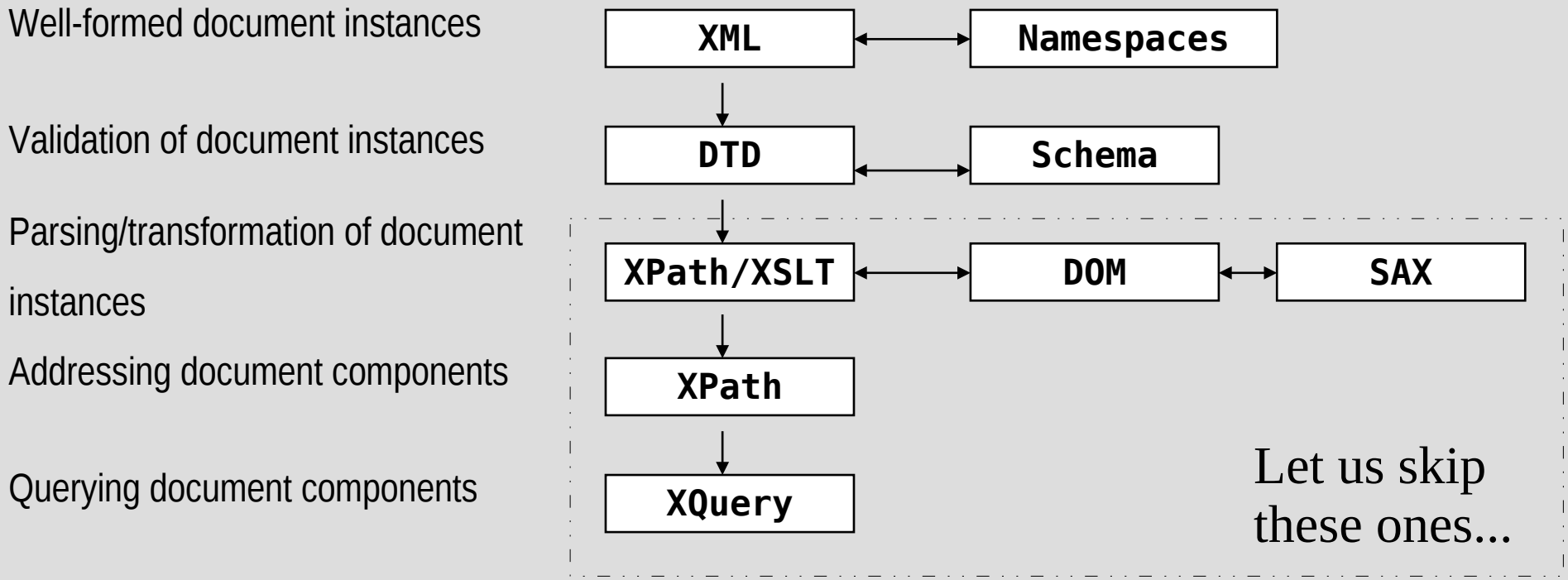
Service Oriented Computing: Web Service standards

Dr. Cristian Mateos Diaz
(http://users.exa.unicen.edu.ar/~cmateos/cos)
ISISTAN-UNICEN-CONICET

XML family of standards

- Domain-specific XML-based standards
  - e.g., MathML, DrawML, RSS, XHTML, SOAP/WSDL/WADL
- A number of general standards:



SOAP/WSDL/WADL: Enabling technologies

- XML
- XML Schemas (rather than DTDs)
- Namespaces:

```
<x xmlns:edi='http://ecommerce.org/schema'>
    <!-- the "edi" prefix is bound to http://ecommerce.org/schema for the
    "x" element and contents -->
</x>
```

which is equivalent to:

```
<root xmlns:edi='http://ecommerce.org/schema'>
    ...
    <edi:x> ... </edi:x>
    ...
</root>
```

Web Services

- Self-contained, self-describing, modular applications that can be published, located, and invoked across the Web
  - More properties: loosely coupled, reusable components, programmatically accessible over ubiquitous protocols, **interoperable**
  - Close resemblance with component-oriented software

- Types of Web Services
  - Simple content-provider implementation

    (stock quote, weather, geo-localization)
  - Complex process or world-altering

    (hotel and ticket booking, resource handling)
  - GET vs POST in Restful services

Components: A simple definition

- Component = Class (from OO) + structural conventions
- Real-world example: JavaBeans; OSGi
  - A JavaBean must contain a default constructor
  - A JavaBean must be serializable
  - JavaBean properties must be accessed via getters/setters

```
public class PersonBean implements java.io.Serializable {
   private String  name; private int age;
   public PersonBean() {}
   public void setName(String n) { this.name = n; }
   public void setAge(int a) { this.age = a; }
   public String getName() { return (this.name); }
   public int getAge() { return (this.age); }
}
```

- Other component models usually define other conventions, for example forbid data sharing between components

Components vs Web Services

- Software components are reusable

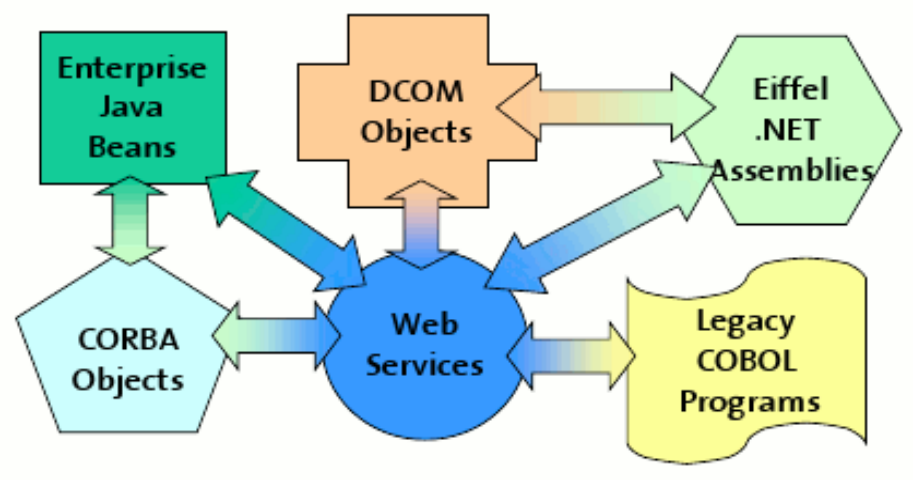
- To be used, a component must:
  - be packaged to be deployed as part of some larger application
  - fit with the existing framework used to develop the system
- Pre-FOOS: Components were sold

- Web services are reusable too

- To be used a Web Service must:
  - be published on the Web
  - Composed; no need to download

- Web Services can be sold too (e.g. Twitter) otherwise *You are the product*!

Many component frameworks for building distributed systems exist (J2E, DCOM, .NET, CORBA, etc.) but they are **not compatible** -->

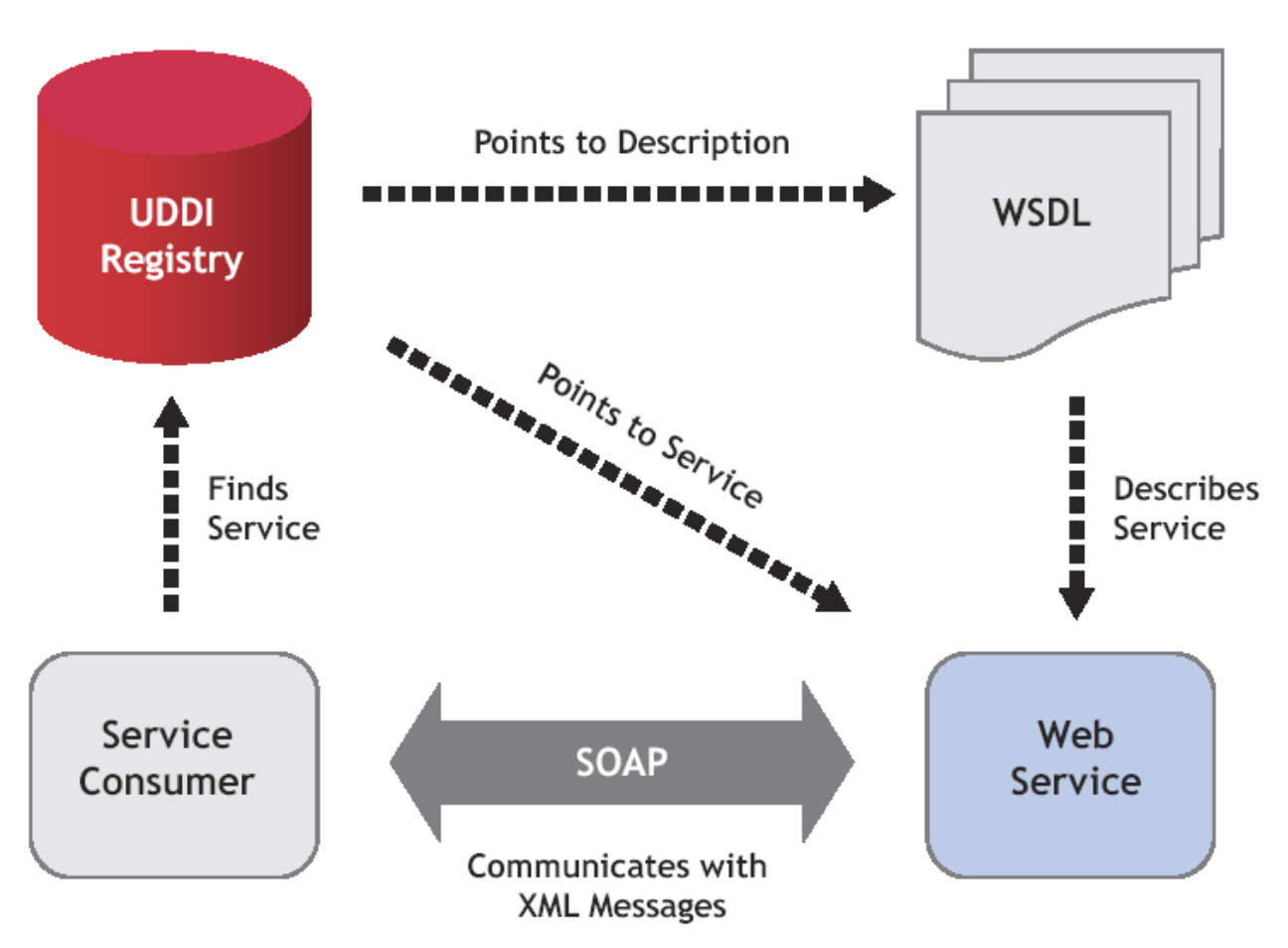


Web Services: Standards organizations

When reading about Web Services, you will surely encounter the words "standard", "specification" and "extension" (i.e., WS-)...*

	W3C	OASIS	WS-I
Established	1994	1993 as the SGML Open, 1998 as OASIS	2002
Approximate membership	400	600	200
Overall goal (as it relates to SOA)	To further the evolution of the Web, by providing fundamental standards that improve online business and information sharing.	To promote online trade and commerce via specialized Web services standards.	To foster standardized interoperability using Web services standards.
Prominent deliverables (related to SOA)	XML, XML Schema, XQuery, XML Encryption, XML Signature, XPath, XSLT, WSDL, SOAP, WS-CDL, WS-Addressing, Web Services Architecture	UDDI, ebXML, SAML, XACML, WS-BPEL, WS-Security	Basic Profile, Basic Security Profile

Web Services: Architecture



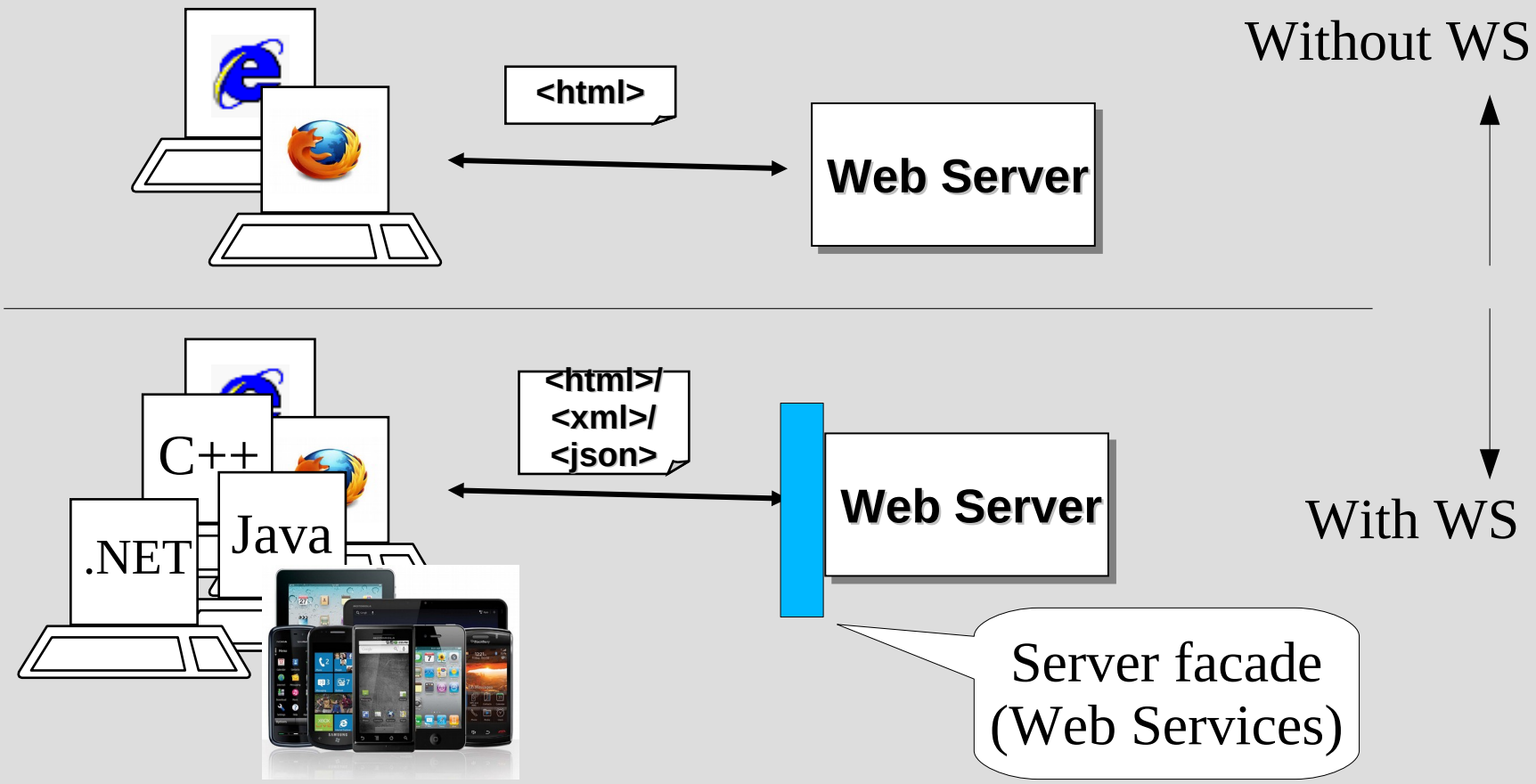
A de facto standard materialization:

- Find/Publish/ Unpublish (UDDI or Syntactic registries)

- Bind (SOAP+WSDL, JSON+WADL/Swagger)

Web Services: Service consumers

PCs/Browsers are not the only way to access Web information!



Web Services: Some links

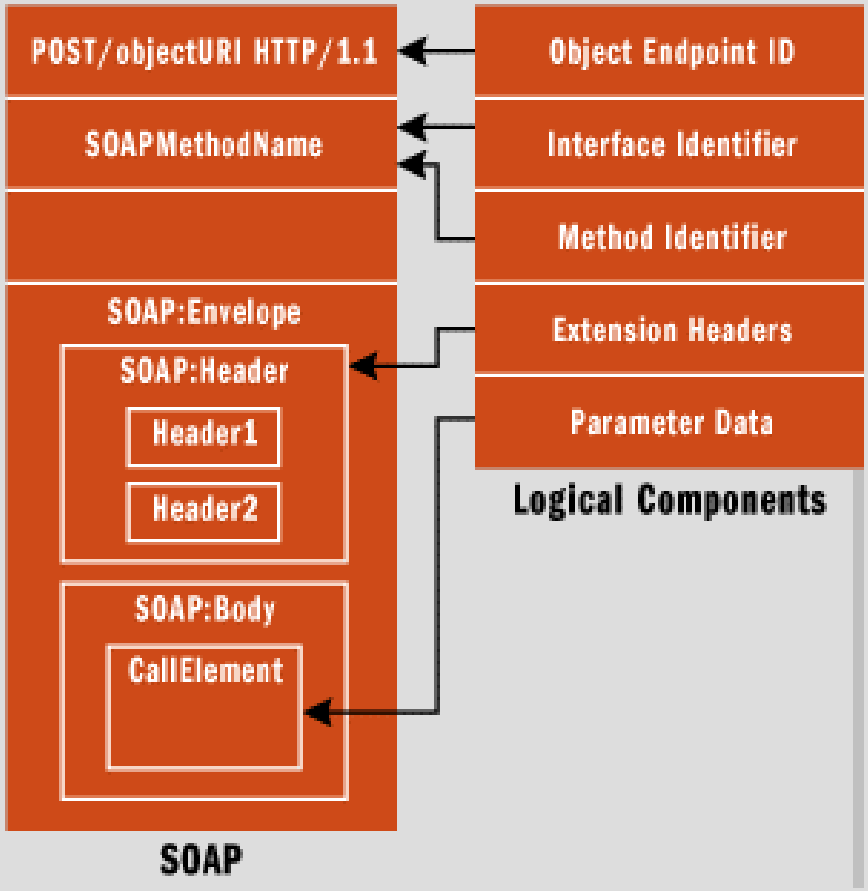
- **Specifications**
  - SOAP: *http://www.w3.org/TR/soap*
  - WSDL: *http://www.w3.org/TR/wsdl*
  - UDDI: *http://xml.uddi.org*
  - WADL: *http://www.w3.org/Submission/wadl/*
- **Java libraries**
  - Axis2: *http://ws.apache.org/axis2*
  - *UDDI4J*: *http://uddi4j.sourceforge.net*
  - jUDDI: *http://ws.apache.org/juddi*

Web Services: SOAP

- SOAP stands for “Simple Object Access Protocol”
- W3C Recommendation

SOAP is essentially an XML transport protocol:

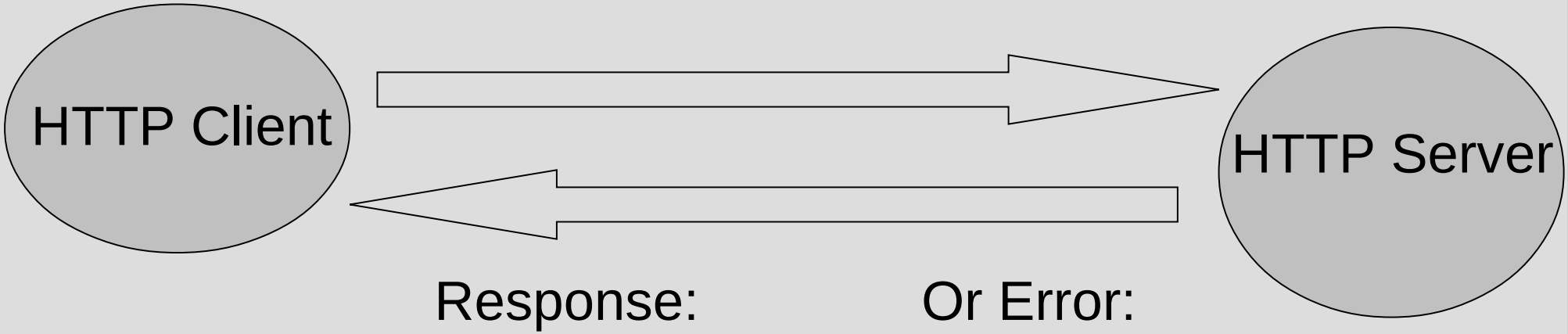
- There is a sender and a receiver
- Content is interchanged between these two



SOAP: Relationships with HTTP

Request:

```
POST /soap HTTP/1.1
Host: 216.128.29.26
Content-Type: text/plain
Content-Length: 200
```



Response:

```
200 OK
Content-Type: text/plain
Content-Length: 200
```

Or Error:

```
400 Bad Request
Content-Length: 0
```

SOAP: Relationships with HTTP (cont.)

```
POST /InStock HTTP/1.1
Host: www.stock.org
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn
```

```
<?xml version="1.0"?>
<soap:Envelope
 xmlns:soap="http://www.w3.org/2001/12/soap-envelope">
 <soap:Body xmlns:m="http://www.stock.org/stock">
        <m:GetStockPrice>
           <m:StockName>IBM</m:StockName>
        </m:GetStockPrice>
  </soap:Body>
</soap:Envelope>
```

SOAP: Relationships with HTTP (cont.)

```
HTTP/1.1 200 OK
Content-Type: application/soap; charset=utf-8
Content-Length: nnn
```

```
<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2001/12/soap-envelope">
    <soap:Body xmlns:m="http://www.stock.org/stock">
        <m:GetStockPriceResponse>
            <m:Price type="xsd:float">34.5</m:Price>
        </m:GetStockPriceResponse>
    </soap:Body>

</soap:Envelope>
```

SOAP faults

- Errors ocurred during message processing
  - One per SOAP message
  - Optional (non-mandatory)
  - Linked to 500 to 599 HTTP status code

- Contain:
  - **<faultCode>** (SOAP-ENV:Client and SOAP-ENV:Server
  - **<faultString>**
  - **<faultActor>**
  - **<detail>** (application-specific detailed information)

- Transformed to language-specific exception mechanisms (e.g. Axis2 SOAPFault)

SOAP faults: Relationships with HTTP

```
HTTP/1.1 500 OK
Content-Type: application/soap; charset=utf-8
Content-Length: nnn
```

```
<?xml version = '1.0' encoding = 'UTF-8'?>
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV = "http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi = "http://www.w3.org/1999/XMLSchema-instance"
  xmlns:xsd = "http://www.w3.org/1999/XMLSchema">
  <SOAP-ENV:Body>
    <SOAP-ENV:Fault>
      <faultcode xsi:type = "xsd:string">SOAP-ENV:Client</faultcode>
      <faultstring xsi:type = "xsd:string">
        Failed to locate method (ValidateCreditCard) in class (examplesCreditCard) at
          /usr/local/ActivePerl-5.6/lib/site_perl/5.6.0/SOAP/Lite.pm line 1555.
      </faultstring>
    </SOAP-ENV:Fault>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

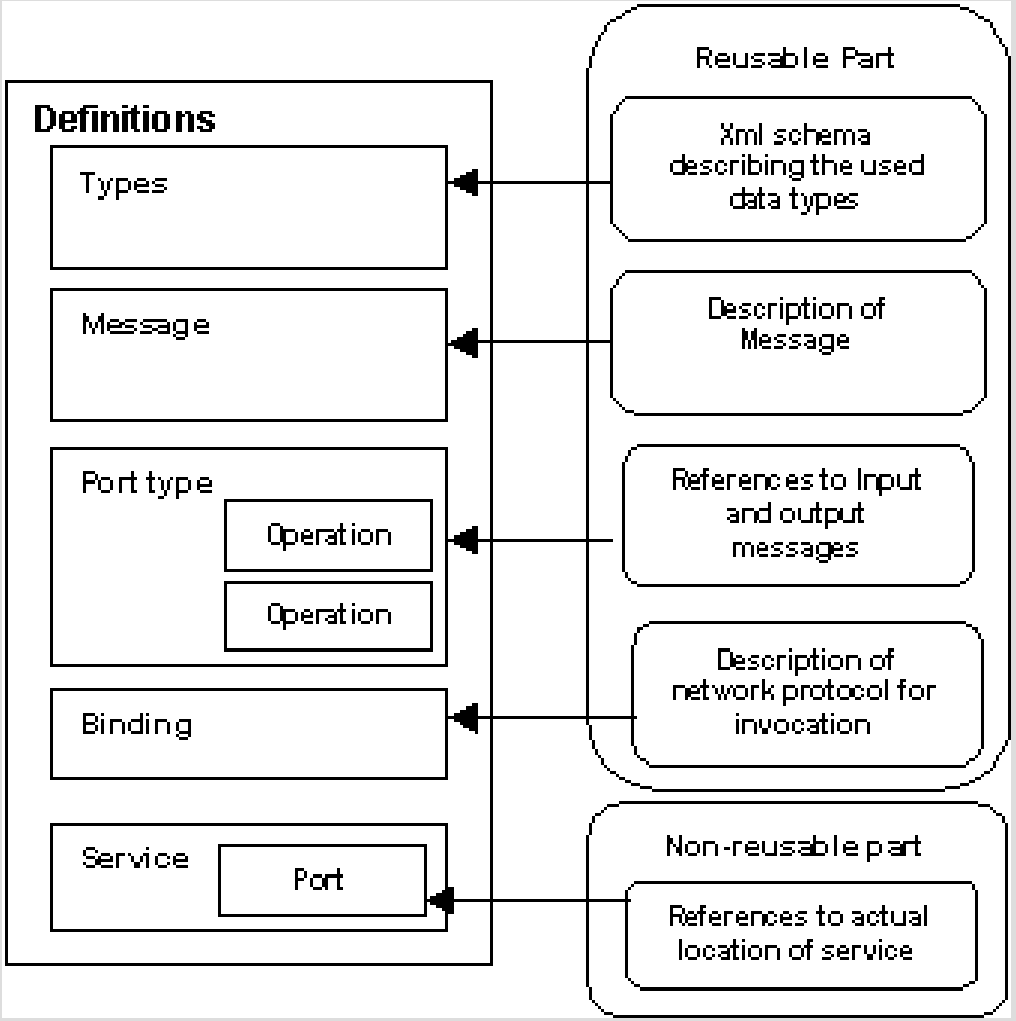
“xsd” versus “xsi” preffix: https://tinyurl.com/ybbj39ca

Web Services: WSDL

- WSDL stands for “Web Service Description Language”
- W3C standardization effort

A WSDL definition is an XML document describing the interface of a SOAP Web Service:

- Interface: (operations; input/output)
- Access (protocol binding)
- Endpoint (location of service)

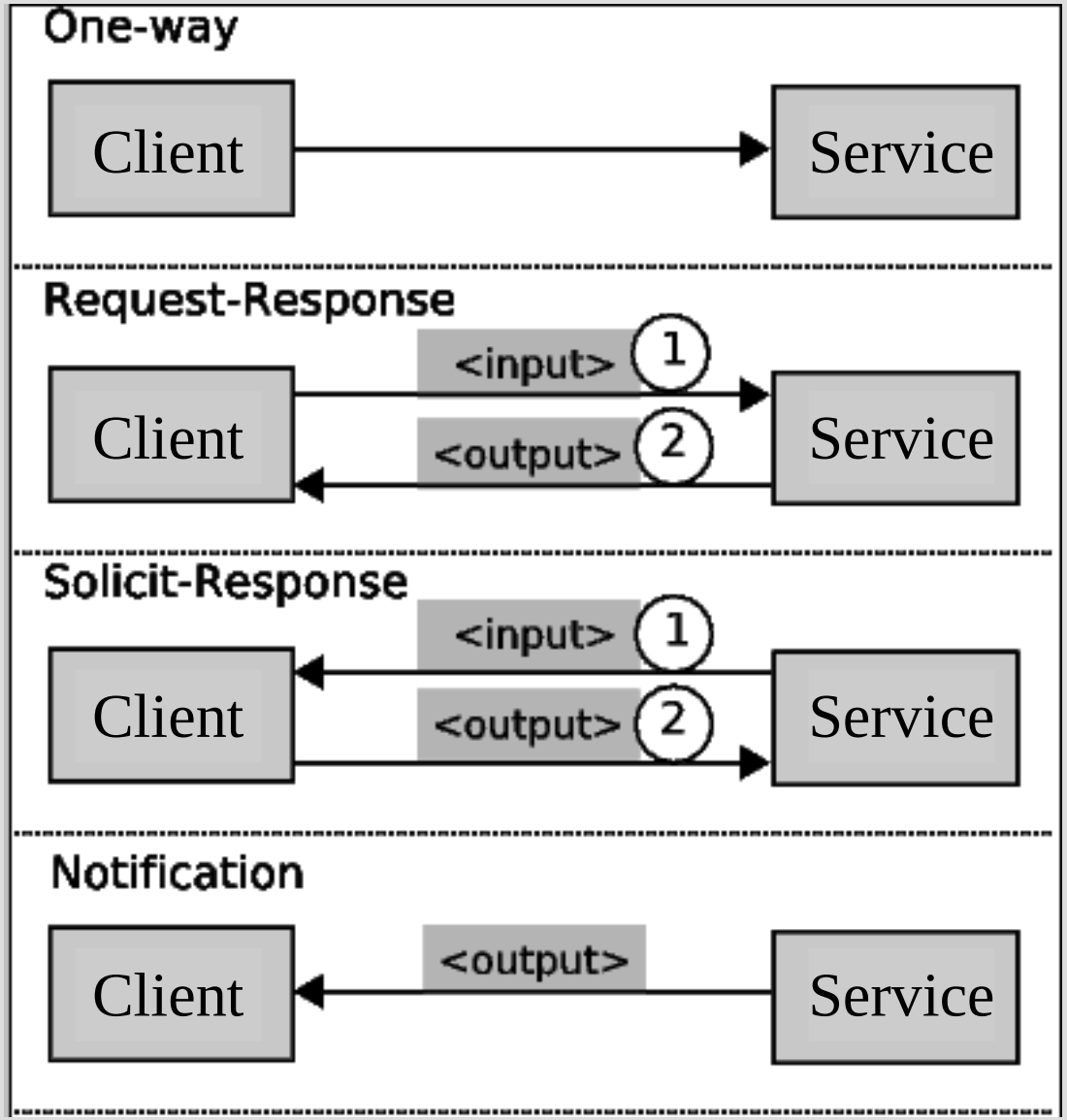


WSDL vs Java

WSDL		Java
WSDL document	◄——►	*Java interface (explicit)/ Java class (implicit)*
Port type	◄——►	*Method signatures (behavior)*
Messages	◄——►	*Individual method parameters + return types*
Data types	◄——►	*Argument types*
Endpoint & binding?		

WSDL 1.0: Message exchange patterns

Similar to calling methods in Java



Less used interaction patterns...

WSDL 2.0: Message exchange patterns

- In-Only
- In-Out
- Out-In
- Out-Only
- *Robust In-Only*
- *In-Optional-Out*
- *Robust Out-Only*
- *Out-Optional-In*

Same as previous slide

WSDL: One-way operation example

```
...
<message name="updateStock">
   <part name="quote" type="xsd:string"/>
   <part name="price" type="xsd:float"/>
</message>

<portType name="dictionary">
   <operation name="updateStock">
      <input name="newStockPrice" message="updateStock"/>
   </operation>
</portType>
...
```

WSDL: Request-response operation example

```
...
<message name="getStockQuoteRequest">
    <part name="quote" type="xsd:string"/>
</message>

<message name="getStockQuoteResponse">
    <part name="price" type="xsd:float"/>
</message>

<portType name="StockQuotePortType">
    <operation name="getStockQuote">
            <input message="getStockQuoteRequest"/>
            <output message="getStockQuoteResponse"/>
    </operation>
</portType>
...
```

WSDL: Summary

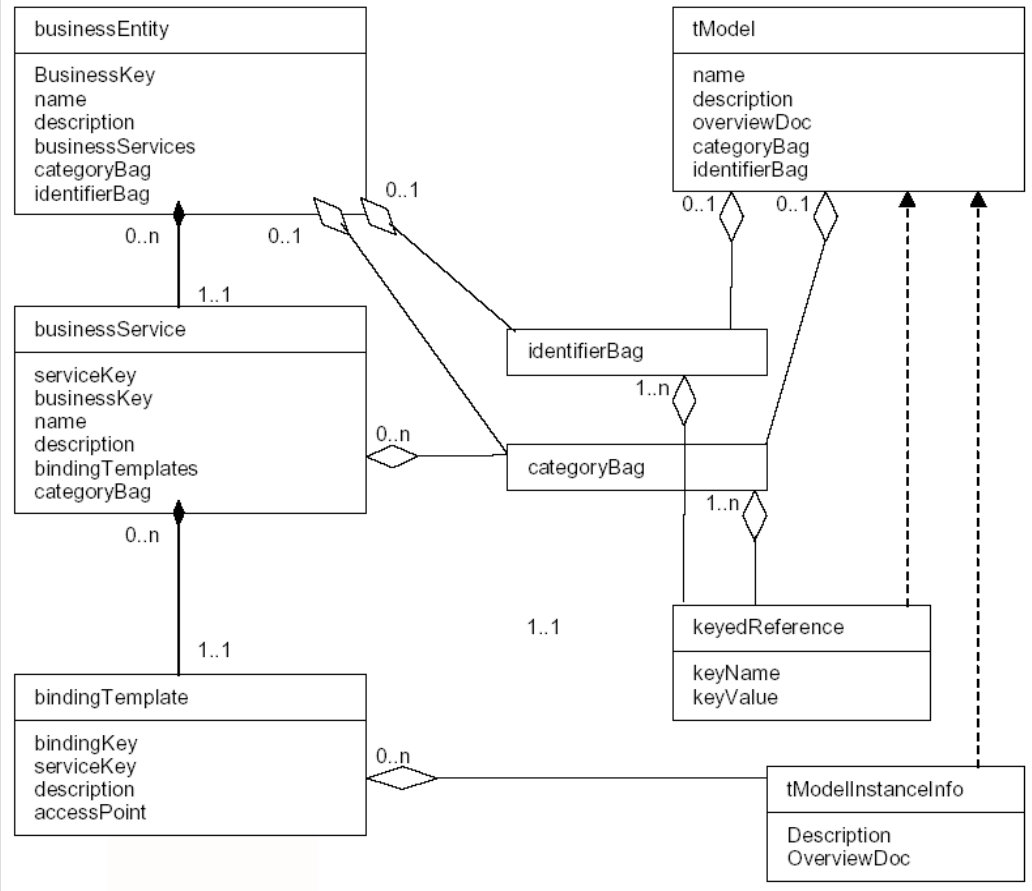
- Language and platform independent
- Multiple operation patterns (MEPs)
- Operations can have multiple inputs and outputs
- Support for multiple bindings (SOAP, RMI, CORBA, Rest)
- More than one binding for the same port type

Web Services: UDDI

- UDDI stands for “Universal Description, Discovery, and Integration Protocol”
- OASIS standardization effort

A UDDI node represents a registry for Web Services:

- Business
- Service information (taxonomies)
- Technical details



UDDI: Inspection

- Browser-based (e.g., Eclipse WTP)
- Programmatic (e.g., using uddi4j):

```
UDDIProxy proxy = new UDDIProxy();
proxy.setInquiryURL(
    "http://www-3.ibm.com/services/uddi/ testregistry/inquiryapi");
proxy.setPublishURL(
    "https://www-3.ibm.com/services/uddi/ testregistry/protect/publishapi");

BusinessList bl = proxy.find_business("Business", null, 0);
Vector businessInfoVector = bl.getBusinessInfos().getBusinessInfoVector();
for (int i = 0; i < businessInfoVector.size(); i++) {
    BusinessInfo businessInfo = (BusinessInfo)businessInfoVector.elementAt(i);
    System.out.println(businessInfo.getNameString());
}
```

Web Services: Search engines

- UDDI is more like a structured search engine in the sense that search criteria are prescribed
- Alternatively, Web Services search engines provide a Google-like interface for looking for services
  - Most of them rely on text processing techniques
  - Performance heavily depends on contract quality
- Some examples:
  - Woogle: http://db.cs.washington.edu/webService
  - WSQBE: http://dx.doi.org/10.1016/j.scico.2008.02.002
  - Swoogle: http://swoogle.umbc.edu
  - WSCE (syntactic search on top of UDDI): http://www2007.org/poster968.php
  - ProgrammableWeb.com, Mashape.com

Search engines: Mashape.com

Mashape Acquires Gelato.io to provide Dev Portals for everyone. Read more about the announcement here

Marketplace | Search APIs | Explore APIs | Docs | Features | Add Your API | Sign Up Free | Login

CATEGORIES

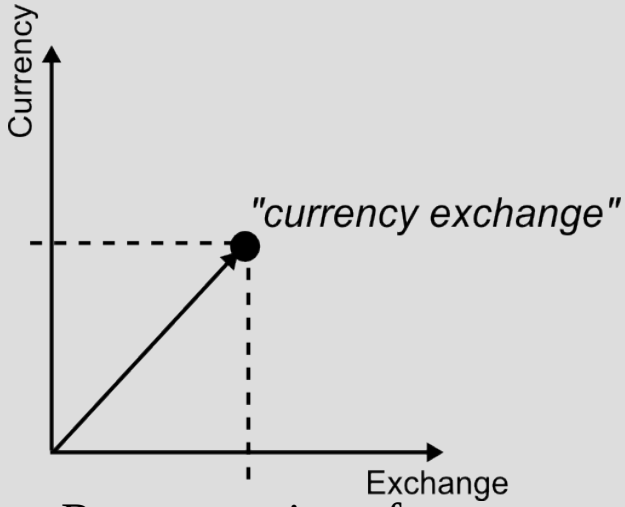
- Tools
- Education
- Devices
- Finance
- Advertising
- Commerce
- Other
- Location
- Business
- Social
- Communication
- Entertainment
- Media
- Medical
- Sports
- Reward

HIDE FILTERS | Sort By: Popular | Price Range: All

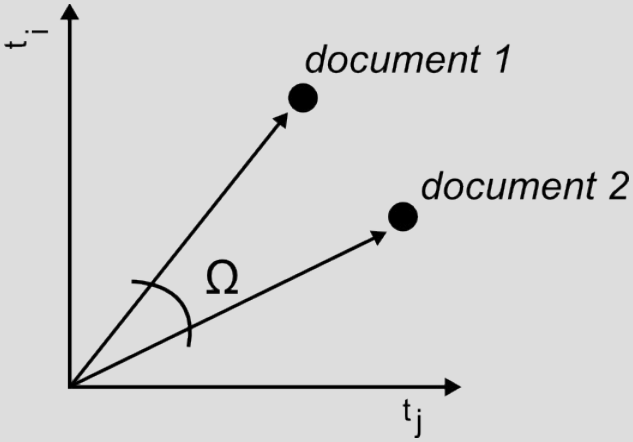
Weather By IP *by onesoft* — The Weather By IP API allows you to find weather information such as Weather...	FREEMIUM	7 developers	5 followers	100% uptime	
Weather *by Weather2020* — Weather2020 is the most popular long range weather forecast API available! Used by...	FREEMIUM	10 developers	11 followers	100% uptime	
WeatherSpot *by pl12133* — WeatherSpot https://github.com/hak8or/WeatherSpot/	FREE	29 developers	28 followers	100% uptime	
Weather *by ericpeng* — Get weather forecast for a specific location (WOEID).	FREEMIUM	4 developers	5 followers	100% uptime	
Weather *by fyhao* — Display Weather forecast data by latitude and longitude. Get raw weather data OR...	FREE	1053 developers	625 followers	100% uptime	

Syntactic search engines: Basics

- Syntactic service registries represent a Web Service as a bag of words obtained from a WSDL document.
- Queries are transformed to a bag of words
- Web Services are ranked by their similarity with a query
- Similarity is measured by the number of shared words between the WSDL document and the query



Representation of
“currency exchange”



Ω represents the difference between document 1 and 2

Syntactic search engines: Basics (cont.)

- How to obtain vectors? Given a set of words:

  - Stop-words removal (e.g. “message”)

  - Porter’s stemming (e.g. “provider/provide” → “provid”)

  - TF-IDF(t): TF(t) * IDF (t)

    - TF(t) = (Number of times term *t* appears in a description) / (Total number of terms in the description)
    - IDF(t) = log_e(Total number of descriptions / Number of descriptions with term *t* in it).

Query-service (or service-service) similarity is determined using the cosine between the two *n*-dimensional vectors, where *n* depends on the **vocabulary size**

Syntactic search engines: TF-IDF example

- A service description containing 100 words
- The word “weather” appears 3 times
- TF(weather) is then (3 / 100) = 0.03
- We have 10 million services and the word “weather” appears in one thousand of these
- IDF(weather) is log(10,000,000 / 1,000) = 4
- TF-IDF(weather) = 0.03 * 4 = 0.12

Syntactic search engines: Basics (cont.)

```
<message name="ChangeVolumeUnitHttpPostIn">
  <part name="VolumeValue" type="s:string" />
  <part name="fromVolumeUnit" type="s:string" />
  <part name="toVolumeUnit" type="s:string" />
</message>
<message name="ChangeVolumeUnitHttpPostOut">
  <part name="Result" element="s0:double" />
</message>
<portType name="VolumeUnitHttpPost">
  <operation name="ChangeVolumeUnit">
    <input message="s0:ChangeVolumeUnitHttpPostIn" />
    <output message="s0:ChangeVolumeUnitHttpPostOut" />
  </operation>
</portType>
```

Stop-word removal + Porter's stemmer
+ TF-IDF

<volum,0.811182256933513 2>, <unit,0.5242440213277584>,
<chang,0.258312567411114>, <valu,0.01996212802225258>,
<result,0.005261473239922817>

Syntactic search engines: Basics (cont.)

How to evaluate Web Service registries performance?

- Recall-at-*n*: Computes the proportion of retrieved relevant documents (RetRel) within a result list of size=*n*, where R represents all relevant documents in the evaluation-set → RetRel_n/R

- Example: 15 relevant documents, 3 retrieved with n=10 → Recall-at-10=3/15=20%

- Precision-at-*n*: Computes precision at different cut-off points of the result list (RetRel_n/n)

- Example: 5 relevant documents in the first 5 positions with n=10 → Precision-at-5=100%, Precision-at-10=50%

Syntactic search engines: Basics (cont.)

- F1-measure: 2*[(Recall-at-*n* * Precision-at-*n*] /
[(Recall-at-*n* + Precision-at-*n*]

- nDCG (per query): DCGp/IDCGp

$$\mathrm{DCG_p} = \sum_{i=1}^{p} \frac{rel_i}{\log_2(i+1)} = rel_1 + \sum_{i=2}^{p} \frac{rel_i}{\log_2(i+1)}$$

$$\mathrm{IDCG_p} = \sum_{i=1}^{|REL|} \frac{2^{rel_i} - 1}{\log_2(i+1)}$$

Syntactic search engines: nDCG example

- A registry returns for a given query 6 descriptions D1, D2, …, D6 with relevance scores 3, 2, 3, 0, 1, 2

$$\mathrm{DCG}_6 = \sum_{i=1}^{6} \frac{rel_i}{\log_2(i+1)} = 3 + 1.262 + 1.5 + 0 + 0.387 + 0.712 = 6.861$$

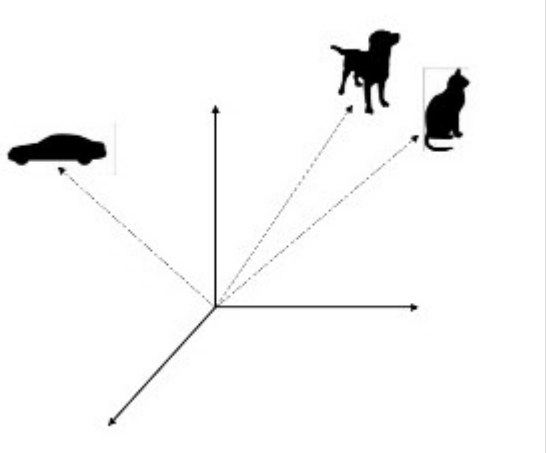
- Finally, ideal DCG (IDCG6) is computed assuming as if the order was 3, 3, 2, 2, 1, 0

- nDCG is thus between 0 and 1

Syntactic search engines: Basics (cont.)

Engine performance is conditioned by the **vocabulary problem** (more on this later)

- Ambiguous acronyms
- Synonyms (“tv” versus “television”)
- Polysemy (words having different meanings)
- Quasi-synonyms (“disease” and “disorder”)



Questions?