Dynamic and Agile SOA using SAWSDL

Karthik Gomadam  
*Wright State University - Main Campus*

Kunal Verma  
*Wright State University - Main Campus*

Amit P. Sheth  
*Wright State University - Main Campus, amit@sc.edu*

Follow this and additional works at: [https://corescholar.libraries.wright.edu/knoesis](https://corescholar.libraries.wright.edu/knoesis)

Part of the Bioinformatics Commons, Communication Technology and New Media Commons, Databases and Information Systems Commons, OS and Networks Commons, and the Science and Technology Studies Commons

Repository Citation  
[https://corescholar.libraries.wright.edu/knoesis/759](https://corescholar.libraries.wright.edu/knoesis/759)

This Presentation is brought to you for free and open access by the The Ohio Center of Excellence in Knowledge-Enabled Computing (Kno.e.sis) at CORE Scholar. It has been accepted for inclusion in Kno.e.sis Publications by an authorized administrator of CORE Scholar. For more information, please contact library-corescholar@wright.edu.
Dynamic and Agile SOA using SAWSDL

Karthik Gomadam\textsuperscript{1}, Kunal Verma\textsuperscript{2} and Amit P. Sheth\textsuperscript{1}

\textsuperscript{1} Services Research Lab, kno.e.sis center, Dayton, OH

\textsuperscript{2} Accenture Technology Labs, Palo Alto, CA
Introduction

• Organizations are often involved in complex business transactions with various partners across the world
  – For example, the business decisions are made in the US, technical and support services are in India and suppliers come from China.

• Variety of factors can affect the business objectives of an organization.

• Business processes need to be more agile and dynamic
Initially, Supplier 1 is cheaper. Change in Chinese Currency Makes supplier two cheaper!!

If the manufacturer cannot relate to this change and react to it, the process of part procurement will be sub-optimal.

The change in system, however, must be done by Technical Services partner in India.

CHALLENGE is to:
1. Create enactment consistent with business objectives
2. Correlate and reflect changes across different participating entities
3. Be able to create agile and dynamic processes
Another Example

- Production flow (a)
- Production flow (b)
- Add a new product line (b)
- Inventory Management (c)
The Hard Problem

• Create partner-level requirements that are consistent with those of the business process
• Select and configure the partners at runtime
• Identify and adapt efficiently to the various events that affect the optimality of the business process
Path to Agility Nirvana

- SAWSDL and Semantic Templates
  - Main memory representation for SAWSDL and Semantic Templates
- Dynamic Business Process
- Experiences in extending open source frameworks to support SAWSDL
Semantics in Services: 4 of a Kind Jackpot

- What does the service offer?
- Response time, Cost, QoS Metrics
- When things go south
- Not just model, Express your data
- What does the service offer?
A Four Tiered Approach

- Business Specification Tier
- Process Enactment Tier
- Partner Services Tier
- Infrastructure Tier
What Semantics, Where?

Amit P. Sheth, Karthik Gomadam: The 4 × 4 semantic model - exploiting data, functional, non-functional and execution semantics across business process, workflow, partner services and middleware services tiers. ICEIS (1) 2007: 1-4
Why Semantics?

• Specification Tier:
  – Need to capture the functional and non-functional specification.
    • Functional and non-functional semantics.

• Enactment Tier:
  – Captures the data flow, control flow and the partner level specifications.
  – Also addresses adaptation.
    • Four types of semantics.
Why Semantics?

• Partner Services Tier:
  – Description of partner services including inputs, outputs, functional and the non-functional guarantees and requirements.
    • Data, Functional and Non-Functional semantics

• Middleware Tier:
  – Middleware capabilities and the policies
  – Data mediation as a middleware level service.
  – Adaptation capabilities must be built into middleware.
    • All four semantics are needed at this level.
Over the Next Three Hours

• Representation
  – SAWSDL
  – Semantic Template

• Manipulation
  – SAWSDL4J
  – ST4J

• Utilization
  – jUDDI (Discovery)
  – Apache Synapse (Dynamic Binding)
**SAWSDL : The facts**

- **Standard Activity**
  - W3C SAWSDL Working group
    - [http://www.w3.org/2002/ws/sawsdl/](http://www.w3.org/2002/ws/sawsdl/)
  - W3C WSDL-S member submission Web page
    - [http://www.w3.org/Submission/WSDL-S/](http://www.w3.org/Submission/WSDL-S/)

- **Tools**
  - Radiant: WSDL-S/SAWSDL Annotation Tool by University of Georgia
  - Semantic Tools for Web Services by IBM alphaWorks
  - WSMO Studio by DERI

- **Some Relevant Papers**
SAWSDL: The Objectives

- Offer an evolutionary and compatible upgrade of existing Web services standards
- Externalize the semantic domain models
  - agnostic to ontology representation languages (although W3C recommended RDFS or OWL are likely to be often used)
  - reuse of existing domain models (in some domains, usable ontologies have been built, eg life science and health care)
  - allows annotation using multiple ontologies (same or different domain)
Guiding principles...

• Support semantic annotation of Web Services whose data types are described in XML schema

• Provide support for rich mapping mechanisms between Web Service schema types and ontologies
Why use SAWSDL

• Build on existing Web Services standards using only extensibility elements

• Mechanism independent of the semantic representation language (though OWL is supported well)

• SAWSDL provides an elegant solution
  – Help integration by providing mapping to agreed upon domain models (ontologies, standards like Rosetta Net, ebXML)
  – Better documentation by adding functional annotation

• Ease in tool and framework upgrades
  – e.g. woden, WSDL4J, JUDDI, Neethi…
SAWSDL Scope

No SAWSDL annotations defined for these WSDL components

Annotated using modelReference
Annotated using modelReference and schemaMapping
SAWSDL at a Glance

Semantics:
- ontology classes
  - discovery, composition
  - filtering, ranking
- lifting/lowering mappings
  - mediation, invocation
- functionality categories
  - publishing, discovery, composition
- anything, really

Image Courtesy:
• **modelReference**: This can be used to specify the association between a WSDL or XML Schema component and a concept in some semantic model.
  – It can be used to annotate the following:
    • WSDL components
      – Interfaces
      – Operations
      – faults
    • WSDL Type Definitions
      – XML Schema complex type definitions
      – Simple type definitions
      – element declarations
      – attribute declarations

• **liftingSchemaMapping**: This can be used to specify mappings between WSDL Type Definitions in XML and semantic data.
• **loweringSchemaMapping**: This can be used to specify mappings between semantic data and WSDL Type Definitions in XML.
The annotation of the operation element carries a reference to a concept in a semantic model that provides a high level description of the operation, specifies its behavioral aspects or includes other semantic definitions.
Using modelReference to annotate faults

The annotation of the fault element carries a reference to a concept in a semantic model that provides a high level description of the fault and can include other semantic definitions.

```xml
<wSDL:interface name="Order">
  <wSDL:fault name="ItemUnavailableFault" element="AvailabilityInformation"
</wSDL:interface>
```
Annotating Types

- SAWSDL specification allows annotation of XML schema elements
  - XML Schema complex type definitions
    - Bottom-level annotation
    - Top level annotation
  - Simple type definitions
  - element declarations
  - attribute declarations
Annotating types

1. **modelReference** to establish a semantic association
2. **liftingSchemaMapping and loweringSchemaMapping** to provide mappings between XML and semantic model
<complexType name="POItem">
  <all>
    <element name="dueDate" nillable="true" type="dateTime"
      sawsdl:modelReference=""
      http://www.w3.org/2002/ws/sawsdl/spec/ontology/purchaseorder#DueDate"/>
    <element name="qty" type="float"
      sawsdl:modelReference=""
      http://www.w3.org/2002/ws/sawsdl/spec/ontology/purchaseorder#Quantity"/>
    <element name="EANCode" nillable="true" type="string"
      sawsdl:modelReference=""
      http://www.w3.org/2002/ws/sawsdl/spec/ontology/purchaseorder#ItemCode"/>
    <element name="itemDesc" nillable="true" type="string"
      sawsdl:modelReference=""
      http://www.w3.org/2002/ws/sawsdl/spec/ontology/purchaseorder#ItemDesc"/>
  </all>
</complexType>
<complexType name="POItem"
  sawsdl:modelReference="http://www.w3.org/2002/ws/sawsdl/spec/ontology/purchaseorder#DueDate">
  <all>
    <element name="dueDate" nillable="true" type="dateTime" />
    <element name="qty" type="float" />
    <element name="EANCode" nillable="true" type="string" />
    <element name="itemDesc" nillable="true" type="string" />
  </all>
</complexType>
Any mapping language can be used for **liftingSchemaMapping**

- Recommended languages: XSLT, Xquery

Any mapping language can be used for **liftingSchemaMapping**

- Recommended languages: SPARQL to query ontology, followed by XSLT, Xquery

```xml
<complexType name="POAddress"
  sawsd1:modelReference="http://www.w3.org/2002/ws/sawsdl/spec/ontology/purchaseorder#Address"
  sawsd1:liftingSchemaMapping="http://www.w3.org/2002/ws/sawsdl/spec/mapping/POAddress2Ont.xslt"
  sawsd1:loweringSchemaMapping="http://www.w3.org/2002/ws/sawsdl/spec/mapping/Ont2POAddress.xslt">
  <all>
    <element name="streetAddress" type="xsd:string" />
    <element name="poBox" type="xsd:string" />
    <element name="city" type="xsd:string" />
    <element name="zipCode" type="xsd:string" />
    <element name="state" type="xsd:string" />
    <element name="country" type="xsd:string" />
    <element name="recipientInstName" type="xsd:string" />
  </all>
</complexType>
```
Lifting SchemaMapping example

<xsl:transform version="2.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:po="http://www.w3.org/2002/ws/sawsdl/spec/wsdl/order#"
xmlns:POOntology="http://www.w3.org/2002/ws/sawsdl/spec/ontology/purchaseorder#">
<xsl:output method="xml" version="1.0" encoding="iso-8859-1" indent="yes" />
<xsl:template match="/">
<POOntology:OrderConfirmation>
<POOntology:Address rdf:ID="Address1">
<POOntology:has_StreetAddress rdf:datatype="xs:string">
<xsl:value-of select="concat(POAddress/streetAddress)"/>
</POOntology:has_StreetAddress>
<POOntology:has_City rdf:datatype="xs:string">
<xsl:value-of select="POAddress/city"/>
</POOntology:has_City>
<POOntology:has_Zip rdf:datatype="xs:string">
<xsl:value-of select="POAddress/zip"/>
</POOntology:has_State>
</xsl:template>
</xsl:transform>
Using schemaMapping with modelReference (heterogeneity)

```xml
<complexType name="POAddress"
  sawsdl:modelReference="http://www.w3.org/2002/ws/sawsdl/spec/ontology/purchaseorder#Addresd"
  sawsdl:loweringSchemaMapping="http://www.w3.org/2002/ws/sawsdl/spec/mapping/Ont2POAddress.xslt">
  <all>
    <element name="streetAddr1" type="xsd:string"/>
    <element name="streetAddr2" type="xsd:string"/>
    <element name="poBox" type="xsd:string"/>
    <element name="city" type="xsd:string"/>
    <element name="zipCode" type="xsd:string"/>
    <element name="state" type="xsd:string"/>
    <element name="country" type="xsd:string"/>
    <element name="recipientInstName" type="xsd:string"/>
  </all>
</complexType>
```

Data level heterogeneity

OWL ontology

WSDL complex type element
<xsl:transform version="2.0"
    <xsl:output method="xml" version="1.0" encoding="iso-8859-1" indent="yes" />
    <xsl:template match="/">
        <POOntology:OrderConfirmation>
            <POOntology:Address rdf:ID="Address1">
                <POOntology:has_StreetAddress rdf:datatype="xs:string">
                    <xsl:value-of select="concat(POAddress/streetAddr1, POAddress/streetAddr2)"/>
                </POOntology:has_StreetAddress>
                <POOntology:has_City rdf:datatype="xs:string">
                    <xsl:value-of select="POAddress/city"/>
                </POOntology:has_City>
            </POOntology:Address>
        </POOntology:OrderConfirmation>
    </xsl:template>
</xsl:transform>
Using modelReference to annotate interfaces

A `modelReference` on a WSDL `interface` element provides a reference to a concept or concepts in a semantic model that describe the Interface.
<wsdl:description targetNamespace="http://www.w3.org/2002/ws/sawsdl/spec/wsdl/order#"
    xmlns:sawsdl="http://www.w3.org/2002/ws/sawsdl/spec/wsdl">
    <wsdl:types>
        <xs:element name="processPurchaseOrderResponse" type="xs:string"
            sawsdl:modelReference="http://www.w3.org/2002/ws/sawsdl/spec/ontology/rosetta#PurchaseOrderResponse"
            ......
        </xs:element>
    </wsdl:types>
    <interface name="PurchaseOrder"
        sawsdl:modelReference="http://example.org/categorization/products/electronics" />
    <operation name="order" pattern=wsdl:in-out
        sawsdl:modelReference="http://www.w3.org/2002/ws/sawsdl/spec/ontology/rosetta#RequestPurchaseOrder"">
        <input messageLabel = "processPurchaseOrderRequest"
            element="tns:processPurchaseOrderRequest"/>
        <output messageLabel = "processPurchaseOrderResponse"
            element="processPurchaseOrderResponse"/>
    </operation>
    <operation name="cancel" pattern=wsdl:in-out
        sawsdl:modelReference="http://www.w3.org/2002/ws/sawsdl/spec/ontology/rosetta#CancelOrder" />
</wsdl:description>
Moving on…

• Representation
  – SAWSDL
  – Semantic Template

• Manipulation
  – SAWSDL4J
  – ST4J

• Utilization
  – jUDDI (Discovery)
  – Apache Synapse (Dynamic Binding)
Semantic templates

• A way of capturing data / functional / non-functional / execution semantics
• Techniques for adding semantics follow SAWSDL principles
Semantic Template: Key Concepts

• Template Term
  – Functional requirement (as Operation)
  – Data requirement (as Inputs and Outputs)

• Term Policy
  – Non-Functional requirement
  – Assertions and constraints
Structural Example

Semantic Template

Template Term
Domain: NAICS#ComputerRAM

Term policy
QoS#Supplier Status = Preferred

Term policy
QoS#ResponseTime < 10 Seconds

Operation
Rosetta#RequestPurchaseOrder
Input: Rosetta#PurchaseOrderInput
Output: Rosetta#PurchaseOrder

Operation Policy
QOS#SupplyTime < 5 days
QOS#Cost < 100000

Operation
Rosetta#CancelOrder
Input: Rosetta#CancelOrderInput
Output: Rosetta#CancelOrderOutput

Operation Policy
QOS#CancellationFee <= 5%
Annotation Example

Semantic Template

ServiceLevelMetaData (SLM)
Category= NAICS:Electronics
ProductCategory= EUNIS:RAM
Location= Athens, GA

SemanticOperation Template (SOTP1)
Action= Rosetta:RequestPurchaseOrder
Input= Rosetta:PurchaseOrder_Input
Output= Rosetta:PurchaseOrder_Output
OLP= {Encryption = RSA, ResponseTime< 5 Sec}

SemanticOperation Template (SOTP2)
Action= Rosetta:CancelOrder
Input= Rosetta:CancelOrder_Input
Output= Rosetta:CancelOrder_Output
OLP= {Encryption = RSA, ResponseTime< 5 Sec}
Part 2: Manipulation

• We will now discuss two object models
  – SAWSDL4J
  – ST4J

• Allows us to create main memory SAWSDL and Semantic Template Objects
SAWSDL4J

• Clean object model for SAWSDL documents
• *Extends* the WSDL4J object model
• Loosely coupled with WSDL4J
  – EX: Used with Woden to support WSDL 2.0
• *ModelReference* as a first class object
Starting up - what you need

- SAWSDL4J libraries
- WSDL4J / Woden libraries
- Jaxen
  - Xpath support
- If building from source
  - Maven
- Get it from
Using SAWSDL4J

• Handling ModelReference
  – Get and set modelReference methods for interface/PortType, Operation elements

• Schema using XPath

```java
Types types = sawsdlDefinition.getTypes();
List < Schema > schemaList = SchemaUtils.getSchemas(types);
Schema s = schemaList.get(0);
try {
  Set < ModelReference> modelReferences =
  s.getModelReferences(s.getContaining(), "//xsd:schema/xsd:element[@name="OrderRequest"]", sawsdlDefinition);
```
ST4J: Semantic Template4J

- Clean object model for semantic templates
- Uses
  - SAWSDL4J and dependencies
- Allows
  - Creating template terms, operations
  - Externalizing data elements
Zero Sum Impact

• Manipulation models allow us to plug SAWSDL / Semantic templates into existing frameworks.

• In past we have extended
  – jUDDI
    • SAWSDL Publication and Discovery
  – Apache Synapse
    • Dynamic Binding
  – Apache Axis 2.0
    • Data Mediation
The Last Leg

• Representation
  – SAWSDL
  – Semantic Template

• Manipulation
  – SAWSDL4J
  – ST4J

• Utilization
  – jUDDI (Discovery)
  – Apache Synapse (Dynamic Binding)
SAWSDL in Everyday Life 😊

• Guiding principle
  – Evolution
  – Hospitality
  – Consistency
Enhancing UDDI

- Objectives
  - Support discovery and publication of semantic Web services
  - Use SAWSDL for service description
  - Use Semantic Templates for requirements description
Two Approaches

• Semantic layer over UDDI
  – Accommodate additional semantic information in existing UDDI data structures

• Semantic extensions to UDDI
  – Extend UDDI data structures to capture the semantics more “natively”
The Layering Approach

• Discussed in detail by
  – Verma Et al. METEOR-S Web Service Discovery Infrastructure
  – Sivashanmugam Et al.
  – Paolucci Et al.
Mapping WSDL-S (SAWSDL) to UDDI

WSDL-S Web Service

Service Level Metadata

Collection of semantically defined operations (Sop)

Operation 1

Operation: FunctionalConcept
Input: SemanticType
Output: SemanticType

Operation 2

Operation: FunctionalConcept
Input: SemanticType
Output: SemanticType

Business Service

CategoryBag

Type = service
name = localName
SLM pairs

BindingTemplate

Operation1 = x
Operation2 = y

Tmodel(id = x)

CategoryBag

Type = Operation
Operation = FunctionalConcept
Input = SemanticType
Output = SemanticType

Tmodel(id = y)

CategoryBag

Type = Operation
Operation = FunctionalConcept
Input = SemanticType
Output = SemanticType
Extending UDDI

• SEMRE
  – Research currently in progress
  – Open source download at
    • http://knoesis.wright.edu/research/srl/oss/semre
Extending UDDI

- Registry integrated with a semantic store
- Separate constructs to model
  - Semantic annotations
  - Relationships
- Ability to pre-compute relationships between interfaces
- Use native SW querying and reasoning frameworks in the registry
And it is Fairly Efficient to Publish

Service Publication Time

- With Interfaces published
- Without Interfaces

Time - Milliseconds

Number of services
And to Discover
SEMRE: Quick Facts

• Extends jUDDI source
• Integrated with JENA
  – Supports SPARQL and SPARUL
• Semantic Registry
  – Services are published under semantic interfaces
    • Not syntactic Tmodel compliance
i-bind: Dynamic Binding Framework

- **i-bind**
  - Enables dynamic binding
  - Uses Semantic Template and SAWSDL
  - Extends Apache Synapse
Apache Synapse

• What is Synapse?
  – Lightweight, High performance
  – Enterprise Service Bus (ESB)
  – Transports and Mediators
  – Integration, Gateway
  – HTTP, SOAP, SMTP, JMS, FTP Transports
  – WS- Addressing and WS-Security
  – Non-blocking HTTPS
Starting off with i-bind

• Create semantic templates for requirements
  – Semantic templates can be saved and re-used

• Deploy semantic templates
  – Returns a virtual endpoint
What is a Virtual Endpoint?

- Mechanism to externalize the semantics of requirements from a process
- Templates can be included in processes
  - Process engines "invoke" the template via virtual endpoint
- The actual partner EPR is "bound" to the virtual point by i-bind
Virtual Endpoint in Action

BPEL Engine  i-bind  Actual Service

Service Provider
i-bind and Registry

**BPEL Engine**

i-bind

SEMRE

**Actual Service**
• Static Binding
  – Discover partners during design time
  – Bind Templates to Process
  – All instances use the same set of partners
• Dynamic Binding
  – Create process with templates
  – Find-N-Bind during runtime
Code Walkthroughs

• SAWSDL4J
• ST4J
  – Hands on for the daring
• Synapse Modification
  – Code walkthrough of existing system
• SEMRE
  – Code and design walkthrough