INF5120 – Model-Based System Development

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

28 March 2011

Based on material developed in the ATHENA (IST-507849), COMBINE (IST-1999-20839), INTEROP (IST-508011), MODELWARE (IST-511731) and SHAPE (ICT-2007-216408) research projects.

INF5120 - Lecture plan - 2011

1: 24/1: Introduction to Model Based System Development (INF5120)

Part I: MDE – Model Driven Engineering
- 2: 31/1: MDE I: Metamodels, Domain specific languages and UML profiles
- 3: 7/2: MDE II: Metamodeling, MDLE and DSL Tools (EMF, GMF, ATL, Kermeta)
- 4: 14/2: MDE III: Model transformations - MOFScript,
- 6: 28/2: MDE IV: Method Engineering with SPEM / EPF/SEMAT

Part II: SSI – Service Innovation and Engineering
- 5: 21/2: SIE I: Service Innovation and CSI, Enterprise Architecture and Service methodologies
- 7: 7/3: SIE II: Business Process Modeling with BPMN 2.0
- 9: 14/3: SIE III: User-oriented design – with Use cases and UI models
- 9: 21/3: SIE IV: Service modeling with SoaML,
- 10: 28/3: SIE V: SOA technologies - with WSDL, XML, BPEL and Patterns

Part IV – Model Driven Interoperability
- 12: 11/4: MDI I: Semantic technologies, Ontologies and Semantic annotations
- 12: 11/4: MDI II: Model Driven Service Interoperability
- 13: 25/5: MDI III: ADM and Migration to Cloud computing

14: 9/5: Conclusion and Summary for INF5120 - Preparation of Exam

Exam: May 30th, 2011 (Monday), 0900-1300 (4 hours)
Outline

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

- Patterns

Service-oriented architecture (SOA)
Different kinds of architectures

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

EA – SOA – WSA

- **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

- Mission
- Vision
- Strategy
- Goals
  - as is
  - to be

**Vision**

as is to be

enterprise architecture
domain/aspect architectures
culture leadership
culture people

**Strategy**

Actions

Operations

- people
- IT
- products
- processes

Describing coherence

- Information architecture
- Product architecture
- Process architecture
- Application architecture
- Technical architecture

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**
- **Service description**
- **Real world effect**
- **Execution Context**
- **Interaction**
- **Contract & Policy**
Principal concepts

**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.

**Service**: The means by which the needs of a consumer are brought together with the capabilities of a 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 use of a capability offered, usually across an ownership boundary, in order to achieve a particular desired real-world effect.

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

**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.

Visibility

- Awareness
- Willingness
- Reachability
- Service
- Real world effect
- Interaction
Visibility

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.

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.

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.

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.

Interacting with services

Real world effect

Service description

Action model

Behavior model

Process model

Interaction

Information model

Structure

Semantics
Interacting with services

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.

Interacting with a service 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 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

Visibility

Service

Shared state

Real world effect

Interaction
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.

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.

Service description

Visibility

Reachability

Service

Service interface

Contract & Policy

Functionality

Interaction

Renewal model

Information model

Real world effect
Service description

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.

Execution context

The execution context may include service descriptions, service contracts, and behavior models, which are used to define the interaction and information models of the service.
Execution context

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
Modeling approach for SOA

- Enterprise Model
- UML Profile for SOA (SoaML)
- Model to Model Transformation
- Model to Text Transformation
- Web Service Specification Model
- Agent Specification Model
- BPEL Specification Model
- P2P Specification Model
- Web Service Execution Artefacts
- Agent Execution Artefacts
- BPEL Execution Artefacts
- P2P Execution Artefacts
- Execution Infrastructure
- Evaluation & Negotiation of Available Functionality
- Enhanced Service Interconnection Bus
- Cross-org.
- Intra-org.
- Existing Enterprise Applications
- Public Infrastructure Services
- Service Wrappers (Enterprise A)
- Service Wrappers (Enterprise X)
- Service Wrappers (Enterprise Y)
- Internal Infrastructure Services
- Process Execution Platform (BPEL)
- Goal-oriented Adaptive Execution Platform (Agents)
- Goal-oriented Adaptive Execution Platform (Agents)
- Active Model Platform (AKM)
- Active Model Platform (AKM)
- Message-Oriented Platform (MQSeries)
- Server-side Component Platform (.NET, J2EE)
- Composed WebService Platform (Web Services)
- Business Process/Agent Active (Business) Model
- Web/Server Component Middleware Process/Agent Middleware Component
- Adaptive Distributed Resource Mgt Platform (P2P)

- Semantic Space
- Web Service Ontology
- Service-Oriented Architecture Ontology
- Reference Ontology

ICT Model Transformations

- CIM
- PIM
- PSIMs

Symbols:
- Metamodel
- Concept
- Relationship
- Correspondence

Metamodels:
- "EA" Metamodel
- SOA Metamodel
- Web Services Metamodel
- Agent Metamodel (AgentMM)
- P2P Metamodel
- Grid Metamodel
**SOA Framework: Process + Applications + Data**

- **Enterprise / Information Workplace**
  - User Interaction module
  - Business process workflow definitions

- **Business Unit Services**
  - Business service
  - Business service
  - Business service
  - Business service

- **Enterprise Services**
  - Business service
  - Business service
  - Business service
  - Business service
  - Utility services

- **Information Fabric**
  - Databases
  - Files
  - Devices

- **Virtualized Data**

---

**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
- 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

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/](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
- WS-Reliable Messaging (WS-RM)
- WS-Transactions
- WS-Choreography (WS-CHORE)
- WS-Common (WS-CONS)
- WS-Interoperability (WS-INT)
- WS-Policy (WS-POL)
- WS-Metadata Exchange (WS-MED)
- 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
- SOAP
- Raw XML
- "Binary"
- EDI
- ebXML

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

(Syntactic) Web Service Interface Description
- WSDL
- XSD

Interaction Sequencing (Co)Constraints
- WS-CHORE

Bindings and Endpoint Descriptions
- WSDL
- XSD

XML Message Schema Definition
- XSD
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

<Model>
name description URL pointers to specifications
Simple Object Access Protocol (SOAP)

Use of SOAP
SOAP envelope

Making a SOAP function call over HTTP
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/supersedes 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

```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 gives contextual support, e.g. like auto-completion, suggestions for elements, etc., as well as validation of the XML document.
Visual language for schema design
Supported by e.g. XMLSpy

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.
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>namespace</td>
<td>As defined in XML Schema specification</td>
</tr>
<tr>
<td></td>
<td></td>
<td>processContents</td>
<td>As defined in XML Schema specification</td>
</tr>
<tr>
<td></td>
<td></td>
<td>values=&quot;skip</td>
<td>lax</td>
</tr>
<tr>
<td></td>
<td></td>
<td>default</td>
<td>As defined in XML Schema specification</td>
</tr>
<tr>
<td></td>
<td></td>
<td>form</td>
<td>Overrides the attributeFormDefault for this schema</td>
</tr>
<tr>
<td></td>
<td></td>
<td>use</td>
<td>As defined in XML Schema specification</td>
</tr>
<tr>
<td></td>
<td></td>
<td>values=&quot;prohibited</td>
<td>optional</td>
</tr>
<tr>
<td></td>
<td></td>
<td>fixed</td>
<td>As defined in XML Schema specification</td>
</tr>
<tr>
<td></td>
<td></td>
<td>form</td>
<td>Overrides the attributeFormDefault for this schema</td>
</tr>
<tr>
<td></td>
<td></td>
<td>values=&quot;qualified</td>
<td>unqualified&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>default</td>
<td>As defined in XML Schema specification</td>
</tr>
<tr>
<td>&lt;&lt;attribute&gt;&gt;</td>
<td>Property</td>
<td>default</td>
<td>As defined in XML Schema specification</td>
</tr>
<tr>
<td></td>
<td></td>
<td>fixed</td>
<td>As defined in XML Schema specification</td>
</tr>
<tr>
<td></td>
<td></td>
<td>form</td>
<td>Overrides the attributeFormDefault for this schema</td>
</tr>
<tr>
<td></td>
<td></td>
<td>use</td>
<td>As defined in XML Schema specification</td>
</tr>
<tr>
<td></td>
<td></td>
<td>values=&quot;prohibited</td>
<td>optional&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>fixed</td>
<td>As defined in XML Schema specification</td>
</tr>
<tr>
<td>&lt;&lt;choice&gt;&gt;</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>&lt;&lt;complexType&gt;&gt;</td>
<td>Class</td>
<td>ComplexType definition generated in XML Schema</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>memberNames</td>
<td>Overrides the package-level default for naming complexType definition</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>Determines whether this element may contain mixed element and character content.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>values=&quot;true</td>
<td>false&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>default=&quot;false&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>modelGroup</td>
<td>Overrides the package-level default model group</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>values=&quot;all</td>
<td>sequence</td>
<td>choice&quot;</td>
</tr>
<tr>
<td>&lt;&lt;element&gt;&gt;</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></td>
<td>anonymousRole</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></td>
<td>values=&quot;true</td>
<td>false&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>default=&quot;false&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>anonymousType</td>
<td>The class type will be anonymous for XML documents generated by the schema</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>values=&quot;true</td>
<td>false&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>default=&quot;false&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>form</td>
<td>Overrides the elementFormDefault for this schema</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>values=&quot;qualified</td>
<td>unqualified&quot;</td>
<td></td>
</tr>
<tr>
<td>position</td>
<td>If assigned, indicates position in the sequence model group</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;&lt;facet&gt;&gt;</td>
<td>Property</td>
<td>A facet is a single defining aspect of a value space. Generally speaking, each facet</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: Message exchange patterns

- Request-Response
- One-Way
- Solicit-Response
- Notification

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, typed definition of the data being communicated

An abstract, description of an action supported by the service
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 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. • default=&quot;rpc&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>transport</td>
<td>The transport attribute specifies the type of binding to be used. • default=&quot;<a href="http://schemas.xmlsoap.org/soap/http">http://schemas.xmlsoap.org/soap/http</a>&quot;</td>
</tr>
<tr>
<td>&lt;&lt;definition&gt;&gt;</td>
<td>Class</td>
<td>targetNamespace</td>
<td>TargetNamespace is an URI (Uniform Resource Identifier). It is mandatory and identifies the namespace which it will belong all of the component names.</td>
</tr>
<tr>
<td>&lt;&lt;element&gt;&gt;</td>
<td>Class</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 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 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>&lt;&lt;partElement&gt;&gt;</td>
<td>Association</td>
<td></td>
<td>An &lt;&lt;partElement&gt;&gt; association represents a relationship between a part component of the WSDL metamodel and</td>
</tr>
</tbody>
</table>

### UML profile for WSDL (2)

| location | Location is an URI. It is optional and indicates the location of some information for the namespace. |
| namespace | 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. |
| <<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. |
| type | 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. |
| <<partElement>> | Association | An <<partElement>> association represents a relationship between a part component of the WSDL metamodel and |
UML profile for WSDL (3)

**Text representation**

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 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**
- **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

Receive Purchase Order

Initiate Price Calculation

Complete Price Calculation

Decide On Shipper

Arrange Logistics

Complete Production Scheduling

Invoice Processing

PO : POMessage

receive

requestShipping

shipInfo

requestProductScheduling

response

Invoice : InvMessage
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

<message name="POMessage">
    <part name="customerInfo" type="sns:customerInfo"/>
    <part name="purchaseOrder" type="sns:purchaseOrder"/>
</message>

<message name="InvMessage">
    <part name="IVO" 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

<process name="purchaseOrderProcess" xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/">
    <partnerLinks>
        <partnerLink name="purchasing">
            partnerLinkType="lns:purchasingLT"
            myRole="purchaseService"/>
        </partnerLink>
        <partnerLink name="invoicing">
            partnerLinkType="lns:invoicingLT"
            myRole="invoiceRequester"
            partnerRole="invoiceService"/>
        </partnerLink>
        <partnerLink name="shipping">
            partnerLinkType="lns:shippingLT"
            myRole="shippingRequester"
            partnerRole="shippingService"/>
        </partnerLink>
        <partnerLink name="scheduling">
            partnerLinkType="lns:schedulingLT"
            partnerRole="schedulingService"/>
    </partnerLinks>

</process>
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>
      ...  
  </flow>
</sequence>

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>&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 then 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>
<tr>
<td>&lt;&lt;data&gt;&gt;</td>
<td>Class</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>Package</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>Action</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>Action</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>Class</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 (2)

<table>
<thead>
<tr>
<th>Stereotype</th>
<th>UML construct</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;&lt;data&gt;&gt;</td>
<td>Class</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>Package</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>Action</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>Action</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>Class</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>
### SensorWeb

- Web based access and modeling of Sensors
- Relate to Oblig2 and sensor models?
- See [www.opengis.org](http://www.opengis.org)
References