WSCI

Chia-Chun Shih, NTUI M

2004/4/28
Outline

• Introduction
• A simplified Example
• Language Overview
• An example
• References
Introduction
Introduction

• WSCI stands for “Web Service Choreography Interface”
• A W3C Note
• Current version: 1.0 (August 2002)
• Co-developed by BEA Systems, Intalio, SAP AG, and Sun Microsystems
The Web Service Choreography Interface (WSCI) is an XML-based interface description language that describes the flow of messages exchanged by a Web Service participating in choreographed interactions with other services.
Introduction (Cont’d)

• It describes how Web Services participate in choreographed, long-lasting and stateful message exchanges.

• It provides the external, observable view of the service behavior (IDL).

• It does not address the definition of the process driving the message exchange or the definition of the internal behavior of each Web Service (IDL).

• Works in conjunction with WSDL.

• It does not assume that Web Services are from different companies: usable for EAI.
Introduction (Cont’d)

- Organize operations into meaningful processes (Choreography)
- Describe the interdependences among operations (Correlation)
- Context-awareness
- Abstract from implementation (IDL)
- Other features, such as transaction processing and exception handling, are included
WSCI: Dynamic, Choreographed Web Service Interface

Activity

Action

Action

References

Operation Type

Operation

Input/Output

Message

WSDL: Static Web Service Interface
A simplified Example
A simplified Example
<?xml version="1.0"?>
<definitions name="Travel Agent Static Interface"
    targetNamespace="http://example.com/consumer/TravelAgent"
    xmlns:xsd="http://www.w3.org/2000/10/XMLSchema"
    xmlns:tns="http://example.com/consumer/TravelAgent"
    xmlns="http://schemas.xmlsoap.org/wsdl/"
>
  <!-- *************************************************************** -->
  <!-- *************************************************************** -->
  <!-- MESSAGE3 ********************************************************* -->
  <!-- *************************************************************** -->
  <!-- *************************************************************** -->
  + <message name="tripOrderRequest">
  + <message name="tripOrderAcknowledgement">
  + <message name="bookingRequest">
  + <message name="bookingConfirmation">
  + <message name="statement">
    <!-- *************************************************************** -->
    <!-- *************************************************************** -->
    <!-- TRAVEL AGENT PORT TYPES *************************************** -->
    <!-- *************************************************************** -->
    <!-- *************************************************************** -->

  <portType name="TATOTraveller">
    <documentation>
      This port type references the operations the Travel Agent performs with the Traveler service</documentation>
    <operation name="OrderTrip">
      <input message="tns:tripOrderRequest"/>
      <output message="tns:tripOrderAcknowledgement"/>
    </operation>
    <operation name="bookTickets">
      <input message="tns:bookingRequest"/>
      <output message="tns:bookingConfirmation"/>
    </operation>
    <operation name="SendStatement">
      <output message="tns:statement"/>
    </operation>
  </portType>
</definitions>
<?xml version="1.0"?>
<wsdl:definitions name="Travel Agent Dynamic Interface"
   targetNamespace="http://example.com/consumer/TravelAgent"
   xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
  xmlns:xsd="http://www.w3.org/2000/10/XMLSchema"
  xmlns:tns="http://example.com/consumer/TravelAgent"
  xmlns="http://www.w3.org/2002/07/wsci10">
  <!-- WSDL complex types -->
  <!-- WSDL message definitions -->
  <!-- WSDL operations and port types -->
  <!-- selectors -->
  <correlation name="itineraryCorrelation" property="tns:itineraryID"/>
  <interface name="TravelAgent">
    <process name="PlanAndBookTrip" instantiation="message">
      <sequence>
        1 <action name="ReceiveTripOrder" role="tns:TravelAgent"
          operation="tns:TAToTraveler/OrderTrip" />
        2 <action name="ReceiveConfirmation" role="tns:TravelAgent"
          operation="tns:TAToTraveler/bookTickets"/>
          <correlate correlation="tns:itineraryCorrelation"/>
        <call process="tns:BookSeats"/>
        </sequence>
    </process>
    <process name="BookSeats" instantiation="other">
      <action name="bookSeats" role="tns:TravelAgent" operation="tns:TAToAirline/bookSeats"/>
    </process>
  </interface>
</wsdl:definitions>
Language Overview
Language Overview

- Interface
- Process
- Action
- Context
- Property & Selector
- Exception
- Transaction
- Global Model

while, foreach, switch, all, sequence, etc.

while, foreach, switch, all, sequence, etc.

while, foreach, switch, all, sequence, etc.
Language Overview (Cont’d)

- **Interface**

  ```xml
  <interface name = NCName>
    Content: (documentation?, process+)
  </interface>
  ```

- **Process**

  ```xml
  <process name = NCName instantiation = message|other :message>
    Content: (documentation?, context?, {any activity}+)
  </process>
  ```

  - Execute in sequential order
  - Can be re-used through
    - Call
    - Spawn (Executed in another thread)
  - Activity includes both process and action
Language Overview (Cont’d)

• **Context**

```
<context>
  Content: ((process | property)*, exception?, transaction?)
</context>
```

- Nested process

• **Action**

```
<action name = NCName operation = QName/NCName role = QName >
  Content: (documentation?, correlate*, call?)
</action>
```

- Use **Choreography Elements** (eg. while, switch, sequence, foreach, etc.) to organize actions into meaningful processes
• Correlate and Correlation

```xml
<correlation name = NCName
    property = list of QName
    extends = QName>

Content: (documentation?)
</correlation>
```

```xml
<correlate correlation = QName
    instantiation = (true|false):false />
```

Language Overview (Cont’d)
Language Overview (Cont’d)

Process instantiation = message

Message 1

Message 1

Message 2

Id: 000
Name: Jennifer
Phone: 25352578

Id: 001
Name: Vivian
Phone: 25364578

TransId: 1234
PersonId: 001
ItemId: 9999

Id: 000
Name: Vivian
Phone: 25364578

Id: 000
Name: Jennifer
Phone: 25352578

Id: 001
Name: Vivian
Phone: 25364578

Process

Action 1

Process

Action 1

Action 2

Process

Action 1

Action 2

Action 3

20
Language Overview (Cont’d)

```xml
<interface name = "Traveler">
  <process name = "PlanAndBookTrip">
    <action name = "bookTickets" role = "tns:Traveler"
         operation = "tns:TravelerToTA/bookTickets">
      <correlate correlation = "defs:bookingCorrelation"
               instantiation = "true"/>
    </action>
    <all>
      <action name = "ReceiveTickets" role = "tns:Traveler"
              operation = "tns:TravelerToAirline/ReceiveTickets">
        <correlate correlation = "defs:bookingCorrelation"/>
      </action>
      <action name = "ReceiveStatement" role = "tns:Traveler"
              operation = "tns:TravelerToTA/ReceiveStatement">
        <correlate correlation = "defs:bookingCorrelation"/>
      </action>
    </all>
  </process>
</interface>
```
Language Overview (Cont’d)

• Property and Selector

```xml
<property name = QName select = XPath>
   Content: (documentation?, value?)
</property>

<value>
   Content: {mixed}
</value>

<selector property = QName
   element = QName
   type = QName
   xpath = expression {extension attribute}>
   Content: (documentation?, {extension element}?)
</selector>
```
Language Overview (Cont’d)

<complexType name="itinerary">
  <sequence>
    <element name="itineraryID" type="xsd1:itineraryID"/>
    <element name="leg" type="xsd1:leg" maxOccurs="unbounded"/>
    <element name="comments" type="string" />
  </sequence>
</complexType>

<complexType name="itineraryID">
  <restriction base="string"/>
</complexType>

<selector property="tns:itineraryNo"
         type="tns:itineraryID"
         xpath="./text()" />

<selector property="tns:itineraryNo"
         element="tns:itinerary"
         xpath="./itineraryID/text()" />
Language Overview (Cont’d)

• **Exception**

```xml
<exception>
    Content: ((onMessage | onTimeout | onFault){+})
</exception>
```

• **Transaction** *(Example)*

```xml
<transaction name = NCName
    type = atomic | open : atomic
    retries = QName>
    Content: (compensation?)
</transaction>
```

```xml
<compensation>
    Content: (documentation?, context?, {any activity}+)  
</compensation>
```
Language Overview (Cont’d)

• Global Model

```
<model name = NCName>
  Content: (documentation?,interface{2,n},connect+)
</model>

<interface ref = QName/>

<connect operations = twoOpName>
  Content: {extension element}? 
</connect>

<connect operations = "tra:TravelerToTA/PlaceItinerary ta:TAtoTraveler/ReceiveTrip"/>
```
Example
An Example

• A Travel Reservation System
• Three parties involved
  - Traveler, Travel agent, and Airline
• Steps
  1. Traveler orders a trip (Traveler → Travel agent)
  2. Travel agent arranges a feasible itinerary (Travel agent ↔ Airline)
  3. Travel agent reports the itinerary to the traveler (Travel agent → Traveler)
  4. Traveler agrees, changes or cancels the itinerary (Traveler → Travel agent)
  5. Travel agent reserves the tickets (Traveler → Travel agent)
  6. Travel agent asks Traveler to confirm (Travel agent → Traveler)
  7. Traveler confirms or cancels the itinerary (Traveler → Travel agent)
  8. Travel agent books the tickets (Travel agent → Airline)
  9. Airline sends the tickets to the Traveler (Airline → Travel agent)
  10. Travel agents send statement to the traveler (Travel agent → Traveler)
Use Case definition
An Example (Cont’d)

• WSCI
  - Travel Agent Interface
  - Traveler Interface
  - Airline Interface
  - Definition of Correlation and Selector
  - Global Model
References

• **WSCI 1.0 Spec**
  ([http://www.w3.org/TR/wsci/](http://www.w3.org/TR/wsci/))

• **WSCI 1.0 FAQ**