Modeling Web Services with UML

OMG Web Services Workshop 2002

Chris Armstrong
ATC Enterprises, Inc.
1751 West County Road B, Suite 310
Roseville, MN 55113
651.633.1818
www.atcenterprises.com
Agenda

- What are Web Services?
- What is SOAP?
- What is UDDI?
- What is WSDL?
- What does all of this have to do with each other?
- UML modeling for SOAP, WSDL, and UDDI
What is a Web Service?

- Represents a specific business function
- Exposed by a company
  - Usually through an Internet connection
- Provided to another company or software application to consume

“Web services are becoming the programmatic backbone for electronic commerce…”

UDDI Technical White Paper

uddi.org
What are Web Service Applications?

- Hyper-applications that are:
  - Designed
  - Assembled
  - Executed
dynamically at run-time using web services
  - Especially in a B2B environment

- Use emerging industry standards:
  - Extensible Markup Language (XML)
  - Simple Object Access Protocol (SOAP)
  - Web Services Description Language (WSDL)
  - Universal Description, Discovery, and Integration (UDDI)
Web Services Business Model

- Service Provider
  - Publish Web Service
  - Register Web Service
  - Find Web Service

- Service Requester
  - Find Web Service
  - Consume Web Service

- Service Broker
Fundamental Web Service Concepts

<table>
<thead>
<tr>
<th>Service</th>
<th>Standard</th>
</tr>
</thead>
<tbody>
<tr>
<td>Publish</td>
<td>UDDI</td>
</tr>
<tr>
<td>Find</td>
<td>UDDI, WSDL, DISCO</td>
</tr>
<tr>
<td>Bind</td>
<td>WSDL, SOAP</td>
</tr>
</tbody>
</table>
Realization of Web Services Business Model

- Publish Web Service
- Register Web Service
- Find Web Service
- Consume Web Service

This is not complete – only shows relationships to industry standards.

- UDDI
- WSDL
- SOAP
Web Services Architectural Mechanisms

- Web Discovery Services
- Web Description Services
- Web Wire Services
Web Wire Services

<<analysis mechanism>>
Inter Process Communication

<<analysis mechanism>>
Security

<<analysis mechanism>>
Reliability

<<analysis mechanism>>
Routing
SOAP Wire Services

SOAP

HTTP

XML
Modeling Web Services with UML

Web Description Services

<<analysis mechanism>>
Workflow Automation

<<design mechanism>>
Business Process Orchestration

<<implementation mechanism>>
BizTalk

<<analysis mechanism>>
Message Sequencing

<<design mechanism>>
Service Capability Configuration

<<implementation mechanism>>
WSFL

<<analysis mechanism>>
Application Integration

<<design mechanism>>
Service Description

<<implementation mechanism>>
UDDI

<<design mechanism>>
Service Description

<<implementation mechanism>>
WSDL
Web Discovery Services

<<analysis mechanism>>
Discovery Services

<<design mechanism>>
Directory Service

<<design mechanism>>
Inspection Service

<<implementation mechanism>>
UDDI
What is SOAP?

- Simple Object Access Protocol (SOAP)
  - Standard for invoking services across the web
  - Uses HTTP for transport
  - Uses XML for data encoding
  - Extensible
Benefits of SOAP

- Can invoke components residing in many architectures
  - DCOM
  - CORBA
  - EJB
  - Perl

⇒ Heterogeneous “glue”
- Cross-platform, cross-architecture integration
Benefits of SOAP

- Uses industry standards
  - HTTP
    - Works through firewalls!
  - XML
- Platform-independent
  - Can be invoked from any platform
    - Browser
    - Desktop application
    - Server component
Web Application Extensions (WAE) for UML

Applet

Application

Client Page

COM Object

EJB Object

Form

Server Page

Session
Standard HTTP Form POST

Form

Client Page

<<submit>>

<<build>>

Client Page Renderer
SOAP Request from Browser - Structure

ISOAPProxy
- newRequest() : SOAP Request
- postRequest(SOAP Request) : SOAP Response
+ invokeMethod()

XML Parser

SOAP Request (from SOAP)

SOAP Response (from SOAP)

Client Page
- <<submit>>
- <<build>>

Client Page Renderer

SOAP Request (from SOAP)

Client Page

Form

<<interface>>
SOAP Request from Browser - Behavior

1: invokeMethod() → 2: newRequest() → 3: new

4: encode → 5: postRequest(SOAP Request) → 6: receive(SOAP Request)
SOAP Request on Server - Structure

<<interface>>
ISOAPTranslator

+ receiveRequest(SOAP Request)
- sendResponse() : SOAP Response
- locateComponent()
- buildResponse()

SOAP Response
(from SOAP)

<<interface>>
IComponentInterface
(from SOAP)

+ method()

SOAP Request
(from SOAP)

XML Parser

+ decode()
+ encode()
SOAP Request on Server - Behavior

ISOAPProxy

ISOAPTranslator

XML Parser

IComponentInterface

1: receiveRequest(SOAP Request)
2: decode()
3: locateComponent()
4: method()
5: buildResponse()
6: sendResponse()
SOAP Process View

- Client Process
- HTTP Server
- SOAP Server
- EJB Server
- CORBA Server
- COM Server
SOAP Deployment View

Client Machine
Browser Process

Firewall

Web Server
Web Server Process
Translator Process

Application Server
Component Process
What is WSDL?

- Web Services Description Language (WSDL)
- Way to describe web services
  - Where do they reside locally
  - What are they called
  - How are external operations bound to internal operations
  - How data is encoded
  - How operation is invoked
- WSDL is similar to an interface in a component architecture, but also includes information on
  - Protocol bindings
  - Deployment
What is WSDL?

- **History**
  - Started as Service Description Language (SDL)
    - Part of early SOAP efforts
  - Turned into Service Contract Language (SCL)
  - Currently called Web Services Description Language (WSDL)
- **In Apache SOAP, use XML Deployment Descriptors**
  - Register with ServiceManagerClient in org.apache.soap.server package
- **WSDL extensions defined**
  - HTTP GET
  - HTTP POST
  - SOAP
WSDL Generation

- Development tool vendors provide WSDL generation tools
  - Microsoft VisualStudio 6.0
    - Uses Web Services Meta Language (WSML) for COM binding on server side
  - Microsoft VisualStudio.NET
  - IBM Web Services Tool Kit (WSTK)
- Also provide tools to generate client-side proxies
WSDL Schema – High-Level

```
WSDL schema
documentation

<table>
<thead>
<tr>
<th>definitions</th>
</tr>
</thead>
<tbody>
<tr>
<td>targetNamespace</td>
</tr>
<tr>
<td>0..1</td>
</tr>
</tbody>
</table>

| message |
| name |
| documentation |
| 0..n |

| types |
| documentation |
| 0..1 |

| binding |
| name |
| 0..n |

| portType |
| name |
| 0..n |
```
WSDL Schema – Types

- **types documentation**
- **schema**
  - attributeForm Default
  - elementForm Default
  - targetNamespace
- **element**
  - name
  - minOccurs
  - maxOccurs
  - nillable
- **sequence**
  - 0..1
- **complexType**
  - +type
  - 0..1
  - 0..n
WSDL Schema – Messages

message

name
documentation

part

name

0..n

element

name
minOccurs
maxOccurs
nillable

complexType

+type

0..1

0..1
UML for WSDL

- Visually model key portions of WSDL definition
  - Other details can be modeled using tagged values
- Use one class for overall service description
  - Use <<wsdl service>> stereotype
  - For each element, use <<element>> stereotype on attributes
    - For elements that are complexTypes, model as separate class with <<element>> stereotype
  - For each message, use <<message>> stereotype on operations
**UML for WSDL**

- Use one class for each portType
  - Use <<wsdl portType>> stereotype
  - Attach to service with aggregation
  - For each operation, use an operation

- Use one class for each binding
  - Use <<wsdl binding>> stereotype
  - Attach to service with aggregation with <<port>> stereotype
  - For each operation, use an operation
Sample UML for WSDL – High-Level

<<wsdl service>>
ZIPCodeResolver

<<wsdl portType>>
ZIPCodeResolverHTTPGet

<<wsdl portType>>
ZIPCodeResolverHTTPPost

<<wsdl portType>>
ZIPCodeResolverSOAP

<<wsdl binding>>
ZIPCodeResolverHTTPGet

<<wsdl binding>>
ZIPCodeResolverSOAP

<<wsdl binding>>
ZIPCodeResolverHTTPPost

<<wsdl binding>>
ZIPCodeResolverHTTPPost
Sample UML for WSDL – Detailed

<<wsdl service>>
ZIPCodeResolver

<<element>> FullZipCode
<<element>> FullZipCodeResponse
<<element>> CorrectedAddressHTML
<<element>> CorrectedAddressHTMLResponse

<<message>> FullZipCodeHttpGetIn()
<<message>> FullZipCodeHttpGetOut()
<<message>> FullZipCodeHttpPostIn()
<<message>> FullZipCodeHttpPostOut()
<<message>> FullZipCodeSoapIn()
<<message>> FullZipCodeSoapOut()

<<port>>

<<wsdl binding>>
ZIPCodeResolverSOAP

FullZipCode()
ShortZipCode()
CorrectedAddressHTML()
CorrectedAddressXML()
VersionInfo()

<<wsdl portType>>
ZIPCodeResolverSOAP

FullZipCode()
ShortZipCode()
CorrectedAddressHTML()
CorrectedAddressXML()
VersionInfo()

<<element>>
USPSAddress

<<element>> Street
<<element>> City
<<element>> State
<<element>> ShortZIP
<<element>> FullZIP
Sample UML for WSDL – Context

<<wsdl service>>
ZIPCodeResolver

<<Interface>>
IZIPCodeResolver

- FullZipCode()
- ShortZipCode()
- CorrectedAddress()
- CorrectedAddress()
- VersionInfo()

<<derive>>

ZIPCodeResolver

ZIPCodeResolver Proxy
What is UDDI?

- Universal Description, Discovery and Integration (UDDI)
- A specification for distributed Web-based information registries of Web services
- Collaborative initiative
  - IBM
  - Microsoft
  - Ariba
- Create a global, platform-independent, open framework to rapidly accelerate the global adoption of B2B e-commerce
  - Enable businesses to discover each other
  - Define how they interact over the Internet
  - Share information in a global registry
What is UDDI?

- Similar to DNS for domain names
  - Except works for web services instead
- Business service registry
  - Logically centralized
  - Physically distributed
- API defined in an XML Schema
UDDI Leaders

- IBM
  - UDDI4J
- Microsoft
  - UDDI for .NET
  - SDK for Visual Studio 6.0
UDDI4J Overview

- Open source implementation of UDDI for Java by IBM
- Provides an API to interact with a UDDI registry
- Generates and parses messages sent to and received from a UDDI server

Requirements
- Apache SOAP 2.1 or later
- JDK 1.2.2 or later
- Ant
UDDI4J Key Components

- **uddi.client** package
  - Especially **UDDIProxy** class
  - Methods map to the UDDI Programmer’s API Specification

- **uddi.datatype** package
  - Contains core UDDI datatypes to send and receive data from a UDDI server

- **uddi.request** package
  - Used internally by UDDIProxy for sending messages to a UDDI server

- **uddi.response** package
  - Used internally by UDDIProxy for receiving messages from a UDDI server
Sample UDDI Usage – Inquire – Behavior

: Get Business Name
: UDDIProxy
: BusinessList
: BusinessInfos
BusinessInfo : Vector
: BusinessInfo

1: UDDIProxy( )
2: setInquiryURL(url : String)
3: find_business(name : String, findQualifiers : FindQualifiers, maxRows : int)
4: getBusinessInfos( )
5: getBusinessInfoVector( )
6: elementAt(arg0 : int)
7: getNameString( )
Sample UDDI Usage – Inquire – Structure

- `UDDIProxy`
  - `find_business()`
  - `setInquiryURL()`

- `Get Business Name`

- `inquiryURL`

- `URL`

- `BusinessList`
  - `getBusinessInfos()`

- `Business Infos`
  - `getBusinessInfoVector()`

- `Vector`
  - `elementAt()`

- `BusinessInfo`
  - `getNameString()`
Apache SOAP Overview – Main Packages

- soap (from apache)
- rpc
- server
- util
- messaging
- providers
- transport
- encoding
Apache SOAP Overview – Main Classes

- Envelope
- Header
- Body
- Fault
- AttributeHandler
- Constants
- Utils
- SOAPException
- attrHandler
- attrHandler
- attrHandler
- attrHandler
- attrHandler
- header
- body
SOAP Usage by UDDI – Part 1

: Get Business Name

1: find_business(name : String, findQualifiers : FindQualifiers, maxRows : int)

: UDDIProxy

2: FindBusiness()

3: setName(s : String)

4: setFindQualifiers(s : FindQualifiers)

5: setMaxRows(s : int)

6: send(el : UDDIElement, inquiry : boolean)
SOAP Usage by UDDI – Part 2

1: send(el : Element, inquiry : boolean)

2: Envelope()

3: Body()

4: Vector()

5: add(arg0 : Object)

Add passed Element
SOAP Usage by UDDI – Part 3

: UDDIProxy

sendBody : Body

soapMessage : Message

response : Element

1: setBodyEntries(bodyEntries : Vector)

2: Message()

3: get transport

4: setSOAPTransport(st : SOAPTransport)

5: Element()
SOAP Usage by UDDI – Part 4

1: send(url : URL, actionURI : String, env : Envelope)
2: receiveEnvelope()
3: getBody()
4: getBodyEntries()
5: firstElement()
SOAP Usage by UDDI – Structure

Get Business Name

FindBusiness

Vector

Message

Message()
receiveEnvelope()
send()

UDDIProxy

find_business()
send()

st

transport

<<Interface>>

SOAPTransport

Element

Element()

Envelope

Envelope()

-body

Body

Body()
getBodyEntries()
setBodyEntries()

SOAPTransport

find_business()
send()
SOAP Message Sending – Part 1

1: send(url : URL, actionURI : String, env : Envelope)

2: SOAPHTTPConnection()

SOAP Message Sending – Part 2

: Message

  st : SOAPHTTPConnection

payloadSW : StringWriter

env : Envelope


  2: StringWriter()

3: marshall(sink : Writer, xjmr : XMLJavaMappingRegistry, ctx : SOAPContext)
SOAP Message Sending – Part 3

```java
st : SOAPHTTPConnection
headers : Hashtable

1: Hashtable()
2: put(arg0 : Object, arg1 : Object)
3: get userName
4: get password
5: put(arg0 : Object, arg1 : Object)
```

Constants.HEADER_SOAP_ACTION
Constants.HEADER_AUTHORIZATION
1: TransportMessage(envelope : String, ctx : SOAPContext, headers : Hashtable)

2: save( )

Web Resources

- **UDDI**
  - www.uddi.org
  - uddi.microsoft.com
  - www.ibm.com/services/uddi

- **SOAP**
  - www.soap.org
  - www.soaprpc.com
  - www.soapware.org
  - www.soapclient.com
  - www.soap-wrc.com

- **Web Services**
  - www.webservices.org
  - www-106.ibm.com/developerworks/webservices
  - www.xmlmodeling.com