Model Driven Interoperability through Semantic Annotations using SoaML and ODM

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

* SINTEF, Pb. 124 Blindern, NO-0314 Oslo, Norway
  (e-mail: JiuCheng.Xu@gmail.com, morningbzy@gmail.com, Arne.J.Berre@sintef.no)
** Cicero Consulting, Postboks 3963, Ullevål Stadion 0805 Oslo, Norway
  (e-mail: oddchrister@gmail.com).

Abstract: Enterprise Interoperability is increasingly being enabled by use of service oriented architectures for the IT systems involved. A combined use of enterprise models, ontologies and service models can facilitate better semantic interoperability of the services being provided and requested. The focus of this paper is to show that service interoperability can be supported through a model driven approach with service oriented systems being described with service models (in SoaML) with semantic annotations to and from ontology models (in ODM)

Keywords: Enterprise Interoperability, Semantic Interoperability, Model Driven Interoperability, Service oriented architecture.

1. INTRODUCTION

1.1 Motivation of Research

The industry is increasingly interested in executing business processes that spans multiple applications. So today’s enterprises, no matter how big or small, have to meet the same challenges - bringing disparate systems together and making their mission-critical applications collaborate seamlessly.

Probably the most important issue related to interoperability is economical in its nature. It refers to obtaining and performing systemic behaviours within specific business environments even if the basic components and subsystems have been developed independently or in different kinds of technological or business environments. Interoperability makes it possible to take advantages of scale and scope economics within the development of the components, and to avoid too high cost of development and/or integration of changes, in a business and technological perspective.

A model-driven approach including semantic annotation of service models with ontologies and corresponding mapping can be used to support semantic interoperability in service oriented systems. The main question is how an annotated model can be transformed into another representation, e.g. Java or an external Domain Specific Language, using the annotations and how its definitions of the lifting and lowering operations between the models and ontology can support the construction of a tool that handles these annotations. In addition the tool will also need to map and validate the models and their annotated concepts from ontologies that are available to the business domain at hand.

1.2 Structure of the Paper

The first section is the motivation of the research. In section 2 we present the context of our work. Background knowledge of related work which leads the readers to better understand will be presented in section 3. We will briefly give introduction to Service Oriented Architecture (SOA), Service oriented architecture Modelling Language (SoaML) and ODM (ontology definition meta-model). Section 4 illustrates our model-driven method that employs semantic annotation using SoaML and ontology and gives an example to estimate the approach at the end. Section 5 summarizes the benefits of this approach and discusses further work.

2. DEFINITION AND CHALLENGES OF INTEROPERABILITY

2.1 Definition of Interoperability

Tanenbaum gave the definition of interoperability as “Interoperability characterizes the extent by which two implementations of systems or components from different manufacturers can coexist and work together by merely relying on each other’s services as specified by a common standard” (Tanenbaum, 2006). IEEE defines interoperability as “the ability of two or more systems or components to exchange information and to use the information that has been exchanged”. (IEEE 1990)

With these two definitions from different points of view, both of them describe the interoperability in the domain of two (or more) separate systems or two parts from same system. However, differences do exist: the first one focuses on the results of the interoperability and considers the integration between systems as a service; the second one pay more attention to the ability of information transformation and reuse between different systems. The notion of interoperability is not only limited to computer and information technology systems, but
also is concerned about concerns the business processes and the business context of an enterprise.

2.2 Challenges of Interoperability

The most important challenge is how to increase the interoperability and integration of information systems. Service and process usually span different domains and organizations. Information (messages) of each distinct domain is always recorded and used with a lot of, often custom, vocabulary and proper noun inside the business or company.

It is difficult for customers without the same background and knowledge to understand and comprehend these languages and take the advantages of the services offered by the business or company. This ability problem can lead to lower adoption rate of the service and lower efficiency of the cooperation between companies (organizations) in both different and the same domains.

Today, most of the information transformation between different systems (organizations) does not have any associated semantic. This can lead to scenarios where information may be changed and lost during the process of transformation. This can for instance happen because two developers will not interpret the available information and make the same decisions about it. This situation leads to a huge work to make specification for each step of the transformation.

This is a kind of problem that can be solved semi-automatically through the means of annotations with embedding of the formal semantics that define the services and the information exchanged between e.g. customers and suppliers. This thus leads to a need to clearly define the semantics both for the information and the processes using the service, and by that allow the automation of services, integrations and transformations.

Software engineering technologies such as model-driven architecture has already given a direction in this field, in the form of ODM, semantic web services. W3C also provides standardized technologies such as OWL/RDFS for creating semantic description of different formal levels and SAWSDL for doing semantic annotation. This paper takes a model driven architecture approach to interoperability, based on the use of ODM for SAWSDL architecture.

3. OVERVIEW OF SoaML and ODM

3.1 Overview of SOA

Service Oriented Architecture (SOA) has gained popularity as an architectural approach for realising a system through a set of provided and required services. Some services might encapsulate parts of legacy systems, while other services might be developed from scratch. Different technologies, such as web services and others might be used to support a SOA implementation.

SOA is a conceptual, technology independent, business architecture that allows business functionality or application logic to be available through reusable IT services. It is a concept, guideline, pattern, an approach and a philosophy of how IT functionality can be planned, designed and delivered as a service.

3.2 Overview of SoaML

The recent SoaML (Service oriented architecture Modelling Language) standard specification from OMG (OMG 2008) describes a UML profile and Metamodel for the specification of services within a service oriented architecture. This is done as a Platform Independent Model (PIM) following the principles of the OMG Model Driven Architecture approach (MDA). The goals of SoaML are to support the activities of service modelling and design and to fit into an overall model-driven development approach.

The SoaML profile supports the range of modelling requirements for service-oriented architectures, including the specification of systems of services, the specification of individual service interfaces, and the specification of service implementations. The SoaML meta-model is based on the UML 2.0 metamodel and also provides an UML 2.0 profile. Figure 1 defines how to use SoaML in UML based on the services defined in the SoaML profile.

Fig. 1. SoaML (Service oriented architecture Modelling Language) Profile of UML

SoaML provides an approach for platform independent description of services being provided and requested through service interfaces between participants.

3.3 Overview of Ontology Definition Metamodel (ODM)
Ontologies are a knowledge representation system that allows us to express knowledge through logical constructs. Ontology Definition Metamodel (ODM) is a standard from OMG for a set of meta-models for different ontology languages. (OMG 2009)

The meta-model consists of several packages: Common Logic, Topic Maps, RDF, OWL and Description Logic. The main focuses in our work are the RDFS and OWL packages. These are a reflection of the W3C recommendations, and provide a way to build a model driven environment around these.

4. USING SEMANTIC ANNOTATION FOR MODEL DRIVEN INTEROPERABILITY

This section outlines the proposal for SASO (Semantic Annotation Tool using SoaML and ODM), which we describe in this paper.

4.1 Vision

The vision for SASO is to provide a tool-set for semantic annotation of models on the PIM (platform independent model) level and between PIM and PSM levels which is shown in figure 2. This will be accomplished through a tool-set that provides:

- An annotation editor
- An ODM implementation
- A component providing infrastructure for transformation based on the annotations.

Fig. 2. Overall SASO (semantic annotation tool using SoaML and ODM) vision

This tool will be open and flexible with regard to future improvements in the technology stack. To achieve this openness and flexibility, it is developed on the Eclipse platform (Eclipse 2009). This platform provides the infrastructure, tools and standard implementations which are needed for the development. It has a vibrant and living ecosystem that gives a certain guarantee that the platform will undergo further improvement for years and allow further improvement of SASO.

4.2 Architecture

To fulfil the vision stated previously core architecture needs to be established. The architecture of SASO will have three main components:

- ODM Implementation and Editor, is considered as one of the three main components, where the ODM meta-model for ontologies is an essential implementation. It needs to feature the meta-model itself and facilities for manipulating, saving and loading files. The editor will be a simple editor based on the graphical formalism of Concept Maps. The implementation will be layered on the top of ODM meta-model.

- Semantic Annotation Model (SAM) Implementation and Editor, allows relating the different PIM models and the common ontology. The sole purpose of this component is to represent the binding between ontologies and PIMs in order to provide facilities for import and export of these.

- SAM Mapping Service, will provide the necessary services to facilitate (automatic) mapping of models and ontologies. This service is responsible for providing services for automatic mappings and exporting these mappings to target platform specific models. This component will need to cater to different types of transformation languages and different transformations.

Fig. 3. Overall SASO architecture

In addition, SASO will take advantage of an existing UML2 implementation. We are using the part of the Eclipse project named UML2 tools. It will provide the necessary parts to access UML2 PIMs, validating and editing them. It is implemented in the Eclipse APIs (EMF, GEF and GMF). The whole architecture of SASO is shown in figure 3.

4.3 Design

Eclipse Modelling Framework (EMF) is an umbrella project offering several different APIs/frameworks for use in metamodeling. The projects range from distributed models to validation. We are in particular building on top of the Graphical Modeling Framework (GMF) which extends upon Graphical Editor Framework (GEF).
After the introduction of the concepts of platform, we give the detail design of the three main components of the SASO:

- **ODM Implementation and Editor**, is done as an Ecore model using EMF. Constraints are expressed by using OCL, and utilizing the OCL plug-in for Eclipse and the resulting EMF code will adhere to these. Code generation is performed to produce both the model domain code and the model manipulation and I/O packages. The ODM editor is built by GMF and set up to feature the simple graphical notation concept maps, as well as RDF and OWL. A simple ontology example of Customer/Supplier is shown in figure 4. It contains entities and relationships among them and shows the key elements related to customer-supplier case, such as company, address, order, contract and invoice, etc.

Fig. 4. Ontology example of Customer/Supplier

- **Semantic Annotation Model (SAM) Implementation and Editor**, the meta-model of SAM will use SAWSDL as its starting point (W3C, 2008). Our approach is to create a model-based version of the SAWSDL architecture. SAWSDL provides an architecture for semantic annotations with modelReference, lifting and lowering references that point towards some resource with functions of mappings. This semantic annotation editor is designed as either a plugin of Eclipse or independent software which will realize the basic functions of semantic annotation of both PSM and PIM models, especially SOA models. It should have the following function:

  - Read an ontology which is presented by OWL/RDF standard provided by W3C organization.
  - Parse and structure the ontology into a tree view.
  - Read different kinds of models and represent them in different views according to the model type.
  - Allow to drag and drop to link ontology elements to the model objects.
  - Output the model into a new file which contains the annotation with defined tags.
  - **SAM Mapping Service**, the mapping service should be open for use with any transformation language. It can be plugged into the extension point provided by the mapping service. The goal of the mapping service is to provide an abstraction between the actual transformations and the abstract mappings done in the SAM Editor. It is realized in a common service that provides the necessary interfaces which is needed to provide transformation. It will feature service interfaces that provide the following public functionality: Transformation directory service interface; Transformation of SAM and source/target models interface and Transformation registration (extension point) interface.

4.4 Realization

- **ODM editor realization**, is done by the use of GMF for the creation of a concept diagram editor and through the use of an ODM UML profile for the creation of OWL diagrams.

- **SAM editor realization**, is done as an Eclipse application by JAVA. Ontology (OWL file) can be structured in tree view showing at the left part in Fig 5. On the right the model is shown together with the annotation added through the tool. Since a tree view editor is simple, easy to comprehend and navigate, the proposal chooses this method to visualize the model. The tree views editor need to display: the annotations of ontology and models and the mappings between the ontology and a model element.

Fig. 5. SAM editor realized in tree views

4.5 Customer/Supplier Example

In this paper, a simple Customer/Supplier case is used to illustrate the SASO approach. The case is the service interoperability for two ERP systems in a Customer/Supplier scenario.

The Customer/Supplier ontology has been defined in OWL. OWL files are text based representation of ontology. As illustration, there are a number of properties that are associated to the element “Address”. So, the “Address” property is chosen as an example for the annotations of ontology and models.
and the mappings between the ontology and a model element in the following. A detail view of the “address” and related elements are created and shown in Fig 6 below.

![Fig. 6. Detail view of “Address” in the ontology](image)

In the semantic annotation editor, the ontology in OWL format can be parsed and represented as a tree view which contains not only all the elements described in ontology, but also the object properties and data properties.

![Fig. 7. Semantic Annotation Editor](image)

After the definition of ontology, a real data model from an ERP system is taken into account. The data model is described as an XML file which contains lots of instances of its model. It is necessary to abstract a meta-model from the data model in XML schema to be described as an .xsd file. In this semantic annotation editor, XML schema can be displayed as a tree view as well composed of data range names and values.

Two important inputs of semantic annotation editor, ontology and model, are both accessed by the editor. Then, the connections can be established between ontology and model by combining the elements or properties from ontology and the attributes in the data model. Fig 7 shows the annotations on the model at the right side, which are the elements or properties coming from the ontology tree view in the left.

After annotation by this editor, a new .xsd file which representing the same metamodel as before but contains the annotations added through the editor, i.e. a set of new attributes, shown as follows, are inserted into the original file.

```xml
<sa:Name name="Address" ontology="http://com.examples.owl/EXAMPLE.owl">
    ...
</sa:Name>
```

If several of the models are annotated with the same ontology, they can be connected to each other easily according to these annotations in the models. We can see the “one to one” relationship between the attributes of source and target in this example.

The SAM mapping services has two steps: Transformation from PSM level to PIM level and Transformation between models on the PIM level.

Transformation from PSM level to PIM level: Generating ERP.uml and CRM.uml from their schemas. It is a snapshot about address in the ERP systems about transformation from schema to model is shown in figure 8 (Berre 2009).

Transformation between models on the PIM level: We used ATL to implement transformation between models. The preparations of ATL are meta-models of ontology, source and target. First of all, generate meta-models: ontology.ecore, ERP.ecore and CRM.ecore from UML files at the first step. Then, create mapping rules from source to ontology and target to ontology using ATL. Figure 9 shows transformation.
rules about address information between source to ontology, and ontology to target.

Fig. 9. “Address” in the source and target transformation rules

At last, we do transformations from source.xml to ontolog.xml and ontolog.xml to target.xmi. Figure 10 shows transformation rules about address information between source to ontology, and ontology to target.

Figure 10 “Address” transformations from source.xml and target.xmi

5. CONCLUSIONS AND DISCUSSION

5.1 Conclusion

The conclusion of this paper is that the use of semantic annotation from a platform independent service model in SoaML and from an appropriate ontology represented with ODM will be able to support service interoperability, as long as mappings can be done through an appropriate ontology. This statement is made in the light of the discussion in section 1 and especially in the light of the proposal and application of conceptual solution that have been done. It shows the possibility of creating reference models with ontologies and annotating UML diagrams with the concepts contained within the ontologies. It also shows that a mapping operation scheme makes it possible to lift and lower the representation from and to the ontology level and thereby creating a “bridge” between two models. We are currently experimenting with the approach for different usage scenarios, and also plan to apply this to a set of industrial case examples from the ERP domain. We are dependent on a developer defining these semantics in the correct way and annotating the models precisely.

5.2 Discussion of Further Work

There are several areas that we are considering for further work: Annotation and ontology repository - the most interesting is to possibly develop a repository for ontologies and the annotation models that can be distributed over a network using web services; Semi-Automatic support for doing semantic annotation and model matching in the repository; Extensible validation - the validation should in all instances be plug-in-based so that it would be possible to add new validates as needed; Relations between specific regions in transformations and models - the purpose is to allow specific regions of code to be related to specific mappings; expanding the tool to support source code; extending the ODM editor and detailing the mapping language (Brovig 2008).

REFERENCES


Eclipse document (2009). org.eclipse.platform.doc.user.3.1
http://www.eclipse.org/documentation

http://electronics.ihs.com/document/abstract


http://www.omg.org/docs/omg/09-03-03.pdf


http://www.w3.org/TR/sawSDL