INF5120 – Model-Based System Development

Lecture #10: SOA, Web services architecture, XSD, WSDL, BPEL

23 March 2009
Brian Elvesæter, SINTEF ICT

Based on material developed in the ATHENA (IST-507849), COMBINE (IST-1999-20839), INTEROP (IST-500011), MODELWARE (IST-511731) and SHAPE (ICT-2007-216408) research projects.

Outline

- Service-oriented architecture (SOA)
- MDD for SOA & SOA platforms
- Web services architecture
  - UDDI
  - SOAP
  - XSD
  - WSDL
  - BPEL
- References
Service-oriented architecture (SOA)

Different kinds of architectures

Enterprise architecture

Integration architecture

Business architecture

Knowledge architecture

Conceptual architecture

Logical architecture

Functional architecture

Information architecture

Architecture framework

Realisation architecture

Service-oriented architecture

ICT architecture

Web services architecture
Enterprise architecture (EA) is the practice of applying a method for describing a current and/or future structure and behaviour for an organization's processes, information systems, personnel and organizational sub-units, so that they align with the organization's core goals and strategic direction.

- Holistic view of the enterprise and all its important assets.

Service-oriented architecture (SOA) is a paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains. [OASIS 2006]

- Architectural style for designing (technical) systems.

Web services architecture (WSA) intends to provide a common definition for understanding Web services. A Web services architecture involves many layered and interrelated technologies. [W3C 2004]

- A set of enabling Web technologies for implementing software systems.

Role of enterprise architecture

Describing coherence


Basic service-oriented model

- Service provider
  - Provides software applications for specific needs as services.
- Service requester
  - A requester could be a human user/application program/another service accessing the service through a desktop or a wireless browser; it could be an application program.
- Service broker:
  - A service broker provides a searchable repository of service descriptions.
  - Examples of service brokers are UDDI (Universal Description, Discovery, and Integration).
OASIS Reference Model for Service Oriented Architecture 1.0

- OASIS
- Abstract framework.
  - Understanding significant entities and relationships between them within a service-oriented environment.
  - Development of consistent standards or specifications supporting service-oriented environment.
- Based on unifying concepts of SOA and may be used by architects developing specific service-oriented architectures in training and explaining SOA.
- Reference model not directly tied to any standards, technologies or other concrete implementation details
- Provide a common semantics that can be used unambiguously across and between different implementations.
- The reference model focuses on the field of software architecture.

What is an SOA

- Service-oriented architecture (SOA) is a paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains.
- Visibility, interaction, and effect are key concepts for describing the SOA paradigm.
  - Visibility refers to the capacity for those with needs and those with capabilities to be able to see each other.
  - Whereas visibility introduces the possibilities for matching needs to capabilities (and vice versa), interaction is the activity of using a capability.
  - The purpose of using a capability is to realize one or more real world effects. At its core, an interaction is "an act" as opposed to "an object" and the result of an interaction is an effect (or a set/series of effects).
Principal concepts

- Visibility
- Service description
- Service
- Interaction
- Contract & Policy
- Real world effect
- Execution Context

**Service:** The means by which the needs of a consumer are brought together with the capabilities of a provider.

**Visibility:** The capacity for those with needs and those with capabilities to be able to interact with each other.

**Service description:** The information needed in order to use, or consider using, a service.

**Execution context:** The set of technical and business elements that form a path between those with needs and those with capabilities and that permit service providers and consumers to interact.

**Real world effect:** The actual result of using a service, rather than merely the capability offered by a service provider.

**Policy:** A statement of obligations, constraints or other conditions of use of an owned entity as defined by a participant.

**Interaction:** The activity involved in making using of a capability offered, usually across an ownership boundary, in order to achieve a particular desired real-world effect.
For a service provider and consumer to interact with each other they have to be able to 'see' each other. Visibility is the relationship between service consumers and providers that is satisfied when they are able to interact with each other. Preconditions to visibility are awareness, willingness and reachability.

Associated with all service interactions is intent – it is an intentional act to initiate and to participate in a service interaction. The extent of a service participant’s willingness to engage in service interactions may be the subject of policies.

Reachability is the relationship between service participants where they are able to interact; possibly by exchanging information. Reachability is an essential pre-requisite for service interaction – participants MUST be able to communicate with each other.

Both the service provider and the service consumer MUST have information that would lead them to know of the other’s existence. Service awareness requires that the service description and policy – or at least a suitable subset thereof – be available.
Interacting with services

Interaction involves performing actions against the service. In many cases, this is accomplished by sending and receiving messages. Key concepts that are important in understanding what it is involved in interacting with services revolve around the service description – which references a information model and a behavior model.

The second key requirement for successful interactions with services is knowledge of the actions invoked against the service and the process or temporal aspects of interacting with the service.

The formal descriptions of terms and the relationships between them (e.g., an ontology) provides a firm basis for selecting correct interpretations for elements of information exchanged.

Knowing the representation, structure, and form of information required is a key initial step in ensuring effective interactions with a service.

The information model of a service is a characterization of the information that may be exchanged with the service.

The action model of a service is the characterization of the actions that may be invoked against the service.

The process model characterizes the temporal relationships and temporal properties of actions and events associated with interacting with the service.
Real world effect

A real world effect can be the response to a request for information or the change in the state of some defined entities shared by the service participants.

In this context, the shared state does not necessarily refer to specific state variables being saved in physical storage but rather represents shared information about the affected entities. In addition, the internal actions that service providers and consumers perform as a result of participation in service interactions are, by definition, private and fundamentally unknowable. By unknowable we mean both that external parties cannot see others’ private actions and, furthermore, SHOULD NOT have explicit knowledge of them.
The service description represents the information needed in order to use a service. A service description SHOULD include sufficient data to enable a service consumer and service provider to interact with each other. This MAY include metadata such as the location of the service and what information protocols it supports and requires. It MAY also include dynamic information about the service, such as whether it is currently available.

A service description SHOULD unambiguously express the function(s) of the service and the real world effects that result from it being invoked.

The service interface is the means for interacting with a service. It includes the specific protocols, commands, and information exchange by which actions are initiated that result in the real world effects as specified through the service functionality portion of the service description.

A service description MAY include support for associating policies with a service and providing necessary information for prospective consumers to evaluate if a service will act in a manner consistent with the consumer’s constraints.
The execution context of a service interaction is the set of infrastructure elements, process entities, policy assertions and agreements that are identified as part of an instantiated service interaction, and thus forms a path between those with needs and those with capabilities.
MDD for SOA & SOA platforms
SOA platform consolidation

- Data and information integration ➔ **Information Fabric**
  - EII: Enterprise information integration
  - ETL: Extract, transform and load
- Application integration ➔ **Integration Suite**
  - EAI: Enterprise application integration
  - B2Bg: Business-to-business gateway
  - ESB: Enterprise service bus
- Applications and Processes ➔ **Business Process Management Suite**
  - BPM: Business process management
  - B2Bi: Business-to-business integration
- Enterprise workplace ➔ **Interaction Platform**
Integration suite services

- **Goal:** Composite applications
- **Components:** EAI, BPM, B2B, B2Bi
- **Extensions:** Adapter, collaboration, analysis, reporting, development, monitoring, contracts, SOA standards, ...

Business process management suite & interaction services

- **Goal:** Continuous process improvement
- **Components:** BPM
  - human-centric: people-intensive processes
  - Integration-centric: system-intensive processes
Information fabric services

- **Goal:** Holistic view of data (information virtualisation)
- **Components:** DBMS, EII + ETL + replication
- **Extensions:** Distributed meta-data repository, distributed data access, integrated data management

Architecture of Web-based solutions

- **Client Tier:**
  - Application Client
  - Dynamic HTML Pages
- **Web Tier:**
  - JavaBeans (optional)
  - JSP / JSF Pages
- **Business Tier:**
  - Web Services
  - EJB Session Beans
  - BPEL Processes
  - Persistent Endless
- **Database Tier:**
  - Database
Web services architecture

What is a Web service?

- The term "Web services" is confusing.
- There are many things that are referred to as “Web services”.
- Adding to the confusion is the term “services” which is interpreted differently by different people.
What is a Web service?

**Web service**

- Web is short for World Wide Web.
- Work performed or offered by a software system (possibly including human resources as well.)
- Software services performed or offered on the Web, using open Internet standards and technologies.

Definition (W3C): Web service

“A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It 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.”

- W3C Web Services Glossary, [http://www.w3.org/TR/ws-gloss/](http://www.w3.org/TR/ws-gloss/)
Characteristics of a basic Web service

- Two fundamental requirements:
  - It sends and receives data formatted as XML documents using SOAP over HTTP.
  - It provides a WSDL service description.
- Additionally, it is common for a Web service to:
  - Be registered with a discovery agent through which it can be located, typically UDDI.

Web services stack

- Service Composition
- Composable Service Assurance
- Description
- Messaging
- Transports

Conceptual stack

- WS-Security
- Web Service Reliable Messaging (WSRM)
- WS-Transactions

Technology stack

- XSD
- WSDL
- UDDI
- WS-Policy
- WS-Metadata Exchange
- XML
- SOAP
- WS-Addressing
- HTTP
- HTTPS
- SMTP
Web services – a conceptual view

Underlying Protocols
- HTTP/WEB
- VANs
- FTP
- SMTP/EMAIL
- MQ-Series

Messaging Encoding
- SOAP
- Raw XML
- ‘Binary’
- EDI
- ebXML
- "Binary"
- Raw XML
- ebXML

Business Entities
Web Service Interfaces

Web Services Architecture

BPEL
- Service Flow

Static → UDDI
- Service Discovery

Direct → UDDI
- Service Publication

WSDL
- Service Description

SOAP
- XML-Based Messaging

HTTP, FTP, email, MQ, IIOP, etc.
- Network

Quality of Service
- Security
**WS-* stack to-be**

- Simplified version of the to-be WS-* stack
  - Families of related specs not expanded
  - Competing spec families not shown
  - "Historical" or abandoned specs not shown

**WS-* stack as-is**

- Complete version of the as-is WS-* stack
  - The 3 widely-accepted specs today are the same as 5 years ago
  - BPEL and WS-Security is gaining momentum
  - Orchestration, discovery and brokering do not exist in today’s world
Model-driven Web Services – Two alternatives

1. Transformation in two steps via UML profile
2. Transformation in one step

Web service metamodels
Universal Description, Discovery and Integration (UDDI)

**UDDI Registry**

*Universal Description, Discovery and Integration*

1. SW companies, standards bodies, and programmers populate the registry with descriptions of different types of services

2. Businesses populate the registry with descriptions of the services they support

3. UBR assigns a programmatically unique identifier to each service and business registration

4. Marketplaces, search engines, and business apps query the registry to discover services at other companies

5. Business uses this data to facilitate easier integration with each other over the Web
Registry data

- Businesses register public information about themselves
- Standards bodies, Programmers, Businesses register information about their Service Types

White Pages

Yellow Pages

Green Pages

Service Type Registrations

The global UDDI business registry

Application

Application

Application

Business Service Registrator
UDDI – Four information types

- `<businessEntity>`: name, contacts, description, identifiers, categories
- `<businessService>`: name, description, categories
- `<bindingTemplate>`: technical information
- `<tModel>`: name, description, URL pointers to specifications

Simple Object Access Protocol (SOAP)
Use of SOAP

SOAP envelope

SOAP Envelope

SOAP Header

Header Block

... Header Block

SOAP Body

Body Block

... Body Block
Making a SOAP function call over HTTP

HTTP Request
Header
Body
XML Data

HTTP Response
Header
Body
XML Data

The SOAP Envelope

```xml
<SOAP:Envelope>
  <SOAP:Header/>
  <SOAP:Body>
    <m:FunctionName>
      <paramName1>paramValue1</paramName1>
      <paramName2>paramValue2</paramName2>
    </m:FunctionName>
  </SOAP:Body>
</SOAP:Envelope>
```
XML Schema Definition (XSD)

Description

- An XML schema describes the structure of an XML document.
- XSD is a comprehensive data modelling language for XML documents.
- The one XML schema specification that has received the broadest industry support.
- The XML schema definition language is also referred to as XML Schema Definition (XSD).
- XML schema is an XML-based alternative to DTD. It replaces/superseeds DTD.

XSD: Purpose

- Define the legal building blocks of an XML document:
  - Defines elements that can appear in a document.
  - Defines attributes that can appear in a document.
  - Defines which elements are child elements.
  - Defines the order of child elements.
  - Defines the number of child elements.
  - Defines whether an element is empty or can include text.
  - Defines data types for elements and attributes.
  - Defines default and fixed values for elements and attributes.

XSD: Approaches for specification

- Several different ways of specifying an XSD.
  - XML text editor
  - XML schema design editor
  - UML profile for XSD
XSD: XML text editor

Can also be built using simple text editors
XML editors gives contextual support, e.g. like auto-completion, suggestions for elements, etc., as well as validation of the XML document.

<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="XMLRequest">
    <xs:complexType>
      ...
    </xs:complexType>
  </xs:element>
</xs:schema>

XSD: XML schema design editor (1/3)

Visual language for schema design
Supported by e.g. XMLSpy
XSD: XML schema design editor (2/3)

**Element types**

- **Terminal:** This is the graphical representation of terminal elements. They usually contain text, numbers or another type of basic data.

- **Intermediate:** The intermediate elements represent the structure of the document. The "+" symbol indicates us that the element can contain more elements.

**Relationship types**

- **Sequence:** This schema represents that bankingInformation contains element accountNumber and bankData.

- **Choice:** This schema represents that fileId contains fileUri or (exclusive or) fileName.

XSD: XML schema design editor (3/3)

**Cardinality modifier**

We have seen the types and relationship between elements, but not the times that an element can appears.

- **Zero or one:** Tell us that the element is optional (can appear zero or one times).

- **One or more:** Tell us that the element appears at least one time, but can appear all times we want.

- **Zero or more:** Tell us that the element appears zero or more times.
**XSD: UML profile for XSD**

- **UML representation of XML schema.**
- Useful in a UML-centric development method if the modelling environment supports generation/import of XSD documents.

---

**XSD metamodel**
XSD metamodel (simplified)

UML profile for XSD (1)

<table>
<thead>
<tr>
<th>Stereotype</th>
<th>UML construct</th>
<th>Tagged value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;&lt;any&gt;&gt;</td>
<td>Class, Property</td>
<td>The stereotyped class or attribute will be replaced by an 'any' or 'anyAttribute' element. The tagged values are copied into the corresponding attributes of the generated element</td>
<td></td>
</tr>
<tr>
<td>namespace</td>
<td>As defined in XML Schema specification</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| processContents | As defined in XML Schema specification  
• values="skip | lax | strict"  
• default="strict" |
| <<attribute>>   | Property       | Assigned to UML attribute or association end. Indicates item is to be generated as an attribute within complexType and not as an element  
| default         | As defined in XML Schema specification |
| fixed           | As defined in XML Schema specification |
| form            | Overrides the attributeFormDefault for this schema  
• values="qualified | unqualified" |
| use             | As defined in XML Schema specification  
• values="prohibited | optional | required"  
• default="optional" |
| <<choice>>      | Class          | Elements marked with this stereotype represent a Choice model group contained within a complexType definition |
| <<complexType>>| Class          | ComplexType definition generated in XML Schema |
| memberNames     | Overrides the package-level default for naming complexType definitions  
• values="qualified | unqualified" |
### UML profile for XSD (2)

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mixed</td>
<td>Determines whether this element may contain mixed element and character content.</td>
</tr>
<tr>
<td>values</td>
<td>• values=&quot;true</td>
</tr>
<tr>
<td>default</td>
<td>• default=&quot;false&quot;</td>
</tr>
<tr>
<td>modelGroup</td>
<td>Overrides the package-level default model group</td>
</tr>
<tr>
<td>values</td>
<td>• values=&quot;all</td>
</tr>
<tr>
<td>anonymousRole</td>
<td>The class type will be directly embedded within the complexType definition. Omit attribute or role type wrapper</td>
</tr>
<tr>
<td>values</td>
<td>• values=&quot;true</td>
</tr>
<tr>
<td>form</td>
<td>Overrides the elementFormDefault for this schema</td>
</tr>
<tr>
<td>values</td>
<td>• values=&quot;qualified</td>
</tr>
</tbody>
</table>

### UML profile for XSD (3)

**UML representation**

**Text representation**

If assigned, indicates position in the sequence model group.

A facet is a single defining aspect of a value space. Generally speaking, each facet...
Web Services Description Language (WSDL)

- Purpose
  - Web services need to be defined in a consistent manner so that they can be discovered by and interfaced with other services and applications.
  - The Web Services Description Language is a W3C specification providing the foremost language for the description of Web service definitions.

WSDL: Description

- XML-based language for describing functional properties of Web services.
- A service consists of a collection of message exchange end points.
- An end point contains an abstract description of a service interface and implementation binding.
- The abstract description of a service contains:
  - (i) definitions of messages which are consumed and generated by the service
  - (ii) signatures of service operations.
- The implementation binding provides a means to map abstract operations to concrete service implementations.
  - It essentially contains information about the location of a binding and the communication protocol to use (e.g., SOAP over HTTP) for exchanging messages.

WSDL: Conceptual view

- Business Entities
- Web Service Interfaces
- Messaging Encoding
- Underlying Protocols

- SOA
- Raw XML
- ebXML
- “Binary”
- EDI
- HTTP/WEB
- SMTP/EMAIL
- FTP
- VANs
- MQ-Series
WSDL: Conceptual model

- **WS Client**
- **WS Provider**
- **Porttype**
  - **Operations**
  - **Concrete Endpoint Address**
  - **(Reusable) Binding**
    - **Concrete Message Encoding**
    - **Concrete Messaging Protocol**

WSDL: Message exchange patterns

- **Time**
- **Request-Response**
- **One-Way**
- **Solicit-Response**
- **Notification**
Changes in WSDL 2.0

- Removal of Operation overloading
- PortType renamed to Interface
  - Interface inheritance
- Port renamed Endpoint
- Extended repertoire of Message Exchange Patterns.
WSDL 2.0 metamodel

UML profile for WSDL (1)

<table>
<thead>
<tr>
<th>Stereotype</th>
<th>UML construct</th>
<th>Tagged value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;&lt;binding&gt;&gt;</td>
<td>Class</td>
<td></td>
<td>A concrete protocol and data format specification for a particular port type. A concrete protocol and data format specification for a particular port type. A &lt;&lt;binding&gt;&gt; class represents a binding component of the WSDL metamodel.</td>
</tr>
<tr>
<td>&lt;&lt;definition&gt;&gt;</td>
<td>Class</td>
<td></td>
<td>A &lt;&lt;definition&gt;&gt; class represents a definition component of the WSDL metamodel.</td>
</tr>
<tr>
<td>&lt;&lt;element&gt;&gt;</td>
<td>Class</td>
<td></td>
<td>An &lt;&lt;element&gt;&gt; class represents an element of the XML Schema.</td>
</tr>
<tr>
<td>&lt;&lt;fault&gt;&gt;</td>
<td>Association</td>
<td></td>
<td>An &lt;&lt;fault&gt;&gt; association represents a relationship between an operation and a message in the WSDL metamodel.</td>
</tr>
</tbody>
</table>
UML profile for WSDL (2)

<table>
<thead>
<tr>
<th>location</th>
<th>Location is an URL. It is optional and indicates the location of some information for the namespace.</th>
</tr>
</thead>
<tbody>
<tr>
<td>namespace</td>
<td>Namespace is an URI (Uniform Resource Identifier). It is mandatory and indicates that the containing WSDL document can contain references to the WSDL definitions in that namespace.</td>
</tr>
</tbody>
</table>

<<input>> Association

An <<input>> association represents a relationship between an operation and a message the WSDL metamodel.

<<message>> Class

An abstract, typed definition of the data being communicated. A <<message>> class represents a message component of the WSDL metamodel.

<<operation>> Class

An abstract, description of an action supported by the service. An <<operation>> class represents an operation component of the WSDL metamodel.

<<output>> Association

An <<output>> association represents a relationship between an operation and a message the WSDL metamodel.

<<part>> Class

A <<part>> class represents a part component of the WSDL metamodel.

<<part>> Class

A <<part>> class represents a part component of the WSDL metamodel.

<<partElement>> Association

An <<partElement>> association represents a relationship between a part component of the WSDL metamodel and another part component.

UML profile for WSDL (3)

<table>
<thead>
<tr>
<th>UML representation</th>
<th>Text representation</th>
</tr>
</thead>
</table>

SINTEF
Web Services Business Process Execution Language (WS-BPEL)

- Description
  - WS-BPEL (or BPEL for short) is a language based on XML that allows for controlling the process flow of a set of collaborating Web services.
  - It can be seen as a (business) extension to the Web services paradigm.
  - Partner interaction is based on the notion of peer-to-peer interaction between Web services.
  - BPEL introduces concepts to express the peer-to-peer conversational relationships between services.
  - Partner links specify the services that a business process interacts with and is introduced as a WSDL extension element.

BPEL language

- XML notation
- Interaction with other Web services:
  - `<receive>`. Wait for an incoming message. Typically at the process start
  - `<invoke>`. Call another Web service
  - `<reply>`. Send a response message from the entire BPEL service
- Control flow
  - `<sequence>`. Sequential control flow
  - `<flow>`. Parallel control flow
  - `<switch>`. Conditional branching
  - `<while>`. Loop
- Data flow
  - `<variable>`. Defines the data objects involved
  - `<assign>`. Copy a data object from one variable to another possibly w/ data transformation

Web service composition

- BPEL is a Web service composition language.
- It defines how to compose other Web services so to accomplish a more complex task.
- A BPEL engine is capable of executing the composite service described by BPEL.
- The outcome will be a composite BPEL-defined Web service which itself can be regarded as a Web service.
**BPEL simplified view**

A BPEL process is a composite Web service with a WSDL description.

**BPEL roles in the Web services world**

- As a public specification of abstract services protocols
  - business partners can supply precise information about semantics of a service and its message properties...
  - ...without revealing internal (opaque) details of implementation
- As an intermediate language for implementing business processes
  - relatively portable
  - extensible for platform-specific operations possible
- As a programming language for Internet-scale distributed applications
  - messaging
  - concurrency
  - error handling
  - ...
BPEL foundations

- WSDL
- XPath
- XSD
- XML

message "signatures": message names, components, simple msg. exchange patterns

data manipulation expressions
type definitions

common "syntax" for WS components and runtime data-as-documents

BPEL details

- Two Uses
  - Executable process descriptions
  - Business protocol descriptions – Abstract processes

- Partner links
  - Paired WSDL interfaces
  - Correlation sets
  - Bind messages to process/activity instances.
  - Endpoint references

- Partner
  - Grouping constraint on partner links to a single business partner.

- Process Activities
  - Basic - assign, throw, terminate, wait, empty, compensate
  - Partner interaction - receive, reply, invoke
  - Structured - sequence, switch, while, pick, flow, scope
BPEL process and scope activities

- **Partner Links**: only in `<process/>`
- **Variables**
  - Correlation Sets
  - Fault Handlers
  - Compensation Handler
  - Event Handlers

**Primary Activity**

- Message variables shared by activities in `<scope/>`
- Correlation sets for associating messages with process/activity instances
- Install special purpose activities in scope
- Compensation of **completed** scopes

BPEL example

1. Receive Purchase Order
2. Initiate Price Calculation
3. Decide on Shipper
   - Complete Price Calculation
4. Arrange Logistics
   - Complete Production Scheduling
5. Invoicing Processing

Diagram showing the flow of activities in a BPEL process.
WSDL port type XML syntax

```xml
<portType name="purchaseOrderPT">
    <operation name="sendPurchaseOrder">
        <input message="pos:POMessage"/>
        <output message="pos:InvMessage"/>
        <fault message="pos:orderFaultType"/>
    </operation>
</portType>
```
Partner link types

Each role specifies exactly one WSDL portType.

Purchase order WSDL

```xml
<message name="POMessage">
  <part name="customerInfo" type="sns:customerInfo"/>
  <part name="purchaseOrder" type="sns:purchaseOrder"/>
</message>
<message name="InvMessage">
  <part name="IVC" type="sns:Invoice"/>
</message>
<message name="orderFaultType">
  <part name="problemInfo" type="xsd:string"/>
</message>
<message name="shippingRequestMessage">
  <part name="customerInfo" type="sns:customerInfo"/>
</message>
<message name="shippingInfoMessage">
  <part name="shippingInfo" type="sns:shippingInfo"/>
</message>
<message name="scheduleMessage">
  <part name="schedule" type="sns:scheduleInfo"/>
</message>
```
BPEL Process

```xml
<process name="purchaseOrderProcess"
    xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/>

<partnerLinks>
    <partnerLink name="purchasing"
        partnerLinkType="lns:purchasingLT"
        myRole="purchaseService"/>
    <partnerLink name="invoicing"
        partnerLinkType="lns:invoicingLT"
        myRole="invoiceRequester"
        partnerRole="InvoiceService"/>
    <partnerLink name="shipping"
        partnerLinkType="lns:shippingLT"
        myRole="shippingRequester"
        partnerRole="ShippingService"/>
    <partnerLink name="scheduling"
        partnerLinkType="lns:schedulingLT"
        partnerRole="schedulingService"/>
</partnerLinks>
```

BPEL process

```xml
<sequence>
    <receive partnerLink="purchasing"
        portType="lns:purchaseOrderPT"
        operation="sendPurchaseOrder"
        variable="PO"/>

    <flow>
        <links>
            <link name="ship-to-invoice"/>
            <link name="ship-to-scheduling"/>
        </links>
    </flow>

    <sequence>
        <assign>
            <copy>
                <from variables="PO" part="customerInfo"/>
                <to variable="shippingRequest" part="customerInfo"/>
            </copy>
        </assign>
    </sequence>

    ...
```
### BPEL process

```xml
<invoke partnerLink="shipping"
    portType="lns:shippingPT"
    operation="requestShipping"
    inputVariable="shippingRequest"
    outputVariable="shippingInfo">
    <source linkName="ship-to-invoices"/>
</invoke>

<receive partnerLink="shipping"
    portType="lns:shippingCallbackPT"
    operation="sendSchedule"
    variable="shippingSchedule">
    <source linkName="ship-to-scheduling"/>
</receive>
```

### UML profile for BPEL (1)

<table>
<thead>
<tr>
<th>Stereotype</th>
<th>UML construct</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;&lt;assign&gt;&gt;</td>
<td>Action</td>
<td>An assign activity maps to a BPEL assign activity with each entry action mapping to a copy element. The right-hand side of each assign statement provides the details of a from element and the left-hand side provides the details of the to element.</td>
</tr>
<tr>
<td>&lt;&lt;catch&gt;&gt;</td>
<td>Action</td>
<td>When an invoked operation throws an exception, or a throw activity explicitly throws an exception, normal execution within the containing scope is terminated. An exception can be caught within the containing scope so that error recovery behavior can be performed. This is modelled as an &lt;&lt;catch&gt;&gt; activity.</td>
</tr>
<tr>
<td>&lt;&lt;compensate&gt;&gt;</td>
<td>Action</td>
<td>Compensation can be triggered by a compensate activity. We follow the BPEL semantics for compensation and when it can be triggered. In particular, a compensate activity is only permitted in the following places:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• In a catch activity of the scope that immediately encloses the scope for which compensation is to be performed.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• In the compensation handler of the scope that immediately encloses the scope for which compensation is to be performed.</td>
</tr>
<tr>
<td>&lt;&lt;compensationHandler&gt;&gt;</td>
<td>Action</td>
<td>Compensation handler activities can be defined to reverse the work performed by a scope, if necessary. Compensation handler activities are not executed when control reaches the parent activity. If the parent activity completes successfully then the compensation handler is installed.</td>
</tr>
<tr>
<td>&lt;&lt;correlation&gt;&gt;</td>
<td>Class, Property</td>
<td>A correlation set is defined by a class stereotyped by &lt;&lt;correlation&gt;&gt; containing attributes with names and types matching those of properties defined within its namespace. A process specifies that it uses a correlation set through an attribute with the type of the correlation set. The stereotype &lt;&lt;correlation&gt;&gt; can also be applied redundantly to the attribute to distinguish it from other attributes.</td>
</tr>
</tbody>
</table>
### UML profile for BPEL (2)

<table>
<thead>
<tr>
<th>Stereotype</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;&lt;data&gt;&gt;</td>
<td>A message type has a number of parts, each of which is of a specified data type. Data types are represented by classes stereotyped as &lt;&lt;data&gt;&gt;.</td>
</tr>
<tr>
<td>&lt;&lt;external&gt;&gt;</td>
<td>External packages contain elements that are defined in another model or elements that are defined directly as platform-specific artifacts (such as Web services or BPEL documents). External packages are not mapped to platform-specific artifacts. Elements that are reused can be modeled explicitly and placed in a package stereotyped with &lt;&lt;external&gt;&gt;.</td>
</tr>
<tr>
<td>&lt;&lt;invoke&gt;&gt;</td>
<td>An invocation of an operation on a partner is represented as an activity with stereotype &lt;&lt;invoke&gt;&gt; with an entry action indicating the operation to be invoked and the attribute containing the input message. For two-way messages, assignment notation is used to indicate the attribute that is updated with the reply message.</td>
</tr>
<tr>
<td>&lt;&lt;loop&gt;&gt;</td>
<td>A looping node is shown as an activity with the stereotype &lt;&lt;loop&gt;&gt;, which contains a decision node and an activity to be repeated, with a control link from the decision node to the activity. The guard on the control link provides the Boolean expression which determines whether the activity is executed each time round the loop.</td>
</tr>
<tr>
<td>&lt;&lt;messageContent&gt;&gt;</td>
<td>A message type has a number of parts, each of which is of a specified data type. Message types are represented by classes stereotyped as &lt;&lt;messageContent&gt;&gt;.</td>
</tr>
</tbody>
</table>

---

### UML profile for BPEL (3)

The diagram illustrates the behavior of the system, showing the interactions between different roles and processes. Each node represents an activity, and the arrows indicate the flow of control and data. The diagram is not described in the text, but it shows the dynamic behavior of the system as defined in BPEL.
References