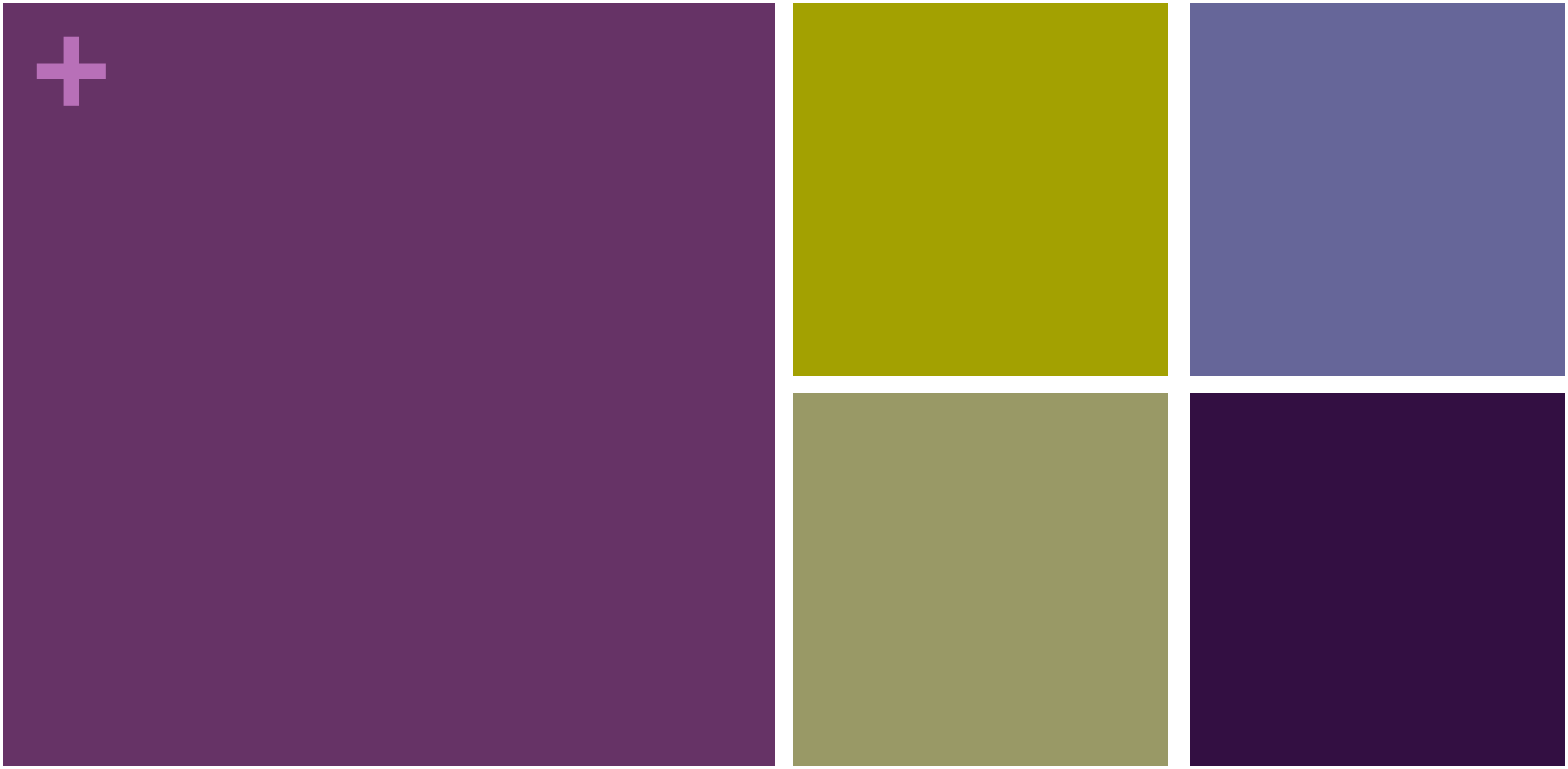


Sistemas Distribuidos
Servicios web

Rodrigo Santamaría

Servicios web

Introducción

- Definición
- Características
- Aplicaciones

IDL
SOAP
REST
XML/JSON-RPC

Introducción

Tipos de middleware

Middleware	Enfoque	Aparición	Comentarios
Petición-respuesta	Paso de mensajes	70s	Inicios, patrón todavía válido
RPC	Uso de métodos	1984	Prog. estructurada Muy utilizado
RMI	Uso de objetos	1995	Muy utilizado
Servicios Web	Métodos como servicios	1998 SOAP 2000 REST	Simple, muy utilizado
P2P	Arquitecturas horizontales	1999 (Napster)	Horizontal, muy utilizado

- Los middleware se adaptan a las necesidades y tecnologías del momento
- Algunos tipos de middleware no son necesariamente excluyentes, p. ej.:
  - El protocolo de petición-respuesta se aplica en servicios web o RMI
  - Un middleware P2P puede dar soporte horizontal a un servicio web

Introducción

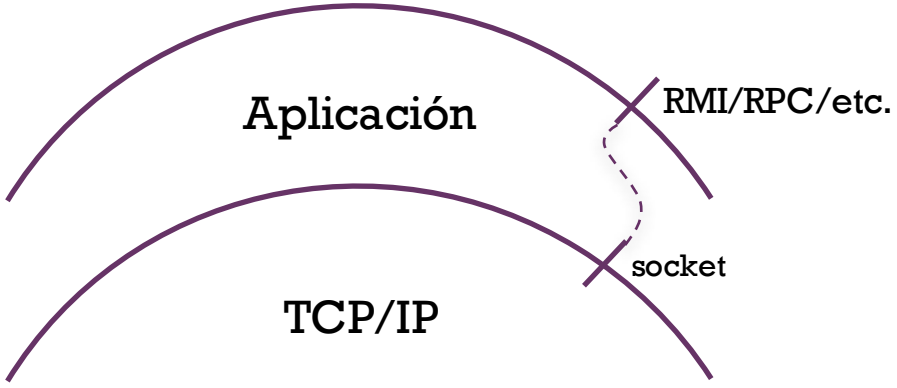
Servicio web

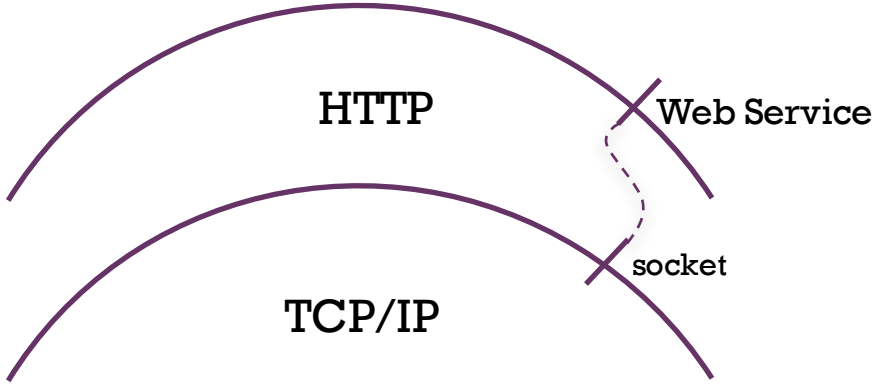
- Middleware que provee una interfaz entre clientes y servidores **más general** que la de un navegador web (HTTP)
- Los clientes acceden a las operaciones publicadas en la interfaz del servicio web a través de **peticiones XML, transmitidas vía HTTP**
- Los servicios web tienen cada vez más importancia en Internet, y suponen la infraestructura *middleware* básica para la computación grid y la computación en la nube.

Introducción

TCP/IP, middleware y servicios web

- **TCP/IP**: transmisión de bytes *mediante* de TCP/UDP+IP
- **Middleware**: transmisión de mensajes *sobre* TCP/UDP+IP
  - **Servicio web**: aprovechando HTTP





https://www.quora.com/What-is-the-difference-between-a-socket-and-a-web-service

Introducción

Servicio web: HTTP

- HTTP es un protocolo por el que los clientes (navegadores) pueden acceder a páginas web y otros recursos, a través de referencias a URIs
  - Aunque las páginas web puedan contener applets, etc. la funcionalidad es reducida
- Los servicios webs permiten más variabilidad en los servicios ofertados
  - La localización de los servidores y el direccionamiento de peticiones y respuestas se realiza por protocolos como HTTP

Introducción

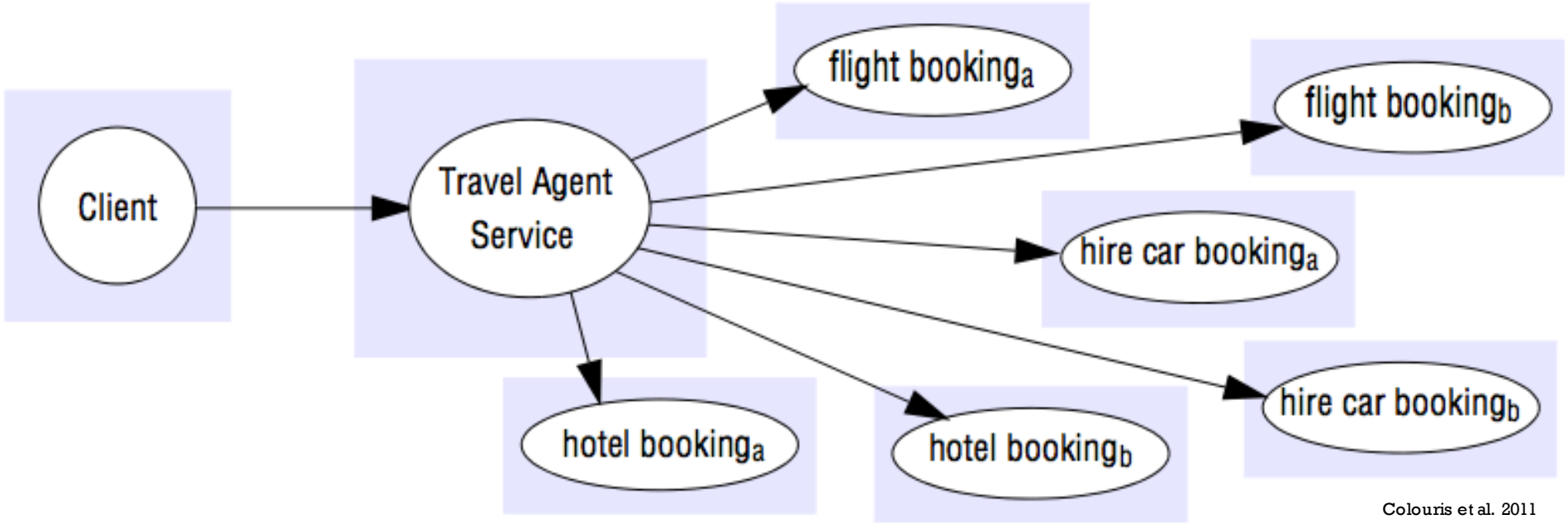
Servicios web: características

- **Combinación**: un servicio se puede combinar con otros
- **Patrones de comunicación**: síncrono, asíncrono o por eventos
- **Desacoplamiento**: minimización de dependencias entre servicios
- **Representación de mensajes**: textual (lo normal) o binario
- **Referencias al servicio**: mediante URIs
- **Transparencia**: XML u otros formatos de representación y URIs

Introducción

Servicios web: combinación

- Las operaciones de un servicio web pueden utilizar otros servicios web para sus operaciones



Colouris et al. 2011

Servicios web

Servicios web: patrones de comunicación

- Petición-respuesta síncrona
  - Es el patrón más usual
  - Invocamos al servicio y esperamos la respuesta a la petición
- Comunicación asíncrona
  - Se envía la petición y se continúa la ejecución
  - Incluso en patrones de petición-respuesta
- Mediante eventos
  - El cliente se suscribe a eventos ofrecidos por el servicio
  - P. ej., servicio web UTC que da el tiempo UTC periódicamente

Servicios web

Servicios web: desacoplamiento

- El desacoplamiento (*loose coupling*) se refiere a la minimización de las dependencias entre los servicios
  - Para ofrecer una mayor flexibilidad en la arquitectura
- Se consigue de dos modos
  - *Interfaces*: el uso de servicios a través de su interfaz permite desacoplar el servicio de su implementación
  - *Minimización de las interfaces*: reduce la dependencia de nombres de operación específicos
    - *Servicios orientados a datos*: los datos se vuelven más importantes que la interfaz y contienen la semántica de interoperabilidad

Servicios web

Servicios web: representación de mensajes

- Textual
  - SOAP representa los servicios y los mensajes en XML
  - Más espacio y tiempo de traducción
  - "Legible"
- Binario
  - Menos espacio
  - Más rápido (no hay traducción)
  - Ilegible

Servicios web

Servicios web: referencia y activación del servicio

- Los servicios se referencian generalmente mediante una URL
  - La referencia del servicio se conoce como punto final (*endpoint*)
- Un servicio web puede ejecutarse en la máquina de punto final, o en servidores secundarios en contacto con ella
  - Por ejemplo, en el caso de servicios con mucha demanda
- Un servicio web puede estar ejecutándose de manera continua, o sólo activarse cuando se realiza una petición
  - En cualquier caso, la URL del punto final está siempre disponible

Servicios web

Servicios web: transparencia

- Protege al programador de los detalles de la representación de los datos y asemeja una petición local a una remota
  - No es soportado por los servicios web directamente
  - P. ej.: SOAP requiere que sus mensajes se escriban en XML
- Existen APIs locales en cada lenguaje de programación para ocultar estos aspectos
  - Por ejemplo WSDL2java y Java2WSDL para Java+SOAP

Servicios web

Aplicaciones

- **Service Oriented Architectures** (SOA)
  - Principios de diseño para desarrollar sistemas distribuidos mediante servicios independientes que se publican de manera dinámica
  - Su implementación se realiza fundamentalmente a través de servicios web
- **Grid computing**
  - Un grid (parrilla) es un middleware para compartir recursos a gran escala (ficheros muy grandes/cálculos muy complicados)
    - Ámbito académico/científico/profesional (cálculo y datos intensivos)
  - Los servicios web permiten que los datos se analicen y filtren en remoto y sólo se devuelva al cliente la parte que le interese
- **Cloud computing**
  - Concepto que evoluciona de grid computing, más centrado en el ámbito de negocio (muchos usuarios, facilidad de uso)
  - Su implementación natural es mediante servicios web
    - Amazon Web Services (AWS): http://aws.amazon.com

Servicios web

- Introducción
- WSDL
- SOAP
- REST
- XML/JSON-RPC

WSDL

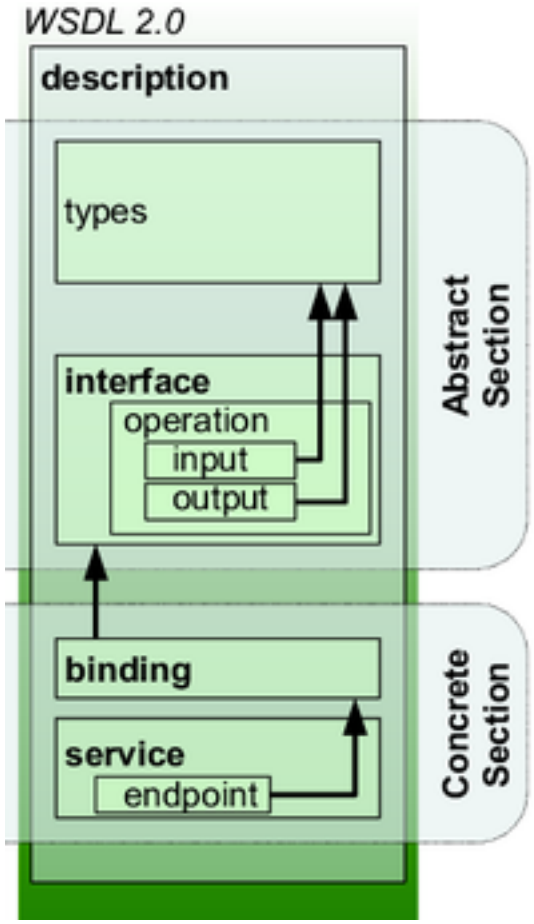
IDL

- Interface Definition Language
  - Lenguaje para especificar la interfaz de componentes software
- **Neutral** respecto al lenguaje de programación
  - Permite la comunicación entre componentes escritos en distintos lenguajes
- Uso común en llamadas a procedimientos remotos (RPCs)
  - CORBA IDL
  - Servicios Web (**WSDL**)
  - Facebook (Thrift)

WSDL

Web Service Definition Language

- Define un esquema XML para descripción de servicios:
  - **Servicio**: descripción del servicio
    - **Punto final**: dirección de acceso (típicamente una URL)
    - **Enlace (binding)**: modo de comunicación subyacente (típicamente HTTP)
    - **Interfaz**: definición de las operaciones y argumentos
      - **Operación**: cada una de las operaciones, y su E/S
      - **Tipo**: de los argumentos de E/S
- Palabras reservadas: xs (XML schema) y tns (target namespace)



http://en.wikipedia.org/wiki/Web_Services_Description_Language

WSDL

Secciones

- Sección **abstracta**
  - Definición de los tipos utilizados por el servicio (los tipos de las variables intercambiadas en los mensajes)
    - Un int de Java se transforma en su equivalente XML: `xs:int`
    - Una clase más compleja en Java se traduce a un objeto `xs:complexType` en XML
- Sección **concreta**
  - Define los puntos finales y su protocolo de enlace
- WSDL es **modular**
  - Un servicio puede darse bajo varios puntos de acceso o protocolos de direccionamiento

WSDL

Herramientas

- Hay herramientas para generar WSDLs a partir de las definiciones de interfaz escritas un determinado lenguaje
  - De modo gráfico
  - Sin necesidad de entrar en las complejidades de su estructura
  - WSDL4J (wsdl4j.sourceforge.org)
  - Java2WSDL
- Otras herramientas generan, a partir de un WSDL, un cliente para un determinado lenguaje de programación
  - WSDL2Java
- soapUI (http://www.soapui.org/)
  - Herramienta gratuita para testear WSDLs de SOAP

Servicios web

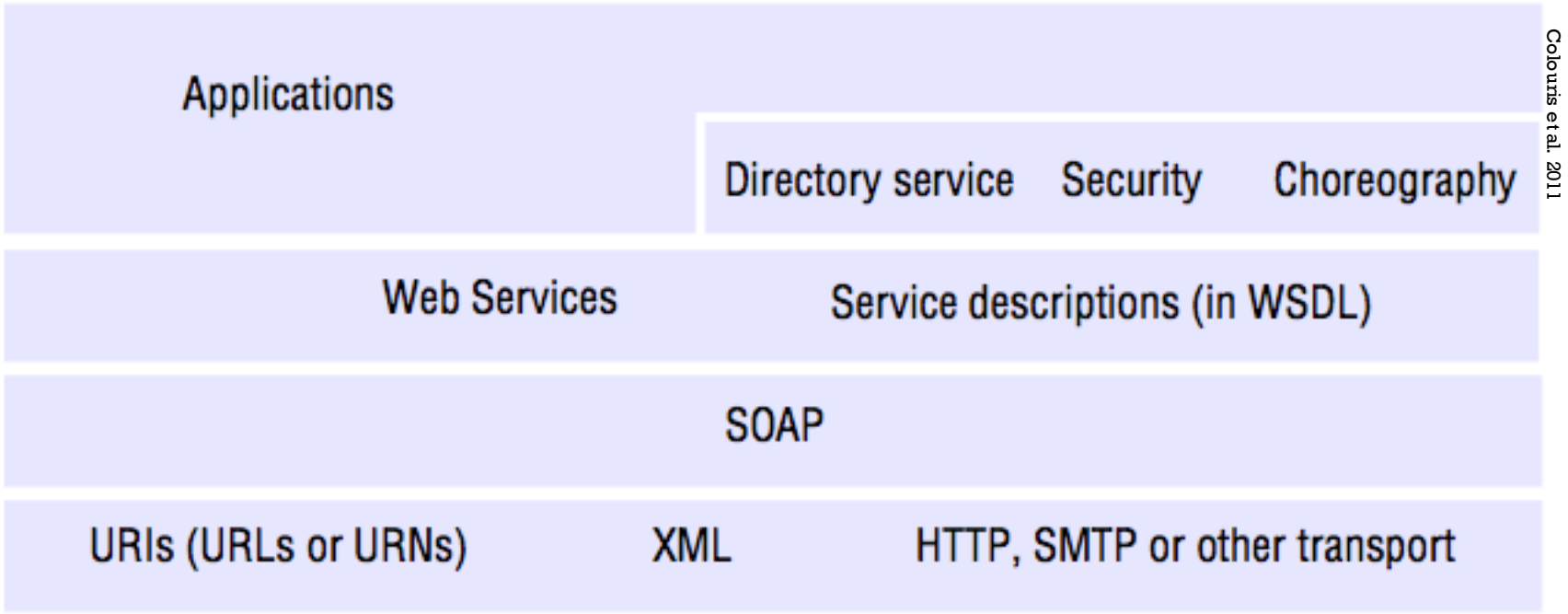
- Introducción
- IDL
- SOAP
- REST
- XML/JSON-RPC

SOAP

- **Simple Object Access Protocol** (Dave Winer et al. 1998)
  - Especificación para el intercambio de información estructurada en servicios web, a través de redes de ordenadores
- Se basa en tres componentes principales
  - **WSDL**: lenguaje de descripción del servicio
  - **HTTP/SMTP**: protocolo de comunicación
  - **XML**: lenguaje de especificación de peticiones y respuestas
- **Independiente**: puede usarse sobre servicios escritos en cualquier lenguaje
- **Neutral**: puede usarse sobre cualquier protocolo de transporte

SOAP

Arquitectura

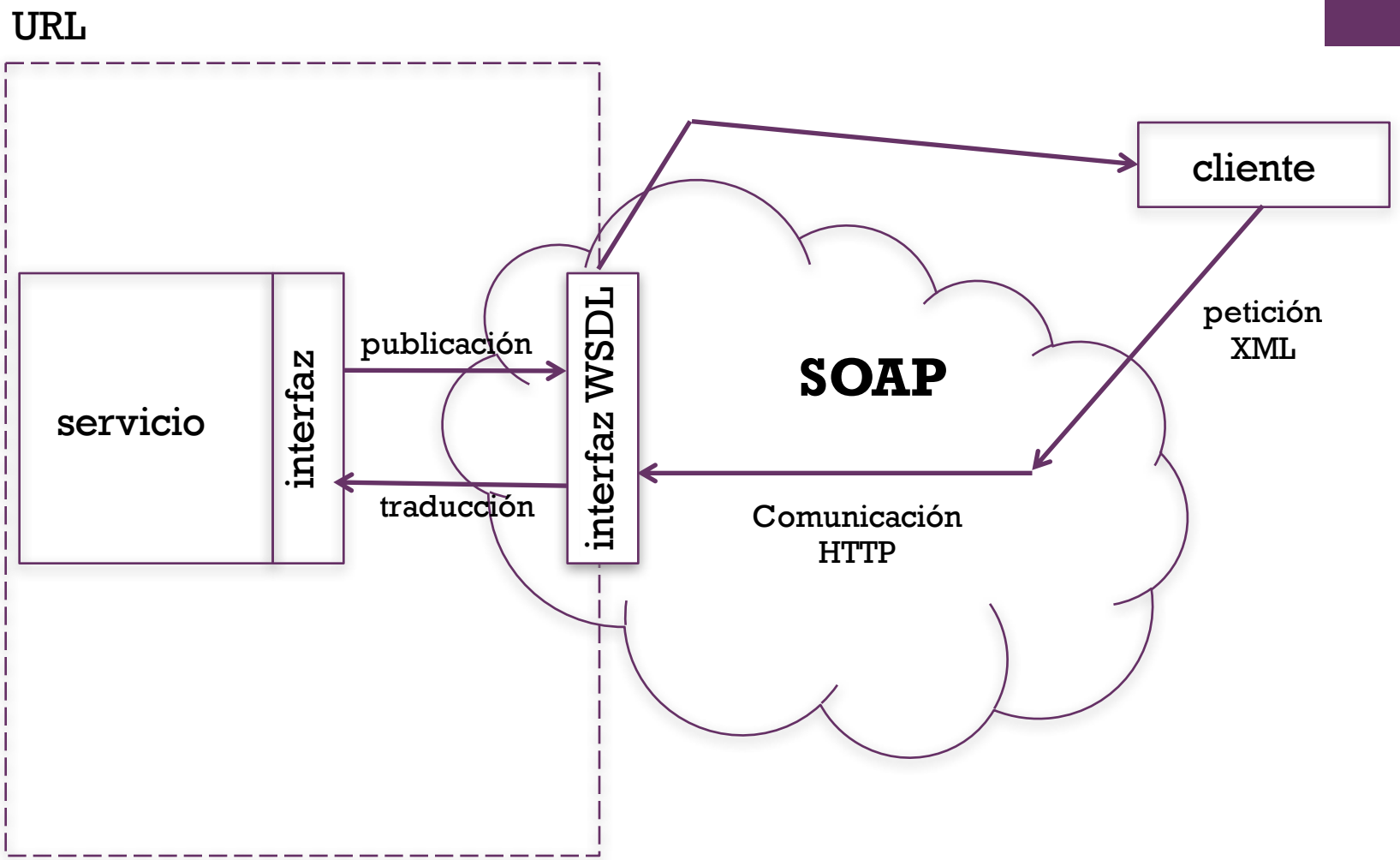


Colouris et al. 2011

- Las aplicaciones pueden ser otros servicios web
- Los servicios llevan asociadas descripciones en un lenguaje (IDL) neutral
- El servicio usa un protocolo para aceptar peticiones y ofrecer resultados en XML, basado en un direccionamiento mediante URIs y protocolos tipo HTTP

SOAP

Modo de funcionamiento



Servicios web

- Introducción
- IDL
- SOAP
- **REST**
- XML/JSON-RPC

REST

- REpresentational State Transfer (Roy Fielding, 2000)
- Arquitectura similar a la definida por SOAP
  - Basado en HTTP y URLs
  - Mensajes en XML (generalmente)
- **Utiliza la potencia de la URI** como WSDL
- **Enfocado a los datos** en vez de a las interfaces
  - Un servicio REST es *RESTful* si cumple que:
    - La URI del servicio es http://foo.com/resources
    - Los datos están en un formato estándar (XML, JSON)
    - Sólo hay operaciones HTTP: GET, PUT, DELETE, POST
    - API accesible a través de un navegador (hipertexto)

REST

Recursos y representaciones

- Un **recurso** es una fuente de información específica
  - Referenciada por un identificador global (URI)
- Una **representación** de un recurso es la forma en la que se intercambia dicho recurso
- Recurso:
  - Un círculo
- Representación:
  - Centro y radio en un .svg
  - Tres puntos de su circunferencia en un .csv

REST

Operaciones

	Colección (http://foo.com/resources)	**Elemento** (http://foo.com/resources/item17)
GET	**Lista** las URIs de los elementos de la colección	**Recupera** la representación del elemento
PUT	**Reemplaza** la colección entera por otra	**Reemplaza** el elemento (si no existe, falla)
POST	**Crea** un nuevo elemento en la colección, le asigna una URI automáticamente y devuelve esa URI	Trata el elemento como una colección, y **crea** un nuevo elemento en ella
DELETE	**Elimina** la colección entera	**Elimina** el elemento

REST

Modelo centrado en servicios (SOAP)

- Método: getUserDetails
  - argumento: ID del usuario
- getUserDetails(1235):

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-
encoding">
 <soap:body pb="http://www.acme.com/phonebook">
  <pb:GetUserDetails>
   <pb:UserID>12345</pb:UserID>
  </pb:GetUserDetails>
 </soap:Body>
</soap:Envelope>
```

Modelo centrado en datos (REST)

- Dato: detalles del usuario 1235

http://www.acme.com/phonebook/**UserDetails/12345**

- Dato: detalles del usuario de nombre John y apellido Doe

http://www.acme.com/phonebook/UserDetails**?firstName=John&lastName=Doe**

- Dato: detalles del usuario de nombre John y apellido Doe en formato JSON

http://www.acme.com/phonebook/UserDetails**.json**?firstName=John&lastName=Doe

REST

REST vs SOAP

	REST	SOAP
Alcance	Arquitectura	Protocolo
Estándar?	No	Sí
Mensajes	HTML o XML	XML
Tipado	No (HTML) o fuerte (XML)	Fuerte (XML)
Ancho de banda	Menor en HTML	Tags adicionales de XML
Testeo	Navegador	Herramientas tipo soapUI
Bibliotecas	No (URIs)	Sí
Petición usual	GET	POST
Cliente	Sencillo	Algo más complejo

http://geeknizer.com/rest-vs-soap-using-http-choosing-the-right-webservice-protocol/

REST

REST vs SOAP

http://nicolas-zozol.developpez.com

REST

REST no es perfecto

- REST no siempre es mejor decisión de diseño que SOAP
  - Las URIs tienen un tamaño máximo
    - No soportan argumentos muy largos
    - Para adjuntar datos (texto o binarios) necesitamos usar POST
      - SOAP es mejor al estar estandarizado
  - Las URIs son públicas
    - Problema si los argumentos son delicados
  - Tipado fuerte
    - REST lo soporta (vía XML) pero no tiene un estándar de tipos
- Generalmente, REST es una solución más sencilla y limpia
  - Suficiente para servicios web poco complejos

REST

Ejemplos*

- Twitter
  - REST es la API original de Twitter, y todavía hoy la más popular entre desarrolladores
  - Listado de recursos: https://dev.twitter.com/docs/api
- Google
  - Tuvo una REST API hasta 2010
    - https://developers.google.com/web-search/docs/
  - Ahora utiliza JSON/Atom Custom Search API
    - http://code.google.com/apis/customsearch/v1/overview.html

* Una buena cantidad de APIs, la mayoría REST, se pueden consultar http://programmableweb.com

REST

Filosofía

- La filosofía subyacente a REST es que una orden (verbo) sobre un recurso web (objeto) debería devolver una versión u otra dependiendo del usuario (sujeto)
- Por ejemplo, en el caso de GET
  - Si el usuario es una persona, devuelve un fichero legible
    - Típicamente un documento HTML como hasta ahora
  - Si el usuario es una máquina, devuelve un fichero en un lenguaje
    - Por ejemplo XML, JSON, etc.
- La web actual, dado un concepto (información), usa una sola representación (página web). Con varias representaciones, la web se está convirtiendo en una herramienta más completa.

Servicios web

- Introducción
- IDL
- SOAP
- REST
- XML/JSON-RPC

XML-RPC

- Creado en 1998 por Dave Winer
- RPC que usa
  - XML como sistema de codificación de las peticiones
  - HTTP como mecanismo de envío
- Evolucionó hasta convertirse en SOAP
  - Más sencillo, no requiere un WSDL
- Como estándar, no es muy evolucionado
  - Simplemente define tipos de datos básicos y peticiones en XML

XML-RPC

Tipos de datos

```
<int>42</int>
<double>-12.53</double>
<boolean>1</boolean>
<string>Hello world!</string>
<nil/>

<base64>eW91IGNhbid0IHJlYWQgdGhpcyE=</base64>

<array>
  <data>
    <value><i4>1404</i4></value>
    <value><string>Something</string></value>
    <value><i4>1</i4></value>
  </data>
</array>
```

XML-RPC

Invocación y respuesta

```
<?xml version="1.0"?>
<methodCall>
  <methodName>examples.getStateName</methodName>
  <params>
    <param>
        <value><int>40</int></value>
    </param>
  </params>
</methodCall>
```

```
<?xml version="1.0"?>
<methodResponse>
  <params>
    <param>
        <value><string>Dakota</string></value>
    </param>
  </params>
</methodResponse>
```

JSON-RPC

- Similar a XML-RPC, pero datos y mensajes se especifican en JSON, mucho menos verboso
  - JSON para definición de datos y mensajes
  - HTTP o sockets TCP/IP para envío
- Ejemplos

```
--> {"jsonrpc": "2.0", "method": "subtract", "params": [42, 23], "id": 1}
<-- {"jsonrpc": "2.0", "result": 19, "id": 1}
```

```
--> {"jsonrpc": "2.0", "method": "foobar", "id": 10}
<-- {"jsonrpc": "2.0", "error": {"code": -32601, "message": "Procedure not found."}, "id": 10}
```

JSON-RPC

- De alguna manera, JSON-RPC cierra el círculo:

Método	Año	WSDL	Recursos	Datos
XML-RPC	1998	No	No	XML
SOAP	1998	Sí	No	XML/WSDL
REST	2000	No	Sí	Cualquiera
JSON-RPC	2005	No	No	JSON

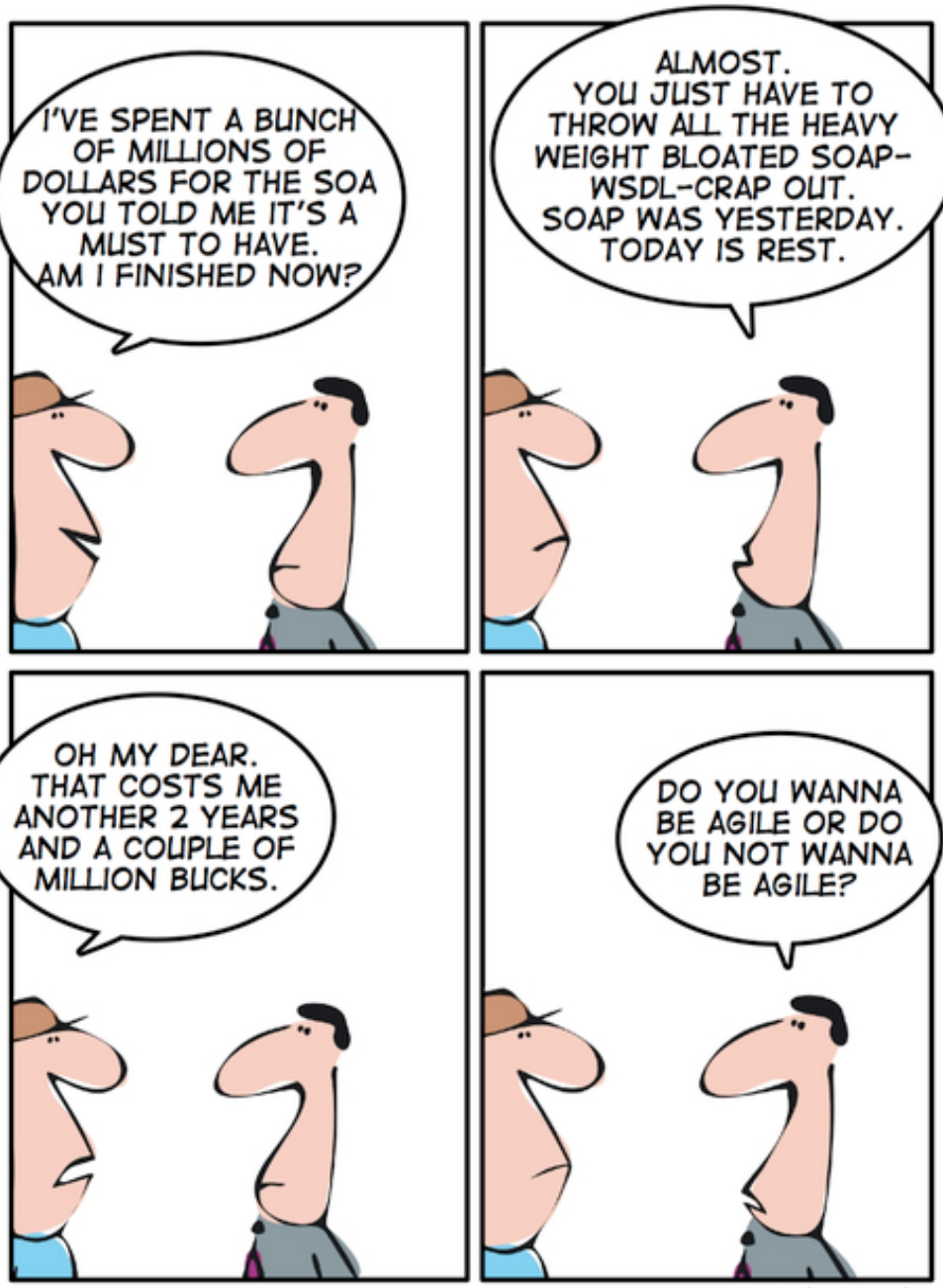
Resumen

- Los servicios web son un tipo de middleware que busca la **simplicidad** en base a las características de **Internet**
- Se centran en un paso de mensajes mediante **HTTP** y uso de datos en texto plano con estructura (**XML, JSON**)
- **SOAP** se centra en el uso de un XML propio para cada servicio, escrito bajo un estándar **WSDL**
- **REST** se centra en el uso de la propia estructura de **URIs** en HTTP para las peticiones, y en un formato abierto a definir en la respuesta (típicamente JSON)
- Un servicio es **RESTful** si cumple con un intercambio basado en **recursos**, entendidos como objetos de datos, que se modifican sólo mediante las cuatro acciones de HTTP (**GET, POST, PUT, DELETE**)
- Todo servicio web necesita en el fondo definir sus **mensajes** y **datos** compartidos, es responsabilidad del desarrollador elegir aquél que mejor se adapte a sus sistema

Servicios web

Bibliografía

- Colouris. Distributed Systems, 5th ed. (2011)
  - Ch 9: Web services
- Wikipedia (inglés)
  - Entradas "REST", "SOAP", "XML-RPC", "JSON-RPC"
- Learn REST: A tutorial
  - http://rest.elkstein.org/
- How I explained REST to my wife
  - http://tomayko.com/writings/rest-to-my-wife
- REST vs SOAP: the right web service
  - http://geeknizer.com/rest-vs-soap-using-http-choosing-the-right-webservice-protocol/



THE CONSULTANTS HANDBOOK PART 2:
MAKE SURE YOUR COSTUMER IS ALWAYS AGILE

http://geekandpoke.typepad.com/