INF5120 – Model-based System Development

Lecture 15

May 5th, 2014
Arne J. Berre, SINTEF ICT
Contents

- System Architecture models
- ADM – Architecture Driven Modernisation
- MDI – Model Driven Interoperability
- Comparison Semantic mappings
- Conclusion & Further work
INF5120 - Lecture plan - 2014

1 (13/1): Introduction – overview of the course. Enterprise Architecture with UML and BPMN and DSLs
2 (20/1): Business Architecture – Business Model Canvas and Business Model Innovation with Value Networks, Strategyzer tool. BPMN modeling, MagicDraw EA tool
3: (27/1): Service Innovation and Service Design, AT ONE, Smaply – BPMN Examples
4 (3/2): User experience and Touchpoints/UI Design – Balsamiq/WebRatio
5 (10/2): UML and Req.Modeling – Agile User stories versus Use cases 2.0
6 (17/2): ServiceML, Requirements Modeling, Goal Modeling, BMM, and Non Functional requirements – Requirements Engineering

7 (24/2): UI Models, WebML and IFML, Process models (WebRatio) (for Oblig 3)
8 (3/3): Model driven engineering – Metamodels, DSL, UML Profiles (for Oblig 2)
9 (10/3): Model driven engineering, transformation technologies (for Oblig 2)
10(17/3): Method Engineering, SW Process frameworks , SPEM/EPF, ISO 24744, FACESEM/ESSENCE (Brian Elvesæter)
11(24/3): Enterprise Architecture, UPDM and SysML
12(31/3): System Architecture and Information/Ontology modeling, UML, ISO 19103
EASTER
14(28/4): Platform models for the Cloud, with CloudML (Alessandro Rossini)
15(5/5): System realisation models (MDA-ADM, SBVR, MDI – Oblig 3 example, Oblg 2 review
16(12/5): Conclusion and Summary for INF5120 - Oblig 3 delivery and review
17(19/5): Preparation for Exam

Exam: Monday June 2nd, 2014, (4 hours)
Model Driven Enterprise Architecture

- Arne.J.Berre@sintef.no
- OMG standards related to the Zachman framework
# Zachman with OMG standards

<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>List of things important to business</td>
<td>List of processes that the business performs</td>
<td>List of locations which the business operates</td>
<td>List of events/cycles important to the business</td>
<td>List of organizations important to the business</td>
<td>List of business goals/strategies</td>
</tr>
<tr>
<td>SBVR</td>
<td>VDM</td>
<td>VDM</td>
<td>OSM</td>
<td>DTFV</td>
<td>BMM</td>
</tr>
<tr>
<td>ODM, IMM (CWM)</td>
<td>BPMN, CMPM</td>
<td>SoaML, UML</td>
<td>SoaML, UML</td>
<td>SoaML, UML</td>
<td>IMM (CWM), UML</td>
</tr>
<tr>
<td>Technology (Physics)</td>
<td>System Design</td>
<td>Technology Architecture</td>
<td>Presentation Architecture</td>
<td>Control Structure</td>
<td>Program</td>
</tr>
<tr>
<td>IMM (CWM), UML</td>
<td>SoaML, UML</td>
<td>SoaML, UML</td>
<td>Architecture</td>
<td>BPMN, CMPM, DTFV</td>
<td>UML</td>
</tr>
<tr>
<td>Component (Assemblies)</td>
<td>Data Definition</td>
<td>Network Architecture</td>
<td>Security Architecture</td>
<td>Timing Definition</td>
<td>Data</td>
</tr>
<tr>
<td>IMM (CWM), UML</td>
<td>UML</td>
<td>UML</td>
<td>Architecture</td>
<td>DTFV</td>
<td>Schedule</td>
</tr>
<tr>
<td>Operation (Instances)</td>
<td>Function</td>
<td>Network</td>
<td>Organization</td>
<td>Rule Definition</td>
<td>Function</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Strategy</td>
</tr>
</tbody>
</table>
OMG standards coverage

- **Scope** (Contexts)
- **Business** (Concepts)
- **System** (Logic)
- **Technology** (Physics)
- **Component** (Assemblies)
- **Operation** (Instances)

- **Data** (What)
- **Function** (How)
- **Network** (Where)
- **People** (Who)
- **Time** (When)
- **Motivation** (Why)

- **SBVR**
- **VDM**
- **OSM**
- **BMM**
- **SBVR**

- **ODM**
- **BPMN**
- **CMPM**

- **IMM (CWM)**
- **SoaML**
- **UML**
- **DTFV**

- **Data Definition**
- **Presentation Architecture**
- **Rule Design**

- **Program**
- **Security Architecture**
- **Rule Definition**

- **Network Architecture**
- **Organization**

- **Operation (Instances)**
- **Data**
- **Function**
- **Schedule**
- **Strategy**

---

Telecom and Informatics
3-tier System Reference architecture

Application Layer:
- UIC
- User Service

Business Layer:
- Service
- Entity

Data Layer:
- DataService

Legacy

Service Infrastructure
JEE: Java Enterprise Edition

- Java for 3 tiers architectures
- Includes
  - Java Libraries (Mail, Protocols, etc.)
  - Tools, compilers, code generators,
  - Application Server
- JEE 6.x
References

- SiSaS Methodology Wiki
  - [http://sisas.modelbased.net/](http://sisas.modelbased.net/)

- JEE 6.0
  - First cup of JEE 6
    - [http://docs.oracle.com/javaee/6/firstcup/doc/](http://docs.oracle.com/javaee/6/firstcup/doc/)
  - Official JEE 6 tutorial
    - [http://docs.oracle.com/javaee/6/tutorial/doc/](http://docs.oracle.com/javaee/6/tutorial/doc/)
WebML models and concepts
- WebRatio model based development -
Preview of WebML concepts

- Site = Content + Composition + Navigation + Presentation

- entities, relationships
- units, pages, links
- navigation + composition
- site views
- user models
- styles
- presentation

marco.brambilla@polimi.it
http://home.dei.polimi.it/mbrambil
ADM – Architecture-Driven Modernization (Reverse MDA)

- Standards on Architecture-Driven Modernization
- Solutions Directory
  - [www.systemtransformation.com/transfor.htm](http://www.systemtransformation.com/transfor.htm)
- Modernization Process Framework & Articles
  - [www.systemtransformation.com/itArchitecture_transformation.htm](http://www.systemtransformation.com/itArchitecture_transformation.htm)
- Reference Book
  - Note: New book on case studies (Ulrich / Newcomb) in the works for 2010 from Elsevier / Morgan Kaufmann / OMG Press
Modernization horse shoe model

- Longer Journey / Greater Impact
- Shorter Journey / Lesser Impact

- Existing Solution
- Target Solution

- Business Architecture
- Application/Data Architecture
- Technical Architecture
Modernization scenarios

- Replacement/Rewrite
- COTS Deployment
- Architecture Consolidation
- MDA / SOA Migrations
- Data Architecture Migration
- User Interface Refactoring / Migration
- Language & Platform Migration
- Application Improvement / Refactoring
- Language / Source Artifact Transliteration & Platform Migration
- Data Conversion

Business Domain

Existing Solution

Target Solution
ADM Framework

Enterprise Assessment: Planning Level analysis that is broad in scope and light on depth

- Enterprise Analysis Planning
- Technical Architecture Assessment
- Modernization Planning
- Business Architecture Assessment
- Data Architecture Assessment

Project-Level Assessment: Detailed analysis to support refactoring & transformation

- Objective Setting
- Proposal Development
- Technical Assessment
- IS Infrastructure Assessment
- Integration/Feasibility Analysis
- Architectural Assessment

Refactoring: Applies dramatic improvements to existing IT architecture

- Application Staging
- Language Change/upgrade
- Remodularization
- Middleware Enabling
- Validation
- Code Stabilization
- Data Definition Standardization

Transformation: Redeploy essence of existing system within a new target architecture

- Data Consolidation & Abstraction
- Rule/Data Synchronization
- Business Rule Capture & Consolidation
- Deploy New & Deactivate Legacy Architecture

Any combination of tasks / subtasks may be combined to create given project scenario.
ADM Standards in OMG

- Business Modeling Standards
- SwA Evidence
- ADM Pattern Recognition
- SW Metrics
- KDM
- ASTM

Existing Solution

Target Solution
KDM & ASTM

#1: ADM: Knowledge Discovery Meta-Model (KDM)
- Established metamodel that allows modernization tools to exchange application meta-data across applications, languages, platforms and environments
- Foundation for subsequent ADM standards
- KDM adopted in 2006 and will become ISO standard in 2009
- Deployed by several vendors and service companies

#2: ADM: Abstract Syntax Tree Meta-Model (ASTM)
- Builds upon KDM to represent software at very granular level
- ASTM supports automated system to system transformations, generally involving language and platform changes
- ASTM passed the architecture board in 2008 is being rolled out in 2009
Pattern Recognition and SMM

#3: ADM: Pattern Recognition

- Pattern Recognition will facilitate examination of structural meta-data with the intent of deriving patterns and anti-patterns about existing systems
- To be used to determine refactoring and transformation requirements and other alignment opportunities
- Analysis could be extended to business architecture
- Discussions proceeding in Fall of 2009

#4: ADM: Structured Metrics Metamodel (SMM)

- SMM is a general purpose metrics standard that supports multiple types of metric libraries
- Software Evidence, ADM Patterns Recognition, Sustainability Metamodel and Business Architecture all considering use of SMM
- Passed in 2008 and will be finalized in 2009
Patterns: From Analysis to Implementation

Analysis (Domain) Patterns

Architecture Patterns (Macro Architecture)

Domain Framework

Design Patterns (Micro Architecture)

(Idioms (Language dependent patterns)

(POO) Reusable Components
Visualisation, Refactoring and Transformation

#5: ADM: Visualization
- Focuses on ways to depict application meta-data stored within the KDM
- Work has not proceeded on this to date

#6: ADM: Refactoring
- Refactoring defines ways in which the KDM can be used to refactor applications
- Envisions metamodel to metamodel transformations
- There is no target date for this work

#7: ADM: Transformation
- Transformation defines mappings between the KDM / ASTM and transformed versions of those metamodels
- Defines mappings and transformations that may occur between existing applications and top down, target models
SINTEF projects

- REMICS (REuse and Migration of legacy applications to Interoperable Cloud Services);
  - started in 2010, SINTEF lead, two SMEs and other research partners. Focus on migration to SOA and cloud and developing a CloudML language

- NEFFICS (Networked Enterprise transformation and resource management in Future Internet enabled Innovation Clouds)
  - Started in 2010, SINTEF lead, using SaaS platforms combined with an advanced innovation management software platform
The REMICS approach

Knowledge Discovery
Reverse Engineering

Source code, binaries, documentation, users, knowledge, configuration files, execution logs and traces

Legacy Artifacts

Source Architecture

Model Driven Interoperability

Validate, Control and Supervise

Migrate

Forward MDA through PIM4Cloud

Service Cloud Implementation

Target Architecture for Service Cloud platform

SoaML with REMICS extensions for Service Clouds
Links to Business Models

Model Transformation Code Generation Traceability

RESERVOIR Joyant Amazon Google Microsoft

Service mediation for adaptation

SOA and Cloud Computing Patterns applied
Legacy Components Replacement and Wrapping
Design by Service Composition

Knowledge: REMICS KDM
Business Process and Rules
Components SoaML
Implementation UML, U2TP
Project concept

- Migration of legacy systems to service clouds
- Model-driven modernization approach

- Recovery
- Migration
  - Model-driven Interoperability
  - Models @runtime
  - Agile methodologies
  - Model-based testing
  - Requirements engineering

ICT
REMICS challenges regarding cloud

- Whether to have private or public or hybrid cloud. Business models and scalability concerns.

- Which cloud platform to choose?

- What does it mean for the architecture? Data handling (reporting system), authorization for access to data, maintenance and adaptation of the systems.

- Developing a platform independent model for cloud that hides technologies (and related standardization activities such as PIM4Cloud, Interoperability for cloud)
REMICS Metamodel extensions
How does it work?

Recovery
- Code
- Reverse Engineering
- UML Model
- Componentization
- SoaML Model

Migration
- UML and SoaML models
- SOA and Cloud patterns application
- Service Mediation
- PIM4Cloud, deployment generation
- Service Cloud Application

Validation and Control
- UML, SoaML, PIM4Cloud models
- Model-based Testing
- Model metrication
- Models@Runtime
- Performance measurement
Involved tools

Recovery
- Netfective BluAge reverse engineering tool
- WUT Tale GUI recovery tool
- SOFTEAM Modelio reverse engineering tools
- SOFTEAM Architecture Componentization Tool for Modelio

Migration
- SOFTEAM Patterns Designer Tool – SOA and Cloud patterns application
- SINTEF Interoperability Framework
- WUT RedSeeds – application generation tools
- SINTEF, SOFTEAM PIM4Cloud modeling and deployment generation
- UT Desktop to Cloud Migration tool

Validation and Control
- FOKUS!MBT
- FOKUS!Metrino
- WUT RSL Testing Tool
- Open source libraries for Models@Runtime
- Framework for performance measurement of 3-tier applications
Tools involved

- BluAge Reverse Engineering tool
- WUT Tale GUI recovery tool
- Modelio, PIM4Cloud, Componentization and Patterns Designer
- SINTEF, PIM4Cloud DSL
- FOKUS!Metrino
- FOKUS!MBT
- WUT, RedSeeds
- UT Desktop to Cloud Migration tool
- RSL Testing Tool
- Models@Runtime
- Interoperability Framework
SBVR
Brief Introduction into
Semantics of Business Vocabulary and Business Rules (SBVR)

OMG presentation from Nikolai Mansourov
CTO, Hatha Systems
http://www.hathasystems.com
Agenda

- Introduction to SBVR
- Overview of SBVR Structure
- Community – context for meaning
- Business Vocabulary
- Business Rules
- Semantic Formulation
- What next?
EU-Rent Case Study

- EU-Rent is a (fictitious) car rental company, used to provide coherent examples in SBVR (and in this presentation)

- The business requirements are fairly simple:
  - EU-Rent operates in several countries; in each country it has local areas containing branches
  - EU-Rent rents cars to customers from branches; one-way rentals are allowed
  - Rentals may be booked in advance or “walk-in”
  - Cars are owned by local areas and stored at branches
  - Each car is of a given model; car models are grouped into car groups; all the cars in a car group have the same rental tariff
  - Cars are serviced at 5,000 mile intervals
  - EU-Rent notes “bad experiences” with drivers (police action, unpaid parking fines, cars damaged or not returned to EU-Rent branches, etc) and may bar drivers who cause them.
EU-Rent samples: vocabulary

car movement
Definition: planned movement of a rental car of a specified car group from a sending branch to a receiving branch.
Reference Scheme: movement-id
Description: A car movement meets the business requirement that a car of a given group has to be moved between branches (“we need to move a full-size car from the London City branch to the Heathrow Airport branch”). A specific car will be assigned to it at some time, not necessarily when the requirement is first identified.
Note: car movements play roles in both "rental" and "car transfer" and car movements are scheduled in these roles.

car movement has movement-id
Necessity: Each car movement has exactly one movement-id.

car movement has receiving branch
Necessity: Each car movement has exactly one receiving branch.

car movement has sending branch
Necessity: Each car movement has exactly one sending branch.

car movement specifies car group
Synonymous Form: car group is specified in car movement
Necessity: Each car movement specifies exactly one car group.
EU-Rent samples: vocabulary

Figure E.3 - Car Movements
EU-Rent samples: rule

E.2.2.2.3 Rule Set -- Charging / Billing / Payment Rules

It is permitted that a rental is open only if an estimated rental charge is provisionally charged to a credit card of the renter that is responsible for the rental.

Guidance Type: operative business rule

Description: While a renter has possession of a car, there is a provisional charge to EU-Rent against his credit card. This will be replaced by an actual charge at the end of the rental.

Enforcement Level: Strict

Supporting fact types:
- rental has rental charge
- estimated rental charge is provisionally charged to credit card
- renter has credit card
- rental has driver
What will SBVR do? (1)

Support specification of business rules:
- From an organization perspective - not an IT system perspective
- Using the vocabulary of the business - not the vocabulary of its IT models
- Regardless of whether the rules can be, or will be, automated
Transformations will be needed

SBVR Model

Business Rules

Shared Concepts

Transformations / mappings

Business Policy Maker

IT Rules

Database Schema

End-to-end Workflow Model

IT System Model

Customer

Employee

Customers

Employees

Cars
What will SBVR do? (2)

SBVR realizes the ‘Business Rules Mantra’:

“Rules are built on Facts. Facts are built on Terms.”

Base Business Rules on Fact Types
Associate Concepts to define Fact Types
Define Concepts

... to describe businesses, not the IT systems that serve them

... in language understandable by business people

Develop Vocabularies to represent them (starting with terms for the concepts)
Preview: making a business rule

Start with a fact type, e.g.

\texttt{rental has driver}

Add a modal operator (from a limited set: “it is obligatory”, “it is necessary” …), e.g.

\textit{it is obligatory that rental has driver}

Quantify and qualify:

Add quantifiers to roles in the fact type (“each”, “at least one”, “no more than N”, …)

\textit{it is obligatory that each rental has at least one driver}

\textit{it is obligatory that each rental has no more than 4 drivers}

Use additional fact types as qualifiers (“the location of the return branch of the rental …”)

Add conditions based on fact types (“if a rental return is more than 4 hours late …”)
Two kinds of Business Rule in SBVR

- **Structural business rules:**
  - Specify what an enterprise *takes things to be*
  - Cannot be broken (are “true by definition”)
  - e.g. *local area is in exactly one operating country*

- **Operative business rules:**
  - Guide what an enterprise *must do*
  - Can be broken, so need an enforcement regime:
    - Detection of violations
    - Remedial action to restore compliance
    - (perhaps) Application of sanctions
  - e.g. *Each rental car that is assigned to a rental must be at the pick-up branch of the rental.*
Semantic Formulation of a business rule

Each rental car must have exactly one vehicle identification number necessary that each rental car has exactly one vehicle identification number

necessity claim \(\Rightarrow\) rule

\[\forall (\text{rental car}) \exists (\text{vehicle identification number})\]

\('universal' quantification\)

\('exactly one' quantification\)

atomic formulation

\(\text{rental car has vehicle identification number}\)
From Business Rule Statement to XML

1. Start with a business rule statement.
2. Identify symbols in vocabulary.
3. Parse according to language rules.
4. Restate as facts of logical formulation.
5. Represent facts of logical formulation as objects.
6. Write objects as XML.

It is prohibited that a barred driver is a driver of a rental.

An obligation claim embeds a logical negation....
It is prohibited that a **barred driver** *is a** driver of a** rental.*
Logical Formulation

It is prohibited that a **barred driver** *is a driver of* a **rental**.

- **obligation claim**
  - *embeds* a logical formulation that *is a* logical negation
  - *has a* negand that *is an* existential quantification
  - *introduces a* variable
    - *has* the type **barred driver**
    - *scopes over* an existential quantification
    - *introduces a* variable
      - *has* the type **rental**
      - *scopes over* an atomic formulation
        - *is based on* the verb concept: 'rental *has driver*'
        - *has a* role binding
          - *is of the* fact type role that *is 'rental' of* 'rental *has driver*'
          - *binds to* the variable that *has* the type **rental**
            - *has a* role binding
              - *is of a* fact type role that *is 'driver' of* 'rental *has driver*'
                - *binds to* the variable that *has* the type **barred driver**
XML (for Logical Formulation)

```xml
<is-obl>
  <modal>
    <logical>
      <negation>
        <variable>
          <eq />
        </variable>
      </logical>
    </modal>
  </is>
</is-obl>
```

```xml
<is-ex>
  <quant>
    <variable>
      <v1/>
    </variable>
  </quant>
</is>
```

```xml
<is-atomic>
  <role>
    <role>
      <fact>
        <ftr1/>
      </fact>
    </role>
  </role>
</is>
```

```xml
<esbr:thing xmi:id=""/>
```

Model Driven Interoperability
Introduction

- Organizations are collaborating with other organizations in order to meet their business objectives.
- For business optimization, organizations re-structure their business realizations by creating new constellations within an enterprise and across the organizational border that need to interoperate.
- Key issue: service network, who is to produce the service, who is to consume the service, business goals.
- It seems BMM and SoaML can combine these issues through:
  - Align goals with service-centric approach.
ATHENA Interoperability Framework (each system is described by enterprise models and different viewpoints, such as business, process, service, information)
Current MDA Interoperability Architecture

CIM/EM models

Semantic annotation

PIM System models

Semantic annotation

PSM System models

Semantic annotation

System

IF

Ref. ontology

Semantic annotation

CIM/EM models

Semantic annotation

PIM System models

Semantic annotation

PSM System models

Semantic annotation

Semantic annotation

Technical mapping

Interoperability execution
Architecture for semantic annotation and reconciliation

Reference Ontology

SwApp#1

Local Software & Data

Sem Annot Set #1

Sem Rec Rules #1

SwApp#2

Local Software & Data

Sem Annot Set #2

Sem Rec Rules #2

Design-time

Run-time

Internet
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
Model Transformation Services support the runtime lifting and lowering transformations among messages and ontologies based on the Model Map.

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.
The EMPOWER Enterprise Interoperable Services Semantic Map

Business Services Catalogue
- Financial Accounting
- Procurement
- Sales Order Management
- Management Accounting
- Warehouse Management
- After Sales Support

Interoperable Enterprise Semantic Map
- Financials
- Logistics
- Sales

Legacy Systems (pieces of functionality)
- Account Processing
- Inquiry Processing
- Quotation Processing
- Sales Order Processing
- Billing

Legacy System A (IT Assets Inventory)
Semantic Adaptation Architecture

- Semantic Services Registry
  - publishes the semantic profile

- Interoperable Enterprise Service Wrapper
  - navigates, selects and uses ontological concepts for WSDL descriptions annotation
  - selects and visualizes the WSDL descriptions of the native Web Services

- Semantic Profile
  - provides the data annotations for transformations generation

- Services Semantic Annotator
  - refers to executable mediator services

- Mediator Services Web Server
  - deploys the generated service package
  - data transformations

- Transformations Creator
  - stores and deploys transformations
  - communicates

- Transformations Repository
  - handles, edits and manages the EMPOWER Ontology

- Ontology Handling Utilities
  - utilizes predefined data transformations for service execution

- EMPOWER Ontology Repository

- Semantic Adaptation Layer
  - utilizes data and functional semantics for defining the Interoperable Enterprise Services
  - Interoperable Enterprise Services Semantic Map

- System Interoperability Layer
  - WSDL description storage
  - service’s functional requirements
  - executable code for specific functionality extraction

- Web Services Repository
  - refers to

- Wrapper Definition and Customization
  - uses the business services catalogue to retrieve the definitions of the Interoperable Enterprise Services

- Interoperable Enterprise Service Designer

- Legacy System Wrappers

- Legacy System IT Assets
  - (database, pieces of functionality, APIs)

- concrete pieces of business logic

- functional schemas (RPC Signatures, CORBA Services)

- data schemas (databases)

- Interface Repository

- Databases
PIM level use of Ontology mappings
Ontology example
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
“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.

Interface of demo
After annotating and exporting the model, you will get the file with a
additional attribute. The annotations are displayed in red.

```xml
<soaml:Class name="POMessage" saName="PurchaseOrderMessage">
  soaml:sterotype="messageType">
</soaml:Class>
<soaml:Class name="Customer" saName="Customer">
  soaml:sterotype="DataType">
  <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>PSM</th>
<th>PSM</th>
<th>PIM</th>
<th>PIM</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1</strong></td>
<td>Ontology-based</td>
<td>Direct mapping</td>
<td>Ontology-based</td>
<td>Direct mapping</td>
</tr>
<tr>
<td></td>
<td>PSM</td>
<td>PSM</td>
<td>PIM</td>
<td>PIM</td>
</tr>
</tbody>
</table>
Example: Address

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

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

Address in Target.xsd has only one element: Address.
1. PSM: Ontology-based

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

3.1. Address in Source.uml corresponds to Source.xsd
3. PIM: Ontology-based

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

2. Mapping Between Models based on ontology on the PIM level

   - CustomerFile
   - MasterFile
   - Company
   - Supplier -> Company
   - Customer -> Company
   - PaymentAgreement
   - InvoiceRule
   - OrderForm
   - DeliveryContract
   - Item
   - Delivery
   - Address
   - ContactInfo
   - Account
   - ContactName : EString
   - PostCode : EString
   - Address : EString
   - PhoneNumber : EString
   - Email : EString

Step 1: Generate meta-models of models and ontology using EMF
3. PIM: Ontology-based

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

2. Mapping Between Models based on ontology on the PIM level

   **Step 2:** Create mapping rules from source to ontology, and ontology to target using ATL

```java
rule Address2OnAddress{
  from
  sc: ABW!AddressInfo (sc.oclIsTypeof(ABW.AddressInfo))
  to
  t: Ontology!Address(
    addressType <- sc.AddressType,
    addressCountryCode <- sc.CountryCode,
    addressPostCode <- sc.ZipCode,
    addressRegion <- sc.Place,
    addressSequenceNo <- sc.SequenceNo,
    address <- sc.Address,
    addressProvince <- sc.Province,
    addressUpdateFlag <- sc.UpdateFlag,
    contactName <- sc.ContactName,
    contactPosition <- sc.ContactPosition,
    email <- sc.Email
  )
}

helper context Ontology!Address def totalAddress : String =
  self.address.toString() + self.addressProvince.toString() + self.addressRegion.toString();

lazy rule Address2ContactInfo{
  from
  a: Ontology!Address
  to
t: CCS!ContactInfo(
    Address <- a.totalAddress,
    AddressType <- a.addressType,
    Postcode <- a.addressPostCode,
    Email <- thisModule.onemail,
    Telephone <- thisModule.onephone
  )
  --do{thisModule.info<-t}
}
```
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

   ![Example XML and UML diagrams]

   **Step 3:** Transform source into ontology and ontology into target.
4. PIM: 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 between all model pairs will result in N-squared mappings.
## Conclusion

### Mapping PIM-Level VS PSM-Level

<table>
<thead>
<tr>
<th></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><strong>Mapping</strong></td>
<td>2N</td>
<td>N$^2$</td>
<td>2N</td>
<td>N$^2$</td>
</tr>
<tr>
<td><strong>Standard Ontology</strong></td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td><strong>Platform Independent</strong></td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td><strong>Multi-source documents Input</strong></td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td><strong>Multi-target documents Output</strong></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 $N^2$ 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.
NEFFICS video demos

- http://neffics.eu/videos

- NEFFICS Platform – Use of the Innovation Community
- NEFFICS Platform – From Innovation Community to Business Models
- NEFFICS Platform – Linking ideas with Model creation
- NEFFICS Platform – Innovation Community & Process administration

- NEFFICS Platform Business Model – Diagram
- NEFFICS Platform Business Model – Value Proposition
- NEFFICS Platform Business Model – Customer
  - NEFFICS Platform Business Model – Capabilities
- NEFFICS Platform Business Model – Value Formula
- NEFFICS Platform Business Model – Levels and Value Types
- NEFFICS Platform Business Model – Operations and link to executable models
Content

- NEFFICS: www.neffics.eu
- REMICS: www.remics.eu
- ENVISION: www.envision-project.eu
- ENVIROFI: www.envirofi.eu
- EMPOWER: http://www.empower-project.eu/
- MODACLOUDS: http://www.modaclouds.eu/
WireCloud demo from ENVIROFI and FI-WARE projects

- [http://www.youtube.com/watch?v=yEXILQYq7s4](http://www.youtube.com/watch?v=yEXILQYq7s4)

- Example of Application mashup, linking outputs to inputs in a service composition.