

Comprehensive Description and Management of XML Web Services

Dr. Vladimir Tosic

Dep. of Computer Science
Univ. of Western Ontario

London, Ontario, Canada

E-mail: vladat@computer.org

<http://elab.njit.edu/vladimir/>

Prof. Patrick C.K. Hung

Faculty of Business and IT
Univ. of Ontario Institute of
Technology - UOIT

Oshawa, Ontario, Canada

E-mail: Patrick.Hung@uoit.ca

<http://www.cs.ust.hk/~cshck/>

Tutorial Goals

- Introduce participants to the vision of service-oriented computing (publish-find-bind model) and XML Web services
- Give an overview of basic Web service technologies
- Explain that comprehensive description and management are crucial for achieving the vision of service-oriented computing
- Summarize and analyze theoretical concepts, the main past results, and open issues
- Provide a foundation for future research and/or decision-making by the participants

Presentation Outline

- I. **Introduction: Web services and their importance**
- II. Overview of the basic Web service (WS) technologies: WSDL, SOAP, UDDI, BPEL4WS
- III. Importance of comprehensive description and management for Web services
- IV. Approaches and languages for comprehensive description for Web services
- V. Security and privacy technologies for WSes
- VI. Approaches and tools for management of WSes
- VII. Summary: Past results and some open issues

Web Services - Definition

W3C Definition of a Web Service

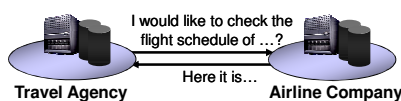
- has a unique Uniform Resource Identifier (URI)
- can be defined, described, and discovered using XML
- supports exchange of XML messages via Internet-based protocols

Supported by all major computing companies, e.g., IBM, Microsoft, Sun, and etc.

Motivation for Web Services

Major drawbacks of traditional business-to-business applications:

- Complex, custom, one-off solutions
- Proprietary end points
- Not scalable
- Costly and time consuming

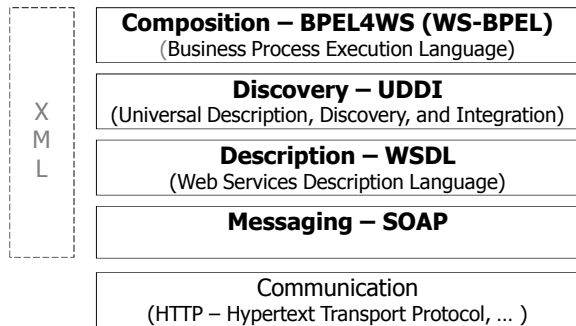


Web Service Standards

Web services are based on *a set of XML standards*:

- Web Services Description Language (WSDL)
- **SOAP** (previously an abbreviation for: Simple Object Access Protocol)
- Universal Description, Discovery and Integration (**UDDI**)
- With other emerging standards such as
 - Web Services Business Process Execution Language (WS-BPEL, previously known as **BPEL4WS**)
 - WS-Security

Stack of Basic Web Service Technologies



Web Services - Explanation

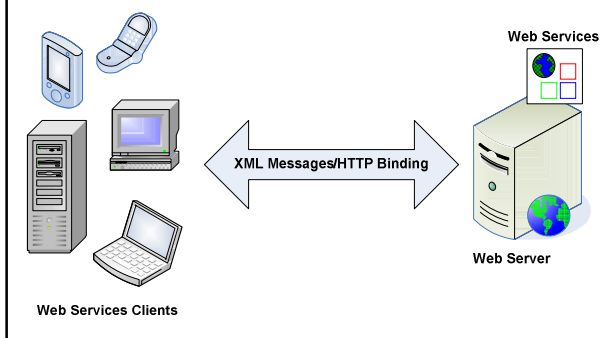
A Web service is a software system designed to support interoperable machine-to-machine/application-to-application interaction over a network.

A Web service has an interface described in a machine-processable format (specifically WSDL).

Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards.

Ref: Web Services Architecture, Editors' copy, Date: 2004/02/05.
<http://dev.w3.org/cvsweb/~checkout-/2002/ws/arch/wsa/wd-wsa-arch-review2.html>

Web Services - Explanation (cont.)



Web Services - Benefits

Some benefits of adopting Web services:

- Platform and vendor independent
- A significant reduction in total cost of development
- Easy to deploy business applications for trading partners
- Convergence of disparate business functionalities

Ref: RATNASINGAM, P. 2002. The Importance of Technology Trust in Web Services Security. Information Management & Computer Security, vol. 10, no. 5, 255-260.

Web Services - Benefits (cont.)

A pool of Web services can provide an easier integration environment:

- Interoperability
- Reusability
- Robustness

Initial applications are usually within businesses (behind the firewall or Intranet) - to gain trust.

An Illustrative Example

Many airline companies have Web sites with information about their services, e.g., flight schedules.

Finding flight and fare information is not simple in automatic business-to-business (B2B) applications.

Ref: SELVIDGE, P. 1999. Student posters: Reservations about the usability of airline web sites Paula Selvidge, CHI '99 extended abstracts on Human factors in computing systems.

An Illustrative Example (cont.)

SCHEDULES AND FARES

From: Hong Kong - HKG
To: Boston - BOS

Round trip / One-way

Departure Date: 25 Mar 04
Return Date: 29 Mar 04

Cabin Class: Economy Restricted

Adults: 1, Children: 0

Show Schedules / Show Fares and Availability

Adapted from: <http://www.cathaypacific.com>

Fare finder

From: Hong Kong To: Boston, MA

Round trip / One way / Multi city

Departure date: Mar 25 Morning
Return: Mar 29 Evening

Search by: Schedule / Price

Electronic certificate number (optional):

Passengers: 1

Book at [united.com](http://www.united.com), earn 1000 bonus miles. [More](#)

Adapted from: <http://www.united.com>

An Illustrative Example (cont.)

The proprietary protocol to invoke the flight schedules operation at:

- www.cathaypacific.com

```
http://www.cathaypacific.com/intl/plan/schedules/result/0,3845,,00.html?fromShortCut=&language=GB&device=&tripType=2&d_city=HKG&a_city=BOS&d_month=200403&d_day=25&r_month=200403&r_day=29&errorListIndex=&errorList=
```

- www.united.com

```
http://www.itn.net/cgi/air?stamp=NEWCOOKY*itn%2Ford%3DNEWREC,itn/air/united&airline=United&persons=1&air_avai=10&air_class=coach+%28lowest+avail.%29&depart=Hong+Kong&dest.0=Boston%2C+MA&mon_abbr.0=Mar&date.0=25&hour_ampm.0=5+am&mon_abbr.1=Mar&date.1=29&hour_ampm.1=5+am&ecert_num=&rt_ow=Round+Trip&best_itins=2&return_to=best_itins
```

An Illustrative Example (cont.)

25 March 2004 (Thursday) Hong Kong (HKG) to Boston (BOS)

Flight	Departure	Arrival	Days of Service	Aircraft	No. of Stops	Duration (Hours)	Notes
CA182	14:14:30	LAX 12:45	SMWTF	EQV	0	12:25	ET
CA588	LAX 15:25	BOS 23:41	SMWTF	EQV	0	05:16	CP

Adapted from: <http://www.cathaypacific.com>

Hong Kong (HKG) to Boston (BOS)

2195.43 USD per person

Leg	Flight info	Date	Depart	Arrive	Stop
1	United Airlines 960	Mar 25	11:00 am HKG	11:40 am BOS	Non-stop
2	United Airlines 582	Mar 25	2:00 pm BOS	5:21 pm BOS	Non-stop

Additional flight options:

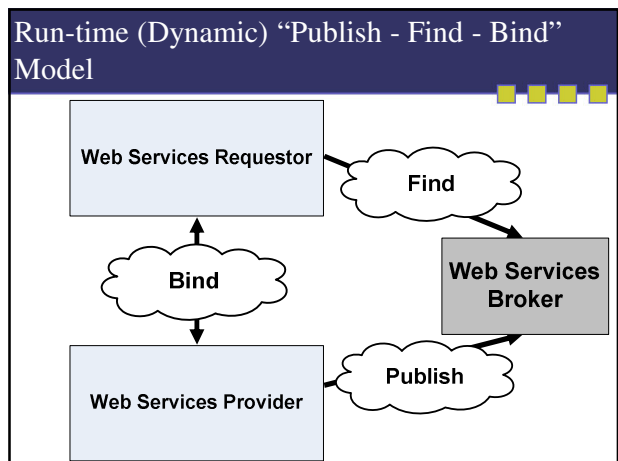
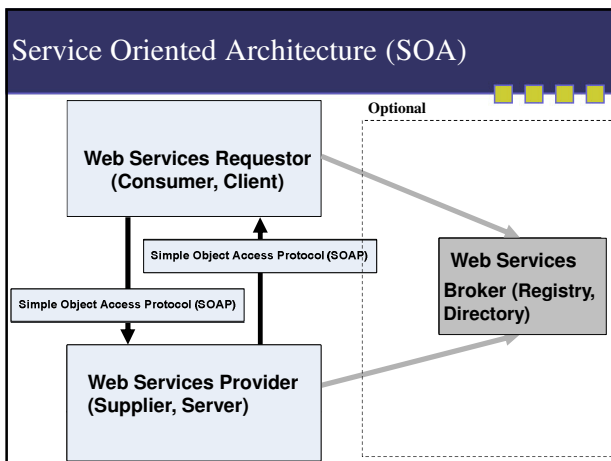
2198.43 USD per person

Leg	Flight info	Date	Depart	Arrive	Stop
1	United Airlines 960	Mar 25	9:45 am HKG	2:30 pm BOS	Non-stop
2	United Airlines 582	Mar 25	5:00 pm BOS	7:21 pm BOS	Plane change

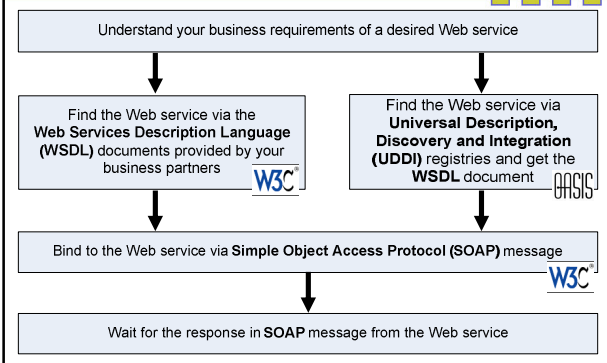
Adapted from: <http://www.united.com>

What's Next?

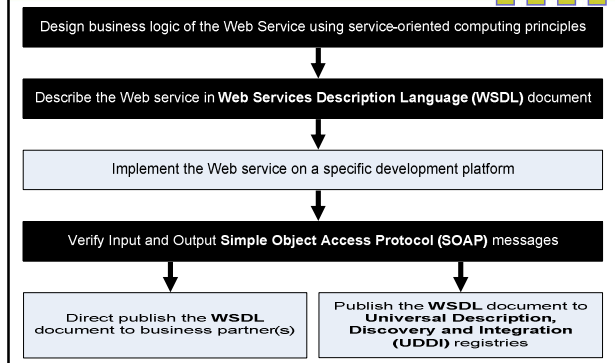
- I. Introduction: Web services and their importance
- II. Overview of the basic Web service (WS) technologies: WSDL, SOAP, UDDI, BPML4WS
- III. Importance of comprehensive description and management for Web services
- IV. Approaches and languages for comprehensive description for Web services
- V. Security and privacy technologies for WSEs
- VI. Approaches and tools for management of WSEs
- VII. Summary: Past results and some open issues



Steps to Be Taken on the Requestor Side



Steps to Be Taken on the Provider Side



An Example of Web Service Description

Web Service Name: *AirLineWS*

Operation Name: *getSchedules*

- **Input Parameter(s):**
 - *origin, destination, day_of_travel, month_of_travel*
- **Output Parameter(s):**
 - *getSchedulesReturn*

Transport Protocol Binding: *HTTP*

URI: *http://www.airlinecompany.com/services/AirLineWS*

Web Services Description Language (WSDL)

Web Services Description Language (WSDL)

describes the Web service's interface:

- what operations the Web service supports
- what protocols to use
- how the data exchanged should be packed

Ref: Web Services Description Language (WSDL) Version 2.0, W3C Working Draft, 10 November 2003. <http://www.w3c.org/TR/wsd20/>

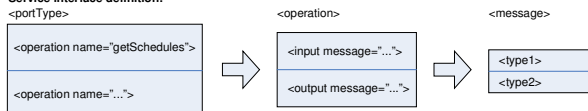
The WSDL document is a contract between the service requestor and provider.



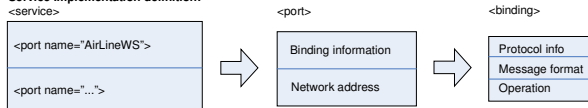
Web Services Description Language (WSDL)

- Details

Service interface definition:



Service Implementation definition:



Web Services Description Language (WSDL)

- Example

```

<wSDL:definitions targetNamespace="AirLineWS" xmlns:impl="AirLineWS" xmlns:intf="AirLineWS"
xmlns:apachesoap="http://xml.apache.org/xml-soap" xmlns:wsdloap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:wSDL="http://schemas.xmlsoap.org/wsdl/" xmlns="http://schemas.xmlsoap.org/wsdl/">
  
```

definitions: It declares the name of the Web service and the namespaces used to define the elements found throughout the remainder of the document.

Web Service Name: *AirLineWS*

Web Services Description Language (WSDL) - Example (cont.)

```

- <wsdl:message name="getSchedulesRequest">
  <wsdl:part name="origin" type="xsd:string" />
  <wsdl:part name="destination" type="xsd:string" />
  <wsdl:part name="day_of_travel" type="xsd:string" />
  <wsdl:part name="month_of_travel" type="xsd:string" />
</wsdl:message>
- <wsdl:message name="getSchedulesResponse">
  <wsdl:part name="getSchedulesReturn" type="xsd:string" />
</wsdl:message>

```

message: For all input, output, and fault messages, it specifies message name and message parts (parameters).

Input Parameter(s):

origin, destination, day_of_travel, month_of_travel

Output Parameter(s):

getSchedulesReturn

Web Services Description Language (WSDL) - Example (cont.)

```

- <wsdl:portType name="AirLineWS_IF">
  <wsdl:operation name="getSchedules" parameterOrder="origin destination day_of_travel month_of_travel">
    <wsdl:input name="getSchedulesRequest" message="impl:getSchedulesRequest" />
    <wsdl:output name="getSchedulesResponse" message="impl:getSchedulesResponse" />
  </wsdl:operation>
</wsdl:portType>

```

operation: It is a description of a web method in terms of the messages that are sent and received.

portType: It is a group of operations.

Operation Name: *getSchedules*

Web Services Description Language (WSDL) - Example (cont.)

binding: It describes the transport protocols, message formats, and message styles supported by a given Web service.

Transport Protocol Binding: HTTP

```

- <wsdl:binding name="AirLineWSSoapBinding" type="impl:AirLineWS_IF">
  <wsdl:soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http" />
  <wsdl:operation name="getSchedules">
    <wsdl:soap:operation soapAction="" />
    <wsdl:input name="getSchedulesRequest">
      <wsdl:soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
        namespace="AirLineWS" />
    </wsdl:input>
    <wsdl:output name="getSchedulesResponse">
      <wsdl:soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
        namespace="AirLineWS" />
    </wsdl:output>
  </wsdl:operation>
</wsdl:binding>

```

Web Services Description Language (WSDL) - Example (cont.)

port: It maps a binding to a network location.

service: It defines the address for invoking the specified service.

URL: *http://www.airlinecompany.com/services/AirLineWS*

```

- <wsdl:service name="AirLineWS_IFService">
  <wsdl:port name="AirLineWS" binding="impl:AirLineWSSoapBinding">
    <wsdl:soap:address location="http://www.airlinecompany.com/services/AirLineWS" />
  </wsdl:port>
</wsdl:service>

```

SOAP

SOAP (previously known as Simple Object Access Protocol) is an XML-based messaging protocol.

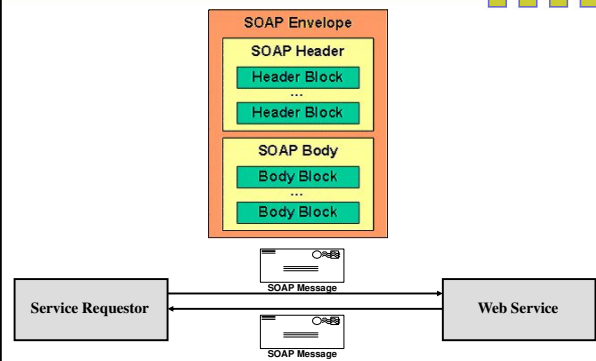
SOAP is independent of the underlying transport protocol:

- HTTP
- SMTP
- FTP.

Ref: SOAP Version 1.2, W3C Recommendation, 24 June 2003.
<http://www.w3.org/2000/xp/Group/>



SOAP - Message Structure



SOAP - Example

The **Input SOAP Message** from **Web Service Requestor** to **Web Service**:

```
<?xml version="1.0" encoding="utf-8" ?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
```

Envelope: defines an overall framework for expressing **what** is in a message **who** should deal with it **whether** it is optional or mandatory

```
</soap:Envelope>
```

SOAP - Example (cont.)

Body: contains call and response information

```
<wsdl:service name="AirLineWS_IFService">
  <wsdl:port name="AirLineWS" binding="Impl:AirLineWSSoapBinding">
    <wsdl:soap:address location="http://www.airlinecompany.com/Services/AirLineWS"/>
  </wsdl:port>
</wsdl:service>

<soap:Body>
  <getSchedules xmlns="http://www.airlinecompany.com/services/AirLineWS">
    <origin>HKG</origin>
    <destination>BOS</destination>
    <day_of_travel>25</day_of_travel>
    <month_of_travel>march</month_of_travel>
  </getSchedules>
</soap:Body>
```

```
<wsdl:message name="getSchedulesResponse">
  <wsdl:part name="origin" type="xsd:string"/>
  <wsdl:part name="destination" type="xsd:string"/>
  <wsdl:part name="day_of_travel" type="xsd:string"/>
  <wsdl:part name="month_of_travel" type="xsd:string"/>
</wsdl:message>
```

SOAP - Example (cont.)

The **Output SOAP Message** from **Web Service** to **Web Service Requestor**:

```
<?xml version="1.0" encoding="utf-8" ?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <getSchedulesResponse xmlns="http://www.airlinecompany.com/services/AirLineWS">
      <getSchedulesReturn>...</getSchedulesReturn>
    </getSchedulesResponse>
  </soap:Body>
</soap:Envelope>
```

Universal Description, Discovery, and Integration (UDDI)

Universal Description, Discovery and Integration (UDDI) is a 'yellow pages'.

UDDI provides a standard means for:

- describing businesses and their services
- allowing the online discovery

Ref: UDDI Version 3.0.1, UDDI Spec Technical Committee Specification, Dated 20031014. <http://uddi.org/pubs/uddi-v3.0.1-20031014.htm>

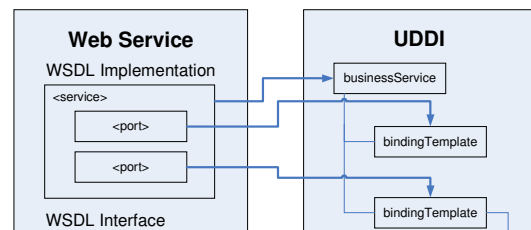


Universal Description, Discovery, and Integration (UDDI) - Details



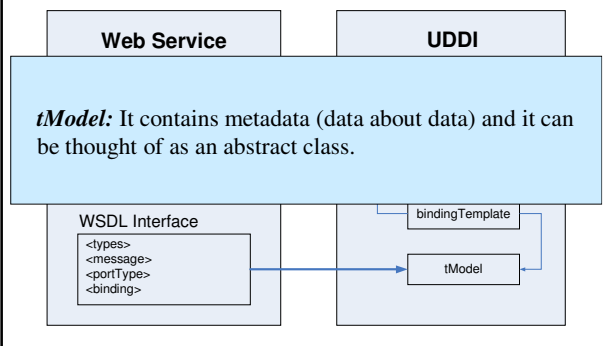
businessService: It contains a service description, a list of associated categories for the service, and a list of pointers to references and information about the service.

Universal Description, Discovery, and Integration (UDDI) - Details (cont.)



bindingTemplate: It contains technical information about a service entry point such as the location or access point in the form of a URL.

Universal Description, Discovery, and Integration (UDDI) - Details (cont.)



Some UDDI Implementations

Live production UDDI registries:

- uddi.microsoft.com
- uddi.ibm.com
- Udditest.sap.com



BPEL4WS (WS-BPEL)

Workflow is a computer supported business process.

The need for coordinating Web services for different business processes in a loosely coupled business-to-business environment.

A *business process* contains a set of activities which represent both business tasks and interactions between Web services.

Web Services Business Process Execution Language (WS-BPEL, previously known as BPEL4WS)

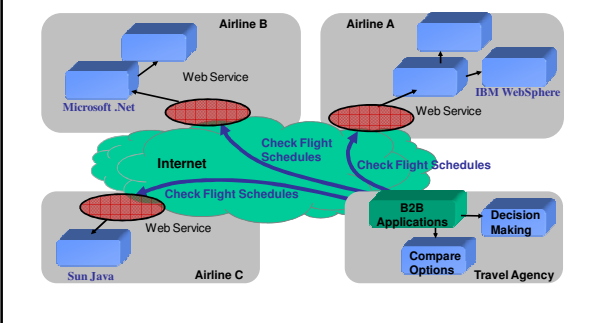
BPEL4WS (cont.)

BPEL4WS defines a model and a grammar for describing the behavior of a business process based on interactions between the process and its Web service interfaces

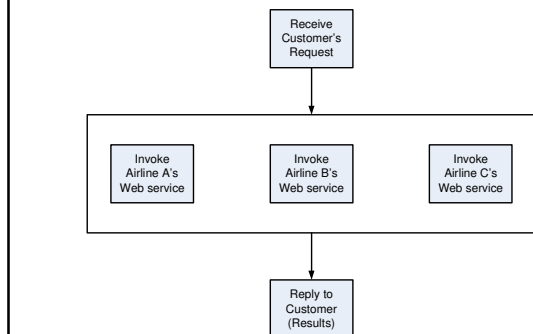
XML workflow language based on WSDL service descriptions with extension elements

In short, a BPEL4WS business process definition can be thought of as a template for creating business process instances.

BPEL4WS - Example



BPEL4WS - Example (cont.)



BPEL4WS - Example (cont.)

```

<process name="CheckFlightSchedulesProcess"
  targetNamespace="http://travelagencies.com/getSchedulesProcessing"
  xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
  xmlns:as="http://airlines.org/wsdl/AirLineWS"
  xmlns:ta="http://travelagencies.org/wsdl/TravelAgencyWS" suppressJoinFailure="yes">
  <partnerLinks>
    <partnerLink name="travelAgency" partnerLinkType="ta:TravelAgencyWS_IF"
      myRole="travelAgency" />
    <partnerLink name="airlineA" partnerLinkType="as:AirLineWS_IF" partnerRole="airlineA" />
    <partnerLink name="airlineB" partnerLinkType="as:AirLineWS_IF" partnerRole="airlineB" />
    <partnerLink name="airlineC" partnerLinkType="as:AirLineWS_IF" partnerRole="airlineC" />
  </partnerLinks>
  <variables>
    <variable name="request" messageType="ta:getSchedulesRequest" />
    <variable name="responseA" messageType="as:getSchedulesResponse" />
    <variable name="responseB" messageType="as:getSchedulesResponse" />
    <variable name="responseC" messageType="as:getSchedulesResponse" />
    <variable name="results" messageType="ta:ReplySchedulesResults" />
  </variables>
  <flow>
    <links>
      <link name="Request-to-AirlineA" />
      <link name="Request-to-AirlineB" />
      <link name="Request-to-AirlineC" />
      <link name="AirlineA-to-Results" />
      <link name="AirlineB-to-Results" />
      <link name="AirlineC-to-Results" />
    </links>
    <receive partnerLink="travelAgency" portType="ta:TravelAgencyWS_IF" operation="request"
      variable="request" createInstance="yes">
      <source linkName="Request-to-AirlineA" />
      <source linkName="Request-to-AirlineB" />
      <source linkName="Request-to-AirlineC" />
    </receive>
  </flow>
</process>

```

BPEL4WS - Example (cont.)

```

<partnerLinks>
  <partnerLink name="travelAgency" partnerLinkType="ta:TravelAgencyWS_IF"
    myRole="travelAgency" />
  <partnerLink name="airlineA" partnerLinkType="as:AirLineWS_IF" partnerRole="airlineA" />
  <partnerLink name="airlineB" partnerLinkType="as:AirLineWS_IF" partnerRole="airlineB" />
  <partnerLink name="airlineC" partnerLinkType="as:AirLineWS_IF" partnerRole="airlineC" />
</partnerLinks>

```

partnerLinks: It defines the different parties that interact with the business process in the course of processing the order.

BPEL4WS - Example (cont.)

```

<variables>
  <variable name="request" messageType="ta:getSchedulesRequest" />
  <variable name="responseA" messageType="as:getSchedulesResponse" />
  <variable name="responseB" messageType="as:getSchedulesResponse" />
  <variable name="responseC" messageType="as:getSchedulesResponse" />
  <variable name="results" messageType="ta:ReplySchedulesResults" />
</variables>

```

variables: It defines the data variables used by the process, providing their definitions in terms of WSDL message types.

BPEL4WS - Example (cont.)

flow: It specifies one or more activities to be performed concurrently.

Links: It can be used within concurrent activities to define arbitrary control structures.

```

<flow>
  <links>
    <link name="Request-to-AirlineA" />
    <link name="Request-to-AirlineB" />
    <link name="Request-to-AirlineC" />
    <link name="AirlineA-to-Results" />
    <link name="AirlineB-to-Results" />
    <link name="AirlineC-to-Results" />
  </links>

```

BPEL4WS - Example (cont.)

received: It allows the business process to do a blocking wait for a matching message to arrive.

Receive
Customer's
Request

```

<receive partnerLink="travelAgency" portType="ta:TravelAgencyWS_IF" operation="request"
  variable="request" createInstance="yes">
  <source linkName="Request-to-AirlineA" />
  <source linkName="Request-to-AirlineB" />
  <source linkName="Request-to-AirlineC" />
</receive>

```

BPEL4WS - Example (cont.)

```

<invoke partnerLink="airlineA" portType="as:AirLineWS_IF" operation="getSchedulesRequest"
  inputVariable="request" outputVariable="responseA">
  <target linkName="Request-to-AirlineA" />
  <source linkName="AirlineA-to-Results" />
</invoke>

```

invoke: It allows the business process to invoke a one-way or request-response operation on a portType offered by a partner.

Invoke
Airline A's
Web service

BPEL4WS - Example (cont.)

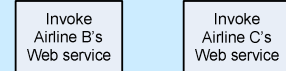
```

- <assign>
  <target linkName="Request-to-AirlineA" />
  <source linkName="AirlineA-to-Results" />
- <copy>
  <from variable="responseA" />
  <to variable="results" part="1" />
</copy>
</assign>

```

assign: It can be used to update the values of variables with new data. An <assign> construct can contain any number of elementary assignments.

BPEL4WS - Example (cont.)



```

- <invoke partnerLink="airlineB" portType="as:AirLineWS_IF" operation="getSchedulesRequest"
  inputValue="request" outputVariable="responseB">
  <target linkName="Request-to-AirlineB" />
  <source linkName="AirlineB-to-Results" />
</invoke>
- <assign>
  <target linkName="Request-to-AirlineB" />
  <source linkName="AirlineB-to-Results" />
- <copy>
  <from variable="responseB" />
  <to variable="results" part="2" />
</copy>
</assign>
- <invoke partnerLink="airlineC" portType="as:AirLineWS_IF" operation="getSchedulesRequest"
  inputValue="request" outputVariable="responseC">
  <target linkName="Request-to-AirlineC" />
  <source linkName="AirlineC-to-Results" />
</invoke>

```

BPEL4WS - Example (cont.)

reply: It allows the business process to send a message in reply to a message that was received through a <receive>.

Reply to Customer (Results)

```

- <reply partnerLink="travelAgency" portType="ta:TravelAgencyWS_IF" operation="reply"
  variable="results">
  <target linkName="AirlineA-to-Results" />
  <target linkName="AirlineB-to-Results" />
  <target linkName="AirlineC-to-Results" />
</reply>
</flow>
</process>

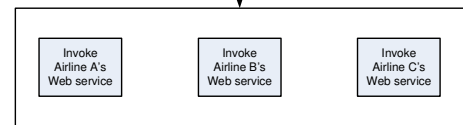
```

BPEL4WS - Example (cont.)

Input Message

Receive Customer's Request

Web Service



Output Message

Reply to Customer (Results)

BPEL4WS - Example (cont.)

Input Message

Receive Customer's Request

The combination of a <receive> and a <reply> forms a request-response operation on the WSDL portType for the process.

Output Message

Reply to Customer (Results)

What's Next?

- I. Introduction: Web services and their importance
- II. Overview of the basic Web service (WS) technologies: WSDL, SOAP, UDDI, BPEL4WS
- III. **Importance of comprehensive description and management for Web services**
- IV. Approaches and languages for comprehensive description for Web services
- V. Security and privacy technologies for WSEs
- VI. Approaches and tools for management of WSEs
- VII. Summary: Past results and some open issues

The Need for Comprehensive Descriptions in the Service-Oriented Architecture (SOA)

- **Comprehensive description** of Web services contains not only functionality, bindings, and location, but also **quality of service (QoS), prices, penalties, security and privacy information, ...**
- On the Web service market, there will be many Web services with similar functionality
- **Comprehensive description is necessary** to:
 1. Publish phase: differentiate from competitors
 2. Find phase: determine the best match
 3. Bind phase: perform management (monitoring and control)

Definition of Management - Monitoring

- **Management = monitoring and control**
 - **Run-time** (and some deployment-time) activities
- **Monitoring** determines state of the system:
 - Measurement or calculation of QoS metrics (measures of QoS): response time, availability, ...
 - Evaluation of conditions (requirements or guarantees): response time < 2 seconds, ...
 - Accounting of invoked operations, consumed resources, measured/calculated QoS metrics, evaluated conditions, taken control actions, billed prices/penalties, ...

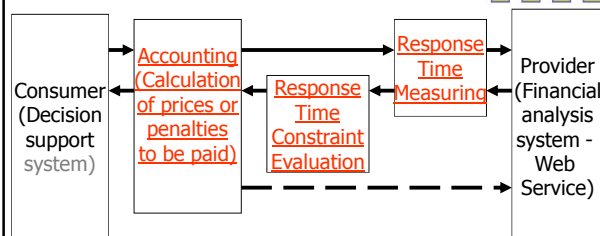
Definition of Management - Control

- **Control** tries to ensure that the managed system is always in its desired state:
 - Starting/stopping the system or its components
 - (Re-)Configuration of the system: setting thread priorities, re-composition of Web services, ...
 - (Re-)Allocation of resources: assigning processing time to requests from different consumers, ...
 - Billing of prices or penalties: penalty for not meeting guaranteed response time is US\$1.00, ...
 - Modification of requirements or guarantees
 - Notification of human administrators

Benefits of Management

- **5 functional areas** of system/network management (**FCAPS**): Fault, Configuration, Accounting, Performance, and Security
- Management helps to:
 - ensure correct operation, attain or surpass guaranteed QoS, discover and fix problems, accommodate change, achieve required security and privacy, balance price/performance ratios, bill consumers, maximize profits, ...
- => **Needed for many business uses of Web services**

An Example of the Need for Web Service Management



- The basic Web service technologies (SOAP, WSDL, UDDI, BPEL4WS) do not address comprehensive description and management!!!

What Has to Be Developed for Web Services' Comprehensive Description and Management?

1. **Well-defined** (ideally: standardized) **formats for specification of various information** (e.g., QoS)
2. Many WS **management algorithms & protocols**
 - Very diverse: selection of WSeS using QoS info, negotiation of QoS of a WS, monitoring of QoS, exchange of run-time QoS info, control to achieve QoS guaranties, adaptation to changes in QoS, ...
 - We will discuss them in the context of specification formats and/or management tools
3. Web service **management infrastructures/tools**

What's Next?

- I. Introduction: Web services and their importance
- II. Overview of the basic Web service (WS) technologies: WSDL, SOAP, UDDI, BPEL4WS
- III. Importance of comprehensive description and management for Web services
- IV. [Approaches and languages for comprehensive description for Web services](#)
- V. Security and privacy technologies for WSEs
- VI. Approaches and tools for management of WSEs
- VII. Summary: Past results and some open issues

Overview of Approaches to Comprehensive Description of Web Services (Example: QoS)

- Management specification = description of what/where/when/how to monitor and/or control
- Management info = descriptions+monitored values
- Classification of Web service **comprehensive description** (management specification) **approaches**:
 1. **Implicit** – built into the implementation (not flexible)
 2. **Contracts** – formal agreements (for QoS, billing, ...)
 - **Service Level Agreements (SLAs)**
 - **Classes of service** – a special type of SLAs
 3. **Policies** – high-level operation & management goals and/or rules (for security, QoS, billing, ...)

Contract

- **Contract** = (binding and enforceable) **formal agreement** between two or more parties
- Defines **requirements & guarantees** of parties
 - Can be used in monitoring and control
- Contracts enable not only comprehensive description, but also **differentiation**
 - Different consumers can have different contracts
- A contract can contain various information: QoS specifications, prices/penalties, security info, ...
 - A WSDL file is a contract

Specification of QoS in Extended WSDL, UDDI, or BPEL4WS Files

- **Strengths**:
 - The extensions can be relatively simple
 - QoS discovery related to Web service discovery
- **Weaknesses**:
 - QoS specification language tied to WSDL (UDDI, BPEL4WS) in terms of tools, evolution, ...
 - Extension mechanisms are limited
 - **Run-time change of QoS information** requires updates of all affected copies of WSDL (UDDI, BPEL4WS) files, which is complicated

Service Level Agreement (SLA)

- A special **type of contract** for QoS (and often price/penalty) requirements & guarantees
- Many different **formats**, one of them is:
 - **Parties** (including supporting management parties)
 - **Service description**
 - **Service operations** – describe available operations
 - **SLA parameters** – define monitoring of QoS metrics
 - **Obligations**
 - **Service Level Objectives (SLOs)** - QoS guarantees
 - **Action guarantees** - specify what happens if SLOs are met or not met

Service Level Agreement (SLA) - A Simple Example

Parties: consumer C and provider P

Service operations: P has one operation (OP1)
float getPrice(String stockName)

SLA parameters: (RT-OP1-C) Response time of operation OP1 measured at consumer C by consumer C

SLOs: (SLO1) For every OP1 invocation by C, RT-OP1-C will be less than or equal to 2 seconds

Action guarantees: (AG1) If SLO1 was met, C pays P price of US\$0.20 per invocation;
(AG2) If SLO1 was not met, P pays C penalty of US\$0.10 per invocation

Service Level Agreement (SLA) - Strengths and Weaknesses

- Strengths:
 - **Formal contract specification** of QoS and related management aspects
 - **Widely used** in computing and communications systems (and now also for Web Services)
- Weaknesses:
 - **Negotiation of custom-made SLAs** can require complex analysis of offers and generation of counter-offers (can be alleviated by using templates)
 - **Management of many concurrent custom-made SLAs** can be complex & with high run-time overhead
 - **Can not be used for QoS-enabled Web service selection**

Class of Service

- A special **type of SLA** that is not custom-made, but **predefined and reusable (anonymous)**
 - **1** provider can offer **many** classes of service that refer to the same functionality, but differ in QoS and prices
 - **1** class of service can be used by **many** consumers
 - **Simple selection** instead of complex negotiation
 - Classes of service already checked for consistency
 - **Strengths:** Usable for QoS-enabled WS selection, no complex negotiation, simpler management, lower run-time overhead, faster adaptation
 - **Weakness:** Discrete differentiation - limited choice

Policies - High-Level Goals and/or Rules

- One **classification of policy types:**
 - **Action:** Describe what should happen - “If-Then” rules (“If response time of operation A is greater than 2 sec, provider pays penalty of US\$0.10”)
 - **Goal:** Describe desired state (“Response time of operation A is less than or equal to 2 sec”)
 - **Utility:** Quantify “goodness” of a particular state (“Add to the goodness measure [2 sec - response time of operation A] * 10 units”) - rarely used
- **SLAs vs. policies:** SLOs can be viewed as goal policies, action guarantees as action policies

QoS Specification Topics Present in All Approaches

- **Where are QoS metrics defined?**
 - There are **no standard QoS metrics** - use, names, and definitions vary! Example: ‘response time’ can have at least 2 different meanings! 4 approaches:
 1. Nowhere (implicit meaning) - not precise
 2. In the QoS language grammar - not flexible
 3. In QoS specification files (e.g., SLAs) - not reusable
 4. In external reusable ontologies (definition files)
 - Other ontologies can define measurement units
- For practical use, QoS specification languages **must** be accompanied by appropriate **tools!**

Why Not Reusing QoS Specification Languages from Other Areas?

- **Many existing QoS specification languages** in multimedia (HQML, ...), distributed objects (QML, QDL, QIDL, ...), and other areas
- **Can not be directly re-used** because of:
 - Incompatibility with WS standards (e.g., WSDL)
 - Heterogeneity of WS implementations and interactions styles used (asynchronous & synchronous; document-based & RPC; ...)
 - Characteristics of WS compositions: business-to-business, Internet scale, dynamism, automatism, ...

Overview of XML Languages for Web Service QoS Specification

- Classified based on the main concept:
 1. **SLA:** WSLA, [SahaiEtAl2002] (WSML), SLAng
 2. **Class of service:** WSOL, WS-QoS, DAML-QoS
 3. **General contract:** OWL-S, WS-Agreement
 4. **Extension of UDDI/BPEL:** UDDIe, [McGregor2003]
 5. **Policy:** WS-Policy
 6. **Manageability capability:** WSDM
- There are some other languages and formats, but these are probably best representatives

Web Service Level Agreement Language (WSLA) - Overview

from IBM Research: H. Ludwig, A. Keller, A. Dan, ...

- **QoS language & management infrastructure**
- Compatible with, but not restricted to WSEs
- **Custom-made SLAs** that have 3 parts:
 1. **Parties** (signatory & supporting) and their management operations
 2. **Service definitions** – service objects (e.g., WSDL operations) and their monitored properties (QoS metrics, SLA parameters, schedules, triggers, ...)
 3. **Obligations** – SLOs and action guarantees

Web Service Level Agreement Language (WSLA) - Language Details

- **SLA parameter** - monitored property; contains 1 QoS metric & extra info for exchange of values
- **QoS metric** – defines where & how to measure or calculate; can be reused across SLA parameters
- An **SLO** contains: **evaluated Boolean expression** (limits values of SLA parameters), obliged party, validity periods, evaluation event or schedule
- An **action guarantee** contains: precondition expression, evaluation event or schedule, **action to be taken**, obliged party, execution modality
- Reusability: SLA templates, metric macros, ...

Web Service Level Agreement Language (WSLA) - Strengths and Weaknesses

- **Strengths** (significantly outweigh weaknesses):
 - **Detailed and precise** description of QoS monitoring and control
 - Several **tools** for SLA creation, deployment, and compliance monitoring – were distributed with IBM's Emerging Technologies Toolkit (ETTK)
 - The **most used** WS QoS specification language
- **Weaknesses** (mainly due to custom-made SLAs):
 - QoS metrics defined within SLAs
 - Can not be used for QoS-enabled WS selection
 - High overhead of supporting custom-made SLAs

Web Service Offerings Language (WSOL) - Overview

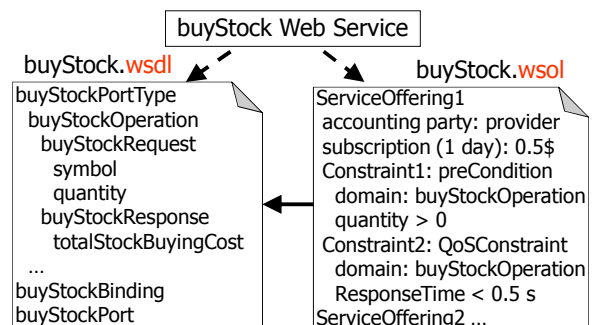
from Carleton Univ.: V. Tosic, K. Patel, B. Pagurek, ...

- plus **Web Service Offerings Infrastructure (WSOI)**
- **Weakness**: parser available, but not a complete compiler generating monitoring code
- **Classes of service** and their **relationships**
- **Multiple categories of constraint**: functional, QoS (including periodic), access rights, context
 - Also: various prices and monetary penalties
- QoS metrics defined in external ontologies
- Its goal was relatively low run-time overhead

Web Service Offerings Language (WSOL) - Language Constructs

- **Service Offering (SO)** - for a class of service
 - **constraints** - expressions (Boolean, arithmetic, ...)
 - **statements** (prices, penalties, management responsibility, disconnection handling)
 - **reusability constructs** (extension, constraint groups, inclusion, instantiation of constraint group templates, ...)
- **Service Offerings Dynamic Relationships (SODRs)** – if a set of constraints from current SO was not met, what is an appropriate replacement SO

Web Service Offerings Language (WSOL) - An Example



Web Service Offerings Language (WSOL) - A Syntax Example

```
<wsol:serviceOffering name="SO1" service=
  "buyStock:buyStockService" accountingParty="WSOL-
  SUPPLIERWS">
  ...
  <wsol:instantiate CGTName="CGT2" resService="..."
    resPortOrPortType="..." resOperation="..."
    resCGName="CG5">
    <wsol:parmValue name="maxResTime">
      <wsol:numberWithUnitConstant>
        <wsol:value>30</wsol:value>
        <wsol:unit type="QoSMeasOntology:millisecond"/>
      </wsol:numberWithUnitConstant>
    </wsol:parmValue>
  </wsol:instantiate>
</wsol:serviceOffering>
```

Web Ontology Language (OWL) - Services (OWL-S)

from the Semantic Web community

- Early version was called DAML-S
- Early version of OWL was called DAML, later DAML+OIL
- Intended for **WS selection**, not QoS monitoring
- **“Service profile”** contains comprehensive description of WSEs, including some QoS, functional constraints, prices, ...
- Can be viewed as a general contract
- Weaknesses: QoS specification is **very weak** – not nearly enough for monitoring activities

Web Services Policy Framework (WS-Policy)

from BEA/IBM/Microsoft/SAP – industry support!

- **General**, flexible and extensible, **framework** for specification of (security) policies for WSEs
- Many good features (e.g., policies can be in or out of WSDL files, some reusability constructs)
- **QoS extension is possible, but missing! To add:**
 - Precise and detailed QoS specification
 - Contracts/SLAs/classes of service and their static and dynamic relationships
 - Standardized expression mechanism

WS-Agreement

from Global Grid Forum (GGF); industry support (IBM,...)

- **General framework** for XML specification of agreements and agreement templates
 - plus a **simple agreement negotiation protocol and run-time agreement monitoring interface**
- Intended for **multiple domains**, not only WSEs
- WS-Agreement allows use of **any language** for the actual contained specifications (including QoS), expressions, QoS metrics, ...
- This **flexibility** can produce **incompatibility**

WS-Agreement - Agreement Template Structure

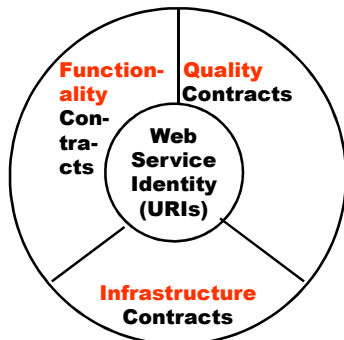
- **Name**
- **Context:** Involved parties (initiator & provider); Expiration time; Template name, Related agreements
- **Terms:** Term compositors ExactlyOne/OneOrMore/All
 - **Service description terms:** Service descriptions, Service references, Service properties
 - **Guarantee terms**, each can have: Service scope, Qualifying condition, **Service level objective (SLO)**, Business value list (Importance, Penalty, Reward, Preference, ...)
- **Creation constraints:** item requirements and/or constraints (in some language)

Web Services Distributed Management (WSDM)

from OASIS Web Services Distributed Management TC

- One of the used inputs: WS-Manageability
- **Two sets of publications:**
 - Management Using Web Services (MUWS) - manageability interfaces of resources exposed as Wses
 - **Manageability capability** – set of operations, properties, events, metadata, and other info. Example: “metrics”
 - Management Of Web Services (MOWS) – WSEs managed as resources and described with MUWS
 - Defines basic metrics, e.g., LastResponseTime
- **Strengths:** Standard management interfaces
- **Weaknesses:** Very limited QoS specification and management support; no specification of guarantees

The "FQI" Classification of (Technical) Contracts for Web Services

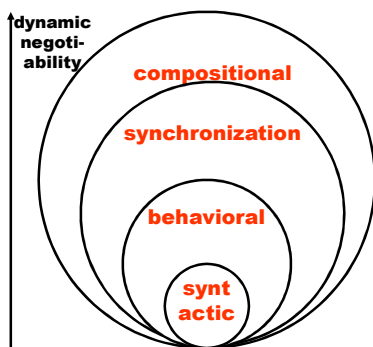


Reference:
Tosic &
Pagurek,
in Proc. of
EEE-05

Functionality Contracts

- Describe **WHAT** a Web Service does
- **Syntactic contracts**: interface names, operation names, parameter names and data types, exceptions, attributes
- **Behavioral contracts**: requirements for and guarantees of correct execution (pre- & post-conditions, invariants; ontological definitions)
- **Synchronization contracts**: dependencies on the order of sent messages (e.g., sequence)
- **Compositional contracts**: flow of messages between composed Web Services

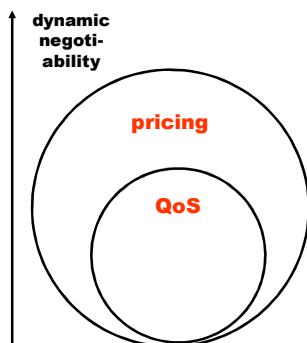
Type of Functionality Contracts



Quality Contracts

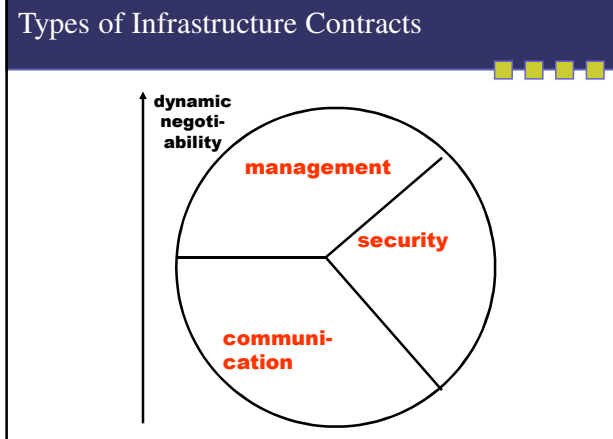
- Describe **HOW WELL** a Web Service performs
- **Enable monitoring & control** of performance, availability/reliability, price/performance ratio
- **QoS (Quality of Service) contracts**: QoS constraints or Service Level Objectives (SLOs)
 - Quality of Web Service (QoWS) - e.g., response time of an on-line operation
 - Quality of Business Service (QoBS) - e.g., delivery time of ordered goods
- **Pricing contracts**: prices (or penalties) for meeting (not meeting) other contracts

Types of Quality Contracts



Infrastructure Contracts

- Describe **BY WHAT MEANS** (protocols, services, entities) a Web Service collaborates
- **Communication contracts**: used communication protocols (e.g., SOAP)
- **Security contracts**: used security and privacy technologies, security and privacy policies
- **Management contracts**: what entities perform monitoring and control, actions to perform in certain situations (e.g., when guarantees were not met)
 - **Contracts have to monitored and enforced!**



Analysis of Some Languages for Description of Web Services

Contract Category	Contract Type	Identity	Functionality				Quality		Infrastructure		
			Syntactic	Behavioral	Synchronization	Compositional	QoS	Pricing	Communication	Security	Management
Language											
WSDL		+	+								+
BPEL4WS					+	+					
WS-CDL					+	+					
WS-Policy				+							+
WSLA							+	+			+
WSOL				+			+	+			+
OWL-S			+	+	+	+					

- ### Existing Web Service Languages vs. "FQI" Contract Types
- At least one language for each contract type
 - No language for all contract types
 - Unfortunately: **compatibility problems!!!**
 - Same or similar concepts defined in different ways
 - Example: WS-Policy policy assertion & WSLA Service Level Objective (SLO)
 - No widely accepted standardization initiative for QoS and pricing contracts
 - **Consequences: no easy combination of languages for all contract types!!!**, reduced usability, contract relationships not captured

- ### What's Next?
- I. Introduction: Web services and their importance
 - II. Overview of the basic Web service (WS) technologies: WSDL, SOAP, UDDI, BPEL4WS
 - III. Importance of comprehensive description and management for Web services
 - IV. Approaches and languages for comprehensive description for Web services
 - V. [Security and privacy technologies for WSEs](#)
 - VI. Approaches and tools for management of WSEs
 - VII. Summary: Past results and some open issues

Role Based Access Control (RBAC): An Illustration

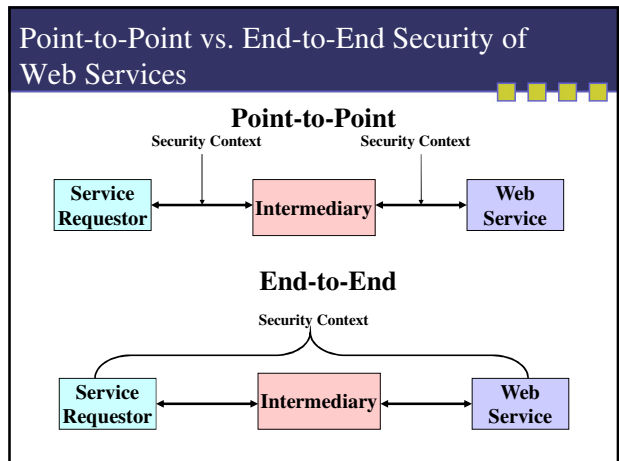
The Need for Security for Web Services

Web services architectures are built on an insecure, unmonitored and shared environment.

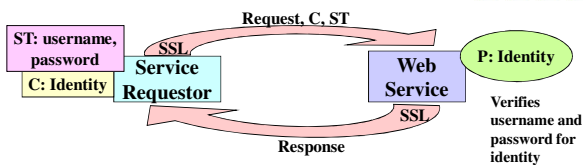
Possible security threats to Web service may include:

- unauthorized disclosure, modification and destruction of information
- unauthorized utilization and misuse of Web services
- interruption, unknown status and repudiation in execution
- denial of service from Web service providers
- corruption of Web services
- more...

Security concerns often cited as a reason for slow adoption of Web service technologies

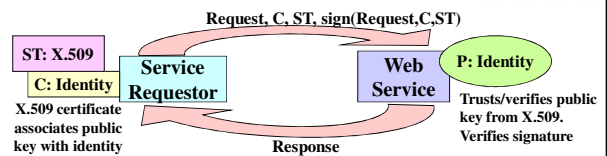


Approaches to Security of Web Services



- Using existing security technologies
- Assumes service requestor and Web service previously established username and password for identity
- SSL/TLS: Secrecy, data integrity and data origin authentication
- Security Token (ST): Entity authentication

Approaches to Security of Web Services (cont.)



- Using signed Security Tokens
- Assume Web service trusts ST or issuer/signer
- Signature: Data origin authentication, data integrity and non-repudiation
- Security Token, Signature: Entity authentication

WS-Security - Overview

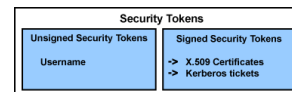
WS-Security describes enhancements to SOAP messaging to provide quality of protection through message integrity, message confidentiality and *single message authentication* that may assume an established session, security context and/or policy agreement.

WS-Security is designed to transmit security data from one application to another application in a SOAP header.

Ref: OASIS Web Services Security Technical Community.
http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss

WS-Security - Overview (cont.)

WS-Security identifies security tokens (i.e., support multiple formats).



Even a client might provide proof of identity and proof that they have a particular business certification.

These mechanisms by themselves do not provide a complete security solution.

WS-Security - Requirements and Security Models

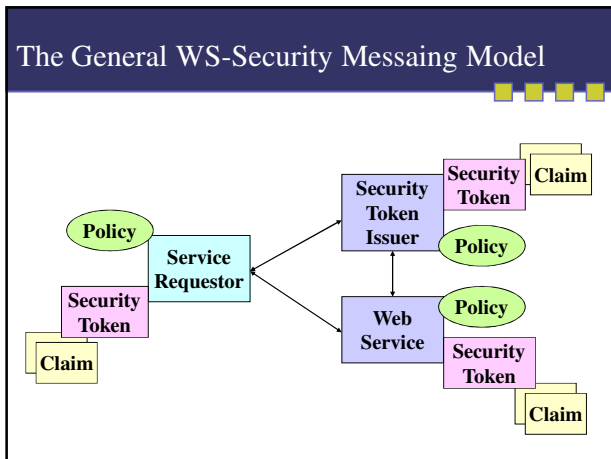
The following list identifies the key driving requirements and security models for WS-Security:

- Multiple security tokens for authentication or authorization
- Multiple trust domains
- Multiple signature formats
- Multiple encryption technologies
- End-to-end message-level security and not just transport-level security

WS-Security - Non-Requirements

The following topics are outside the scope of WS-Security:

- Establishing a security context or authentication mechanisms that require multiple exchanges
- Key derivation, exchange and derived keys
- How trust is established or determined
- Advertisement and exchange of security policy
- Non-repudiation



WS-Security - Details

Policy - The claims and other information required by the Web Service in order to process a message (e.g., requestor has permission to request service).

Claim - A claim is a statement that a client makes (e.g. name, identity, key, group, privilege, capability, etc).

Security Token - A security token represents a collection of claims.

Security Token Issuer

- Security Token: Corroboration/proof of the required claims
- Trust brokers between different trust domains

Security Assertions Markup Language (SAML) - Overview

Single Sign-On (SSO): To log on to a system once and then access different Web services applications without requiring additional logons.

Security Assertions Markup Language (SAML) is a specification that defines a standard way to represent authentication, attribute, and authorization information that may be used in a distributed environment by disparate applications.

The representation of security data is an assertion by a trusted third-party security service.

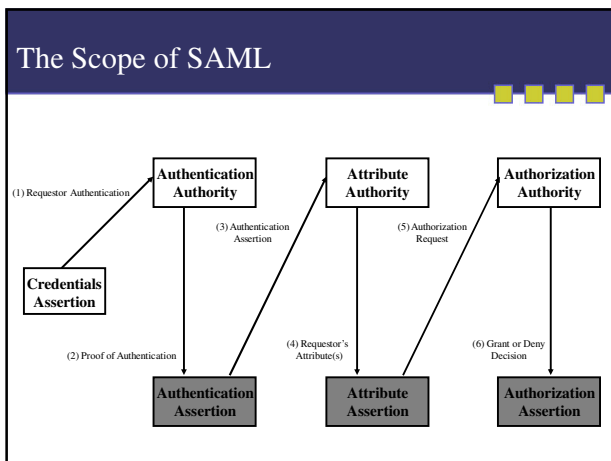
Security Assertions Markup Language (SAML) - Overview (cont.)

The third-party services will act as trusted source to assert the correctness of a particular of a particular authentication, authorization, or attribute activity.

Web services providers submit SAML tokens to security servers for making security decisions.

SAML assertions need a protocol to be transmitted from one application to another in some secure way.

Ref: OASIS. 2002. SAML 1.0 Specification Set: Committee Specifications. http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security



XML Signature

Message integrity is provided by leveraging **XML Signature** in conjunction with security tokens to ensure that messages are transmitted without modifications.

The integrity mechanisms are designed to support multiple signatures, potentially by multiple actors, and to be extensible to support additional signature formats.

Ref: XML Signature Working Group. <http://www.w3.org/Signature/>

WS-Security - Example

```

<?xml version="1.0" encoding="utf-8" ?>
<S:Envelope xmlns:S="http://www.w3.org/2001/12/soap-envelope"
  xmlns:wssse="http://schemas.xmlsoap.org/ws/2002/04/secext/" xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
  xmlns:wssu="http://schemas.xmlsoap.org/ws/2002/04/utility">
  <S:Header>
    <wssse:Security>
      <wssse:UsernameToken Id="MyID">
        <wssse:Username>TravelAgencyABC</wssse:Username>
        </wssse:UsernameToken>
      </wssse:Security>
    </S:Header>
    <S:Body Id="MsgBody">
      <ds:SignatureMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
      <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#hmac-sha1" />
      <ds:Reference URI="#MsgBody">
        <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
        <ds:DigestValue>LylSF0PI4wPU...</ds:DigestValue>
      </ds:Reference>
      </ds:SignatureInfo>
      <ds:SignatureValues>D3bchm5gK...</ds:SignatureValues>
      <ds:KeyInfo>
        <wssse:SecurityTokenReference>
          <wssse:Reference URI="#MyID" />
        </wssse:SecurityTokenReference>
      </ds:KeyInfo>
      </ds:Signature>
    </S:Body>
  </S:Envelope>
  </S:Header>
  <S:Body Id="MsgBody">
    <getSchedules xmlns="http://www.airlinecompany.com/services/AirLineWS">
      <origin>HKG</origin>
      <destination>BOS</destination>
      <day_of_travel>25</day_of_travel>
      <month_of_travel>march</month_of_travel>
      </getSchedules>
    </S:Body>
  </S:Envelope>

```

WS-Security - Example (cont.)

<Security>: It contains security information for an intended receiver.

```

</wssse:Security>
</S:Header>
<S:Body Id="MsgBody">
  <getSchedules xmlns="http://www.airlinecompany.com/services/AirLineWS">
    <origin>HKG</origin>
    <destination>BOS</destination>
    <day_of_travel>25</day_of_travel>
    <month_of_travel>march</month_of_travel>
    </getSchedules>
  </S:Body>
</S:Envelope>

```

WS-Security - Example (cont.)

```

<wssse:UsernameToken Id="MyID">
  <wssse:Username>TravelAgencyABC</wssse:Username>
</wssse:UsernameToken>

```

<UsernameToken>: It defines username of the client using. Note that here we assume the service knows the password - in other words, it is a shared secret.

WS-Security - Example (cont.)

```

<ds:SignatureInfo>
  <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
  <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#hmac-sha1" />
  <ds:Reference URI="#MsgBody">
    <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
    <ds:DigestValue>LylSF0PI4wPU...</ds:DigestValue>
  </ds:Reference>
</ds:SignatureInfo>

```

<SignedInfo>: They describe what is being signed and the type of canonicalization being used.

<Reference>: It selects the elements that are signed.

WS-Security - Example (cont.)

<SignatureValue>: It specifies the signature value of the canonicalized form of the data that is being signed as defined in the XML Signature specification.

```

<ds:SignatureValues>D3bchm5gK...</ds:SignatureValues>

```

WS-Security - Example (cont.)

<KeyInfo>: It provides information, partial or complete, as to where to find the security token associated with this signature.

<SecurityTokenReference>: They indicate that the security token can be found at (pulled from) the specified URL.

```

<ds:KeyInfo>
  <wssse:SecurityTokenReference>
    <wssse:Reference URI="#MyID" />
  </wssse:SecurityTokenReference>
</ds:KeyInfo>

```

XML Encryption

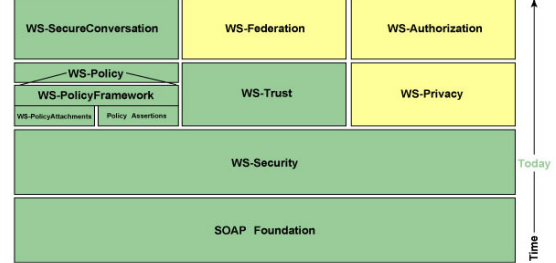
Message confidentiality leverages *XML Encryption* in conjunction with security tokens to keep portions of a SOAP message confidential.

The encryption mechanisms are designed to support additional encryption processes and operations by multiple actors.

Ref: XML Encryption Working Group. <http://www.w3c.org/Encryption/2001/>

A Stack of Envisioned Web Service Security Standards

Figure 1. The evolving WS-Security Roadmap



Adapted from: IBM CORPORATION, 2002. Security in a Web Services World: A Proposed Architecture and Roadmap, White Paper, Version 1.0.
<http://www-106.ibm.com/developerworks/library/ws-secroad/>

WS-Security - Discussion

WS-Security is a building block that can be used in conjunction with other Web service extensions and *higher-level application-specific protocols* to accommodate a wide variety of security models and encryption technologies.

Ref: Web Services Security (WS-Security):
<http://www-106.ibm.com/developerworks/webservices/library/ws-secure/>

WS-Security - Discussion (cont.)

Future standards may allow privacy obligations or restrictions to be added to this data.

Unless such standards are used, the producer must ensure by out-of-band means that the recipient is bound to adhering to all restrictions associated with the data, and the recipient must similarly ensure by out-of-band means that it has the necessary consent for its intended processing of the data.

Ref: Web Services Security (WS-Security):
<http://www-106.ibm.com/developerworks/webservices/library/ws-secure/>

What Is Privacy?

Privacy is a state or condition of limited access to a person.

Ref: SCHOEMAN, E. D. 1984. Philosophical Dimensions of Privacy: An Anthology. New York, NY, Cambridge Univ. Press.

Information privacy relates to an individual's right to determine how, when, and to what extent information about the self will be released to another person or to an organization.

Privacy Requirements for Web Services

The Web Services Architecture (WSA) Requirements includes specific **privacy requirements**:

- **AR020.1** the WSA must enable privacy policy statements to be expressed about Web services.
- **AR020.2** advertised Web service privacy policies must be expressed in P3P.
- **AR020.3** the WSA must enable a consumer to access a Web service's advertised privacy policy statement.
- **AR020.5** the WSA must enable delegation and propagation of privacy policy.
- **AR020.6** Web Services must not be precluded from supporting interactions where one or more parties of the interaction are anonymous.

Online: www.w3.org/TR/2002/WD-wsa-reqs-20021114

Privacy Access Control Issues

Future standards may allow privacy obligations or restrictions to be added to this data.

Unless such standards are used, the producer must ensure by out-of-band means that the recipient is bound to adhering to all restrictions associated with the data, and the recipient must similarly ensure by out-of-band means that it has the necessary consent for its intended processing of the data.

Ref: Web Services Security (WS-Security):

<http://www-106.ibm.com/developerworks/webservices/library/ws-secure/>

Role Based Access Control (RBAC) - An Illustration

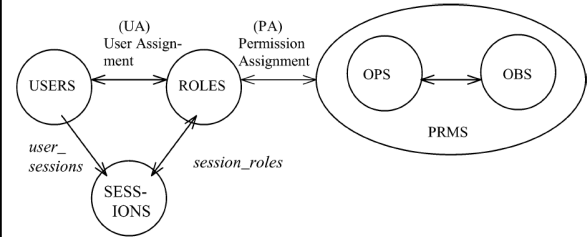
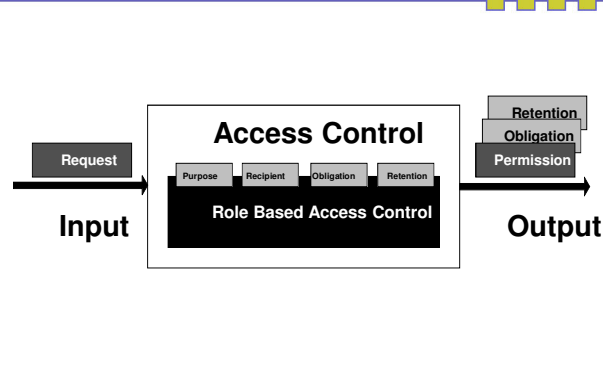


Fig. 1. Core RBAC.

Adapted from: David F. Ferraiolo, Ravi Sandhu, Serban Gavrila, D. Richard Kuhn and Ramaswamy Chandramouli, "Proposed NIST Standard for Role-Based Access Control, ACM Transactions on Information and Systems Security (TISSEC)," Volume 4, Number 3, August 2001.

Role Based Access Control (PBAC) - Another Illustration



Enterprise Privacy Authorization Language (EPAL)

The **Enterprise Privacy Authorization Language (EPAL)** is used to encode an enterprise's privacy-related data-handling policies and practices.

An EPAL policy defines lists of hierarchies of data-categories, data-users, and purposes, and sets of actions, obligations, and conditions.

Online: www.zurich.ibm.com/security/enterprise-privacy/

What's Next?

- I. Introduction: Web services and their importance
- II. Overview of the basic Web service (WS) technologies: WSDL, SOAP, UDDI, BPEL4WS
- III. Importance of comprehensive description and management for Web services
- IV. Approaches and languages for comprehensive description for Web services
- V. Security and privacy technologies for WSEs
- VI. Approaches and tools for management of WSEs
- VII. Summary: Past results and some open issues

Overview of Approaches to (QoS) Web Service Management

- Approaches to **QoS monitoring**
 - SOAP message intermediaries, probes, sniffers
- Approaches to **QoS discovery and selection**
 - Using historical information vs. using contracts
 - Provider as only source of its QoS specifications, UDDI extensions, special QoS information registry
- Some approaches to **QoS control**
 - Using different request queues for different QoS
 - Re-composition of Web services vs. re-negotiation of contracts

Using SOAP Message Intermediaries

- **Exchange of monitored values:** a) in SOAP headers; b) using special push or pull operations
- **Strengths:** Realistic & consumer-specific measures
- **Weaknesses:** High run-time overhead (can be reduced with periodic/occasional monitoring); results depend on network location of measurement

Using Probes

- **Strengths:** Run-time overhead can be lower
- **Weaknesses:** Results not consumer-specific, provider can treat probes in a special way; not possible to use SOAP headers to send monitored values; results depend on network location of probes

Using Sniffers

- **Strengths:** Very low run-time overhead; measures can be realistic & consumer-specific
- **Weaknesses:** Unknown SOAP message's Internet route; WS security technologies can be a problem; not possible to use SOAP headers to send monitored values; results depend on network location of sniffers

Using Historical QoS Information for WS Selection - Possible Approaches

- From the **same consumer**
 - Problem: When consumer did not previously invoke this operation of the provider Web service
- From **probes**
 - Problem: Easy for providers to give excellent QoS to probes, while bad QoS to real consumers
- From **all consumers**
 - Problem: Consumers have different characteristics (e.g., could be located on different continents)
 - Problem: Other consumers' reports can be fake

Using Historical QoS Information for WS Selection - Discussion

- General problem: **Circumstances of different invocations are different!**
 - Example: When the number of provider's concurrent consumers grows, it is likely that QoS perceived by individual consumers will drop
- General problem: Absence of targets/goals to guide control activities (including billing)
- Conclusion: Historical QoS information can be **useful**, but it provides **no guarantees** (and can even be misleading) => **contracts are needed**

Using Provider as the Only Source of Its QoS Specifications

1. Publish: Provider publishes **only its WSDL file** to the registry (e.g., UDDI registry)
2. Find: Consumer searches registry **only based on functionality** and selects one Web service
3. Bind: Consumer and provider **negotiate** QoS (and prices/penalties) that consumer will receive

- **Strengths:** QoS can be dynamic - registry not updated when QoS changes; QoS can be customer-specific; no need to extend UDDI
- **Weaknesses:** QoS not used for selection of WSEs

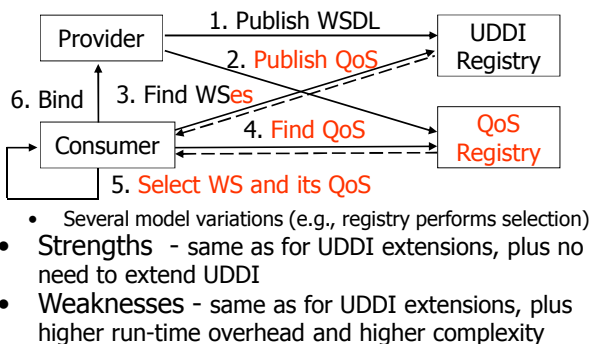
Using UDDI Extensions Describing QoS - Explanation and Some Options

1. Publish: Provider publishes its **WSDL file and QoS specifications** to the extended UDDI registry
 2. Find: Consumer searches registry **based on both functionality and QoS** and selects one WS (and its QoS - if multiple classes of service)
 3. Bind: Consumer binds to the **provider & its QoS**
- Option: Extended UDDI can be a gateway to the real UDDI, which is not changed
 - Option: [Ran2003] suggests a QoS certifier - probe verifying QoS guarantees in the registry

Using UDDI Extensions Describing QoS - Discussion

- **Strengths:** QoS used for selection of Web services; simple selection (not negotiation) of QoS
- **Weaknesses:**
 - QoS changes require updating the registry (most works update only stored QoS specifications)
 - Consumers can have outdated QoS information (unless all are informed about updates, which requires keeping info about all consumer-provider relationships and has high run-time overhead)
 - If no classes of service - no QoS differentiation
- **Issue:** Compatibility with the ordinary UDDI

Using Separate QoS Information Registry



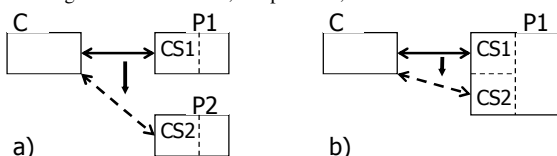
Several Control Approaches That Try to Meet QoS Guarantees

1. **Manipulate which request is processed first**
 - Provider has several different request queues, e.g., one for each class of service
 - Scheduler within the provider decides from which queue to process a request, depending on QoS guarantees, current load, queue lengths, ...
2. **Manipulate thread priorities** for different requests and/or OS scheduling discipline
3. **General approach: Manipulate allocation of resources** for various requests
4. **Load balancing** between replicas

Re-Composition of Web Services vs. Re-Negotiation of Contracts

Run-time adaptation of Web service compositions

- a) **Re-composition of Web services** – more powerful
 - Special case: Switching between WSEs (only 1 change)
 - b) **Re-negotiation of contracts** – faster, simpler, lighter
 - Special case: Switching between classes of service
- Legend: C – consumer; P – provider; CS – class of service



Research and Industrial Tools for Web Service Management

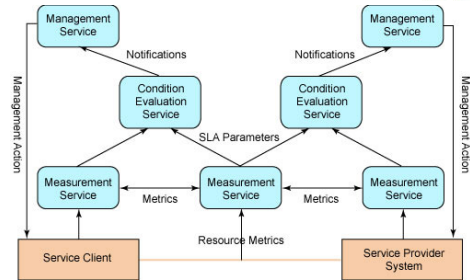
- Published as **research work** in refereed papers
 - **QoS-enabled selection of WSEs:** UDDIe, UX
 - **Contract-based QoS management:** WSLA, WSMF ([SahaiEtAl2002]), WSOI (WSOL), WS-QoS, Cremona (WS-Agreement)
 - Other: Smartware, ... and many others
- **Commercial products** (non-refereed literature)
 - Web service QoS (performance) and/or security management products from specialized companies
 - Large systems management suites
 - Related management products

Web Service Level Agreement (WSLA) Framework - Overview

from IBM Research; uses the WSLA language

- **Modules = services:** 1) Establishment; 2) Deployment; 3) Measurement; 4) Condition Evaluation; 5) Management; 6) Business Entity
 - Prototype: **SLA Compliance Monitor** – module 1 is simple, 2 is implemented, 3 & 4 are general purpose, 5 & 6 missing
 - Service Deployment Information (SDI) is a subset of WSLA
 - Special management port types (e.g., for value exchange)
- **Strengths:** The most widely used WS QoS research infrastructure; comprehensive approach to QoS management; support for management third parties
- **Weaknesses:** Run-time overhead

Web Service Level Agreement (WSLA) Framework - Run-Time Use



Reference/source: A. Keller, H. Ludwig: The WSLA Framework – Specifying and Monitoring Service Level Agreements for Web Services. *Journal of Network and System Management* (11), Nr. 1, Special Issue on E-Business Management. Plenum Publishing Corporation, March 2003

Web Service Offerings Infrastructure (WSOI)

from Carleton Univ.; uses the WSOL language

- Extends open-source **Apache Axis** SOAP engine, run over Apache Tomcat application server
- **Monitoring** of WSOL service offerings
- **Dynamic manipulation** of service offerings
- **Strengths:** Unique management support for WS classes of service, relatively low run-time overhead
- **Weaknesses:** Limited control activities to meet QoS guarantees, not all modules implemented in the prototype, incomplete documentation

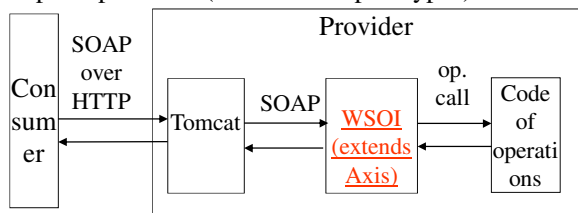
Web Service Offerings Infrastructure (WSOI) - Main Modules

- **Modules for monitoring:**
 - Standard Axis modules (including handlers and chains)
 - **WSOI-specific handlers and chains** (QoS monitoring)
 - WSODEngine modules (for ordering of WSOI handlers)
 - Timer (for periodic activities)
- **Modules for manipulation:**
 - **SOMgmtDecisions** (code of management algorithms)
- **Modules for both monitoring & manipulation:**
 - Modules for **Service Offering Management (SOM)** port types (used in management protocols)
 - Data structures (service offering descriptions, consumer info, accounted management info within sessions, ...)

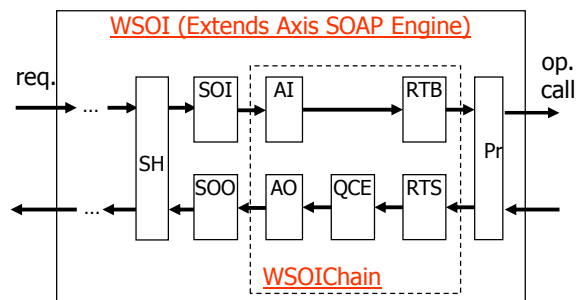
WSOI Inside a Provider Web Service

– Can also be used in consumers and SOAP intermediaries

- Exchange of management (incl. QoS) information: in SOAP headers or using special built-in push or pull operations (one of SOM port types)



Example of Using WSOI: WSOI-Specific Handlers and Chains

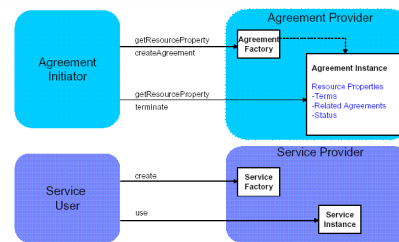


CREation and MONitoring of Agreements (Cremona) - Overview

from IBM Research; uses WS-Agreement

- **Architecture** for WS-Agreement middleware
 - Agreement initiator & agreement provider roles
 - Agreement Management layers: a) ... Protocol Role ... (APRM), b) ... Service Role ... (ASRM), c) Strategic ... (SAM)
- **Java library** that: 1) implements WS-Agreement interfaces; 2) provides management functionality for agreement templates and instances; 3) defines abstractions to be implemented in domain-specific environments
- **Strengths:** Relates agreements with underlying resources; reusable for various domains
- **Weaknesses:** Needs additions to be used for WSEs

CREation and MONitoring of Agreements (Cremona) - Agreement Roles



Agreement roles independent of service roles

Reference/source: H. Ludwig, A. Dan, B. Kearney: Cremona: An Architecture and Library for Creation and Monitoring of WS-Agreements. In: M. Aiello, M. Aoyama, F. Curbera, M. Papazoglou: Proceedings of the 2nd International Conference on Service Oriented Computing (ICSOC 2004), pp. 65 - 74. ACM Press. New York, 2004

Smartware

from Infosys: A. Sharma, H. Adarkar, S. Sengupta

- **QoS control:** Differentiated scheduling of requests based on context priorities
 - Context = info about provider application, user, and client device; sent by consumer in request SOAP header
- Based on **Apache Axis SOAP engine**, adds:
 - Interceptor – reads context info and determines priority
 - Scheduler – puts request into a queue for its priority; based on scheduling policy fetches a request from a queue
 - Dispatcher – forwards request to the provider
- **Strengths:** Rare work that performs QoS control
- **Weaknesses:** Scheduling uses limited information

Some Observations about Industrial Products for Web Service Management

- They **address many practical problems**
 - Academic researchers should be aware of these works and their accomplishments
 - Some works contain advanced solutions that show how SLAs and policies can be used in practice
- Many products have significant **limitations:**
 - Crucial role of human administrators (i.e., not completely automated)
 - Limited/predefined choice of used QoS metrics
 - Lack of formal machine-understandable QoS specification (instead, forms are used)

Some Products for Web Service QoS Management from Specialized Companies

- Often products (1 or more) addressing several management areas, including performance (QoS)
- **Actional SOA Command and Control** (including SOAPStation Web Services Broker) – policies
- **AmberPoint** (including Service Level Manager) – custom-made SLAs
- **Blue Titan** (including Network Director) – policies
- **Infravio Ensemble** (including X-broker) – custom-made Web Services Delivery Contracts (extend SLAs)
- **WestGlobal mScope** (including Performance Management Module - PMM) – custom-made SLAs

Some Products for Web Service Security Management from Specialized Companies

- Some of them are based on hardware appliances
- **Flamenco Networks** – proxy-based solution (can act as a SOAP firewall); centralized Controller uses policies
- **Netegrity TransactionMinder** – SOAP messages intercepted and analyzed by TransactionMinder XML Agent; control by central Policy Server
- **Reactivity XML Firewall** – hardware appliance that acts as a gateway and performs policy-based security management; Web-based management console
- **Vordel** – VordelSecure is a software and/or hardware XML firewall and filters XML traffic; VordelDirector is used for centralized policy-based management

Large Systems Management Suites

- **Contain many different management products**
 - some related to WSeS (or “business services”)
 - some related to QoS or security management of applications, computing systems, networks
- 1. HP: **OpenView** (includes **SOA Manager**)
- 2. IBM: **Tivoli** (includes Business Systems Manager)
- 3. Computer Associates (CA): **Unicenter** (includes **Web Services Distributed Management – WSDM**)
- 4. BMC Software: **Patrol** (includes MAINVIEW)
- 5. Microsoft (includes Application Center)

Some Related Management Products

- Web service infrastructures, such as:
 - **Grand Central Communications** Business Services Network
 - **IONA Artix** (security and also very limited support for QoS)
 - **webMethods** Enterprise Services Platform
- Application performance tools, such as:
 - Many tools use Java Management Extensions (JMX)
 - **Mercury**: Service Level Management, ...
 - **OPNET** (Altaworks): Commander, Panorama, ...
 - **Quest Software**: PerformaSure, Foglight, ...
- Application security tools, such as:
 - **Entegrity Solutions**: AssureAccess, ...

What's Next?

- I. Introduction: Web services and their importance
- II. Overview of the basic Web service (WS) technologies: WSDL, SOAP, UDDI, BPEL4WS
- III. Importance of comprehensive description and management for Web services
- IV. Approaches and languages for comprehensive description for Web services
- V. Security and privacy technologies for WSeS
- VI. Approaches and tools for management of WSeS
- VII. **Summary: Past results and some open issues**

Summary of Properties of Web Services

In summary, the properties of Web Services are:

- **Loosely-coupled**: Web services can run independently of each other on entirely different implementation platforms and run-time environments.
- **Encapsulated**: The only visible part of a Web service is the public interface, e.g., WSDL and SOAP.
- **Standard Protocols and Data Formats**: The interfaces are based on a set of standards, e.g., XML, UDDI, WSDL, SOAP, and etc.

Summary of Properties of Web Services (cont.)

- **Invoked Over Intranet or Internet**: Web services can be executed within or outside the firewall.
- **Components**: The composition of Web services can enable business-to-business transactions or connect the internal systems of separate companies, such as workflow.
- **Ontology**: Everyone must understand the functionality behind how data values are computed.
- **Business Oriented**: Web services are not end-user software! (Although their could be used for interaction with end users)

Summary of Comprehensive Description and Management for Web Services

- QoS, price, security, trust, manageability ... will be **differentiators** in the WS market
 - Determine WS to be chosen among similar ones
- Their **description is the basis for management**
- **Management (monitoring & control) necessary** to meet QoS guarantees, discover & fix problems, accommodate change, provide security/privacy, ...
- The vision of **service-oriented computing can not be achieved without management**
 - But, the basic Web service technologies do not address QoS specification and management

Some Recommendations Related to the Use of Web Services

- Do not wait until everything is finalized. Start now with SOAP and WSDL. Track other standards as appropriate
- Do not let security concerns hold up Web services development... SSL and SAML will suffice for now
- Participate in standards development activities where you have a vested interest in the outcome or to stay abreast of developments
- Do not expect Web services to transform your business; use Web services to transform your processes

Ref: Larry Perlestein, "Web Services Standards: De Facto, De Jure or Defunct?" U.S. Symposium/ITxpo, 2002.

The Main Results on Comprehensive Description and Management of Web Services (beyond the Basics)

Several ...

- ... languages for precise and detailed specification of SLAs or classes of service for Web services
- ... languages in the security and privacy domain
- ... general frameworks that can be extended for QoS, price and/or security specification
- ... industrial standards under development
- ... prototyped solutions for QoS-enabled Web service discovery/selection
- ... research infrastructures that enable Web service QoS management (predominantly monitoring)
- ... industrial management products used in practice

Some Open Topics

- **Standards for Web service QoS and security/privacy specification**
 - Contracts vs. policies; Extending WS-Policy, OWL-S, ...; Equal status of guarantees and requirements; ...
- **Automatic negotiation** of contracts
- **Standards for Web service management interfaces**
 - WSDM, WS-Agreement, ...
- **Control of Web services** to meet contract guarantees
 - Resource capacity planning and management; Building complex control plans; Trust/reputation management; ...
- **Integrated management** of business operations, Web services, and underlying computing & communications infrastructure
 - Standard models and mappings between them
- Solutions for **adaptation to various changes**

Resources

- **Standardization bodies:** W3C, OASIS, WS-I
- **Publications** are scattered between many different conferences, journals, and books
 - Intl. Web Services Quality Workshop (WQW) at WISE (Web Information Systems Engineering) 2003, 2004
 - W3C Workshop on Constraints and Capabilities for Web Services, Oct. 2004 – input into future standardization
 - Annotated bibliography on QoS for WSEs available at: <http://elab.njit.edu/vladimir/QoS4WS.html>
- **Ask the tutorial presenters** (e-mail on Slide 1)
- **Hire one of the tutorial presenters!** (Knowledge, experience, hard work, dedication, passion, ...)