INF5120 – Modellbasert Systemutvikling

F13: Model Driven Interoperability and MDA Industrial Evolution (Genova example)

Lecture 03.05.2010
Arne-Jørgen Berre

INF5120 - Lecture plan - 2010

1: 25/1: Introduction to MBSU, MDA, OO and Service/SOA modeling, Overall EA, 4 parts: MDE/SSS/MS/MDI (AJB)

Part I: MDE – Model Driven Engineering
2: 1/2: MDE I: Metamodeling, DSL and UML profiles, MDA technologies (XMI, Eclipse, EMF/GMF) (AJB/BRE)

Part II: SSS – Service Science and Service/SOA technologies
3: 8/2: SSS I: Service science (top down) - Service and SOA Technologies (bottom up) (AJB)

Part I continued: MDE – Model Driven Engineering
4: 15/2: MDE II: Model transformations with MOFScript, ATL and other technologies (GO/JO)
5: 22/2: MDE III: Code generation with MOFScript, ATL and other technologies (GO/JO)

Part III: MOS – Modeling of Services - with SoaML
6: 1/3: MOS I: Business Process Modeling (CIM) - with BPMN 2.0, and BMM, EA with UPDM (AJB)
7: 8/3: MOS II: SoaML, UML2 and SysML, Modelio SOA and Scope, -Collaboration and Component models (AJB)
8: 15/3: MOS III: SoaML (PIM) and Requirements modeling , CIM->PIM and SoaML (AJB)
9: 22/3: MOS IV: Method Engineering and SPEM / EPF - for Service systems (BRE)

EASTER

Part IV – Model Driven Interoperability
10: 12/4: MS V: SOA and Service Design, MDA and ADM - Intro to MDI (AJB)
11: 19/4: MDI I: Semantic Web with Ontologies and Model Driven Interoperability (TIR)
12: 26/4: MDI II: Semantic Services and Model Driven Interoperability (TIR)
13: 3/5: MDI IV: Model Driven Interoperability and Industrial Evolution of MDA/MDE (AJB), Knut Sagli/ESITO)
14: 10/5: Course summary and preparation for Exam 31/5 (AJB)

Exam: May 31st, 2010 (Monday), 0900-1200 (3 hours)
Agenda

- **Model Driven Interoperability**
  - MODELS 2010, Oslo, 3-8. October 2010
  - Ref. 2 papers on MDI

- **Industrial evolution of MDA/MDE**

- **Modeling tools**: MagicDraw, Enterprise Architect, Modeio, IBM Software Architect, Microsoft Domain Specific Languages, Oracle, ...

- **Guest lecture, ESITO – Genova, Knut Sagli**

---

MODELS 2010, Oslo, 3-8. October 2010

See: [http://models2010.ifi.uio.no](http://models2010.ifi.uio.no)  
Student volunteers?
Model Driven Interoperability

Current MDA Interoperability Architecture
Model Driven Service Interoperability through use of Semantic Annotations

I-ESA 2009 paper
Arne-Jørgen Berre
Fangning Liu
Jiucheng Xu
Brian Elvesæter
SINTEF ICT
Model Driven Interoperability through Semantic Annotations using SoaML and ODM

JiuCheng Xu*, ZhaoYang Bai*, Arne J.Berre*, Odd Christer Brovig**

INCOM'09, Moscow, 3. June. 2009

Contents

- Introduction
- Description of EMPOWER and MEMPOWER
  - EMPOWER Project
  - MEMPOWER Project
- Comparison Semantic mappings
- Conclusion & Further work
EMPOWER

- an innovative framework for interoperability between enterprise systems
- a flexible and extensible architecture
- a system environment

**System Interoperability Layer**
- Interoperable Enterprise Service Wrapper
- Semantic Services Registry
- Mediator Services Web Server
- Transformations Repository

**Semantic Adaptation Layer**
- Services Semantic Annotator (SAWSDL)
- Semantic Services Registry
- Mediator Services Web Server
- Transformations Repository

**Semantic Annotation Model**
- Ontology Definition Meta-model is a family of MOF meta-models, mappings between those meta-models, and a set of profiles that enable ontology modeling through the use of UML-based tools.
- Semantic Annotation Model editor is used to relate different PIM models and ontology. It is used to annotate the SoaML model with Ontology.
- SoaML describes the services models. The Model Mapping in the MEMPOWER includes transformations from models to ontology and ontology to models.

**Model Transformation Services**
- support the runtime lifting and lowering transformations among messages and ontologies based on the Model Map.
The EMPOWER Enterprise Interoperable Services Semantic Map

Semantic Adaptation Architecture

Semantic Services Registry
- Interoperable Enterprise Service Wrapper
  - Semantic Profile
- Services Semantic Annotations
  - WSDL, description storage
- Web Services Repository
  - Refers to executable code for specific functionality extraction

Mediator Services Web Server
- Transforms the generated service package
  - Data transformations
- Metadata Service
- Provides the data annotations for transformations generation

Transformations Creator
- Handles, adds and manages the EMPOWER Ontology
  - Stores and displays transformations
- Ontology Handling Utilities
  - Navigates, selects and uses ontological concepts for WSDL descriptions annotation

Semantic Adaptation Layer
- Utilises predefined data transformations for service execution
- Interoperable Enterprise Service Repository
  - Utilises data and functional semantics for defining the interoperable Enterprise Services

System Interoperability Layer
- Legacy System Wrappers
  - Database, pieces of functionality, APIs
- Legacy System A (IT Assets Inventory)
  - Functional schema (UML, diagrams), data schema (databases)
  - Execute code for specific functionality extraction

Executive Summary
- Account Processing
- Inquiry Processing
- Quotation Processing
- Sales Order Processing
- Billing
SAM – Semantic Annotations tools

(SASO: semantic annotation tool using SoaML and ODM)

Ontology example

(SINTEF ICT)
Address Ontology

Address in Source and UML

Address in Source.uml corresponds to Source.xsd

Address in Source.xsd
“Address” in the source and target transformation rules

Create mapping rules from source to ontology, and ontology to target using ATL.

Source to Ontology

Ontology to Target

“Address” transformations from source.xml and target.xmi

Transform source into ontology and ontology into target.
SAM editor realized in tree views

Ontology is represented as a structured and classified tree view. It shows the properties and relationships between those classes.

A simple example of class annotations on the PIM level
After annotating and exporting the model, you will get the file with an additional attribute. The annotations are displayed in red.

```xml
<soaml:Class name="POMessage" saName="PurchaseOrderMessage"
  stereotype="messageType">
  <soaml:Attribute name="customerId" saName="hasCompanyRegNo"
    type="String" modifier="public" />
  <soaml:Attribute name="name" saName="hasComanyName"
    type="Name" modifier="public" />
  <soaml:Attribute name="address" saName="hasAddress"
    type="String" modifier="public" />
  <soaml:Attribute name="creditScore" type="Integer" modifier="public" />
</soaml:Class>
```

**Semantic Mapping**

- 1. Ontology-based mapping on the PSM-Level (EMPOWER)
- 2. Direct mapping on the PSM-Level
- 3. Ontology-based mapping on the PIM level (MEMPOWER)
- 4. Direct mapping on the PIM level

<table>
<thead>
<tr>
<th>Approach</th>
<th>Ontology-based PSM</th>
<th>Direct mapping PSM</th>
<th>Ontology-based PIM</th>
<th>Direct mapping PIM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example: Address

Ontology

Address in Target.xsd has only one element: Address

Address in Source.xsd is divided into three elements: Address, Place, and Province

Address in Ontology is divided into three elements: Address, Region, and Province

1.PSM: Ontology-based

- Annotation based on ontology on the PSM-level
  --Annotate source.xml and target.xml using Ontology

Source.xsd

Target.xsd

Ontology

Source.xml

Address annotation
2. PSM: Direct Mapping

- Mapping without ontology on the PSM-level
  --Map between source.xml and target.xml (xsl:easy)

3. PIM: Ontology-based

1. Transformation from PSM level to PIM level
  --Generate sources.uml and target.uml from schemas (HyperModel Designer)
3.PIM: Ontology-based

1. Transformation From PSM level to PIM level
   --Generate sources.uml and target.uml from schemas (HyperModel Designer)

2. Mapping Between Models based on ontology on the PIM level
   - Step 1: Generate meta-models of models and ontology using EMF
   - Step 2: Create mapping rules from source to ontology, and ontology to target using ATL

Source-Ontology

Ontology-Target
3. PIM: Ontology-based

- 1. Transformation From PSM level to PIM level
   -- Generate sources.uml and target.uml from schema (HyperModel Designer 3.1)

- 2. Mapping Between Models based on ontology on the PIM level
   - Step 1: Transform source into ontology and ontology into target

3. Direct Mapping

- Transformation Between Models without ontology on the PIM level
  -- Use Semaphore tool to map source to target
Conclusion

- Ontology-based mapping (S-O-T) VS Direct mapping (S-T) on the PIM level
  - 2N vs N²

Mapping between each model and ontology will result in a linear growth of number of mappings.

- Mapping PIM-Level VS PSM-Level

<table>
<thead>
<tr>
<th>Mapping</th>
<th>2N</th>
<th>N²</th>
<th>2N</th>
<th>N²</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard Ontology</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>Platform Independent</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Multi-source documents Input</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Multi-target documents Output</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
</tbody>
</table>
Conclusion & Further work

- Conclusion
  - Ontology-based semantic annotations reduces mapping times from N-squared to 2N, but cost is a standard ontology.
  - Model Driven approach supports the interoperability independent from platform technologies, compared to a platform specific technical approach.

- Further work
  - Implement multiple industrial use cases with five scenarios for comparing EMPOWER and MEMPOWER.

Another example of Ontology-based Service: Message Reconciliation
Ad hoc reconciliation vs Ontology-based Reconciliation

Ad-Hoc
- Based on **ad hoc adapters** between pair of partners
- **Not scalable** respect to the growing of the number of partners

Ontology-based
- **Highly independent** solution, the semantic annotation does not depend on the other business partners
- **Highly scalable**, the complexity of the Semantic Annotation does not depend on the cardinality of the partners

Ontology-based reconciliation
Lossless and Lossy Annotations

- **Lossless SA**: when the annotation *fully captures* the intended meaning
  - A Local Schema (LS) element corresponds exactly to a concept in the RO
  - The meaning of a LS element can be precisely derived from concepts in the RO

- **Lossy SA**: when the annotation *fails to fully representing* the intended meaning
  - The meaning of a LS element does not have a matching concept in the ontology, nor the possibility of deriving it, since:
    - the intended meaning is outside the scope of the RO
    - The LS elem is not sufficiently refined (i.e., it does not match the accuracy level of e ontology) [underspecification]
    - The LS element presents a level of refinement not deemed useful [overspecification]

Example of Mismatch

EnterprA (Buyer)

**Purchase Order**
- Order_Number
- Order_Date
- Buyer_Info
  - Name
  - Address
    - Street_Name
    - Street_No
    - City_Post_Code
  - Telephone
- Products_Info
  - Product_Code
  - Description
  - Quantity
  - Price (unitary)
- Currency (Dollar, Euro, Pound)
- Charge
- RequestedDeliveryDate

EnterprB (Supplier)

**Sale Order**
- Date
- Organization_Name
- Contact_Person
  - Location
    - Street_Address
    - City_Post_Code
    - Country
  - Phone_Number
    - Area_Code
    - Number
    - Ext
- Client_Order_Number
- Order_Lines
  - Product_Code
  - Description
  - Quantity
  - Price (total per line)
- Currency (USD, Euro, Yen)
- Total
Ontology-based Reconciliation Approach

Example of actual reconciliation

Local Schema (LS)

```
Purchase Order (PO)
  ...
  Address
    ...
    City-Post_Code: literal
```

Semantic Annotation

```
LS.PO.Address.City-Post_Code =:
RO. Address.City AND RO.Address.Zip_Code
```

Reconciliation Rule

```
unpack(LS.PO.Address.City-Post_Code, "-") →
  (RO.Address.City, RO. Address.Zip_Code)
```

Run-time Reconciliation

```
{"Rome - 00185"} →{"Rome", "00185"}
```
An example of Transformation Rule in the Jena2 syntax

**SPLIT**
order.has_orderHeader.has_buyerInfo.has_organisationInfo.has_contactPerson.has_name
INTO
PurchaseOrder_BOD.relTo_Buyer.relTo_ContactPerson.hasPart_FirstName
PurchaseOrder_BOD.relTo_Buyer.relTo_ContactPerson.hasPart_Surname

NameSplitting:

```plaintext
[(?x0 rdf:type ai:order) (?x1 rdf:type ai:orderHeader) (?x2 rdf:type ai:buyerInfo) (?x3 rdf:type ai:organisationInfo) (?x4 rdf:type ai:contactPerson) (?x5 rdf:type ai:name)]
→
[(?x0 rdf:type ro:PurchaseOrder_BOD) (?x2 rdf:type ro:Buyer_BA) (?x4 rdf:type ro:ContactPerson_BA) Split(?x4, " ", ?y1, ?y2, 'http://www.w3.org/2001/XMLSchema#string') (?x4 rdf:type ro:hasPart_FirstName ?y1) (?x4 rdf:type ro:hasPart_Surname ?y2)]
```

**Conclusion and outlook**

- BMM can be used to support discussions on Organisational interoperability
- Support for semantics with ontologies and mediation is available now
- Short term benefit can be gained in the area of services for semantic interoperability – through the use of ontologies, and use of mappings and transformations for information and service interoperability
- i.e. – start here from an industrial perspective, establish ontologies, use these directly or mediate through semantic annotation.
- Semantic Web Services and Service-oriented Semantic Architectures (SESA) is a promising future technology
- Longer term benefits can be expected related to matching goals with services for process and service composition and process interoperability