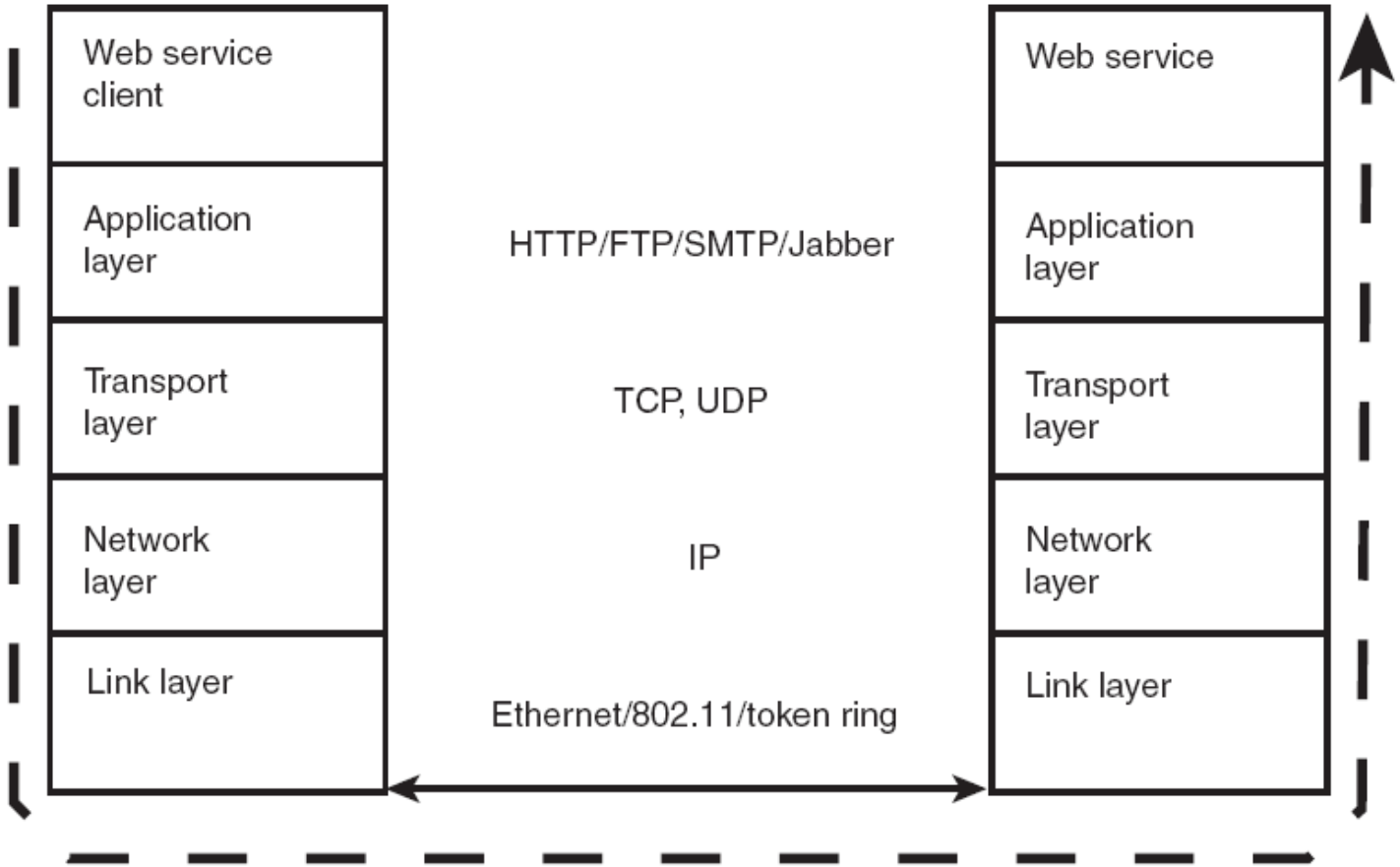
- Son los message exchange patterns (MEP) que soporta SOAP
- Algunas posibilidades
  - Request/Response
    - enviamos el mensaje, y nos quedamos esperando hasta que llegue la respuesta del mismo
  - One-way
    - enviamos el mensaje, y nos olvidamos de la respuesta. También se denomina “fire-and-forget”

Transporte de Mensajes

- SOAP no impone el uso de un determinado protocolo para el intercambio de mensajes

- A través del concepto de "*binding*", SOAP permite especificar:
  - cómo los mensajes SOAP se encapsulan en un protocolo de transporte
  - cómo los mensajes SOAP deben ser tratados con las primitivas del protocolo

Transporte de Mensajes



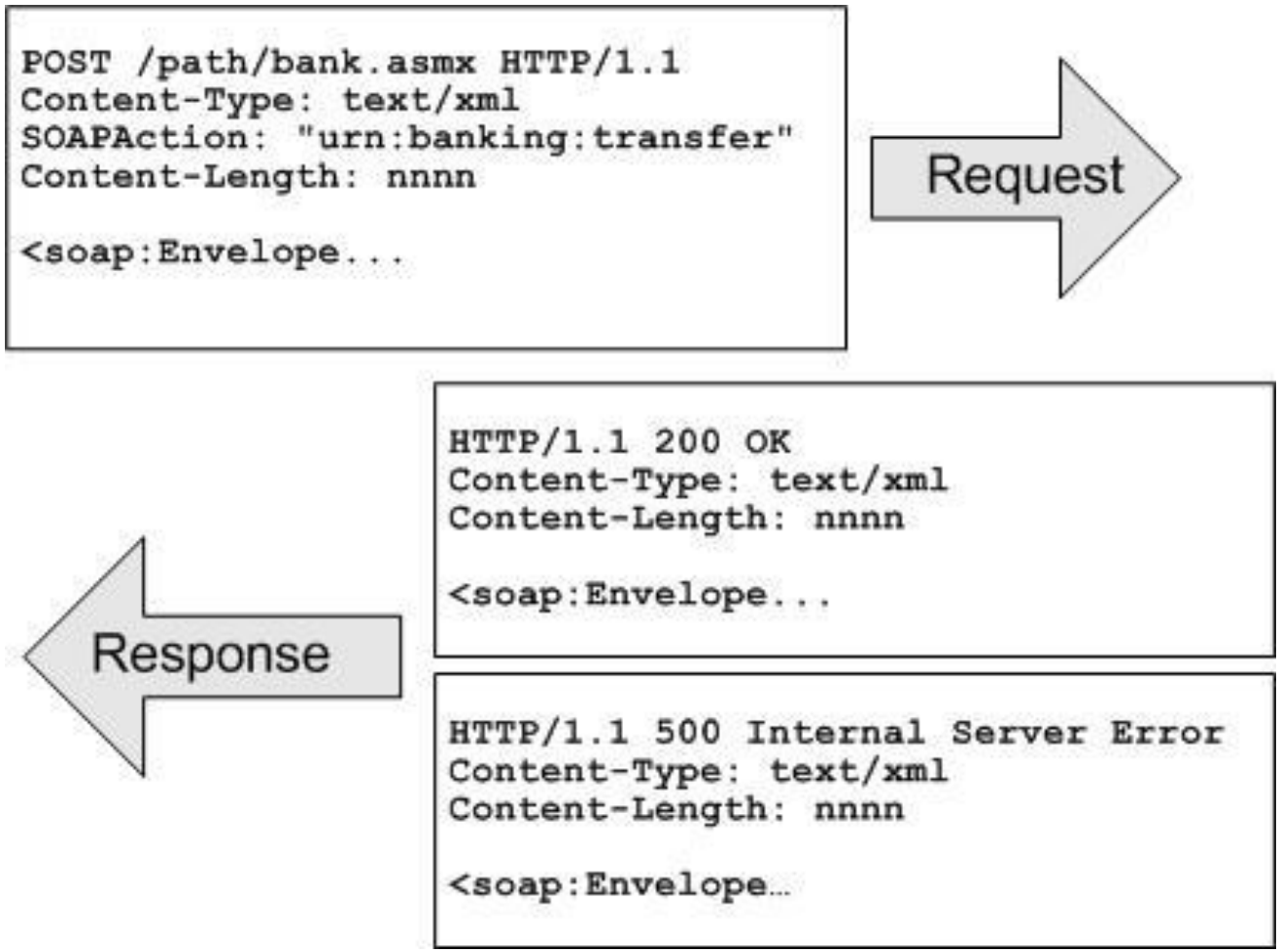
Sams Teach Yourself Web Services in 24 Hours. Stephen Potts, Mike Kopack. Sams Publishing 2003.

SOAP Bindings

- En el estándar SOAP se incluye la especificación de un "*bindings*" para los protocolos http y smtp

- Se dan un conjunto de reglas para definir nuevos "*bindings*"
  - Binding JMS (estándar)
    - http://www.w3.org/TR/soapjms/
  - Binding TCP (propietario)
    - http://blogs.sun.com/oleksiys/entry/soap_tcp_makes_web_services

HTTP SOAP Binding



Web Service Description Language (WSDL)

LINS
Laboratorio de Integracion de Sistemas

WSDL

- Estándar de la W3C que define un lenguaje basado en XML que permite describir la interfaz, formas de acceso y ubicación de un Web Service

- Actualmente, en versión 2.0

Para qué sirve?

- Dónde está ubicado el servicio?
- Qué operaciones tiene?
- Qué mensajes de entrada/salida recibe/responde?
- Cuál es la estructura de los mensajes?
- Qué protocolos de transporte hay que usar (bindings)?

- Un documento WSDL se divide en dos partes:
  - descripción abstracta
  - descripción concreta

WSDL – Descripción Abstracta

- La descripción abstracta describe de forma general la estructura de la interfaz del Web Service, que incluye operaciones, parámetros y tipos de datos abstractos
- Los cuatro elementos XML que componen la descripción abstracta son:
  - <wsdl:types>
  - <wsdl:message>
  - <wsdl:portType>
    - <wsdl:operation>

WSDL – types

- El elemento "*types*" encapsula todas las definiciones abstractas de tipos de datos

```
<wsdl:types>
   <xsd:schema
   targetNamespace="http://..../weatherService/wsdl"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema">
        <xsd:element name="City"
                type="xsd:string"/>
        <xsd:element name="Country"
                type="xsd:string"/>
   </schema>
<wsdl:types>
```

WSDL – messages

- El elemento "*messages*" representa de forma abstracta los parámetros de entrada y salida para una operación

```
<message name="getWeatherIn">
   <part name="CityIn" element="tns:City"/>
   <part name="CountryIn" element="tns:Country"/>
</message>
```

WSDL – portType

- El elemento “portType” es el contenedor de todas las operaciones abstractas y describe una interfaz específica del servicio

```
<portType name="getWeatherPortType">
   <operation name="getWeather">
        <input message="getWeatherIn"/>
        <output message="getWeatherOut"/>
   </operation>
</portType>
```

WSDL – Descripción Concreta

- La descripción concreta asocia a una descripción abstracta una dirección de red concreta, un protocolo de comunicación y estructuras de datos concretas
- Los tres elementos XML que componen la descripción concreta son:
  - <wsdl:binding>
  - <wsdl:service>
    - <wsdl:port>

WSDL – binding

- Este elemento asocia un “portType”, y sus mensajes y operaciones, a un protocolo de transporte y un formato de mensaje

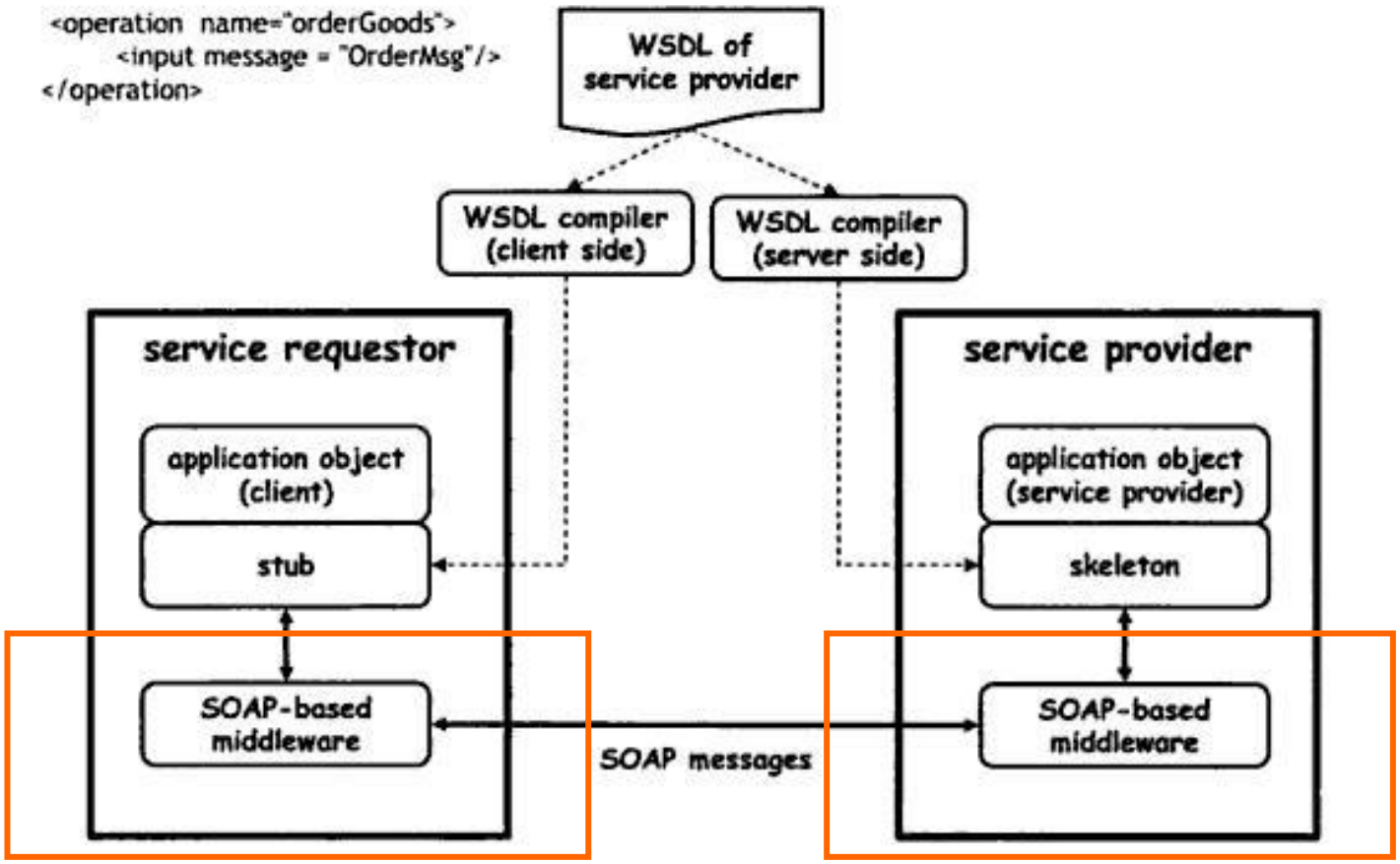
```
<binding name="weatherServiceSoapBinding"
        type="getWeatherPortType" >
        <soap:binding style="rpc" transport="http"/>
        <operation name="getWeather">
              <input soap:body use="encoded"/>
              <output soap:body use="encoded"/>
        </operation>
</binding>
```

WSDL – service y port

- El elemento "*port*" especifica la dirección de red para un determinado "*binding*"
- El elemento "*service*" es un contenedor de elementos "*port*"

```
<service name="WeatherService">
   <port name="weatherServicePort"
         type="tns:weatherServiceSoapBinding">
         <soap:address
               location="http://.../weather/"/>
   </port>
</service>
```

Cómo lo uso?



Proporcionado por plataformas de desarrollo

Universal Description, Discovery and Integration (UDDI)

UDDI

- Estándar de la OASIS que provee una forma estándar de publicar, categorizar y buscar Web Services
  - Actualmente en la versión 3.0.2
- UDDI define
  - Un directorio y un modelo de datos para almacenar información de servicios y negocios
  - tres interfaces para utilizar el registro UDDI
    - Inquiry (Búsqueda de servicios)
    - Publish (Publicación de servicios)
    - Subscribe (Notificación de cambios)

Registro UDDI

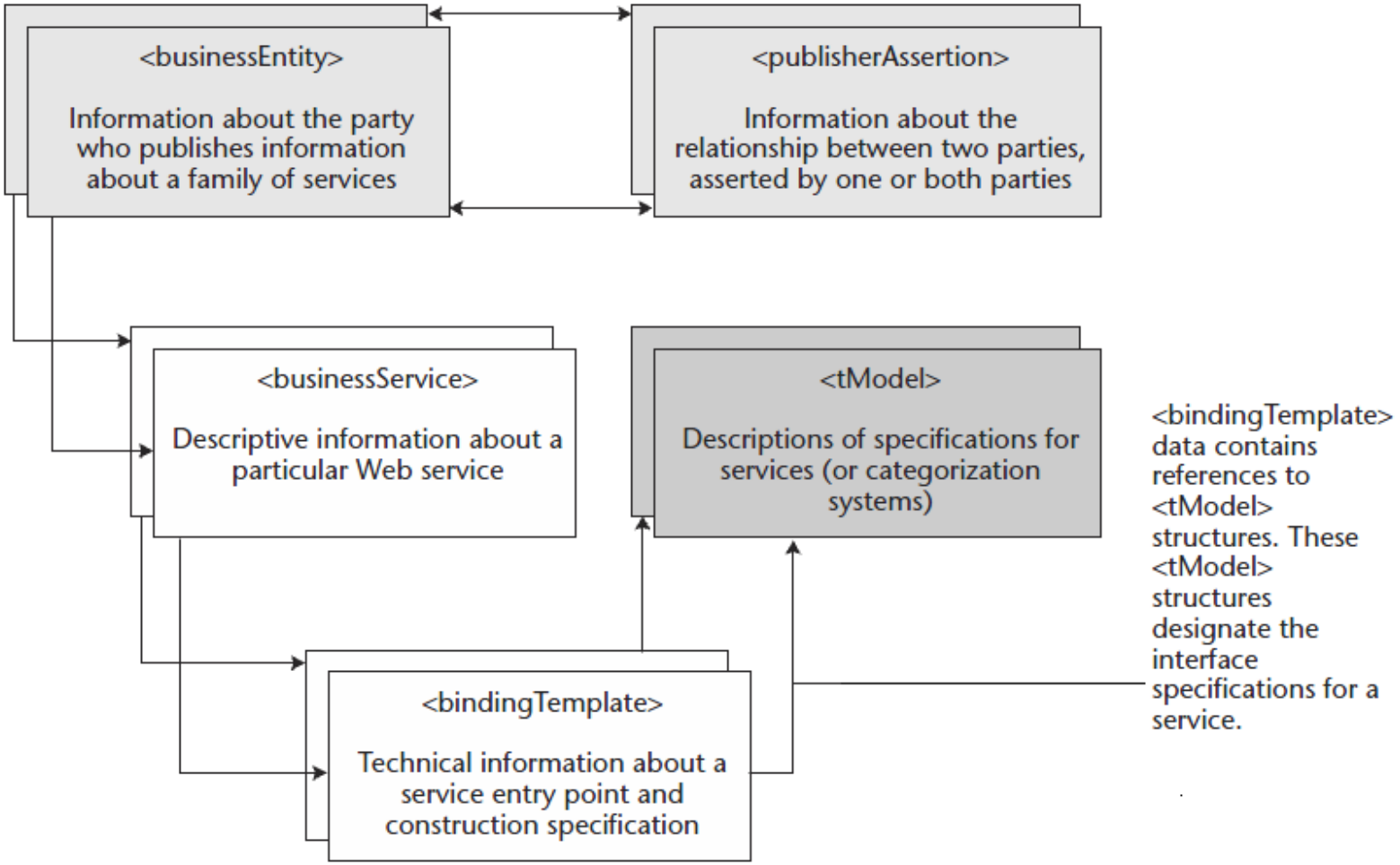
- El registro de un negocio en UDDI tiene tres partes:
  - Páginas blancas - dirección, contacto y otros identificadores conocidos.
  - Páginas amarillas - categorización industrial basada en taxonomías.
  - Páginas verdes - información técnica sobre los servicios que aportan las propias empresas.

Registro UDDI

- El esquema original de classificación de UDDI estaba basado en una taxonomía de negocios del gobierno de US

- Las versiones recientes de UDDI tienen soporte para la definición de taxonomías personalizadas

Modelo de Datos UDDI



Developing Java Web Services. Ramesh Nagappan, Robert Skoczylas, Rima Patel Sriganesh. Wiley Publishing. 2003.

Interfaces UDDI

- El estándar UDDI define interfaces que pueden ser utilizadas, por proveedores y consumidores de Web Services

- Dichas interfaces están descriptas utilizando WSDL y pueden ser accedidas a través de mensajes SOAP sobre HTTP

Interfaces UDDI

- Interfaz de Publicación (publish)
  - Un proveedor de Web Services utiliza esta interfaz para publicar, actualizar o eliminar información en el registro UDDI
  - Dos tipos de operaciones: save y delete
- Interfaz de Búsqueda (inquiry)
  - Un consumidor de Web Services utiliza esta interfaz para buscar Web Services, proveedores de Web Services e información de los mismos
  - Dos tipos de operaciones: find y get

Interfaces UDDI: Subscripción

- Objetivo: proveer un mecanismo a los clientes, para recibir información concerniente a los cambios que se realizan en el registro UDDI

- Dos mecanismos de comunicación
  - Sincrónico
    - Operación getSubscriptionResults
  - Asincrónico
    - Vía mail o Web Services SOAP
    - Clientes pueden implementar la operación notify_subscriptionListener

Sin embargo…

- Complejo de consultar
  - Basado en WSDL y SOAP
- Consultar servicios dinámicamente no es real
  - Ninguna empresa corre este riesgo
    - No es el servicio que se busca, mala performance, etc
- Moraleja:
  - Necesidad de catálogos de servicios
  - UDDI no es la solución
- Catalogo de servicios del Estado no UDDI
  - https://www.gub.uy/agencia-gobierno-electronico-sociedad-informacion-conocimiento/tematica/catalogo-plataforma-interoperabilidad

Resumen

- SOAP
  - Define el formato y transporte de los mensajes
- WSDL
  - Describe las interfaces de los servicios
  - Describe operaciones, mensajes, estructuras de datos, protocolos de transporte, etc
- UDDI
  - Permite el desarrollo de directorios de “Web Services”
  - Clasificación de servicios por parte de proveedores
  - Búsqueda de servicios por clientes/consumidores
  - Conceptos útiles, mala tecnología

Limitaciones

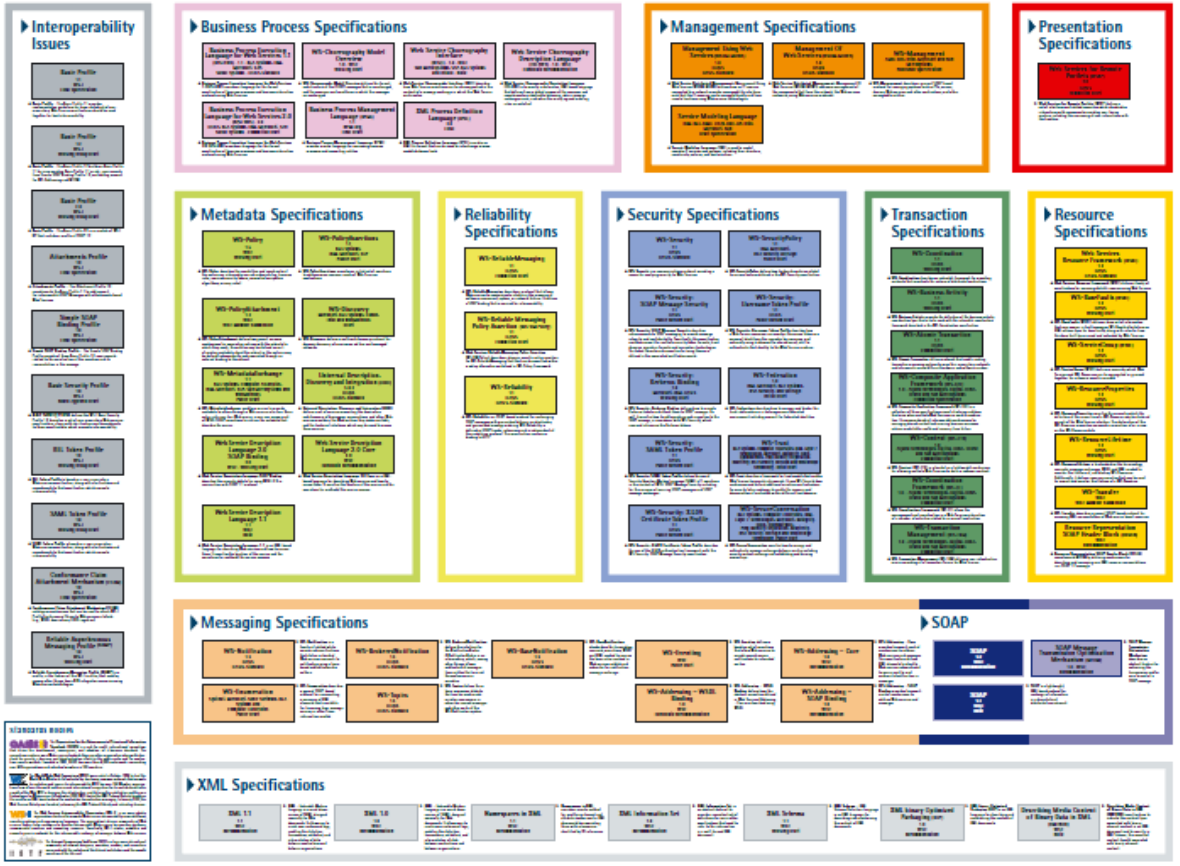
- Serialización a SOAP
- Descripción de características no funcionales en un WSDL
- Seguridad? Transacciones? Mensajería?
  - Los Web Services básicos no proveen soluciones a este tipo de requerimientos empresariales

Tecnologías avanzadas para el desarrollo de Web Services

Estándares Avanzados de WS

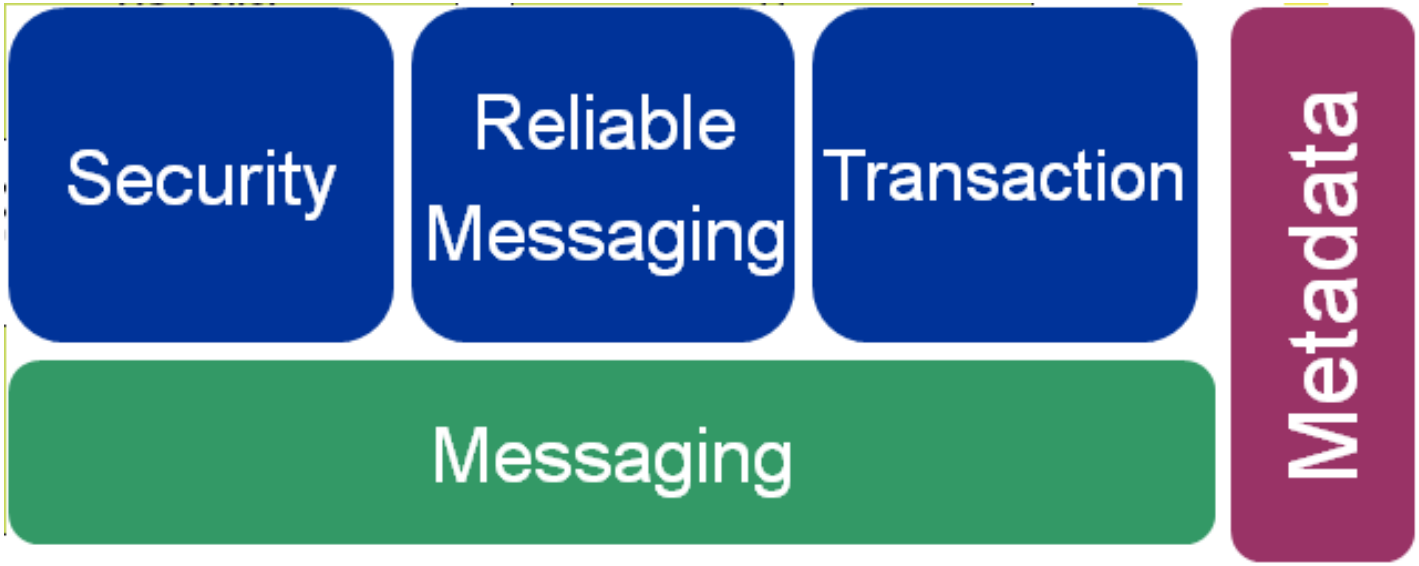
- Los estándares básicos no abordan problemáticas comunes en contextos empresariales
  - Confiabilidad, seguridad, transacciones, etc.
- Surgen entonces un conjunto de nuevas especificaciones (conocidas como WS-* o “segunda generación de estándares”)
- Cada una aborda una problemática específica y están orientadas a bloques y a su composición

Estándares Avanzados de WS



http://www.innoq.com/soa/ws-standards/poster/

Segunda Generación de Estándares para WS



Conceptos básicos en Seguridad

Confidencialidad, integridad, autenticación y no repudio

Requerimientos empresariales

- Confidencialidad de la información
  - ¿Cómo podemos prevenir que terceros visualicen los mensajes?

- Integridad de la información
  - ¿Cómo podemos prevenir que se modifiquen los mensajes entre emisor y receptor?

Requerimientos empresariales

- Autenticación de usuarios
  - ¿Cómo sabemos quién es el emisor del mensaje?
  - ¿Cómo se puede probar que es quién dice ser?

- No repudio
  - ¿Puede el emisor/receptor decir que:
    - no envío el mensaje?
    - envió un mensaje diferente al recibido?

Confidencialidad

- Cifrado simétrico

- Cifrado asimétrico

Cifrado simétrico

- Es la forma tradicional de la criptografía
  - Su origen conocido se remonta al menos a la época de los romanos
- Los algoritmos simétricos utilizan la misma clave para encriptar y desencriptar la información
  - $Dk(Ek(x)) = x$
- A la clave se le denomina “clave secreta”, “Llave secreta”, “secreto compartido” (Secret-Key o shared key en inglés)
- Requieren una comunicación previa de la clave entre las entidades participantes de forma segura (secreta)

Ejemplo: Cifrado César

- Data de la época de Julio César (siglo I A. C.)
- Es un tipo de cifrado por sustitución en el que una letra en el texto original es reemplazada por otra letra que se encuentra un número fijo de posiciones más adelante en el alfabeto.

- Ejemplo:

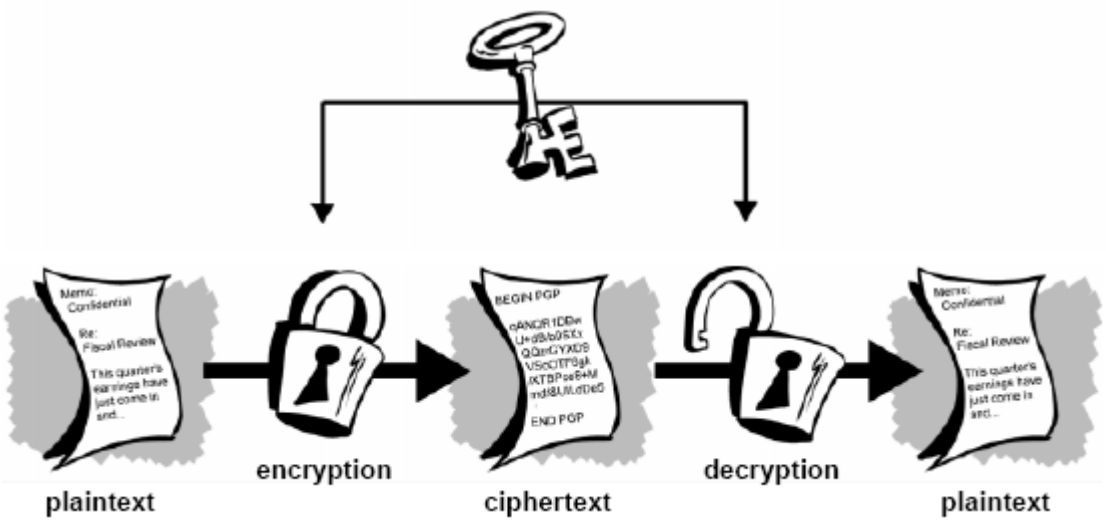
  Texto original: WIKIPEDIA, LA ENCICLOPEDIA LIBRE
  Texto codificado: CÑPÑVKJÑG, QG KSIÑIQUVKJÑG QÑHXK

Cifrado asimétrico

- Los métodos asimétricos, conocidos como de clave pública, se basan en encontrar un criptosistema donde se tienen dos claves distintas para encriptar y desencriptar, y es computacionalmente imposible obtener la clave de descifrado (*dk*) a partir de la clave de cifrado(*ek*) (o viceversa)

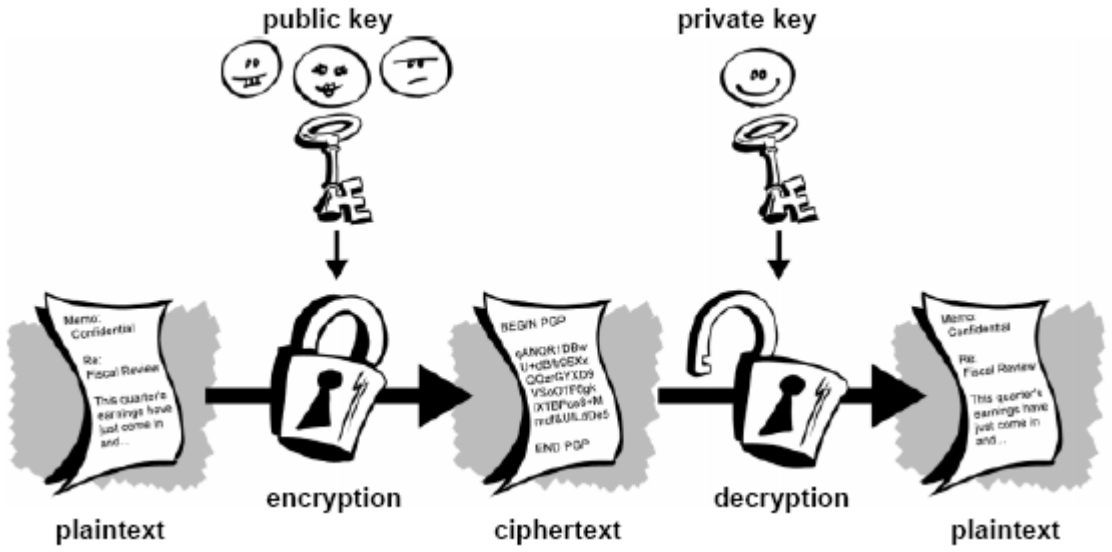
- De esa forma, una de las claves puede ser hecha pública

- Ejemplos: Certificados digitales



Cifrado simétrico

Cifrado asimétrico



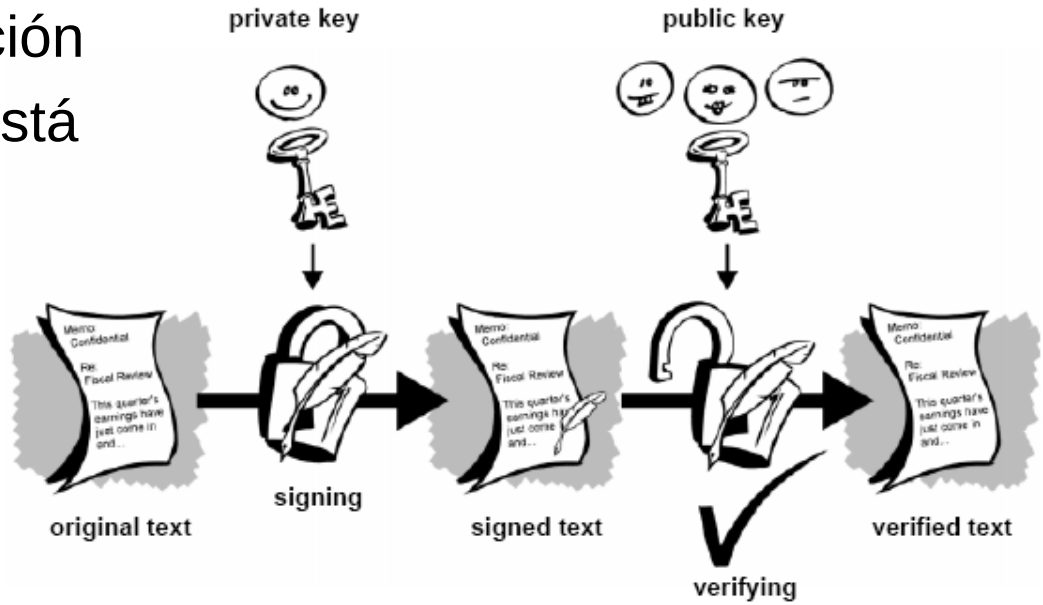
Cifrado asimétrico

- Además de mantener la confidencialidad de los datos…

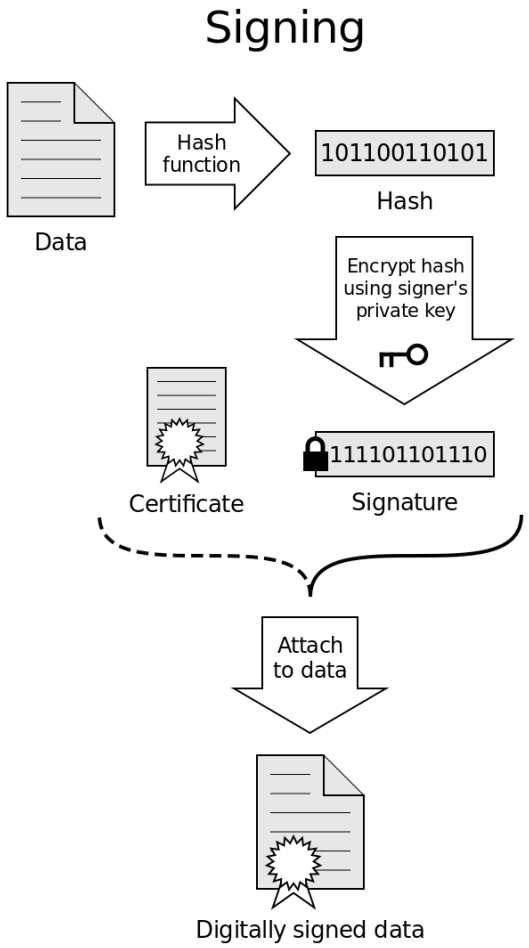
¿para qué se podrían usar las claves públicas y privadas?

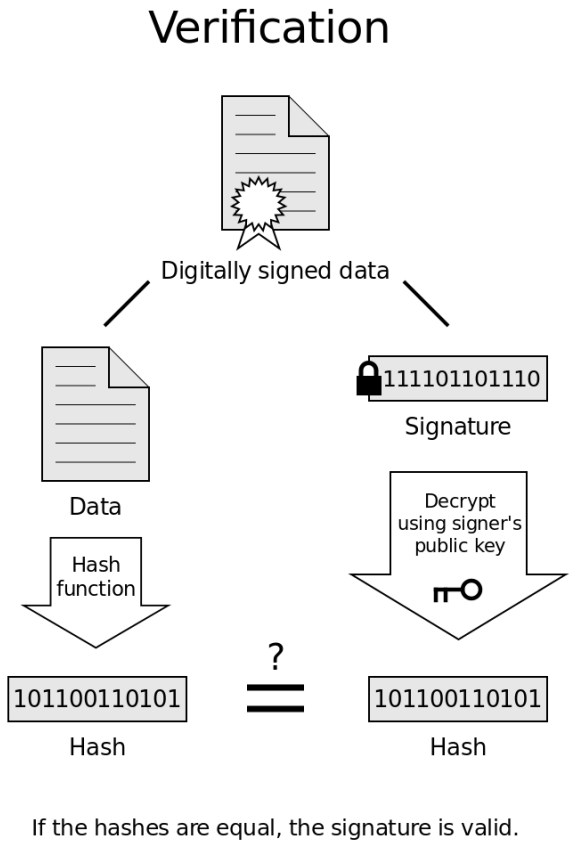
Firma digital

- Las firmas digitales permiten al receptor de la información verificar
  - La autenticidad del origen de la información
  - Que la información está intacta



Firma digital





Firma digital

1. Se calcula el hash de los datos
2. Se encripta el hash con la clave privada
   - A este hash cifrado se le denomina firma
3. Se adjunta la clave pública y firma al mensaje
   - Se obtiene un mensaje firmado

Verificación de firma digital

1. Se calcula el hash de los datos
2. Se desencripta la firma con la clave pública
   - Se obtiene el hash de datos original
3. Se comparan el hash de datos original y calculado
   - En caso de ser iguales, el mensaje llegó incambiado y es de quién dice ser
   - En caso contrario, alguien modificó el mensaje o no está firmado por quién dice ser.

Firma digital

- Los algoritmos asimétricos son lentos
  - Por lo tanto, no se cifra todo el mensaje
  - Se cifra solo el hash del mensaje

- Cifrar todo el mensaje sería costoso
  - Recordar los mensajes pueden variar en tamaño

No repudio

- Servicio de seguridad que previene que un emisor niegue haber remitido un mensaje (cuando realmente lo ha emitido) y que un receptor niegue su recepción (cuando realmente lo ha recibido).

- En el primer caso el no repudio se denomina en origen y en el segundo en destino.

[Ribagorda:1997]

Seguridad en WS

Escenario

- El Web Service requiere autenticar de las aplicaciones clientes
- El Web Service requiere mantener la confidencialidad de cierta información intercambiada
- El Web Service debe rechazar mensajes enviados por el cliente hace más de 5 minutos

Introducción

- Dos alternativas para proveer seguridad entre Web Services:
  - Seguridad en Capa de Transporte (SSL/TLS)
    - Seguridad provista por el canal de comunicación
  - Seguridad a nivel de Mensaje SOAP
    - Seguridad incluida en el propio mensaje

Introducción

- Dos alternativas para proveer seguridad entre Web Services:
  - Seguridad en Capa de Transporte (SSL/TLS)
    - Seguridad provista por el canal de comunicación
  - Seguridad a nivel de Mensaje SOAP
    - Seguridad incluida en el propio mensaje

Seguridad en capa de transporte: Autenticación HTTP

- HTTP soporta diferentes mecanismos para el control de recursos Web
  - Mecanismos basados en cabezales HTTP 401 y WWW-Authenticate.

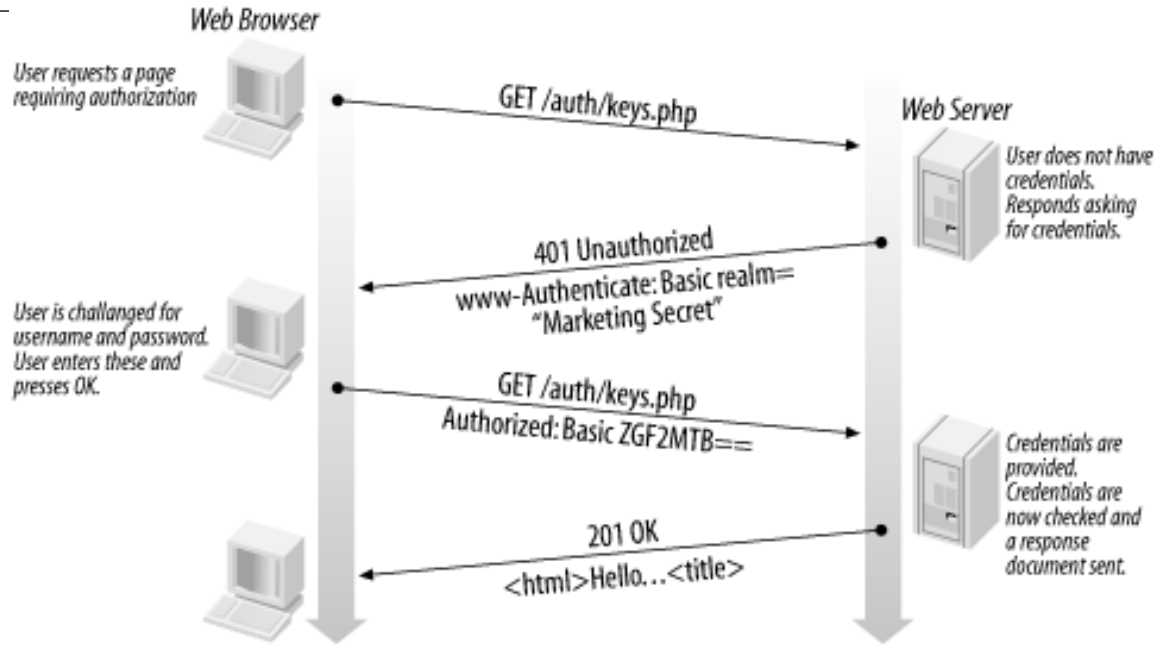
- Opciones: Basic, Digest, NTLM.

Seguridad en capa de transporte: Autenticación HTTP

- HTTP soporta diferentes mecanismos para el control de recursos Web
  - Mecanismos basados en cabezales HTTP 401 y WWW-Authenticate.

- Opciones: Basic, Digest, NTLM.

Basic Authentication



- Seguridad a nivel de protocolo HTTP
- El cliente envía usuario y contraseña codificado en base64
- Contraseña NO cifrada
- Sensible a “Sniffing”

Basic Authentication

- HTTP Basic es un mecanismo simple de seguridad de tipo challenge/response que pueden utilizar los servidores para solicitar información de autenticación a un cliente (usuario/contraseña).

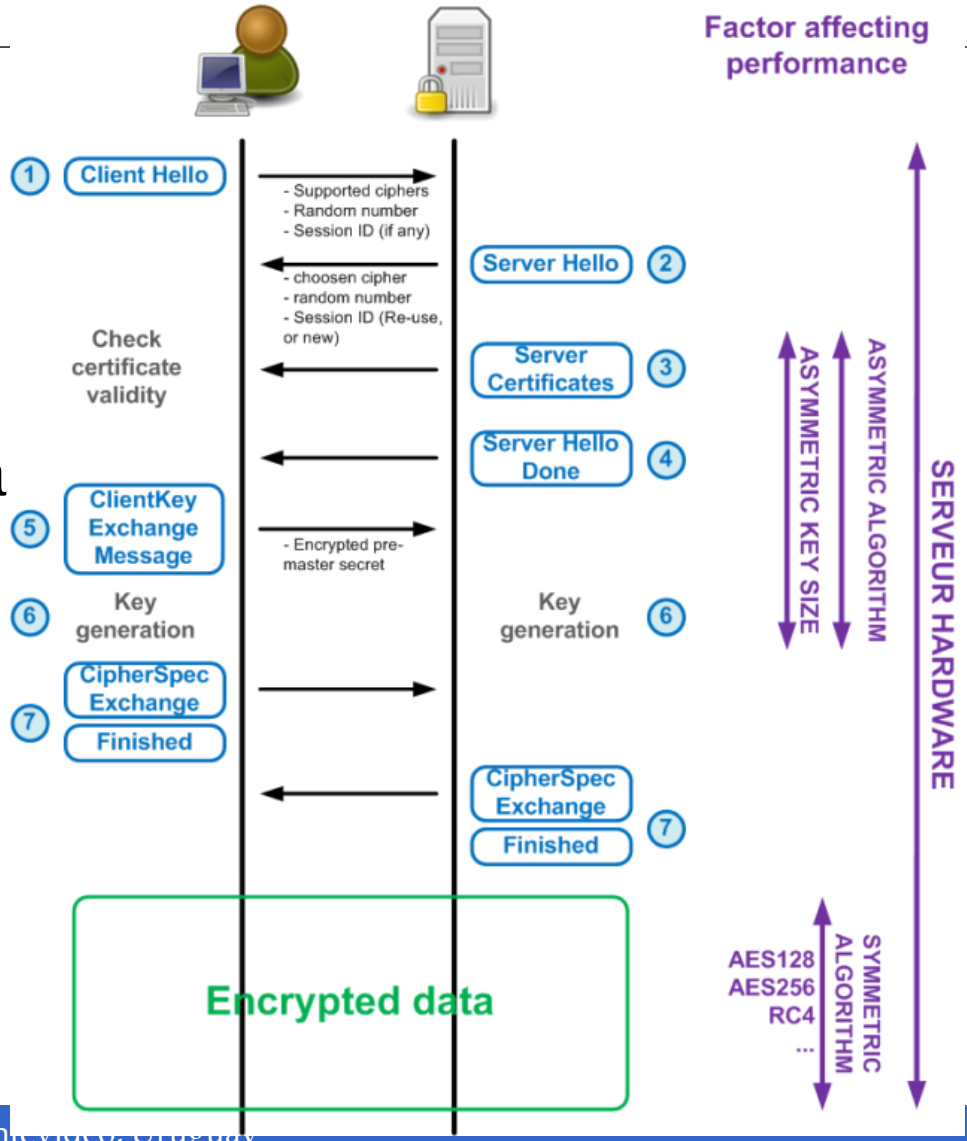
- El cliente envía la información de autenticación en un cabezal http (Authorization) codificada en base64.

Basic Authentication

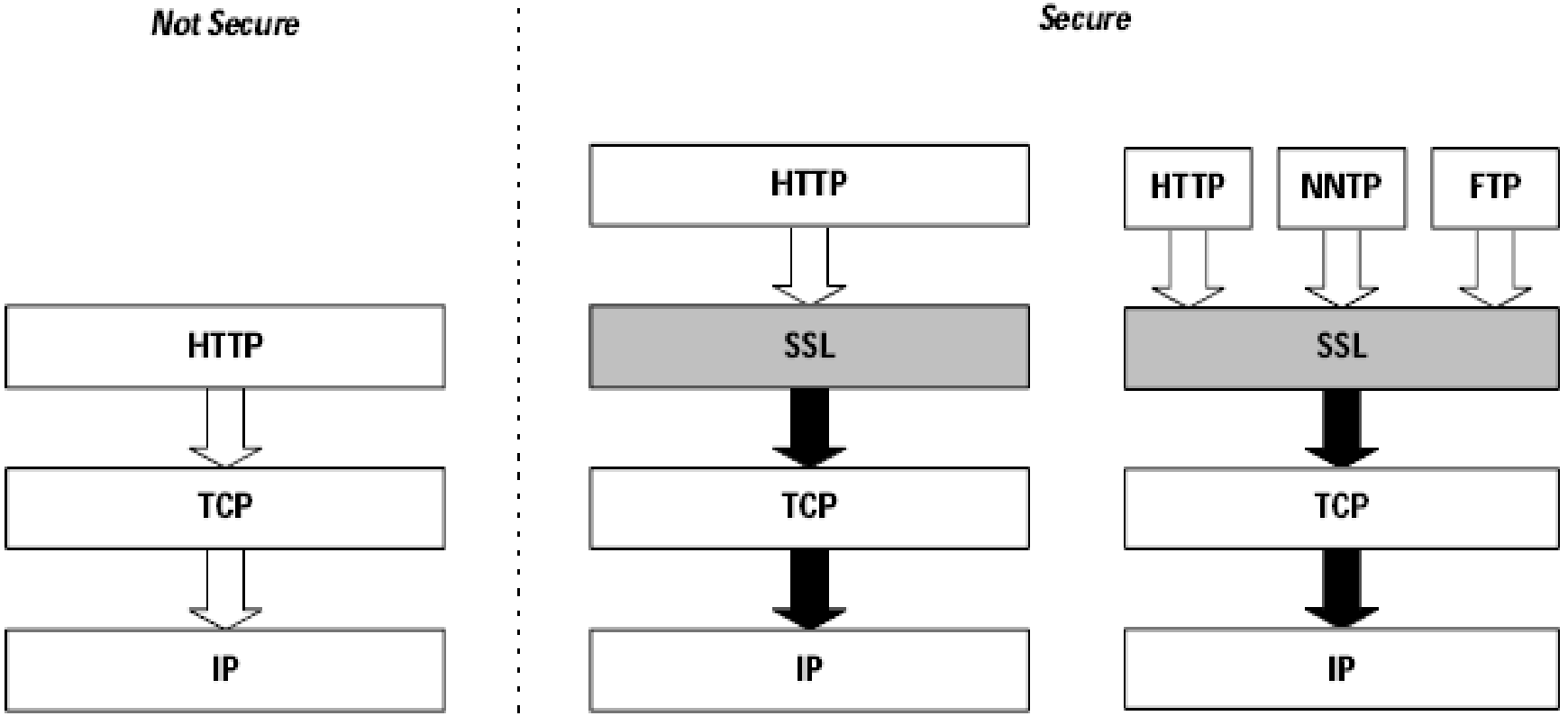
1. El cliente hace una solicitud al servidor
2. El servidor envía una respuesta con un código de estado http 401, un mensaje de error de autenticación y un cabezal http (WWW-Authenticate).
   - P. ej: WWW-Authenticate Basic realm="Mi sitio"
   - El atributo realm identifica el dominio de seguridad
3. La mayoría de los clientes capturan este error y solicitan al usuario final un userID y contraseña.
4. El cliente envía el usuario y contraseña en un cabezal http Authorization
   - El usuario y contraseña viajan codificados en base64 con formato usuario:contraseña
   - P. ej: Authorization: Basic aHR0cHdhdGNoOmY=

Seguridad en capa de transporte: (SSL/TLS)

- Es estándar
- Autenticación mutua de servidores
- Integridad y confidencialidad de la información.
- Sesiones seguras
- Transparente a las aplicaciones
  - Configurado a nivel de servidor Web



Seguridad en Capa de Transporte (SSL/TLS)



SSL & TLS Essentials: Securing the Web. Stephen A. Thomas. John Wiley & Sons. 2000.

Sin embargo…

- Cifrado y firmado de mensajes punto a punto
  - Intermediarios pueden dejar “huecos” de seguridad
- Autenticación de usuarios punto a punto
  - Autenticación del usuarios trabajosa si hay intermediarios
- Cifrado de todo el mensaje
- Dependiente del protocolo de transporte

Seguridad en Web Services

- Dos alternativas para proveer seguridad a Web Services:
  - Seguridad en Capa de Transporte
    - Seguridad provista por el canal de comunicación
    - Autenticación HTTP o SSL/TLS
  - Seguridad a nivel de Mensaje SOAP
    - Seguridad incluida en el propio mensaje
    - WS-Security, WS-SecureConversation, WS-Trust

WS-Security

- Seguridad a nivel de mensaje, no transporte
- Garantiza la confidencialidad e integridad de los mensajes SOAP
- Brinda mecanismos para adjuntar “tokens” de seguridad a los mensajes SOAP
  - Usuario/Password, certificados X.509, etc
  - Permite la autenticación de usuarios!
- Es un estándar
  - Actualmente en versión 1.1

WS-Security

- XML Encryption
  - Estándar para el cifrado XML
  - WS-Security: Versión 1.1 (actual)
- XML Signature
  - Estándar para la firma de documentos XML
  - WS-Security: versión 1.1
  - Actualmente versión 2.0
    - Mejoras de performance, simplicidad y streaming
- WS-Security “ordena” XMLEncryption y XMLSignature para firmar mensajes SOAP

XML Encryption

```
<PaymentInfo
xmlns="http://example.org/paymentv2">
 <Name>John Smith</Name>
 <CreditCard Limit="5,000" Currency="USD">
  <Number>4019 2445 0277 5567</Number>
  <Issuer>Example Bank</Issuer>
  <Expiration>04/02</Expiration>
 </CreditCard>
</PaymentInfo>
```

```
<PaymentInfo xmlns="http://example.org/paymentv2">
 <Name>John Smith</Name>
 <EncryptedData
Type="http://www.w3.org/2001/04/xmlenc#Element"
          xmlns="http://www.w3.org/2001/04/xmlenc#">
  <CipherData>
   <CipherValue>A23B45C56</CipherValue>
  </CipherData>
 </EncryptedData>
</PaymentInfo>
```

XML Signature

```
<Envelope xmlns="urn:envelope"></Envelope>

<Envelope xmlns="urn:envelope">
 <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
  <SignedInfo>
    <CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315#WithComments"/>
    <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
    <Reference URI="">
     <Transforms>
      <Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
     </Transforms>
     <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
     <DigestValue>uooqbWYa5VCqcJCbuymBKqm17vY=</DigestValue>
    </Reference>
  </SignedInfo>
  <SignatureValue>KedJuTob5gtvYx9qM3k3gm7kbLBwVbEQRl26S2tmXjqNND7MRGtoew==</SignatureValue>
  <KeyInfo>
    <X509Data>
      <X509SubjectName>CN=Ed Simon,O=XMLSec Inc.,ST=OTTAWA,C=CA</X509SubjectName>
      <X509Certificate>MIID5jCCA0+gA...lVN</X509Certificate>
    </X509Data>
  </KeyInfo>
 </Signature>
</Envelope>
```

Además…

- Permite adjuntar tokens de seguridad
  - Autenticación de usuarios!
  - Diferentes tipos de token:
    - Username Token
    - Binary Security Token
    - SAML (Security Assertion Markup Language)

- Inclusión de timestamps de los mensajes
  - Creación y vencimiento

Escer
Depos

Mensajes SO
firmados vía
Servicio y cl
vía

```
<envelope>
<header>
…
<security>
 <binarysecuritytoken>…</binarysecuritytoken>
 <signature>…</signature>
</security>
</header>
<body>
  <depositar>
   <EncryptedData><CipherData>
     <CipherValue>A23B4…5C56</CipherValue>
   </CipherData></EncryptedData>
   <fecha>09-03-2009</fecha>
  </depositar>
</body>
</envelope>
```

Security Token

Firma

Cifrado

Ejemplo: Timestamp

- Define fecha de creación y vencimiento de los mensajes

```
<Envelope>
  <Header>
    <Security>
      <Timestamp>
        <Created>2006-06-19T16:22:29.281Z</Created>
        <Expires>2006-06-19T16:27:29.281Z</Expires>
      </Timestamp>
    </Security>
  </Header>
  <Body>
    <numberOfArcticles >42</numberOfArcticles>
  </Body>
</Envelope>
```

Ejemplo: Username Token

- Define mecanismos para autenticar usuarios basado en usuario y contraseña.
  - Username: nombre de usuario
  - Contraseña: texto plano o digest.
  - Nonce: String único para detección de ataques replay.
  - Created: fecha/hora de creación del mensaje.

```
<Envelope>
 <Header>
  <Security>
   <UsernameToken>
    <Username>NNK</Username>
    <Password Type="...#PasswordDigest">weYI3nXd8LjMNVksCKFV8t3rgHh3Rw==</Password>
   <Nonce>WScqanjCEAC4mQoBE07sAQ==</Nonce>
   <Created>2003-07-16T01:24:32Z</Created>
  </UsernameToken>
  </Security>
 </Header>
 <Body>
    ...
 </Body>
</Envelope>
```

SSL vs WS-Security

Características	SSL	WS-Security
Estándar	Sí	Sí
Seguridad	A nivel de transporte	A nivel de applicación
Confidencialidad/ Integridad	Todo el mensaje.	Todo o partes del mensaje
Autenticación de clientes	X.509, Basic Authentication	Usuario/Contraseña, X.509, Kerberos, SAML.
Protocolo	Potencialmente varios. HTTP el más popular.	Potencialmente varios. HTTP el más popular.
Algoritmos de cifrado	Simétricos	Simétricos y asimétricos En gral, asimétrico
Sesiones seguras	Sí	No, pero… WS-SecureConversation

¿Cuándo usar SSL?

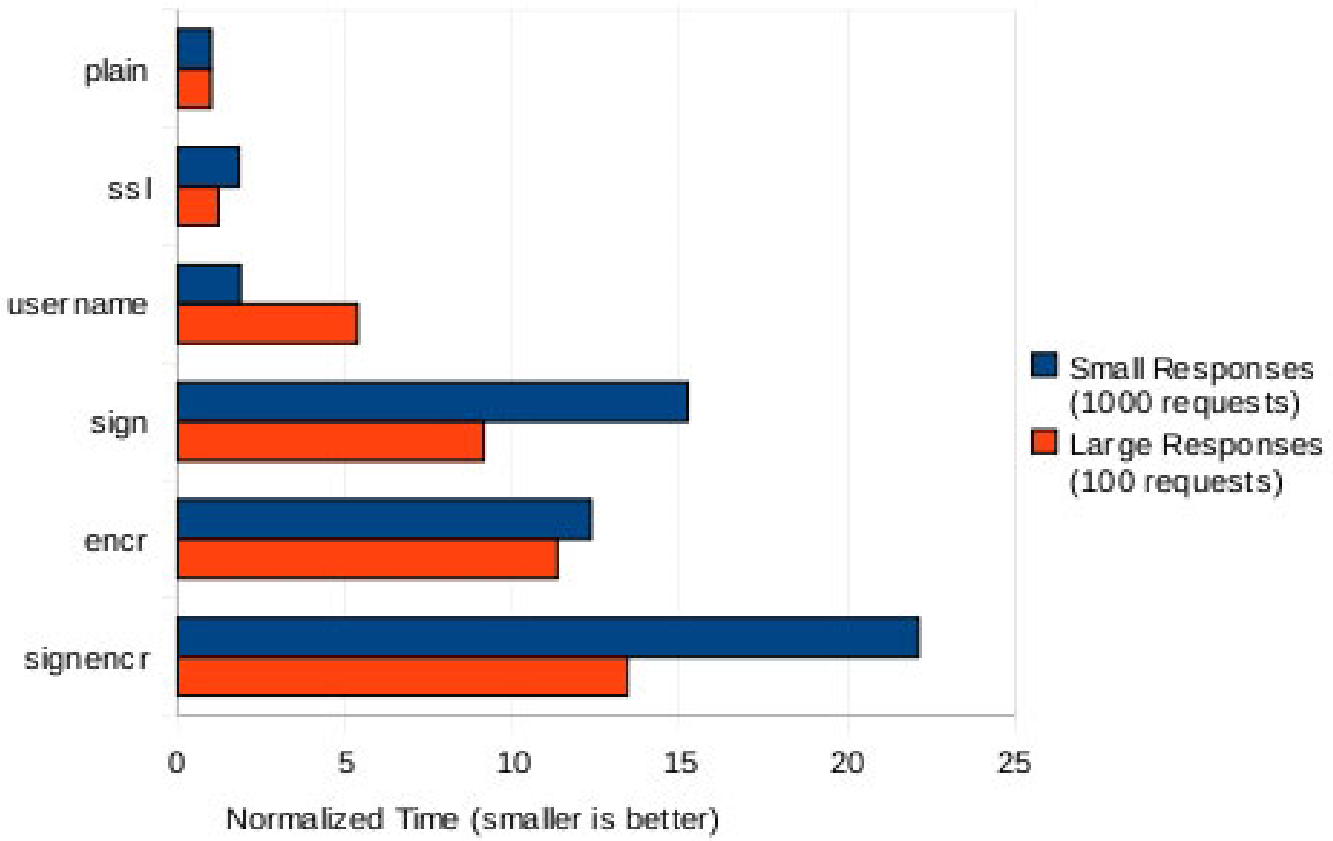
- Es la solución por defecto a utilizar
- Gran volumen de transacciones (rápido y escalable)
- No hay procesamiento por parte de intermediarios
  - p.ej: filtrado o ruteo basado en contenido
- Asegurar adjuntos en Web Services
  - SSL encripta todos los paquetes a nivel de transporte
  - El header, body y attachments SOAP están todos asegurados

- Normalmente es la solución que utilizaremos a no ser existan intermediarios no deseados

¿Cuándo usar WS-Security?

- Los intermediarios encriptan partes de los mensajes y deben dejar otras en texto plano
- Autenticación en el origen del mensaje
  - quién/qué envió el mensaje?
- Seguridad de mensajes persistente
  - La integridad y confidencialidad de los mensajes persiste más allá del mecanismos de transporte

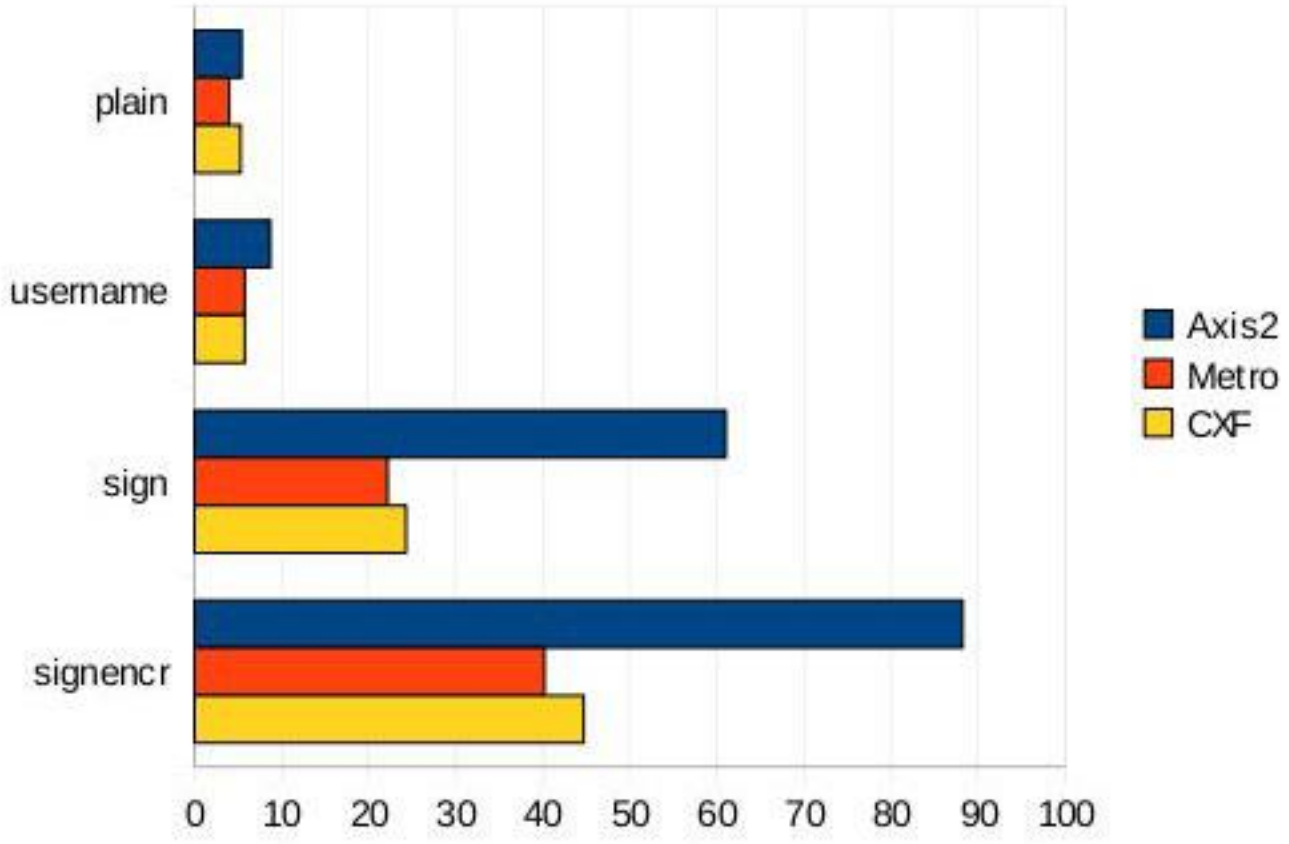
Costos: Performance



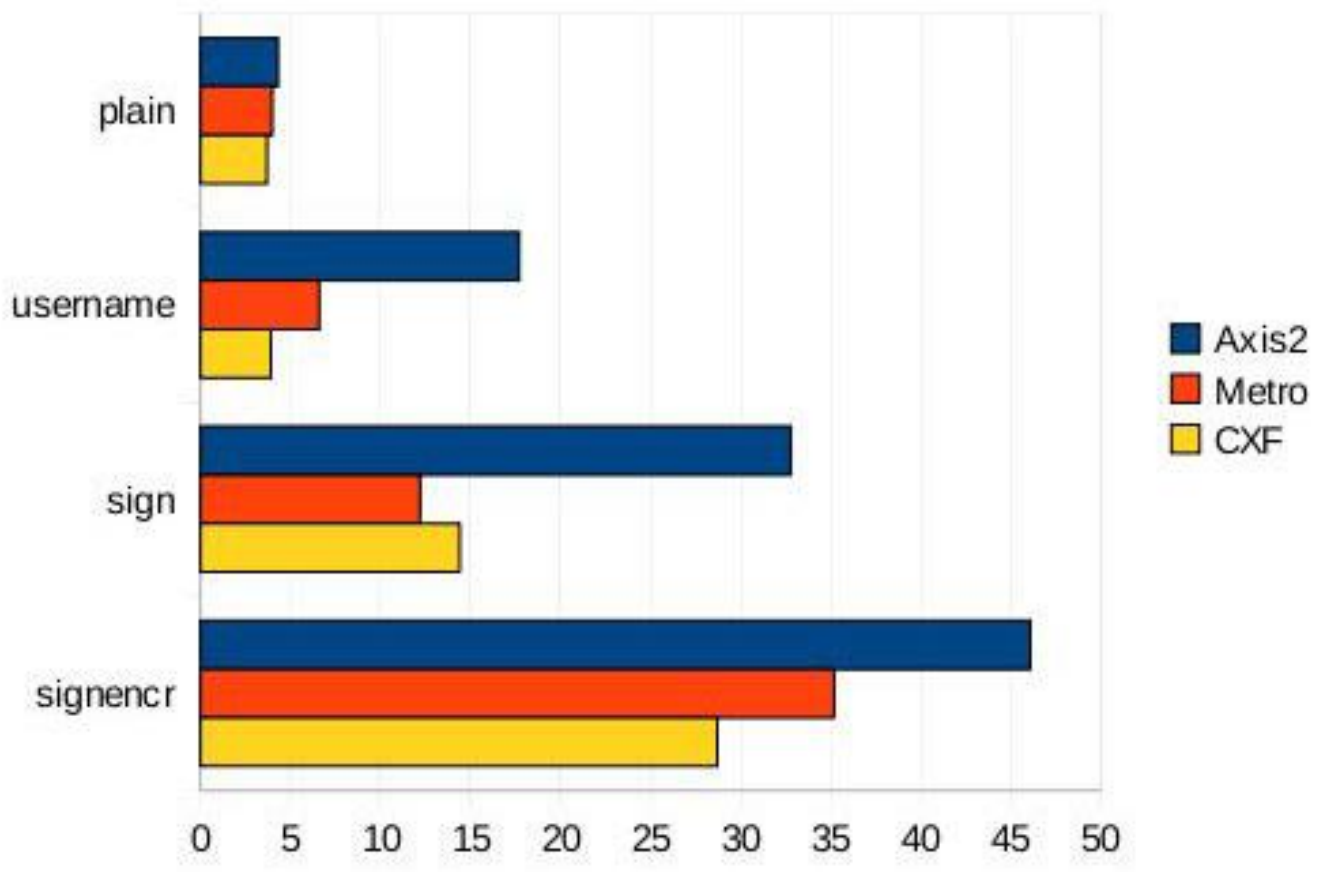
http://www.ibm.com/developerworks/java/library/j-jws6/index.html

Además…

- Small responses



Large responses



http://www.ibm.com/developerworks/java/library/j-jws14/index.html

Escenario

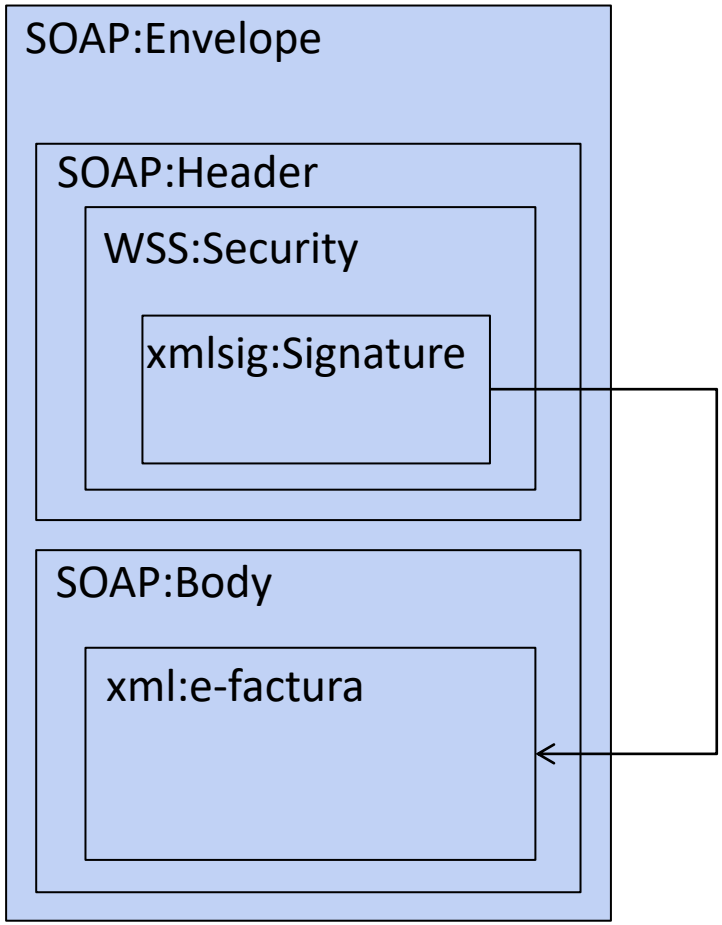
- El Web Service requiere autenticar de las aplicaciones clientes
- El Web Service requiere mantener la confidencialidad de cierta información intercambiada
- El Web Service debe rechazar mensajes enviados por el cliente hace más de 5 minutos

Caso de estudio: e-factura

- DGI publica un Servicio Web externo para factura electrónica que funciona como un **único** canal por el cual se brindan los **distintos servicios** de factura electrónica.
- Seguridad:
  - Autenticación (WS-Security)
  - Integridad (WS-Security)
  - Confidencialidad (SSL)
  - No repudio (Firma digital)
- Referencia: https://www.efactura.dgi.gub.uy/
- Web Service:

  https://efactura.dgi.gub.uy:6443/ePrueba/ws_eprueba?wsdl

Ejemplo de request



Fin

LINS

Laboratorio de Integracion de Sistemas