Model Driven Service Interoperability through use of Semantic Annotations

Arne-Jørgen Berre      Fangning Liu        Jiucheng Xu         Brian Elvesæter
SINTEF, Norway      KTH, Sweden        SINTEF, Norway       SINTEF, Norway
Arne.J.berre@sintef.no  fangning@kth.se  jiucheng.xu@sintef.no  brian.elvesater@sintef.no

Abstract

This paper presents an approach for comparing two architectures for ontology-based semantic annotation for service interoperability, the EMPOWER architecture using platform specific XML-based technologies and the MEMPOWER architecture extending this with platform independent Model Driven Architecture (MDA) based technologies. We will compare the two architectures with respect to pilot requirements and experienced advantages and challenges for model driven systems. The two approaches are being evaluated based on examples from interoperability between ERP-systems in a Buyer/Seller interaction context.

Keywords: Model Driven interoperability, Service interoperability, Semantic annotation

1. Introduction

The move towards a future Internet of Services will lead to an increased number of service-oriented enterprise applications becoming available. The need to achieve service interoperability among these can be supported through the use of ontology-based semantic annotations. A model-driven approach to this can improve the interoperability independent of platform technologies, compared to a platform specific technical approach for this.

MDA is the approach from Object Management Group (OMG) on how model driven engineering can we executed. Doing the semantic annotation at the Platform Independent Model according to MDA will enable us to specify mapping to and from an ontology independent of the technology being used. Further, ontology-based semantic annotation is can support semantic interoperability by introducing common understanding and standardization.

MEMPOWER combines the EMPOWER architecture and The Model Driven Interoperability (MDI), and we will compare the two architectures with respect to pilot requirements and previously experienced advantages and challenges for model driven systems. In this paper, we first introduce the FUSION [1] architecture, and the EMPOWER extensions to FUSION in section 2. MDI framework is described in section 3. In section 4, we describe the main areas of MEMPOWER and how we will compare it to EMPOWER using a pilot to evaluate the advantages and disadvantages of MDI. Finally, we draw conclusions and describe further work.

2. EMPOWER extended from FUSION

FUSION is a finished research project was funded by the European Commission of the 6th Framework Programme. It aims to uplift business collaboration and interconnection between enterprises, by using semantic annotation and discovery of services through a semantic registry for service-oriented legacy systems [2].

EMPOWER contributes an innovative framework, a flexible and extensible architecture and a system environment for facilitating the standardized creation and ad-hoc exposure of pieces of functionality [3]. Its architecture extends FUSION’s architecture and reuses FUSION ontology. It is being funded by the European Commission of the 7th Framework Programme.
2.1 FUSION Architecture

FUSION has Design-time and Execution environments. Using the Design-time environment, user could create the executable semantic profiles and generate executable business processes. The Execution environment is responsible for registering and publishing semantic profiles to a services registry, supporting categorization and search service of the services registry and deploying the grounded business process model to a process execution engine [2]. The FUSION architecture is illustrated in figure 1.

Figure 1. Architecture of FUSION

Three basic units constitute the Execution environment of FUSION. In addition, the Design-time environment consists of four components. Two main units in the FUSION environments are described as following:

**Administration module:** It plays a coordinator role in execution. It manages the registration and publication of semantic profiles in semantic registry. Meanwhile, it manages the deployment of the grounded business process model to the Business Process Execution Environment.

**Semantic Profiler:** This is the main component in FUSION Design-time environment. Its main responsibility is to create semantic profiles of Web services, generating transformations between input/output messages and an ontology, and will trigger the publication process of the Integration Mechanism.

2.2 FUSION Ontology

The FUSION ontology is based on the ENIO ontology [4]. It is used as a resolution of heterogeneity at message level to facilitate semi-automatic process design. ENIO provides four layers and three facets to satisfy the needs. The structure of ENIO is represented in figure 2.

Figure 2. FUSION Ontology

Upper Ontology captures general concepts in the real world and is independent of all domains. Vendor-independent layer models the domain of business applications with particular services and data spaces. Vendor-dependent is an extension of the above layer. It models the specific enterprise applications in particular business domains. The case-specific layer instantiates the vendor dependent layer ontology.

Every layer has three facets to annotate enterprise services. Data facet captures the definition of data in exchanged messages. Functional facet provides semantic meaning to the operations of enterprise services. Process facet contains process template ontology.

2.2 The approach of EMPOWER

EMPOWER system comprises two layered subsystems, System Integration Layer and Semantic Adaption Layer. The System Integration Layer is for exposing web services at enterprise level, and the Semantic Adaptation Layer is for capturing semantic meaning of enterprise services. This architecture is an extension of FUSION using the Ontology Designer, Ontology Repository and Semantic Registry. The Service Semantic Annotator, Transformations Creator and Interoperable Enterprise Wrapper in it can be mapped to the semantic profiler in FUSION. The architecture of this is shown in figure 3. FUSION ontology is used as the basis.
3. Model Driven Interoperability

Model-driven development (MDD), and in particular OMG’s Model Driven Architecture (MDA) [5], is emerging as the state of practice for developing model driven applications and software systems. Several research projects have sought to provide us with a better way of addressing and solving interoperability issues using semantic annotation and ontology, compared with earlier non-modelling approaches. ATEHNA[8] and INTEROP[6] define an interoperability framework for MDD of enterprise applications and software systems to address interoperability. The framework provides a foundation, consisting of a set of reference models for how to apply MDD in software engineering disciplines, in order to support the business interoperability [7].

3.1. Reference Model on MDI

The Model Driven Interoperability framework shown in figure 4 is further described in [7].

MDI introduces different abstraction levels to reduce the gap between enterprise models and code level during model transformations. The definition of the levels was based on three levels of abstraction defined in MDA namely CIM, PIM and PSM. CIM (computation independent model) describes the business context and business requirements for the software systems. PIM (platform independent model) describes software specifications independent of execution platforms. PSM (platform specific model) describes the realization of software systems.

In addition, in order to reduce the gap between the CIM and PIM, the framework includes vertical and horizontal integrations. Vertical integration is about transformation between different levels. Horizontal integration is the interoperation between two enterprises, which can be applied in all abstract levels. During the transformation between each level, MDA and Architecture-Driven Modernization (ADM) could be perceived as a top-down and a bottom-up approach to software development and integration [8]. We use semantic annotations and ontologies, in every level to achieve mutual understanding.

The ATHENA MDI Framework [8] describes four categories (services, information, process and non-functional) of system aspects, in all three abstract levels mentioned above. It specifies software interoperability issues addressed by conceptual integration.

4. MEMPOWER
MEMPOWER is a Model Driven variant of EMPOWER, in order to compare with the advantages and disadvantages of MDI. In this approach, we focus on the model level, which abstracts and represents to the XML level in the EMPOWER. In EMPOWER, we use XML, WSDL and SAWSDL with OWL. The models in MEMPOWER are expressed using UML, Service oriented architecture Modeling Language (SoaML) [14], Semantic Annotation Model (SAM) [10] and Ontology Definition MetaModel (ODM) [11].

In relation to MDA, semantic annotation to ontology can be very beneficial for the development of models [10]. As the extension of EMPOWER, MEMPOWER is supposed to allow for easier, and automated, interoperability between the systems. In this paper, we introduce the architecture of it which at PIM level allows achieving greater interoperability.

4.1. Model-driven Enhancements in MEMPOWER

In MEMPOWER, we have five main areas (1-5) where we introduce platform independent models instead of the platform specific XML level components in the EMPOWER architecture. The architecture of this is shown in figure 5.

In this architecture, the five main areas are SoaML, SAM, ODM, Model Map and Model Transformation Services.

1) SoaML: We use SoaML to describe the mapping between models and ontologies. SoaML is a recently standardised UML profile and metamodel for modeling services at a platform independent level. The Model Mapping includes transformation rules from and to the corresponding platform technology. It corresponds to WSDL, OWL-S and WSML at the XML level in EMPOWER.

2) SAM: Semantic Annotation Model editor is used to relate different PIM models and ontologies. The metamodel for SAM is depicted in figure 6. The source and target are two distinct classes that both inherit from identifiable Element, abstract class. One annotation is a one-to-one relationship between Model Reference Source and Ontology Reference Target, but both of them are allowed to be related to several annotations. Annotation inherits from Named Element, having a distinct name. An annotation is associated with a Mapping. Mapping is composed of a Lowering mapping construct and/or a Lifting mapping construct [10]. It corresponds to SAWSDL in EMPOWER.

3) ODM: It is a family of MOF metamodels, mappings between those metamodels as well as mappings to and from UML, and a set of profiles that enable ontology modeling through the use of UML-based tools. It reflects the abstract syntax of several standard knowledge representation and conceptual modeling languages [11]. ODM is used to select and utilize of specific ontological concepts in the semantic annotation process. It also supports ontology editing capabilities, such as extensions and restrictions [3]. It corresponds to the use of ontologies in OWL in EMPOWER.

4) Model Map: Model Map resolves the mapping rules in a model to model rule language for any mismatches between requested and provided service based on the corresponding annotated SoaML models and the ontology. It corresponds to the use of the Semantic Map in EMPOWER.

5) Model Transformation Services: It supports the
runtime lifting and lowering transformations among the input and output messages and ontologies based on the Model Map. It generates and deploys the runtime transformation code into the transformation repository and extends the semantic services registry of the pre-selected service with the references to the transformation.

In this paper we use a simple buyer-seller case to illustrate the MEMPOWER approach and then discuss both architectures based on this. The case is service interoperability for two ERP systems in a buyer/seller scenario derived from the case in the SoaML standard and semantic annotation to a shared Purchasing ontology. We are looking into a situation where there are semantic differences between the data in requested and provided services.

First we give the annotation and mapping from Model A (which is shown in figure 8) to the ontology. We observe that our metamodel has the capability to annotate different resources through its Reference attribute in the respective classes ModelReferenceSource and OntologyReferenceTarget. This allows connecting the classes in model A with its counterpart in the ontology. In the proposed tree view this will equate to presenting the user with the following hierarchy in the proposed tree view editor:

```
Root
  | Order
  |    | Order:class
  |    | ShippingCompany
  |    |    | ShippingCompany:class
  |    |    | Customer: class
  |    |    | name: Customer
```

This shows us that the metamodel can cater to annotation of concepts in the ontology and of model elements. There’s an ambiguity in this application though. A small ambiguity in this case is the name property of Customer. Since ODM is a representation of OWL, the name property can both be a concept related to ShippingCompany through an objectProperty, or a value property that is contained within the concept itself. This distinction should be made explicitly in the properties of the part of the annotation concerning name:Customer, where we are dealing with equivalent information. In the case where we are mapping between the ontology and the model, this distinction will be inherited in the mappings.

Then we give the annotation and mapping between model B (which is shown in figure 9) and the ontology. We have the following problems at hand this time:

- **Concatenated information in strings:** It will be
split according to a separator or by index.

- **Classes in-cohesive**: The model more than one concern will need to be split up. For example the class Transport from model B contains type information, company and address.

- **Naming differences**: It needs to be resolved according to the references ontology. Example is “Stock” in model B and “Inventory” in model A.

![Figure 8. PIM for Customer](image)

Our conceptual mapping language supports the mapping of all of these operations. A possible annotation structure would look like this:

Root

Inventory

Stock:class

Lowering mapping

Lifting mapping

ShippingCompany

company:Transport

Lowering mapping

Lifting mapping

The mappings in this case for the Inventory would be that the replace operation is run at the ontology name when we lower from the ontology, and vice versa replaced with inventory when we lift to ontology level. The operations from the lowering of company:Transport uses the split operation to attain the name of the company and instantiate as the company of a Transport instance class. Lifting also uses split to extract the company name into an instance of the separate concept ShippingCompany.

The principal approach is the same for both EMPOWER and MEMPOWER, but we can observe the following differences. For service annotation EMPOWER will use annotations through SAWSDL and the MEMPOWER will use SAM. For ontology model expression the EMPOWER will use OWL and MEMPOWER use ODM instead. For the model map rules at design time MEMPOWER will use a model-level transformation language (ATL) instead of XSLT at XML level in EMPOWER. We will also investigate the use of mediator services for runtime that matches the actual requested and provided services in the MEMPOWER architecture. The final goal would be to automatically create such mediator services using an MDI approach.

![Figure 9. PIM for Supplier](image)

In comparing the two approaches we see from the initial analysis that MEMPOWER will suffer from the traditional challenges of model driven systems. We have just finished an experiment comparing a platform specific technology approach and a platform independent model driven approach for the development of service oriented systems in the insurance domain. The model driven approach was in this experiment based on the model driven approach for service development from [13].

We found challenges with the model driven approach in the use of new model-based technology which often needs to be learned by people coming from a programming background only. The initial productivity might be lower as a result. The advantage of the model driven approach comes if we need to
address multiple implementation technologies, including those that might not have direct support for semantic annotation technologies. Model driven tools often lack maturity in terms of support for round trip engineering and team development.

5. Conclusion and further work

In this paper we presented an approach for comparing two architectures for ontology-based semantic annotation for service interoperability. The EMPOWER architecture is using platform specific XML-based technologies and the MEMPOWER architecture is extending this with platform independent MDA-based technologies. Through evaluating MEMPOWER and EMPOWER using a Buyer/ Seller case, it proves that service interoperability can be supported through use of ontology-based semantic annotations, and that a model-driven approach to this can improve the interoperability independent from platform technologies, compared to a platform specific technical approach for this.

The initial analysis shows, however, that a number of challenges need to be overcome for the model-based approach before the full potential of model driven interoperability can be achieved. In the future, we will be working in parallel with both the EMPOWER and MEMPOWER architectures, to explore their use on multiple industrial use cases as well as advance in dealing with the challenges identified for the model driven interoperability approach of MEMPOWER. We appreciate the contributions of the FUSION, SHAPE and EMPOWER project teams and the support from the EU commission to provide the foundation and funding for this work.

References