

Intelligente Systeme im World Wide Web

Web Services - An Introduction

Folien zur Vorlesung im Sommersemester 2006

Anupriya Ankolekar

Institut für angewandte Informatik und Formale
Beschreibungsverfahren (AIFB)

Universität Karlsruhe (TH)

Outline

- Web Services
 - What is a Web Service?
 - Web Service Standards
 - SOAP
 - WSDL
 - BPEL
 - UDDI
 - Why are Web Services interesting?
- Next lecture
 - Semantic Web Services

Prominent Web Services



With Google SOAP Search API, your computer can do the searching for you.

info Web Services

- Amazon Web services
 - <http://aws.amazon.com>
- Google Search API
 - <http://www.google.com/apis/>
- T-info Web Services offered by Deutsche Telekom
 - <http://services.t-info.de/soap/index.jsp>

More possible Web Services ...

- A Web service by a news network (such as CNN) could potentially expose a function that returns a list of headlines, for a given category.
- A calendar management system Web service could allow websites to check availability for a particular user for a meeting.
- An astronomy Web service might provide a function allowing users to call `GetStarInfo("star name")` to get the weight, distance and composition of a given star.

So, what is a Web Service?

A web service is “one or more elements of functionality that are accessible through the use of Internet standards such as XML and HTTP”

- Fundamentally, a Web service represents a business function or business process that is available over the Web as a service
- Software components (application logic) accessible via standard Web protocols
 - “Programming the Web”
 - **Better: “remote procedure calls over the Web”**
 - Web sites with no user interface
- Available to any client that speaks the necessary Web protocols (XML, SOAP)
 - Platform independent components
- Enable highly distributed systems

Exposing business functionality

- Identify and define valuable business processes
- Define loosely coupled, service-based interfaces to those processes
- Describe those interfaces in a Web-based, industry-standard format
- Publish the service interface so that it can be discovered and used by potential customers and partners over the Web
- Accept requests and send replies using standard Web protocols
- Bridge between those external requests (using Web protocols) and the implementation of the Web services using existing or new business functionality (which may use standard or propriety enterprise protocols)

What are the benefits of Web Services?

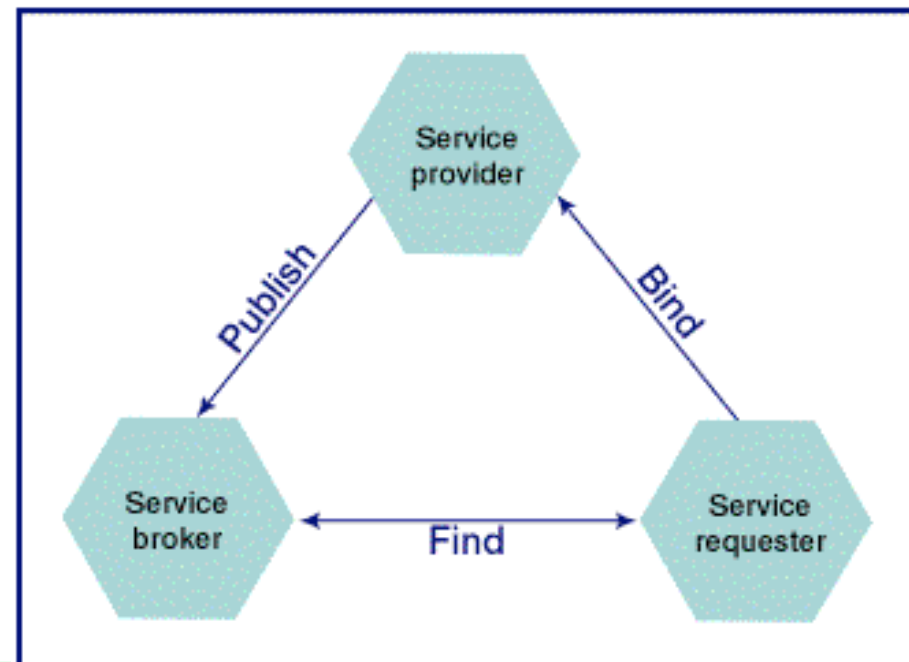
- Just-in-time integration of applications
- Reduce complexity by encapsulation
- Enable interoperability of legacy applications
- Minimize requirements for shared understanding

Web services architecture overview:

<http://www-128.ibm.com/developerworks/webservices/library/w-ovr/>

How can I use a Web Service?

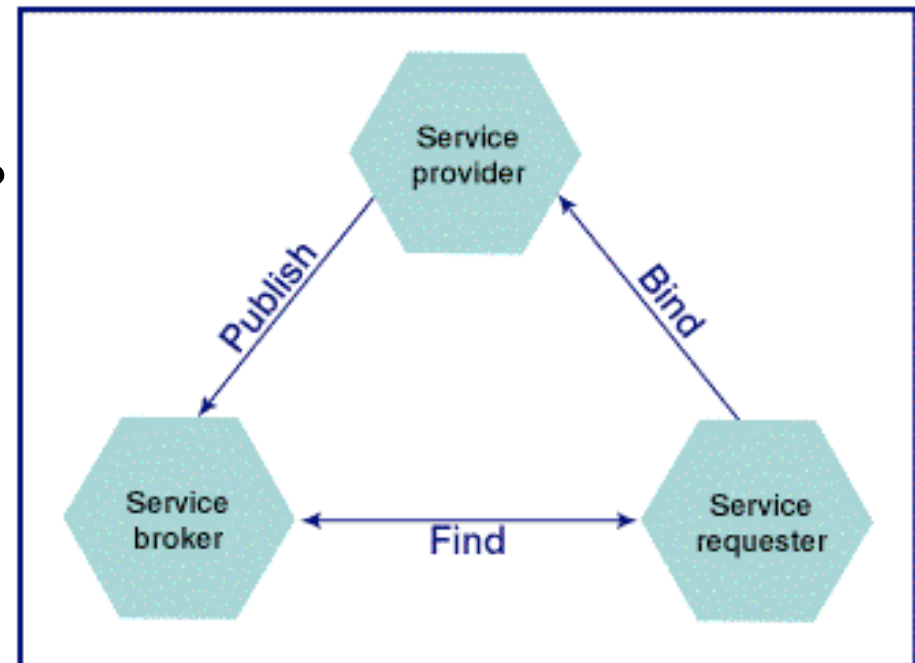
- Clients need answers to following questions
 - What services are available?
 - How do I communicate with this particular service?
 - Let's talk (give me some data)



Steps in using Web services

A web service must be

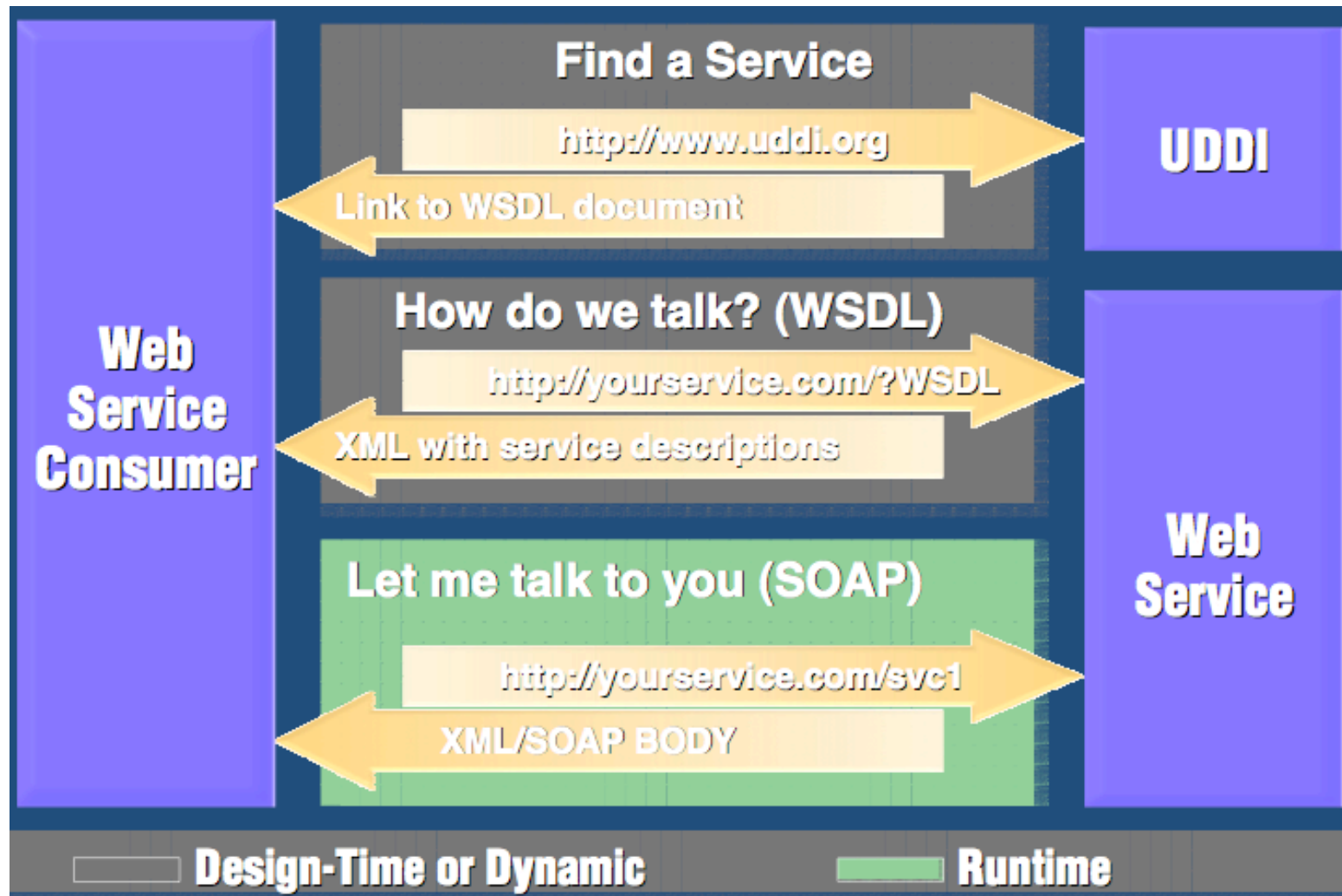
- created, and its interfaces and invocation methods defined
- published to one or more intranet or Internet repositories for potential users to locate
- found to be invoked by potential users
- invoked
- unpublished
 - when it is no longer available?



Web Standards

- UDDI (Universal Description, Discovery and Integration)
 - Yellow pages for services, registry, directory
- WSDL (Web Service Description Language)
 - Document describing the message exchange contract
- SOAP (Simple Object Access Protocol)
 - XML-based messaging protocol

Web Services Standards in Play



SOAP

- A transport-neutral protocol for XML data interchange (but focusing on HTTP)
- Processing model (envelopes, intermediaries, ...)
- Data representation (encoding) and SOAP RPC
- Transport Protocol Bindings (e.g. HTTP)

- Foundation of WS-*

SOAP Message

```
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
  xmlns:w="http://www.widget.inc/shop"
  xmlns:n="http://notaries.example.org">
  <env:Header>
    <w:ticket>54B42CF401A</w:ticket>
    <n:token>
      <n:value>32158546</n:value>
      <n:issuer>"http://notarypublic.example.com"</n:issuer>
    </n:token>
  </env:Header>
  <env:Body>
    <w:buy>
      <w:product>light gadget</w:product>
      <w:amount>430</w:amount>
    </w:buy>
  </env:Body>
</env:Envelope>
```

Why is SOAP not enough?

- SOAP describes the structure of the message, not the content
- How do you pass a person's name?

```
<student>
```

```
  <firstName>Jane</firstName>
```

```
  <familyName>Doe</familyName>
```

```
  ...
```

```
</student>
```

- Or as

```
<student>
```

```
  <name>Doe, Jane</name>
```

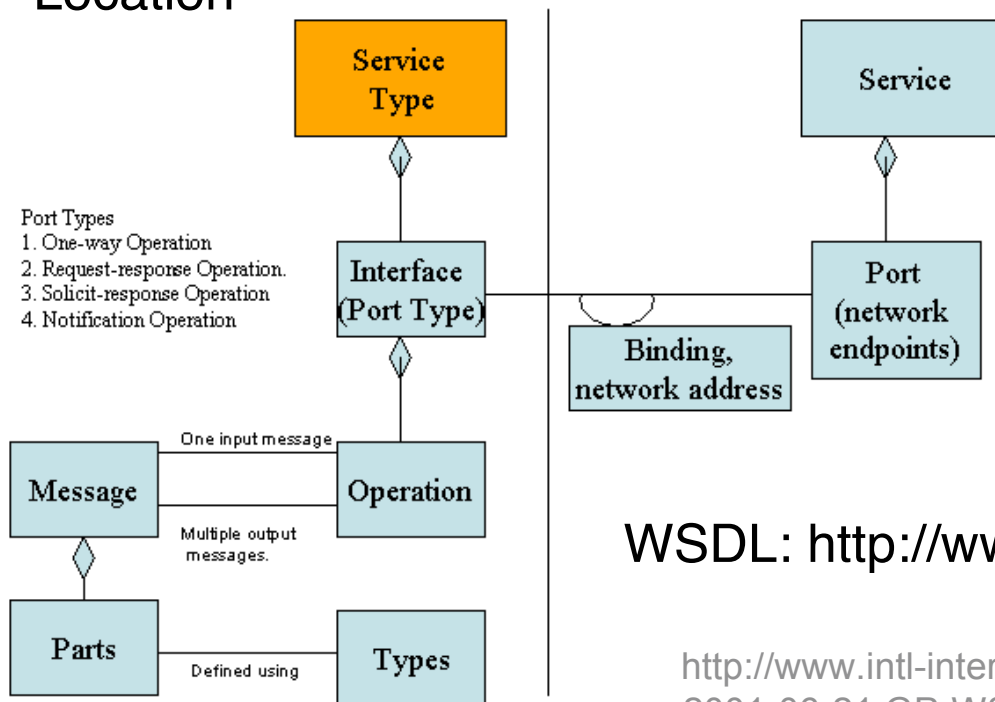
```
  ...
```

```
</student>
```

WSDL

Descriptions of interfaces of Web Services ([How do we talk?](#))

- Message types
- Operations
- Encodings and communication protocols
- Location



- Necessary information for writing clients
- Automatic generation of stubs and skeletons

WSDL: <http://www.w3.org/2002/ws/desc/>

http://www.intl-interfaces.com/servicemodel/2001-06-21.GP-WSDL_object_diagram.png

WSDL Document Structure

```
<description xmlns=http://www.w3.org/2004/08/wsdl targetNamespace="..." ...>
  <types>
    <!--XML Schema description of types being used in messages-->
    ...
  </types>
  <interface name="...">
    <!--list of operations and their input and output-->
    ...
  </interface>
  <binding name="..." interface="..." type="...">
    <!--message encodings and communication protocols-->
    ...
  </binding>
  <service name="..." interface="...">
    <!--combination of an interface, a binding, and a service location-->
    ...
  </service>
</description>
```

Example: Google Search API WSDL

```

<?xml version="1.0"?>

<!-- WSDL description of the Google Web APIs.
The Google Web APIs are in beta release. All interfaces are subject to
change as we refine and extend our APIs. Please see the terms of use
for more information. -->

<!-- Revision 2002-08-16 -->

<definitions name="GoogleSearch"
  targetNamespace="urn:GoogleSearch"
  xmlns:typens="urn:GoogleSearch"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
  xmlns="http://schemas.xmlsoap.org/wsdl/">

  <!-- Types for search - result elements, directory categories -->

  <types>
    <xsd:schema xmlns="http://www.w3.org/2001/XMLSchema"
      targetNamespace="urn:GoogleSearch">

      <xsd:complexType name="GoogleSearchResult">
        <xsd:all>
          <xsd:element name="documentFiltering" type="xsd:boolean"/>
          <xsd:element name="searchComments" type="xsd:string"/>
          <xsd:element name="estimatedTotalResultsCount" type="xsd:int"/>
          <xsd:element name="estimateIsExact" type="xsd:boolean"/>
          <xsd:element name="resultElements" type="typens:ResultElementArray"/>
          <xsd:element name="searchQuery" type="xsd:string"/>
          <xsd:element name="startIndex" type="xsd:int"/>
          <xsd:element name="endIndex" type="xsd:int"/>
          <xsd:element name="searchTips" type="xsd:string"/>
          <xsd:element name="directoryCategories" type="typens:DirectoryCategoryArray"/>
          <xsd:element name="searchTime" type="xsd:double"/>
        </xsd:all>
      </xsd:complexType>
    
```

Example: Google Search API WSDL

```

<!-- Messages for Google Web APIs - cached page, search, spelling. -->

<message name="doGetCachedPage">
  <part name="key" type="xsd:string"/>
  <part name="url" type="xsd:string"/>
</message>

<message name="doGetCachedPageResponse">
  <part name="return" type="xsd:base64Binary"/>
</message>

<message name="doSpellingSuggestion">
  <part name="key" type="xsd:string"/>
  <part name="phrase" type="xsd:string"/>
</message>

<message name="doSpellingSuggestionResponse">
  <part name="return" type="xsd:string"/>
</message>

<!-- note, ie and oe are ignored by server; all traffic is UTF-8. -->

<message name="doGoogleSearch">
  <part name="key" type="xsd:string"/>
  <part name="q" type="xsd:string"/>
  <part name="start" type="xsd:int"/>
  <part name="maxResults" type="xsd:int"/>
  <part name="filter" type="xsd:boolean"/>
  <part name="restrict" type="xsd:string"/>
  <part name="safeSearch" type="xsd:boolean"/>
  <part name="lr" type="xsd:string"/>
  <part name="ie" type="xsd:string"/>
  <part name="oe" type="xsd:string"/>
</message>

<message name="doGoogleSearchResponse">
  <part name="return" type="typens:GoogleSearchResult"/>
</message>

<!-- Port for Google Web APIs, "GoogleSearch" -->

<portType name="GoogleSearchPort">

  <operation name="doGetCachedPage">
    <input message="typens:doGetCachedPage"/>
    <output message="typens:doGetCachedPageResponse"/>
  </operation>

  <operation name="doSpellingSuggestion">
    <input message="typens:doSpellingSuggestion"/>
    <output message="typens:doSpellingSuggestionResponse"/>
  </operation>

```

Example: Google Search API WSDL

```

    <operation name="doGoogleSearch">
      <input message="typens:doGoogleSearch"/>
      <output message="typens:doGoogleSearchResponse"/>
    </operation>
  </portType>

  <!-- Binding for Google Web APIs - RPC, SOAP over HTTP -->
  <binding name="GoogleSearchBinding" type="typens:GoogleSearchPort">
    <soap:binding style="rpc"
      transport="http://schemas.xmlsoap.org/soap/http"/>

    <operation name="doGetCachedPage">
      <soap:operation soapAction="urn:GoogleSearchAction"/>
      <input>
        <soap:body use="encoded"
          namespace="urn:GoogleSearch"
          encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
      </input>
      <output>
        <soap:body use="encoded"
          namespace="urn:GoogleSearch"
          encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
      </output>
    </operation>

    <operation name="doSpellingSuggestion">
      <soap:operation soapAction="urn:GoogleSearchAction"/>
      <input>
        <soap:body use="encoded"
          namespace="urn:GoogleSearch"
          encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
      </input>
      <output>
        <soap:body use="encoded"
          namespace="urn:GoogleSearch"
          encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
      </output>
    </operation>

  <!-- Endpoint for Google Web APIs -->
  <service name="GoogleSearchService">
    <port name="GoogleSearchPort" binding="typens:GoogleSearchBinding">
      <soap:address location="http://api.google.com/search/beta2"/>
    </port>
  </service>
</definitions>

```

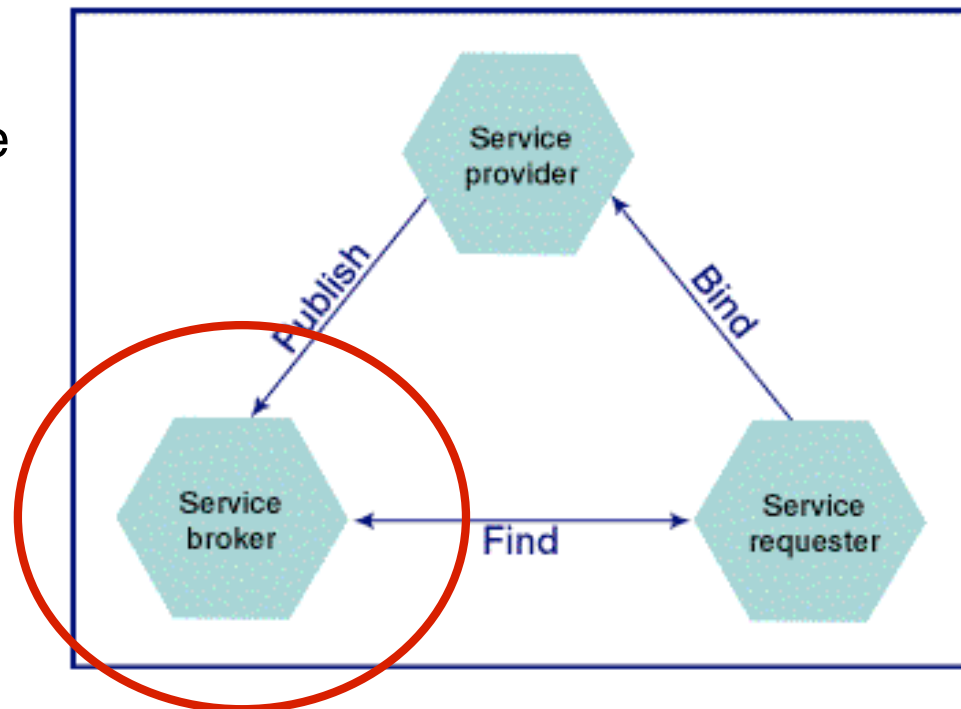
Example: Google Search API WSDL

```
<!-- Endpoint for Google Web APIs -->  
<service name="GoogleSearchService">  
  <port name="GoogleSearchPort" binding="typens:GoogleSearchBinding">  
    <soap:address location="http://api.google.com/search/beta2"/>  
  </port>  
</service>  
</definitions>
```

UDDI

Which services can I use?

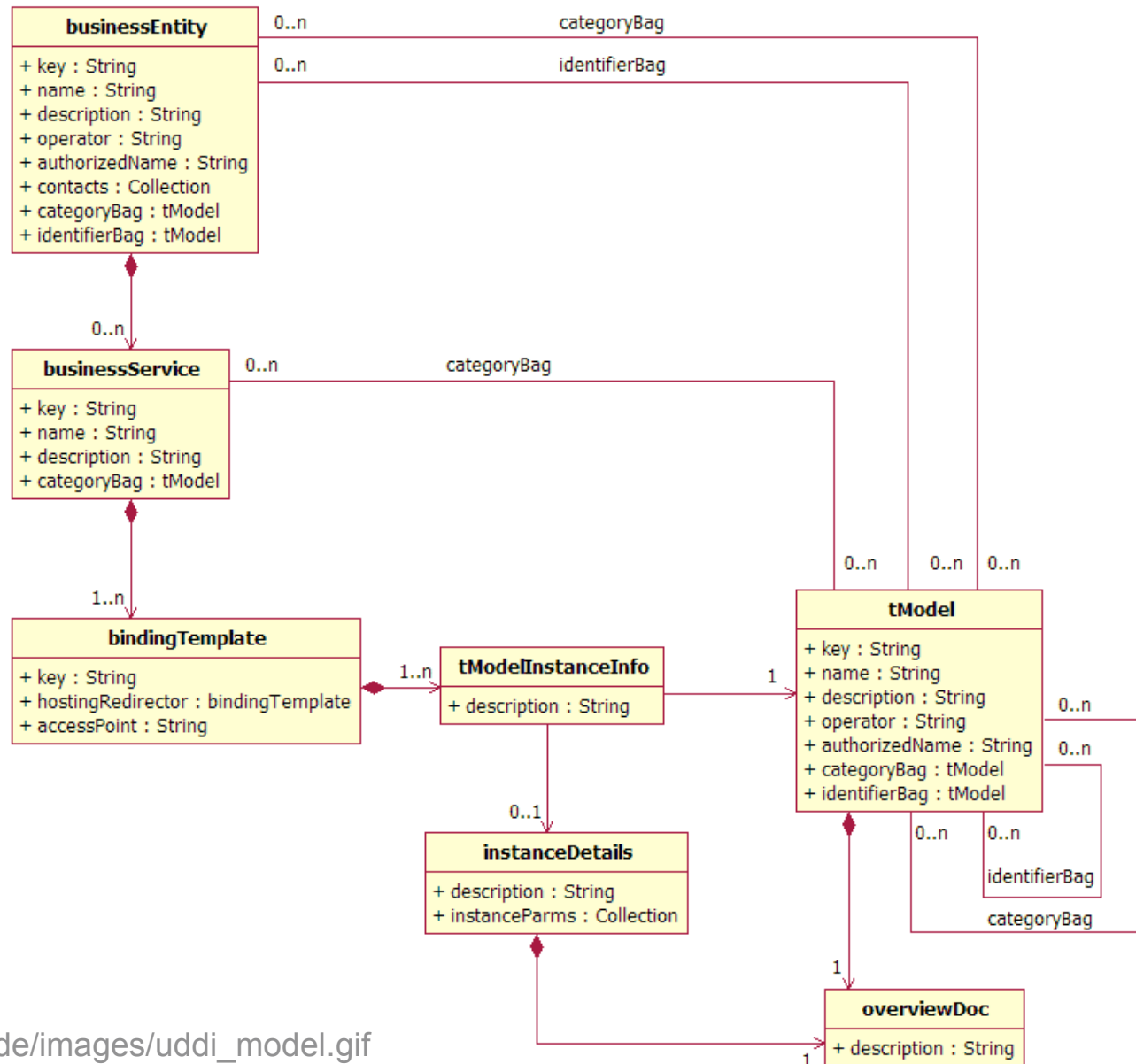
- Provides registries and mechanisms for Web service discovery
- Service providers publish information about their services on UDDI registry
- Service requesters query UDDI registry for information about existing services
- <http://www.uddi.org>





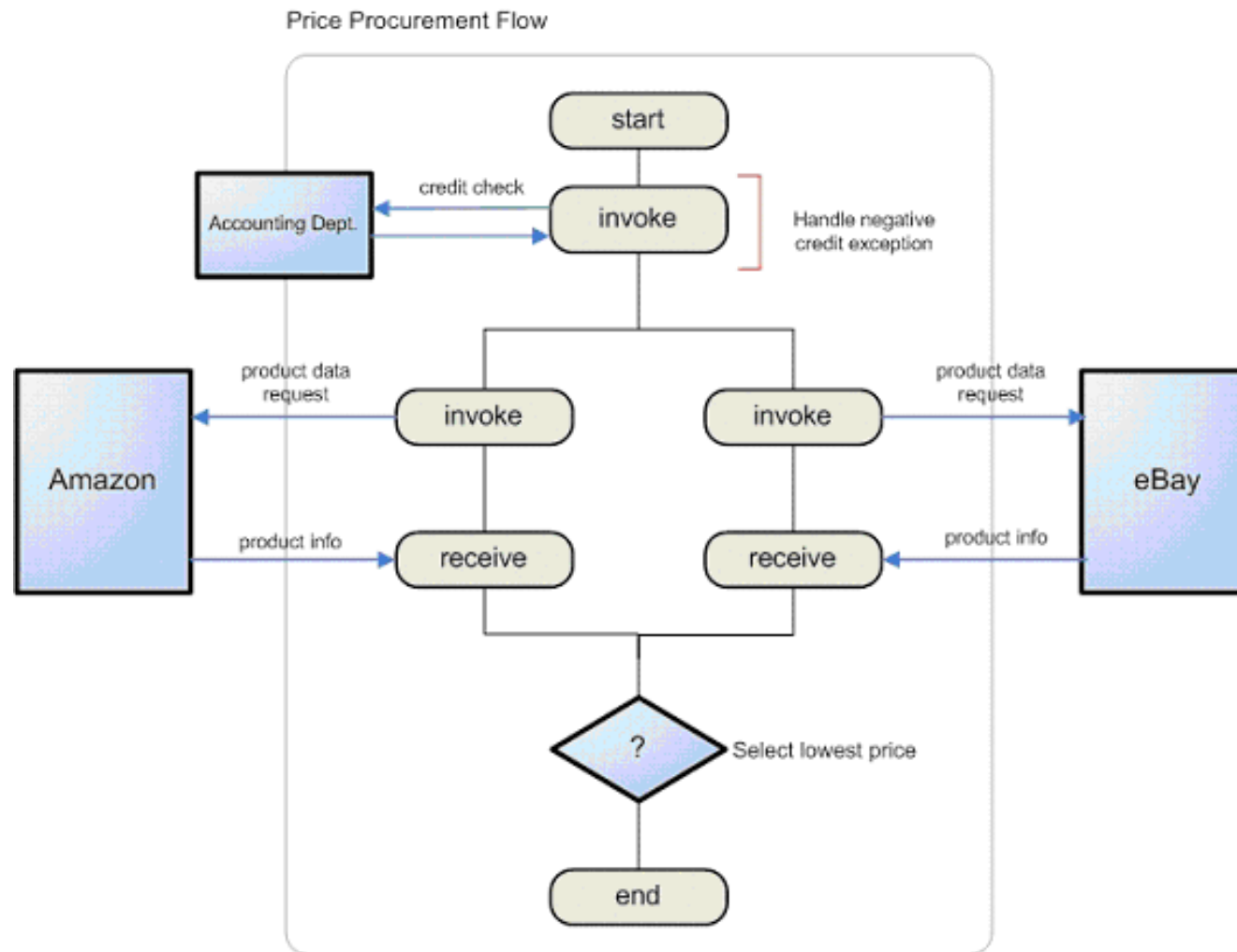
UDDI Model

- **businessEntity**
 - describes a concrete business
- **businessService**
 - describes a Web service
- **bindingTemplate**
 - describes invocation information
- **tModel**
 - technical details, e.g. reference to WSDL description



http://java.boot.by/wsd-guide/images/uddi_model.gif

Complex Business Processes

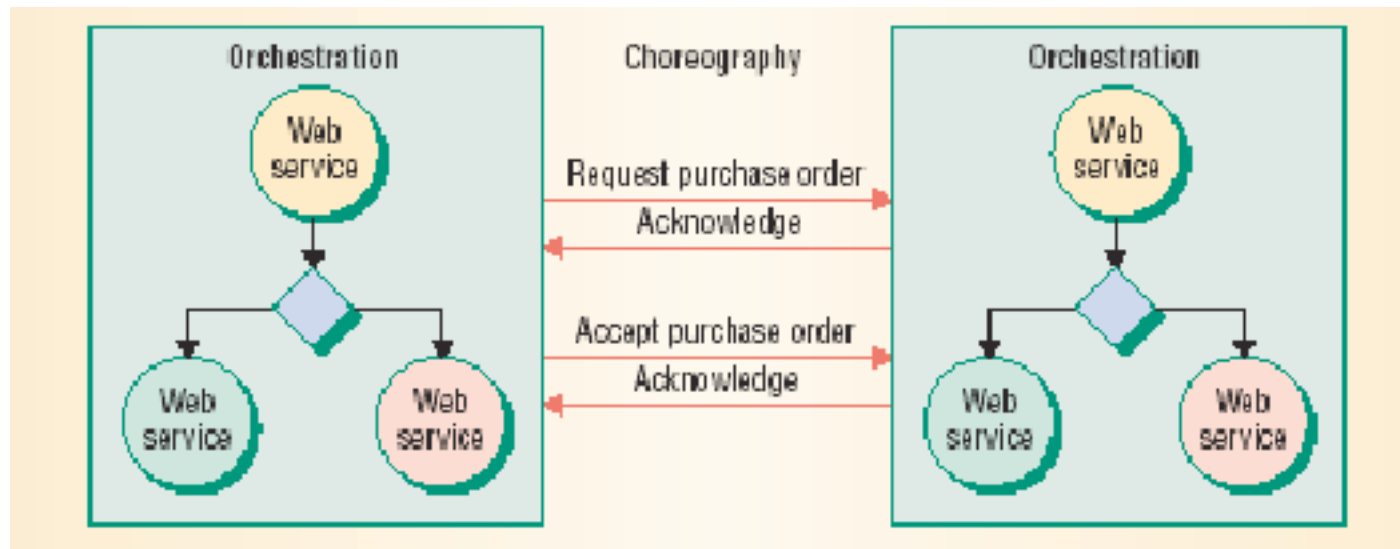


<http://www.devx.com/assets/vendshow/11625.gif>

Requirements for supporting B2B interactions

- Service execution will be subject to service level agreements (SLAs).
- Collaborations between partners will require a shared context to be passed with the business documents.
- Collaborative business processes may take days to execute, so need to support transactions
- The service definitions that are exposed by a business will be composed of internal business processes.
- Many of these "business compositions" will be complex, requiring the use of process automation/workflow for their definition and execution.
- A business composition is described by a process workflow, where the steps of the workflow are business functionality within the enterprise, such as packaged applications accessed through enterprise application integration (EAI) or business objects accessed through service interfaces.

Choreography vs. Orchestration



- Orchestration model: interactions between multiple Web services from the view of single Web service
- Choreography: Global exchange of messages between Web services

BPEL

- Business Process Execution Language
 - An orchestration language
 - provides a means to formally specify business processes and interaction protocols
- extends the Web Services interaction model and enables it to support business transactions
- describes complex work flows of business processes

<http://www-128.ibm.com/developerworks/library/specification/ws-bpel/>

BPEL Structure

Describes

- Partners
- Messages
- Message flow

```
<process name="ticketOrder">
  <partners>
    <partner name="customer"
      serviceLinkType="agentLink"
      myRole="agentService"/>
    <partner name="airline"
      serviceLinkType="buyerLink"
      myRole="ticketRequester"
      partnerRole="ticketService"/>
  </partners>
  <containers>
    <container name="itinerary" messageType="itineraryMessage"/>
    <container name="tickets" messageType="ticketsMessage"/>
  </containers>
  <flow>
    <links>
      <link name="order-to-airline"/>
      <link name="airline-to-agent"/>
    </links>
    <receive partner="customer"
      portType="itineraryPT"
      operation="sendItinerary"
      container="itinerary"
      <source linkName="order-to-airline"/>
    </receive>
    <invoke partner="airline"
      portType="ticketOrderPT"
      operation="requestTickets"
      inputContainer="itinerary"
      <target linkName="order-to-airline"/>
      <source linkName="airline-to-agent"/>
    </invoke>
    <receive partner="airline"
      portType="itineraryPT"
      operation="sendTickets"
      container="tickets"
      <target linkName="airline-to-agent"/>
    </receive>
  </flow>
</process>
```

<http://www-128.ibm.com/developerworks/webservices/library/ws-bpelwp/>

Why are Web Services useful?

For Developers...

- Access to a “Web-wide library of software components”
- Smart development tools can...
 - Help you locate useful Web Services
 - Download service descriptions (WSDL)
 - Automatically generate code from the WSDL to talk to the service using SOAP over HTTP
 - On the server side, automatically generate WSDL for a service from its source code

Use the Web Service in your own programs

```
using System;
using System.IO;

public class Quote {

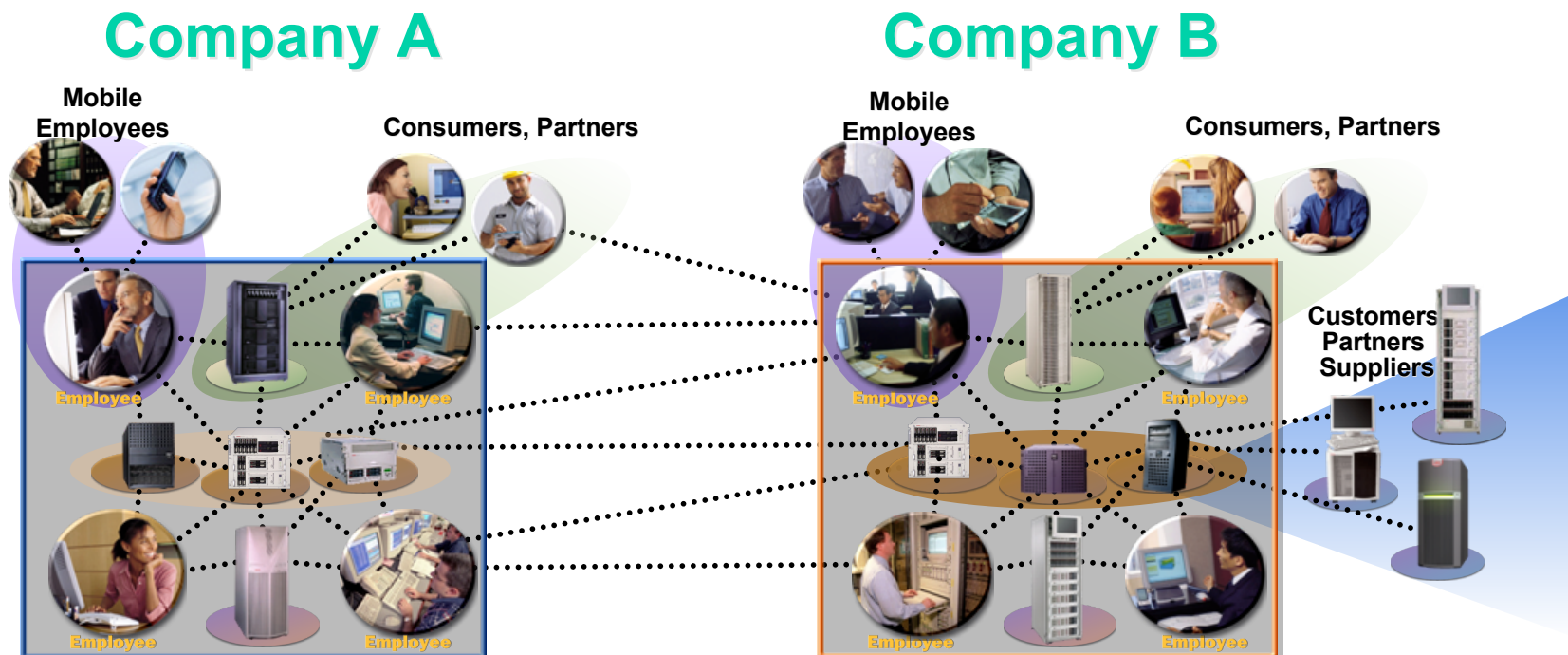
    public static void Main(String[] args) {
        StockQuoteService service = new StockQuoteService();
        float msftPrice = service.getQuote("MSFT");
        Console.WriteLine(msftPrice);
    }
}
```

For Businesses...

- Three keys to next generation applications:
 - “Any-to-Any” integration
 - *Integral assumption of development*
 - *Must tie together “islands of data, devices, OS, businesses, people”*
 - Intelligent devices
 - *Many types, with varying capabilities, but all speak common protocols*
 - *Anytime, anywhere access*
 - Open and accessible to all
 - *Open, internet based standards*
 - *Broad accessibility*

New Applications

- Shift to decentralized/distributed
- Span multiple clients, servers, services
- Federate across organizations
- Build systems that play in larger solutions

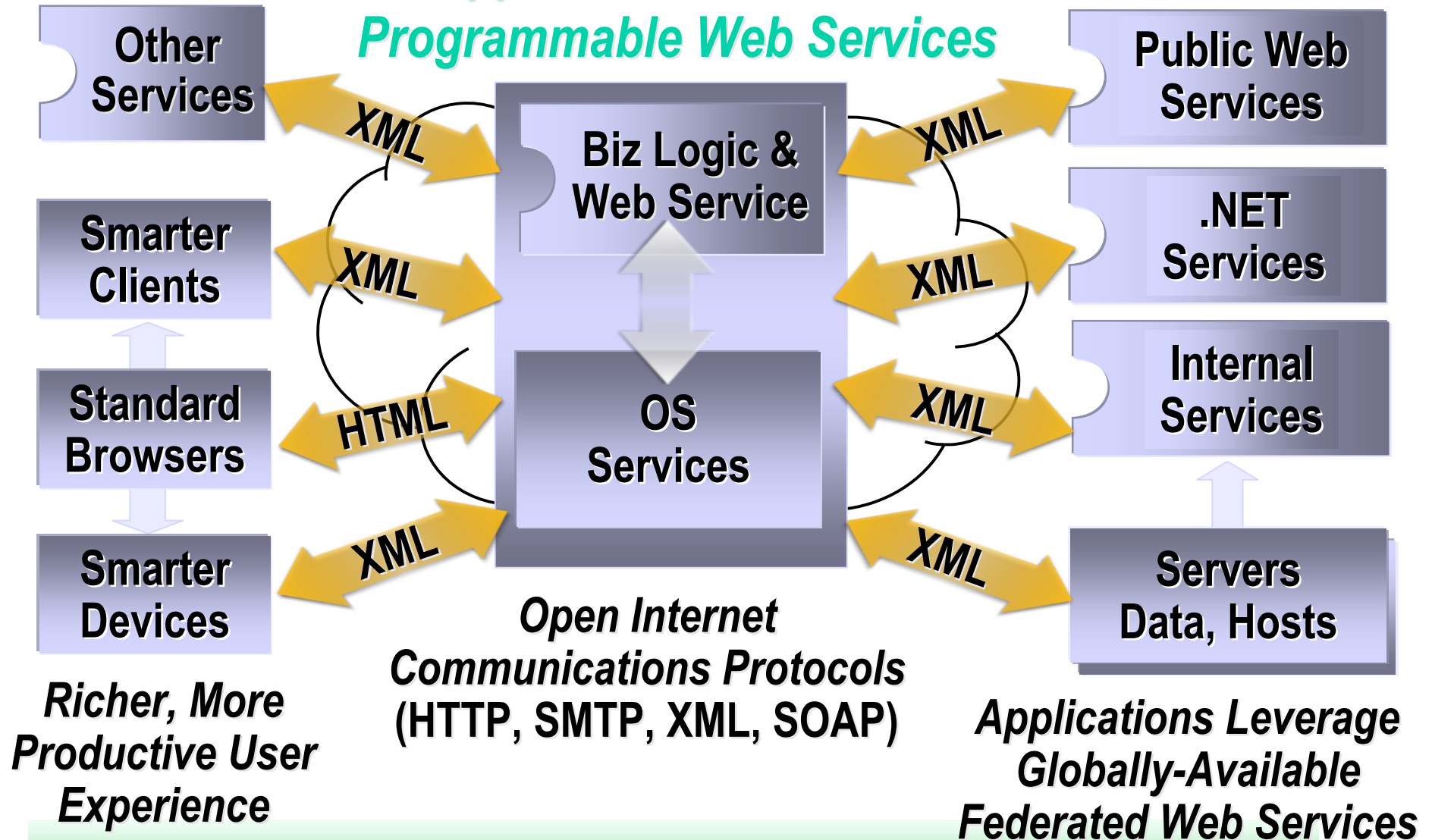


The economic value of Web Services?

- Every data exchange is potentially a revenue opportunity
 - Both the raw data and the exchange/translation can have value
 - *Ex: stock quotes are essentially free, but stock alerts sent to my phone have value*
 - Still need someone willing to buy it
- Web Services help in two ways:
 - Increase availability of data
 - *“It’s on the web!”*
 - Enabled clients = potential customer pool for your data

Next Gen Web Applications

Applications Become Programmable Web Services



Acknowledgements

- Brian LaMachhia, CS155, Spring 03, Yale University
 - <http://zoo.cs.yale.edu/classes/cs155/spr03/lectures.html>
- Mark Slater, CMPS 183, May 2005, UC Santa Cruz
 - <http://www.cse.ucsc.edu/classes/cmcs183/Spring03/>
- An Introduction to XML and Web Technologies -- Web Services, Anders Mueller & Michael I. Schwartzbach 2006, Addison-Wesley

Next Lecture

- Web Services
 - What is a Web Service and how can it be used?
 - SOAP
 - WSDL
 - UDDI
 - BPEL
 - Business case for Web services
- Semantic Web Services
 - Some criticisms of the standards
 - Semantics to reduce interoperability issues
 - OWL-S
 - WSMO

END

