INF5120
"Modellbasert Systemutvikling"
"Modelbased System development"

Arne-Jørgen Berre

arneb@ifi.uio.no or Arne.J.Berre@sintef.no

INF5120 - Lecture plan - 2012

- Part I: SSI - Service Innovation and Agile Service/Software Engineering
- Part II: SSMDE - Model Driven Engineering
- Part III - Model Driven Interoperability and ADM

1: 16/1: Introduction to Model Based System Development (INF5120)
2: 23/1: SIE I: Enterprise Architecture, Role modeling-Collaboration and Value Networks - Verna Allee (VNA)
3: 30/1: SIE II: Business Process Modeling with BPMN 2.0 and Business Model Innovation - Peter Lindgren (BMI)
4: 6/2: SIE III: AT ONE - User-oriented design - with Use cases and user stories
5: 13/2: SIE IV: Service modeling with SoaML - Service modeling - Design, patterns
6: 20/2: SIE V: Precise Modeling in UML with OCL and Design with DCI - Design, patterns
7: 27/2: MDE I: Software Process Model Frameworks - Essence/SEMAT, SPEM, EFP and ISO 24744 - Shihong Huang/Brian Elvesæter/Arne J. Berre
8: 5/3: MDE II: Metamodels, Domain specific languages and UML profiles (Franck Fleurey, Brian Elvesæter)
9: 12/3: MDE III: Metamodeling, MDLE and DSL Tools (EMF, GMF, ATL, Kermeta) (Franck Fleurey)
10: 19/3: MDE IV: Model transformations - MOFScript, QVT DSLs with examples (Franck Fleurey)
11: 26/3: MDE V: Method Engineering and CORAS UML profile - DSL example (Arne J. Berre)
12: 2/4: EASTER
14: 16/4: MDE VII: Information Models, Semantics and Model Driven Interoperability (AJB, Dumitru Roman)
15: 23/4: MDE VIII: Model Driven Interoperability, ADM and MDA/Migration to SOA/SaaS/Cloud
16: 30/4: MDI I: SOA technologies, MDE for SOA (Accenture/NAV) (AJB, Ismar Slomic, Øystein Gisnås)
17: 7/5: MDI II: Model Driven Interoperability, ADM and MDA/Migration to SOA/SaaS/Cloud
18: 14/5: MDI III: Preparation of Exam

Exam: Monday June 4th, 2011, 1430-1830 (4 hours)
INF5120 – Oblig/Exercise plan - 2012

1: 16/1: None
2: 23/1: Guest lecture: Value Networks – Verna Allee (VNA)
3: 30/1: Guest lecture: Business Model Innovation - Peter Landgren (BMI) – Establish groups
4: 6/2: AT ONE initial exercise – overall approach for Oblig 1 – “myServiceFellow”
5: 13/2: Group presentation
6: 20/2: Group presentation
7: 27/2: Group presentation
8: 5/3: MDE Tools – introduction – Oblig 2 intro
9: 12/3: MDE Tools II - EMF
10: 21/3: MDE Transformation tools - Delivery of Oblig 1
11: 26/3: Walk through of Oblig 1
12/4, 9/4: EASTER
12: 16/4: Discussion on Oblig 2 – part 1 and 2
13: 23/4: Discussion on Oblig 2 - part 2 and 3
14: 30/4: Discussion on Oblig 2 - part 3
14: 7/5: Oblig 2 delivery (!)
15: 7/5: Oblig 2 summary
16: 13/5: Conclusion and Summary for INF5120 - Preparation of Exam

Exam: Monday June 4th, 2011, 1430-1830 (4 hours)

Outline

- Service-oriented architecture (SOA)
- EA and MDD for SOA & SOA platforms
- Web services architecture
  - UDDI, SOA, XS, WSDL, BPEL
- SOA Design Patterns
- MDE and SOA in practice (Accenture and NAV)
- AICIT (Accenture Innovation Center for IBM Technology)
- NAV – “The Pension program project”
- Ismar Slomic, Øystein Gisnås
OMG Modeling languages and Zachman Framework

<table>
<thead>
<tr>
<th>Scope (Contexts)</th>
<th>Business (Concepts)</th>
<th>System (Logic)</th>
<th>Technology (Physics)</th>
<th>Component (Assemblies)</th>
<th>Operation (Instances)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SBVR</td>
<td>SBVR</td>
<td>VDM</td>
<td>OSM</td>
<td>BMM</td>
<td>BPIM</td>
</tr>
<tr>
<td><strong>OMG Modeling languages</strong></td>
<td><strong>Zachman Framework</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data (What)</td>
<td>Function (How)</td>
<td>Network (Where)</td>
<td>People (Who)</td>
<td>Time (When)</td>
<td>Motivation (Why)</td>
</tr>
<tr>
<td>Data</td>
<td>Function</td>
<td>Network</td>
<td>People</td>
<td>Time</td>
<td>Motivation</td>
</tr>
<tr>
<td>People</td>
<td>Time</td>
<td>Motivation</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Semantic Model</strong></td>
<td><strong>Business Process Model</strong></td>
<td><strong>Business Logistics</strong></td>
<td><strong>System Workflow Model</strong></td>
<td><strong>Master Schedule</strong></td>
<td><strong>Business Plan</strong></td>
</tr>
<tr>
<td><strong>Logical Data Model</strong></td>
<td><strong>Application Architecture</strong></td>
<td><strong>Distributed System Architecture</strong></td>
<td><strong>Human Interface Architecture</strong></td>
<td><strong>Process Structure</strong></td>
<td><strong>Business Rule Model</strong></td>
</tr>
<tr>
<td><strong>Physical Data Model</strong></td>
<td><strong>System Design</strong></td>
<td><strong>Technology Architecture</strong></td>
<td><strong>Presentation Architecture</strong></td>
<td><strong>Rule Design</strong></td>
<td><strong>Data Definition</strong></td>
</tr>
</tbody>
</table>

CIM-PIM-PSM Reference Matrix

<table>
<thead>
<tr>
<th>Aspect / Level</th>
<th>Information</th>
<th>Service</th>
<th>Process</th>
<th>Rules</th>
<th>Events</th>
<th>Organization</th>
<th>Goals</th>
<th>NFA</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIM</td>
<td>OMG (CIM-ODM)</td>
<td>BPMN FMC</td>
<td>SBVR EFC</td>
<td>BPMN</td>
<td>OSM</td>
<td>SBVR</td>
<td>SBVR</td>
<td>SBVR</td>
</tr>
<tr>
<td>PIM</td>
<td>OMG (PIM-PSM)</td>
<td>OMG (PIM-PSM)</td>
<td>OMG (PIM-PSM)</td>
<td>OMG (PIM-PSM)</td>
<td>OMG (PIM-PSM)</td>
<td>OMG (PIM-PSM)</td>
<td>OMG (PIM-PSM)</td>
<td>OMG (PIM-PSM)</td>
</tr>
<tr>
<td>PSIM</td>
<td>OMG (PSM-PSM)</td>
<td>OMG (PSM-PSM)</td>
<td>OMG (PSM-PSM)</td>
<td>OMG (PSM-PSM)</td>
<td>OMG (PSM-PSM)</td>
<td>OMG (PSM-PSM)</td>
<td>OMG (PSM-PSM)</td>
<td>OMG (PSM-PSM)</td>
</tr>
</tbody>
</table>
with INF5120 Modeling languages

Requirements
- Context and Goals
  - Use cases/stories
  - Role
  - Interaction
  - Structure
  - Function
  - Coordination
  - Information
  - Quality

Design
- UI IFML WebML
  - Models SoaML
    - Roles collaboration
  - BPMN
    - Processes
  - UML
    - Class
    - Information
    - Ontologies
  - OCL
    - EFA
    - Extra Functional Aspects

Implementation
- Interface
  - Actors
  - Tasks
  - Orchestration
  - Model Driven Architecture/MDE
  - Model Driven Architecture/MDE
  - Model Driven Architecture/MDE
  - Model Driven Architecture/MDE

Infrastructure
- Channels
  - Resources
  - Executors
  - Workflows
  - Stores and Messages
  - Monitoring, adaptation

Technologies/Realisation-K
- XML, OWL, BPEL/XPDL, WSDL, OWL-S/WSML, SWRL, Security, QoS

Legacy and New systems/services, ERPs/ESAs

BPDM, SBVR, EDOC, UPMS, PIM4SOA, ODM

BPDM, POP*, ARIS, ArchiMate, GEFERAM, GRAI, Zachman, UML, B.Rules...

UML profiles and metamodels for BPEL, WSDL, XML, XPDL, OWL-S, WSML, WSDL-S

Technologies Reliability Code

UI

Information
- Process
  - Services
  - Rules
  - NFA

Data
- Wflow/Comp
  - Interfaces
  - Rules
  - NFA

CIM-K
- Ontologies
  - Bus.Process
  - Bus.Rules

Goals NFA/Qualities Or

BPMN, POP*, ARIS, ArchiMate, GEFERAM, GRAI, Zachman, UML, B.Rules...
Service-oriented architecture (SOA)

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.

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**

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

MDD for SOA & SOA platforms
Modelling approach for SOA

Model transformations

“EA” Metamodel

PIM

SOA Metamodel

PSM

Web Services Metamodel
Agent Metamodel (AgentMM)
P2P Metamodel
Grid Metamodel

Symbols
- Metamodel
- Concept
- Relationship
- - Correspondence

Model transformations

CIM
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, …
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

Business process management suite & interaction services

- Goal: Continuous process improvement
- Components: BPM
  - human-centric: people-intensive processes
  - Integration-centric: system-intensive processes
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.
### Web services stack

**Conceptual stack**
- Service Composition
- Composable Service Assurance
- Description
- Messaging
- Transports

**Technology stack**
- WS-BPEL
- WS-Security
- Web Service Reliable Messaging (WS-RM)
- WS-Transactions
- XSD
- WSDL
- UDDI
- WS-Policy
- WS-Metadata Exchange
- XML
- SOAP
- WS-Addressing
- HTTP
- HTTPS
- SMTP

### Web services – a conceptual view

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

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

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.

---

**Simple Object Access Protocol (SOAP) and RESTful Services**
Use of SOAP

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.

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

---

**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.
### 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 relaced 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></td>
<td>namespace</td>
<td>As defined in XML Schema specification</td>
<td></td>
</tr>
<tr>
<td></td>
<td>processContents</td>
<td>As defined in XML Schema specification • values=&quot;skip</td>
<td>lex</td>
</tr>
<tr>
<td>&lt;&lt;attribute&gt;&gt;</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></td>
<td>default</td>
<td>As defined in XML Schema specification</td>
<td></td>
</tr>
<tr>
<td></td>
<td>fixed</td>
<td>As defined in XML Schema specification</td>
<td></td>
</tr>
<tr>
<td></td>
<td>form</td>
<td>Overrides the attributeFormDefault for this schema • values=&quot;qualified</td>
<td>unqualified&quot;</td>
</tr>
<tr>
<td></td>
<td>use</td>
<td>As defined in XML Schema specification • values=&quot;prohibited</td>
<td>optional</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>memberNames</td>
<td>Overrides the package-level default for naming complexType definitions</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. • values=&quot;true</td>
<td>false&quot; • default=&quot;false&quot;</td>
</tr>
<tr>
<td>modelGroup</td>
<td></td>
<td>Overrides the package-level default model group • values=&quot;all</td>
<td>sequence</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>anonymousRole</td>
<td></td>
<td>The class type will be directly embedded within the complexType definition. Omit attribute or role type wrapper • values=&quot;true</td>
<td>false&quot; • default=&quot;false&quot;</td>
</tr>
<tr>
<td>anonymousType</td>
<td></td>
<td>The class type will be anonymous for XML documents generated by the schema • values=&quot;true</td>
<td>false&quot; • default=&quot;false&quot;</td>
</tr>
<tr>
<td>form</td>
<td></td>
<td>Overrides the elementFormDefault for this schema • 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>&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: Conceptual view

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

(Syntactic) Web Service Interface Description

Bindings and Endpoint Descriptions

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

- SOAP
- Raw XML
- "Binary"
- EDI
- ebXML

WSDL: Conceptual model

- WS Client
- Operations Invoked through Ports
- Operation
  - Name, Abstract Message Parts Schema
  - Message Exchange Pattern

- WS Interface
- Ports
- Concrete Endpoint Address

- Porttype
- Operation

- (Reusable) Binding
- Concrete Message Encoding
  - Concrete Messaging Protocol
**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;&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;&lt;binding&gt;&gt; class represents a binding component of the WSDL metamodel.</td>
</tr>
<tr>
<td></td>
<td>binding</td>
<td>Binding type</td>
<td>default=&quot;soap:binding&quot;</td>
</tr>
<tr>
<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>
<td>default=&quot;rpc&quot;</td>
</tr>
<tr>
<td></td>
<td>transport</td>
<td>The transport attribute specifies the type of binding to be used.</td>
<td>default=&quot;<a href="http://schemas.xmlsoap.org/soap/http">http://schemas.xmlsoap.org/soap/http</a>&quot;</td>
</tr>
<tr>
<td>&lt;&lt;&lt;definition&gt;&gt;</td>
<td>Class</td>
<td></td>
<td>A &lt;&lt;&lt;definition&gt;&gt; class represents a definition component of the WSDL metamodel.</td>
</tr>
<tr>
<td></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>
<td></td>
</tr>
<tr>
<td>&lt;&lt;&lt;element&gt;&gt;</td>
<td>Class</td>
<td></td>
<td>An &lt;&lt;&lt;element&gt;&gt; class represents an element of the XML Schema.</td>
</tr>
<tr>
<td></td>
<td>baseType</td>
<td>The base type</td>
<td></td>
</tr>
<tr>
<td></td>
<td>minOccurs</td>
<td>The minimum number of occurrences</td>
<td></td>
</tr>
<tr>
<td></td>
<td>maxOccurs</td>
<td>The maximum number of occurrences</td>
<td></td>
</tr>
<tr>
<td></td>
<td>name</td>
<td>The name of the element</td>
<td></td>
</tr>
<tr>
<td>&lt;&lt;&lt;fault&gt;&gt;</td>
<td>Association</td>
<td></td>
<td>An &lt;&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)

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>location</td>
<td>Location is an URI. It is optional and indicates the location of some information for the namespace.</td>
</tr>
<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>
<tr>
<td>&lt;&lt;&lt;input&gt;&gt;</td>
<td>Association</td>
</tr>
<tr>
<td>&lt;&lt;&lt;message&gt;&gt;</td>
<td>Class</td>
</tr>
<tr>
<td>&lt;&lt;&lt;operation&gt;&gt;</td>
<td>Class</td>
</tr>
<tr>
<td>&lt;&lt;&lt;output&gt;&gt;</td>
<td>Association</td>
</tr>
<tr>
<td>&lt;&lt;&lt;part&gt;&gt;</td>
<td>Class</td>
</tr>
<tr>
<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>
</tbody>
</table>
| <<<partElement>> | Association | A <<<partElement>> association represents a relationship between a part component of the WSDL metamodel and...
Web Services Business Process Execution Language (WS-BPEL)

- Direct relationship to BPMN
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 with 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.

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

<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

<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>
  ...
</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>

...
References

Principles of Service Design

- Design principles
- Standardised Service Contracts
- Service Loose Coupling
- Service Abstraction
- Service Reusability
- Service Autonomy
- Service Statelessness
- Service Discoverability
- Service Composability
Service-Orientation and Object-Orientation (T. Erl)

- Service-Orientation and Object-Orientation Part I: A Comparison of Goals and Concepts
- Service-Orientation and Object-Orientation Part II: A Comparison of Design Principles

---

While object-orientation evolved out of approaches that included procedural programming, service-orientation builds upon the object-oriented design paradigm and, together with additional influences, establishes a distinct paradigm of its own.
Vendor-agnostic context pattern

<table>
<thead>
<tr>
<th>Problem</th>
</tr>
</thead>
<tbody>
<tr>
<td>An architectural model based on a single vendor platform can constrain an enterprise to the confines of that platform, resulting in constant limitations that may inhibit business requirements fulfillment.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>The model behind the technology architecture is designed independently from but still in alignment with primary vendor platforms.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Application</th>
<th>Impacts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vendor-agnostic patterns and principles are applied to the design of the services, the service inventory, and its surrounding architecture.</td>
<td>Establishing a vendor-agnostic model increases design and governance effort while reducing the availability of proprietary features.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Principles</th>
<th>Architecture</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service Abstraction, Service Reusability, Service Composability, Service Loose Coupling</td>
<td>Inventory, Enterprise</td>
</tr>
</tbody>
</table>