Semantics to Empower Services Science: Using Semantics at Middleware, Web Services and Business Levels

Amit P. Sheth

Wright State University - Main Campus, amit.sheth@wright.edu

Follow this and additional works at: http://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


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 corescholar@www.libraries.wright.edu.
THE 4 X 4 SEMANTIC MODEL

Amit Sheth*
Kno.e.sis center,
Wright State University,
Dayton, OH

* with Karthik Gomadam
Outline

• Motivation
• The Four Tiers
  – Modeling, Enactment, Partner Services and Execution
• The Four Types Of Semantics
  – Data, Functional, Non-Functional and Execution
• The 4 X 4 Model
  – Unifying the four tiers using the four types of semantics
• The 4 X 4 Model In Action
Motivation

• 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 more agile and dynamic
Motivation

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

Technical Services partner in India.
The Hard Problem

- Create partner-level requirements that are consistent with those of the business process
- Verify the correctness of the enactment with respect to the business process modeling
- Select and configure the partners at run time
- Identify and adapt efficiently to the various events that affect the optimality of the business process
Outline

• Motivation
• The Four Tiers
  – Modeling, Enactment, Partner Services and Execution
• The Four Types Of Semantics
  – Data, Functional, Non-Functional and Execution
• The 4 X 4 Model
  – Unifying the four tiers using the four types of semantics
• The 4 X 4 Model In Action
The Four Tiers

- What do I want to do?

- How am I going to do it?

- Who are my partners in this?

- What is my environment for execution?
The Four Tiers

• Business Specifications Tier (referred to as business process tier in the paper)
  – Functional and non-functional aspects of the business specification are captured at this level.
    Example: Develop a SOA based solution for procuring various components to manufacture gaming hardware requests with the following constraints / requirements
    • Must support XGP graphic processing
    • Minimum 100 Gb disk space
    • Product must never be out of inventory with retailers
    • Level 3 security
• **Workflow Tier**
  - Actual workflow enactment of the specification.
  - Partners based on “What they do” are identified. Not **Who**
    - Example: Partners for the parts ordering specification are
      - Suppliers for Graphics processor, Gaming Chip, Disk drive, Forecasting partner (to give retailer stock information and demand forecasting).
  - Process level specification is broken down into partner level specification
  - What to do when something goes wrong with this enactment (Adaptation and event identification)
• **Workflow Tier** (Contd..)
  – Example partner requirement
  • **Disk Drive Partner:**
    – Must be able to do a purchase order for hard drives
    – Will send PORequest according to Rosetta PO and expect a POResponse conforming to RosettaNet
    – Communication must be over secure 128 bit encrypted channel. (Non-functional requirement)
    – Disk capacity must be at least 100 Gb (non-functional)
The Four Tiers

- **Partner Services Tier**
  - Captures the capabilities and requirements of potential partner services.
  - **Example of Disk drive service**
    - Accept input in Rosetta RequestPO and ebXML RequestPO formats and output in Rosetta POREsponse format (data)
    - Request purchase order for Hard drives (Functional)
    - 128 Bit SSL communication
    - Drives with capacities 80, 100 and 120 Gb
The Four Tiers

- **Middleware Services Tier**
  - Captures the services offered by containing middleware systems.
  - Includes capabilities related to deployment, security, load balancing, message routing and forwarding, service selection and switch, policy based message handling and event management.
Outline

• Motivation
• The Four Tiers Of A Business Process
  – Modeling, Enactment, Partner Services and Execution
• The Four Types Of Semantics
  – Data, Functional, Non-Functional and Execution
• The 4 X 4 Model
  – Unifying the four tiers using the four types of semantics
• The 4 X 4 Model In Action
What does Semantics bring to the table?

• **Better Reuse**
  – Semantic descriptions of services to help find relevant services

• **Better Interoperability**
  – Beyond syntax to semantics, mapping of data exchanged between the services (very time consuming without semantics, just as XML in WSDL gives syntactic interoperability, SAWSDL gives semantic interoperability)

• **Configuration/Composition**
  – Enable dynamic binding of partners

• **Some degree of automation across process lifecycle**
  – Process Configuration (Discovery and Constraint analysis)
  – Process Execution (Addressing run time heterogeneities like data heterogeneities.)
What does Semantics bring to the table?

- **Better Reuse**
  - Semantic descriptions of services to help find relevant services

- **Better Interoperability**
  - Beyond syntax to semantics, mapping of data exchanged between the services (very time consuming without semantics, just as XML in WSDL gives syntactic interoperability, SAWSDL gives semantic interoperability)

- **Configuration/Composition**
  - Enable dynamic binding of partners

- **Some degree of automation across process lifecycle**
  - Process Configuration (Discovery and Constraint analysis)
  - Process Execution (Addressing run time heterogeneities like data heterogeneities.)
Semantics to Web Services: The ingredients

- Conceptual Model/Ontology
  - An agreed upon model that captures the semantics of domain
  - Common Nomenclature
  - Domain Knowledge (facts)

- XML based service description
  - Standards and specifications like WSDL for web service description, WS-Agreement for capturing agreements etc.

- Annotate the service description
SAWSDL at a glance

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

Ack: Jacek Kopecky
Annotating types

1. **modelReference** to establish a semantic association
2. **liftingSchemaMapping** and **loweringSchemaMapping** to provide mappings between XML and semantic model
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 upgrades
  - e.g. wsif / axis invocation
- Is a W3C candidate recommendation
What can we support or demonstrate today

- API for handling SAWSDL documents: SAWSDL4J
- Tool for annotating WSDL services to produce SAWSDL: Radiant and for discovery: Lumina
- Using SAWSDL with UDDI for Discovery: MWSDIr
- Using SAWSDL with Apache Axis for Data Mediation
- Using SAWSDL with WS-BPEL for run-time binding
- Early Examples of SAWSDL annotated services: biomedical research

Also:
- Semantic Tools for Web Services by IBM alphaWorks
- WSMO Studio, more mentioned by Jacek
Modeling: Using Radiant
Execution: WS Discovery using Lumina (MWSDI)
Execution: WS Discovery using Lumina (MWSDI)
Execution: Data Mediation

DATA MEDIATION REQUIRED

OUTPUT FROM WEB SERVICE 1

INPUT TO WEB SERVICE 1

OUTPUT FROM WEB SERVICE 2

INPUT TO WEB SERVICE 1

Address line 1
Address line 2
City_State_zip

Listing Name
First Name
Last Name
Address
City
State
Postal Code
Phone Number
Published

Web service 1
Address Lookup

Web service 2
Geocode Enhancer

Census Track
State Number
County Number
Block Number
Block Group

Telephone Number
- User specified mappings from Web service message element to semantic model concept (say OWL Ontology)
  - upcast: from WS message element to OWL concept
  - Downcast: from OWL concept to WS message element

```xml
<complexType name="Address">
  <sequence>
    <element name="StreetAd1" type="xsd:string"/>
    <element name="StreetAd2" type="xsd:string"/>
    ..........  
  </sequence>
</complexType>
```

```xml
<POOntology:has_StreetAddress rdf:datatype="xs:string">{ fn:concat($a/streetAddr1 , " ", $a/streetAddr2 ) } </POOntology:has_StreetAddress>
```
### Execution: Data Mediation

<table>
<thead>
<tr>
<th>Heterogeneities / Conflicts</th>
<th>Examples - conflicted elements shown in color</th>
<th>Suggestions / Issues in Resolving Heterogeneities</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Domain Incompatibilities</strong></td>
<td><strong>attribute level differences that arise because of using different descriptions for semantically similar attributes</strong></td>
<td>A semantic annotation on the entities and attributes (provided by WSDL-S: modelReference) will indicate their semantic similarities.</td>
</tr>
<tr>
<td><strong>Naming conflicts</strong></td>
<td>* Web service 1: Student(Id#, Name)  * Web service 2: Student(SSN, Name)</td>
<td>* Mapping WS2 Id# to WS1 Id# is easy with some additional context information while mapping in the reverse direction is most likely not possible.</td>
</tr>
<tr>
<td>Two attributes that are semantically alike might have different names (synonyms)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Two attributes that are semantically unrelated might have the same names (homonyms)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Data representation conflicts</strong></td>
<td>* Web service 1: Student(Id#, Name)  * Web service 2: Book (Id#, Name)</td>
<td>* Mapping WS1 Marks to WS1 Grades is easy with some additional context information while mapping in the reverse direction is most likely not possible.</td>
</tr>
<tr>
<td>Two attributes that are semantically similar might have different data types or representations</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Data scaling conflicts</strong></td>
<td>* Web service 1: Marks 1-100  * Web service 2: Grades A-F</td>
<td>* Mapping in both directions will require some additional context information.</td>
</tr>
<tr>
<td>Two attributes that are semantically similar might be represented using different precisions</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Entity Definition</strong></td>
<td><strong>entity level differences that arise because of using different descriptions for semantically similar entities</strong></td>
<td>A semantic annotation on the entities and attributes (provided by WSDL-S: modelReference) will indicate their semantic similarities.</td>
</tr>
<tr>
<td><strong>Naming conflicts</strong></td>
<td>* Web service 1: EMPLOYEE (Id#, Name)  * Web service 2: WORKER (Id#, Name)</td>
<td>* Mapping in both directions will require some additional context information.</td>
</tr>
<tr>
<td>Semantically alike entities might have different names (synonyms)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Semantically unrelated entities might have the same names (homonyms)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Schema Isomorphism conflicts</strong></td>
<td>* Web service 1: PERSON (Name, Address, HomePhone, WorkPhone)  * Web service 2: PERSON (Name, Address, Phone)</td>
<td></td>
</tr>
<tr>
<td>Semantically similar entities may have different number of attributes</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Abstraction Level Incompatibility</strong></td>
<td><strong>Entity and attribute level differences that arise because two semantically similar entities or attributes are represented at different levels of abstraction</strong></td>
<td>* WS2 defines the student entity at a much general level. A mapping from WS1 to WS2 requires adding a Type element with a default ‘Graduate’ value, while mapping in the other direction is a partial function.</td>
</tr>
<tr>
<td><strong>Generalization conflicts</strong></td>
<td>* Web service 1: GRAD STUDENT (ID, Name, Major)  * Web service 2: STUDENT (ID, Name, Major, Type)</td>
<td>* A set-of Professor entities is a Faculty entity. When the output of WS1 is a Professor entity, it is possible to identify the Faculty group it belongs to, but generating a mapping in the other direction is not possible.</td>
</tr>
<tr>
<td>Semantically similar entities are represented at different levels of generalization in two Web services</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Aggregation conflicts</strong></td>
<td>* Web service 1: PROFESSOR (ID, Name, Dept)  * Web service 2: FACULTY (ID, ProfID, Dept)</td>
<td>* Course modeled as an entity by WS1 is modeled as an attribute by WS2. With definition contexts, mappings can be specified in both directions.</td>
</tr>
<tr>
<td>Semantically similar entities are represented at different levels of generalization in two Web services</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Attribute Entity conflicts</strong></td>
<td>* Web service 1: COURSE (ID, Name, Semester)  * Web service 2: DEPT (Course, Sem, ...)</td>
<td></td>
</tr>
<tr>
<td>Semantically similar entity modeled as an attribute in one service and as an entity in the other</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* Interoperation between services needs transformation rules (mapping) in addition to annotation of the entities and/or attributes indicating their semantic similarity (matching).
Web services interoperate by re-using these mappings.

- Ontologies now a vehicle for Web services to resolve message level heterogeneities
Four types of semantics

- Data Semantics: What are the inputs and outputs of a service
- Functional Semantics: What does a service do?
- Non-Functional Semantics: The non-functional requirements and capabilities of a service
- Execution Semantics: What is the execution context and the task skeleton (execution states) associated with this service
Semantics for Technical Services

Development / Description / Annotation
- WSDL, WSDL-S, SAWSDL, WSMO, OWL-S
- METEOR-S
- (MWSAF)

Composition, Configuration and Negotiation
- BPEL, WS-Agreement, WS-Policy
- METEOR-S
- (MWSAF)

Execution, Adaptation and Mediation
- BPWS4J
- activeBPEL
- WSMX
- METEOR-S

Publication / Discovery
- (Semantic) UDDI
- METEOR-S
- (MWSDI)
Semantics for Technical Services

Data/Information Semantics

Execution, Adaptation and Mediation
- BPWS4J, activeBPEL, WSMX, METEOR-S

Composition, Configuration and Negotiation
- BPEL, WS-Agreement, WS-Policy, METEOR-S (MWSCF)

Development/Description/Annotation
- WSDL, SAWSDL, (OWL-S, WSMO, WSDL-S) METEOR-S (MWSAF)

Publication/Discovery
- (Semantic) UDDI METEOR-S (MWSDI)
Semantics for Technical Services

Execution, Adaptation and Mediation
- BPWS4J
- activeBPEL
- WSMX
- METEOR-S

Composition, Configuration and Negotiation
- BPEL, WS-Agreement, WS-Policy
- METEOR-S (MWSCF)

Functional Semantics

Development / Description / Annotation
- WSDL, WSDL-S, SAWSDL, WSMO, OWL-S
- METEOR-S (MWSAF)

Publication / Discovery
- (Semantic) UDDI
- METEOR-S (MWSDI)
Semantics for Technical Services

- **Execution, Adaptation and Mediation**
  - BPWS4J,
  - activeBPEL,
  - WSMX
  - METEOR-S

- **Composition, Configuration and Negotiation**
  - BPEL, WS-Agreement, WS-Policy
  - METEOR-S (MWSCF)

- **Non Functional Semantics**

- **Development / Description / Annotation**
  - WSDL, WSDL-S,
  - SAWSDL, WSMO,
  - OWL-S
  - METEOR-S (MWFAF)

- **Publication / Discovery**
  - (Semantic) UDDI
  - METEOR-S (MWSDI)
Semantics for Technical Services

Execution, Adaptation and Mediation

BPWS4J, activeBPEL, WSMX METEOR-S

Composition, Configuration and Negotiation

BPEL, WS-Agreement, WS-Policy METEOR-S (MWSCF)

Execution Semantics

Development / Description / Annotation

WSDL, WSDL-S, SAWSDL, WSMO OWL-S METEOR-S (MWSAF)

Publication / Discovery

(Semantic) UDDI METEOR-S (MWSDI)
Semantics for Technical Services

Execution, Adaptation and Mediation
- BPWS4J, activeBPEL, WSMX, METEOR-S

Composition, Configuration and Negotiation
- BPEL, WS-Agreement, WS-Policy, METEOR-S (MWSCF)

Development / Description / Annotation
- WSDL, WSDL-S, SAWSDL, WSMO, OWL-S, METEOR-S (MWSAF)

Publication / Discovery
- (Semantic) UDDI, METEOR-S (MWSDI)

Execution Semantics
Data / Information Semantics
QoS Semantics
Functional Semantics

Knowledge Enabled Information and Services Science
Outline

- Motivation
- The Four Tiers Of A Business Process
  - Modeling, Enactment, Partner Services and Execution
- The Four Types Of Semantics
  - Data, Functional, Non-Functional and Execution
- The 4 X 4 Model
  - Unifying the four tiers using the four types of semantics
- The 4 X 4 Model In Action
Semantics for the 4 X 4 Model

• Currently, each tier has its own standard modeling language, e.g. UML or BPMN at Business Process Tier, BPEL for Workflow Enactment Tier, SAWSDL/ WSDL at Partner Services Tier and config files/WSDL at Middleware Services Tier

• Becomes hard to correlate different pieces of the puzzle

• A semantically enriched model that allows us to capture the semantics at each of the four tiers
The 4 X 4 Model

- The 4 X 4 model does not intend to replace any of the current languages. It is a way to add additional description.
- Can be represented by using semantic templates.
  That brings us to
- What are semantic Templates?
Semantic templates

- A way of capturing data / functional / non-functional / execution semantics
Example of a semantic template in the supply chain domain

Semiotic Template

ServiceLevelMetadata (SLM)
Category= NAICS:Electronics
ProductCategory= DUNS:RAM
Location= Athens, GA

SemanticOperation Template (SOPT1)
Action= Rosetta:RequestPurchaseOrder
Input= Rosetta:PurchaseOrder_Input
Output= Rosetta:PurchaseOrder_Output
OLP= {Encryption = RSA, ResponseTime< 5 Sec}

SemanticOperation Template (SOPT2)
Action= Rosetta:CancelOrder
Input= Rosetta:CancelOrder_Input
Output= Rosetta:CancelOrder_Output
OLP= {Encryption = RSA, ResponseTime< 5 Sec}
What semantics at what level?
Why the semantics?

• **Business Specification Tier:**
  – Need is to capture the functional and non-functional specification. Hence we capture functional and non-functional semantics.

• **Workflow enactment Tier:**
  – Captures the data flow, control flow and the partner level specifications.
  – Also addresses adaptation.
    • Hence we need all four types of semantics.
Why the semantics?

• **Partner Services Tier:**
  – Must allow description of partner services including inputs, outputs, what the service offers and the non-functional guarantees and requirements.
    • Data, Functional and Non-Functional semantics

• **Middleware Services Tier:**
  – Must advertise middleware level capabilities and the policies associated with them.
  – Data mediation can be thought of a middleware level service.
  – Adaptation capabilities must be built into middleware.
    • All four semantics are needed at this level.
Outline

• Motivation
• The Four Tiers Of A Business Process
  – Modeling, Enactment, Partner Services and Execution
• The Four Types Of Semantics
  – Data, Functional, Non-Functional and Execution
• The 4 X 4 Model
  – Unifying the four tiers using the four types of semantics
• The 4 X 4 Model In Action
Both SAWSDL and SA-REST are grounded to formal semantic models captured as ontologies.

SAWSDL
SAWSDL
SAWSDL

SA-REST
SA-REST

Partner Services are described using SAWSDL / SAREST + Policies are captured using semantically enhanced Policy / Agreement (Partner Services Tier)

Middleware Services Tier

Middleware services tier provides services for selection, configuration and adaptation of processes. (Middleware Services Tier)

Web processes are composed using services described in SAWSDL (Workflow Enactment Tier)

Workflow Enactment Tier

Tasks in a webprocess/workflow are grounded in semantic partner descriptions as templates.

Semantic templates are captured as ontologies.

Workflow specifications are created in the process modeling tier using semantic templates. (Process Modeling Tier)
4 X 4 Model in Action

- Semantic Templates for capturing process and partner level specifications
- SAWSDL used for SOAP based WS in
  - Semantic publishing and discovery of services
  - Dynamic binding
  - Adaptation
  - Data mediation
- SA-REST (XML + Microformats)
  - Smashups
  - Integration of REST based services
- Enhanced policy descriptions
  - Service selection
  - Process adaptation (Adaptation policies)
A Manufacturer needs to order various components
  – Model business specifications
  – Model Partner specifications
  – Capture adaptation rules and events
    • Needs to include human elements
    • Needs to capture the risk involved in various actions and estimate the probability of various events.
  – Enact and execute business process

How to capture and understand the System, Service and Human aspects?

The 4 x 4 Model presents an unified model that integrates the different tiers, that allows to semantically relate the different components across different layers
Illustrating Dynamic Configuration

- Being able to bind partners to a workflow during execution time

- Key tasks include
  - **Modeling**
    - Creating process and partner level specifications
    - Workflows created with partners described using semantic template
  - **Execution**
    - Discovery of partners (To be able to discover, we need to address publication as well)
    - Address data heterogeneity
    - Optimization and Adaptation
Conclusions: The 4 x 4 Model in a Nutshell

- The four tiers in Business process modeling are identified as
  - Business Process Tier, Workflow Enactment tier, Partner Services Tier and Middleware Services Tier
- Four types of semantics in SOA lifecycle
  - Data, Functional, Non-Functional and Execution
- 4 x 4 Model integrates the four tiers in business process modeling with the four types of semantics
- Creates a unified construct to relate the different tiers
- Can be captured using Semantic Templates
  - For SOAP services, Semantic Templates are defined using SAWSDL and Policy constructs
  - For REST services, Semantic Templates are defined using XML and Microformats (RDFa)
Conclusion: What does Semantics Bring to the Table?

• Better Reuse
  – Semantic descriptions of services to help find relevant services
  – Allows to study data, functional and non-functional variations between the different tiers

• Better Interoperability
  – Beyond syntax to semantics, mapping of data exchanged between the services (very time consuming without semantics, just as XML in WSDL gives syntactic interoperability, SAWSDL gives semantic interoperability)
  – Functional mediation to address different interaction protocols

• Configuration/Composition
  – Enable dynamic binding of partners
  – Create (S)Mashups dynamically
  – Optimization and adaptation during run time
  – Verify enactments against corresponding business process specifications
Conclusion: What does Semantics Bring to the Table?

- Some degree of automation across process lifecycle
  - Process Configuration (Discovery and Constraint analysis)
  - Process Execution (Addressing run time heterogeneities and exceptions)