INF5120 – Model-based System Development

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

7 April 2008
Brian Elvesæter, SINTEF ICT

Based on material developed in the ATHENA (IST-507849), COMBINE (IST-1999-20839), INTEROP (IST-508011), and MODELWARE (IST-511731) research projects.

Outline

- What is an architecture?
- MDD framework for SOA
- SOA and integration
- Web services architecture
  - UDDI, SOAP, XSD, WSDL, BPEL
- References
What is an architecture?

Different kinds of architectures

- Enterprise architecture
- Business architecture
- Conceptual architecture
- Functional architecture
- Architecture framework
- Service-oriented architecture
- Realisation architecture
- ICT architecture
- Integration architecture
- Knowledge architecture
- Logical architecture
- Information 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.

IEEE Std 1471-2000

IEEE Std 1471-2000

- IEEE Recommended Practice for Architectural Description of Software-Intensive Systems
- Adopted September 2000

Architecture definition

- Structure(s) of a system in terms of
  - components,
  - their externally visible properties,
  - their relations,
  - and the underlying principles

Common frame of reference for architectural descriptions

- Common terminology
  - architecture, architectural description, model, view, viewpoint, system, stakeholder, concern, …
The fundamental organisation of a system embodied in its components, their relationships to each other and to the environment, and the principles guiding its design and evolution.

The expression of a systems architecture with respect to a particular viewpoint. Addresses one or more of the concerns of the system stakeholder.

Developed using the methods established by its viewpoint, consisting of views expressing an architectural description.

Architecture of what and for whom?

Virtual enterprise

Business

Software system

Software component

Software object
Why enterprise architecture?

How can I use best practices to ensure the success of the business?

How can I involve my people in improving the performance of the business?

How can I ensure that the IS technology helps the work of my people?

Role of enterprise architecture

Mission
Vision
Strategy
Goals to be
as is
enterprise architecture
domain/aspect architectures
Actions
culture
leadership
people
Operations
... people IT

products processes

Describing coherence

MDD framework for SOA
Johnson and Lyndon provide enactment of all the roles found in an SOA (consumer, provider, intermediary) and flexible communication between Web services through an intuitive user interface.

The WSDL Analyzer tool detected syntactical mismatches between service descriptions and provides a basis for runtime mediation of Web service messages.

The Web service extensions to the JACK autonomous agents platform allow SOAs to use agents for brokering, mediation and negotiation between Web services.

BDI teams provide a flexible and composable alternative to traditional approaches to Web service composition.
Web services enactment framework

- It enables users to enact most of the roles typically found in an SOA.
- It allows sending real SOAP messages between Web services without having to write a single line of code.
- It features a Web-based user interface designed to closely resemble Web-based email applications.
- SOAP messages and Web Services endpoints are used in place of email messages and email addresses.
- The user can see incoming SOAP messages in the Inbox and create outgoing SOAP messages in the Outbox that will be sent to external Web services.
- A powerful user-interface generator relieves the user from having to deal with XML documents by generating forms for displaying and editing any XML-based data type.
SOA and integration

The waves of client/server technology

Base Source: Client/Server Survival Guide, 1994
Robert Orfali, Dan Harkey
OS/2 Edition, VNR Computer library + AJB update
Application architecture vs. SOA

SOA and integration

- Fundamental change for integration: X <-> Y
  - Pre-SOA: outside, after development
  - Post-SOA: inside, integral part of development / computational model
- Consequences
  - How should integration be done?
  - Innovation and experience
  - Competition, expansion, consolidation
- Not understood:
  - IDC Directions 2006 (3/2/06): SOA important but not understood or deployed as claimed
  - Gartner (2/15/06): “Globally, organizations placing minor emphasis on understanding the role of data integration in SOA and creation of data services at the foundation of their architectures”
History of integration

- 1950 – 2006: Integration = develop then integrate
  - 1950s-1970s: Simple, manual integration
  - 1970s-1980s: Distributed Computing
    - Applications (interoperation)
    - Databases (integrate)
- 1990s: Business Driven Integration – concepts, technologies, and tools – increased automation, internet-based computing
  - Concepts: Workflows, Processes, Web,
  - Integration solutions blossom (diverge): ETL, EAI, BPM, …
- 2000: SOA Emerges
  - 2000: Web services
  - 2003: Integration solution evolution accelerates, vendor chaos ensues
  - 2005: Growth in all integration categories

Integration in SOA

- 2006 – 2012: Integration = dominant programming model
  - 2001-2010: Wrapping
  - 2005-2010: Re-Engineering
  - 2006-2008: Consolidation
  - 2006-2008: Research on Semantic SOA
  - 2007-2012: Emergence of SOA Platforms and Solutions
  - 2006-2012: Problem Solving Era: IT/integration relegated to low level function
SOA platform consolidation

- **Data and information integration**  
  - EII: Enterprise information integration  
  - ETL: Extract, transform and load

- **Application integration**  
  - EAI: Enterprise application integration  
  - B2Bg: Business-to-business gateway  
  - ESB: Enterprise service bus

- **Applications and Processes**  
  - BPM: Business process management  
  - B2Bi: Business-to-business integration

- **Enterprise workplace**  
  - Interaction Platform
Goal: Composite applications
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
Trends

- Consolidation ➔ comprehensive platforms
- Merging of Human Workflow and System Orchestration/Process services
- Integration of Business Rules Engines
- Support for Event Notification services (publish and subscribe)
- Integration of Model-generated workplaces and role/task-oriented user interfaces, user interaction services, portals, and multi-device interfaces
- Explicit use of models (Enterprise and System)
- Enterprise architecture + SOA

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.

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/

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

Conceptual stack

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

Technology stack

- WS-Security
- Web Service Reliable Messaging (WS-IRM)
- WS-Transactions
- XSD
- WSDL
- UDDI
- WS-Policy
- WS-Addressing
- XML
- SOAP
- WS-Addressing
- HTTP
- HTTPS
- SMTP

Web services – a conceptual view

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

- WSDL
- XSD
- SOAP
- Raw XML
- BPEL
- Interaction Sequencing (Co)Constraints
- EGO-Centric Workflow Process Description
- Bindings and Endpoint Descriptions
- XML Message Schema Definition
- HTTP/WEB
- SMTP/EMAIL
- FTP
- VANs
- MQ-Series
- ebXML
- "Binary"
**Web Services Architecture**

**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

UDDI – Four information types

- `<businessEntity>`: name, contacts, description, identifiers, categories
- `<businessService>` (1..n): name, description, categories
- `<bindingTemplate>` (1..n): 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:Header>  ← Optional

  <SOAP:Body>
    <m:FunctionName>
      <paramName1>paramValue1</paramName1>
      <paramName2>paramValue2</paramName2>
    </m:FunctionName>
  </SOAP:Body>

</SOAP:Envelope>
```

XML Schema Definition (XSD)
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.

  http://www.w3.org/TR/xmlschema-0/

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

```xml
<?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>
```

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

- **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><code>&lt;&lt;any&gt;&gt;</code></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></td>
<td>As defined in XML Schema specification</td>
<td></td>
</tr>
<tr>
<td>processContents</td>
<td></td>
<td>As defined in XML Schema specification</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• values=&quot;skip</td>
<td>lax</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• default=&quot;strict&quot;</td>
<td></td>
</tr>
<tr>
<td><code>&lt;&lt;attribute&gt;&gt;</code></td>
<td>Property</td>
<td>Assigned to UML attribute or association end. Indicates item is to be generated as an attribute within complexType and not as an element.</td>
<td></td>
</tr>
<tr>
<td>default</td>
<td></td>
<td>As defined in XML Schema specification</td>
<td></td>
</tr>
<tr>
<td>fixed</td>
<td></td>
<td>As defined in XML Schema specification</td>
<td></td>
</tr>
<tr>
<td>form</td>
<td></td>
<td>Overrides the attributeFormDefault for this schema</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• values=&quot;qualified</td>
<td>unqualified&quot;</td>
</tr>
<tr>
<td>use</td>
<td></td>
<td>As defined in XML Schema specification</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• values=&quot;prohibited</td>
<td>optional</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• default=&quot;optional&quot;</td>
<td></td>
</tr>
<tr>
<td><code>&lt;&lt;choice&gt;&gt;</code></td>
<td>Class</td>
<td>Elements marked with this stereotype represent a Choice model group contained within a complexType definition.</td>
<td></td>
</tr>
<tr>
<td>memberNames</td>
<td></td>
<td>Overrides the package-level default for naming complexType definition</td>
<td></td>
</tr>
<tr>
<td>anonymousRole</td>
<td></td>
<td>The class type will be directly embedded within the complexType definition. Omit attribute or role type wrapper.</td>
<td></td>
</tr>
<tr>
<td>anonymousType</td>
<td></td>
<td>The class type will be anonymous for XML documents generated by the schema.</td>
<td></td>
</tr>
<tr>
<td>form</td>
<td></td>
<td>Overrides the elementFormDefault for this schema</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• values=&quot;qualified</td>
<td>unqualified&quot;</td>
</tr>
<tr>
<td>position</td>
<td></td>
<td>If assigned, indicates position in the sequence model group.</td>
<td></td>
</tr>
</tbody>
</table>

### UML profile for XSD (2)

<table>
<thead>
<tr>
<th>Stereotype</th>
<th>UML construct</th>
<th>Tagged value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mixed</td>
<td></td>
<td>Determines whether this element may contain mixed element and character content.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• values=&quot;true</td>
<td>false&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• default=&quot;false&quot;</td>
<td></td>
</tr>
<tr>
<td>modelGroup</td>
<td></td>
<td>Overrides the package-level default model group</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• values=&quot;all</td>
<td>sequence</td>
</tr>
<tr>
<td><code>&lt;&lt;element&gt;&gt;</code></td>
<td>Property</td>
<td>Assigned to UML attribute or association end. Indicates item is to be generated as element within complexType and not as attribute.</td>
<td></td>
</tr>
<tr>
<td>anonymousRole</td>
<td></td>
<td>The class type will be directly embedded within the complexType definition. Omit attribute or role type wrapper.</td>
<td></td>
</tr>
<tr>
<td>anonymousType</td>
<td></td>
<td>The class type will be anonymous for XML documents generated by the schema.</td>
<td></td>
</tr>
<tr>
<td>form</td>
<td></td>
<td>Overrides the elementFormDefault for this schema</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• values=&quot;qualified</td>
<td>unqualified&quot;</td>
</tr>
<tr>
<td>position</td>
<td></td>
<td>If assigned, indicates position in the sequence model group.</td>
<td></td>
</tr>
<tr>
<td><code>&lt;&lt;facet&gt;&gt;</code></td>
<td>Property</td>
<td>A facet is a single defining aspect of a value space. Generally speaking, each facet characterizes a value space.</td>
<td></td>
</tr>
</tbody>
</table>
UML profile for XSD (3)

Web Services Description Language (WSDL)
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

Underlying Protocols

Messaging Encoding

(bindings and endpoint descriptions)

WSDL: Conceptual model

WS Interface

WS Provider

Porttype

Operations

Concrete Endpoint Address

(Reusable) Binding

Concrete Message Encoding

Concrete Messaging Protocol

WS Client

Operations Invoked through Ports

Name, Abstract Message Parts Schema

Message Exchange Pattern
WSDL: Message exchange patterns

WS Client

Time

Request-Response

One-Way

Solicit-Response

Notification

WS Provider

WSDL 1.1 metamodel

A collection of related endpoints

A single endpoint defined as a combination of a binding and a network address

A concrete protocol and data format specification for a particular port type

An abstract set of operations supported by one or more endpoints

An abstract, description of an action supported by the service

An abstract, typed definition of the data being communicated

A container for data type definitions

An abstract, description of a concrete protocol and data format

A container for data type definitions
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>binding</td>
<td>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></td>
<td></td>
<td>style</td>
<td>The style attribute indicates whether the operation is a remote procedure call (RPC) or a document-oriented operation.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>transport</td>
<td>The transport attribute specifies the type of binding to be used.</td>
</tr>
<tr>
<td>&lt;&lt;definition&gt;&gt;</td>
<td>Class</td>
<td>targetNameSpace</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>baseType</td>
<td>The base type</td>
</tr>
<tr>
<td></td>
<td></td>
<td>minOccurs</td>
<td>The minimum number of occurrences</td>
</tr>
<tr>
<td></td>
<td></td>
<td>maxOccurs</td>
<td>The maximum number of occurrences</td>
</tr>
<tr>
<td></td>
<td></td>
<td>name</td>
<td>The name of the element</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>
<tr>
<td>&lt;&lt;import&gt;&gt;</td>
<td>Class</td>
<td></td>
<td>An &lt;&lt;import&gt;&gt; class represents an import component of the WSDL metamodel.</td>
</tr>
<tr>
<td>&lt;&lt;input&gt;&gt;</td>
<td>Association</td>
<td></td>
<td>An &lt;&lt;input&gt;&gt; association represents a relationship between an operation and a message in the WSDL metamodel.</td>
</tr>
<tr>
<td>&lt;&lt;message&gt;&gt;</td>
<td>Class</td>
<td></td>
<td>An abstract, typed definition of the data being communicated. A &lt;&lt;message&gt;&gt; class represents a message component of the WSDL metamodel.</td>
</tr>
<tr>
<td>&lt;&lt;operation&gt;&gt;</td>
<td>Class</td>
<td></td>
<td>An abstract, description of an action supported by the service. An &lt;&lt;operation&gt;&gt; class represents an operation component of the WSDL metamodel.</td>
</tr>
<tr>
<td>&lt;&lt;output&gt;&gt;</td>
<td>Association</td>
<td></td>
<td>An &lt;&lt;output&gt;&gt; association represents a relationship between an operation and a message in the WSDL metamodel.</td>
</tr>
<tr>
<td>&lt;&lt;part&gt;&gt;</td>
<td>Class</td>
<td></td>
<td>A &lt;&lt;part&gt;&gt; class represents a part component of the WSDL metamodel.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>type</td>
<td>Type is a base type XSD. It is optionally and must be defined when the part component uses a base type but not when the part component uses an element of the XML Schema.</td>
</tr>
<tr>
<td>&lt;&lt;partElement&gt;&gt;</td>
<td>Association</td>
<td></td>
<td>A &lt;&lt;partElement&gt;&gt; association represents a relationship between a part component of the WSDL metamodel and...</td>
</tr>
</tbody>
</table>
Web Services Business Process Execution Language (WS-BPEL)
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

BPEL partnerLink and partner definitions
- Web Services Description Language (WSDL)
- XML Path Language (XPath)
- XML Schema (XSD)
- XML

Diagram:
- 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

- 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

- Receive Purchase Order
  - Initiate Price Calculation
  - Complete Price Calculation
  - Invoice Processing

- Decide on Shipment
  - Complete Logistics
  - Complete Production Scheduling

- Initiate Production Scheduling
WSDL port type XML syntax

```xml
<portType name="purchaseOrderPT">
  <operation name="sendPurchaseOrder">
    <input message="pos:POMessage"/>
    <output message="pos:InvMessage"/>
    <fault name="cannotCompleteOrder" 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>
    <sequence>
      <assign>
        <copy>
          <from variable="PO" part="customerInfo"/>
          <to variable="shippingRequest" part="customerInfo"/>
        </copy>
      </assign>
    </sequence>
    ...
</flow>
```

BPEL process

```xml
<invoke partnerLink="shipping"
  portType="lns:shippingPT"
  operation="requestShipping"
  inputVariable="shippingRequest"
  outputVariable="shippingInfo">
  <source linkName="ship-to-invoice"/>
</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><code>&lt;&lt;assign&gt;&gt;</code></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><code>&lt;&lt;catch&gt;&gt;</code></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 <code>&lt;&lt;catch&gt;&gt;</code> activity.</td>
</tr>
</tbody>
</table>
| `<<compensate>>` | Action        | 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:  
  - In a catch activity of the scope that immediately encloses the scope for which compensation is to be performed.  
  - In the compensation handler of the scope that immediately encloses the scope for which compensation is to be performed. |
| `<<compensationHandler>>` | Action | 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. |
| `<<correlation>>` | Class, Property | A correlation set is defined by a class stereotyped by `<<correlation>>` 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 `<<correlation>>` can also be applied redundantly to the attribute to distinguish it from other attributes. |
| `<<data>>`       | Class         | 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 `<<data>>`. |
| `<<external>>`   | Package       | 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 `<<external>>`. |
| `<<invoke>>`     | Action        | An invocation of an operation on a partner is represented as an activity with stereotype `<<invoke>>` 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. |
| `<<loop>>`       | Action        | A looping node is shown as an activity with the stereotype `<<loop>>`, 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. |
| `<<messageContent>>` | Class | 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 `<<messageContent>>`. It is often useful to group a set of receive activities together as an activity. |
UML profile for BPEL (3)

References
References

