Model Driven Ontology:
A New Methodology for Ontology Development

Mohamed Keshk
Sally Chambless
Raytheon Company
Largo, Florida
Mohamed.Keshk@raytheon.com
Sally.Chambless@raytheon.com

Abstract

Semantic technology is becoming a preferable alternative for enterprise-wide applications intertwined with interoperable information sharing, due to the distributed nature of this technology. Ontology is the cornerstone of semantic technology; therefore, a major challenge for the project team is to build a complete and consistent ontology data model that represents the correct business domain. Effective collaboration among customer and team members is essential for the creation of the correct ontology model. Equally necessary is a mechanism to automatically transform this model into ontology script.

Within today's leading organizations using semantic technology, a significant factor in business success rests solely in the hands of the ontologists. It is they alone who are responsible for building the correct ontology data model. Having no other members on the project team capable of verifying and validating the created ontologies may put the entire business at risk.

This paper describes a new methodology, "Model Driven Ontology," in which using a standard modeling activity as a key process for building ontology would effectively and efficiently enhance collaborations between different parties of the project team. This would lead to a consistent ontology model validated and approved by all members of the project team (business experts, intel analysts, DB admins, architects, ontologists, etc.).

Model Driven Ontology uses a UML object model artifact as a starting point to build an ontology data model. This model as a common ground for all team members is then systematically transformed to a formal ontology, facilitating the development of enterprise-wide information exchange and sharing, which can be uniformly developed, centrally maintained, and efficiently reused [6]. This would lead to more efficient and inexpensive information sharing between different information systems, cost effective development and deployment of information systems, and better quality decision making as a result of more timely, accurate, and complete information.

Introduction

The development of large-scale enterprise applications has become increasingly complex due to the massive growth of enterprise data and the constant changing of requirements. Semantic technology has been seen as a crucial alternative for managing this complexity by providing a solid and flexible infrastructure for information exchange, retrieval, sharing, and discovery.

As ontologies play a central role in facilitating semantic technology solutions, it is essential for business to standardize the ways ontologies are developed. The phases of ontology development include analysis, design, coding, validation, execution, and maintenance. Moreover, it is vital for businesses to keep all key players (business

experts, intel analysts, architects, DB admins, ontologists, etc.) closely involved in the development phases.
Organizations using semantic technology, including those in both governmental and private sectors, frequently hand ontology development tasks solely to the ontologists. In most cases, the consequence is a dilemma, since no other team member is capable of validating the ontology script created by the ontologists, and the business might be at great risk if the script does not reflect the correct business model. Therefore, a methodology to standardize the way ontology is developed is badly needed.

This paper sheds light on the value of modeling in the context of ontology development for enterprise applications. It shows how modeling can be an effective way to manage the complexity of ontology development [5], as it fosters better communication by overlooking implementation details that are not relevant to the overall system, and delivers robust design and assessment of requirements and architectures. Despite these virtues, mainstream ontologists have yet to take advantage of modeling in everyday practice [8].

Our approach uses a Unified Modeling Language (UML) object model as the common means for expressing ontology models. As an industry standard, UML graphic models provide a common ground for team members to better understand the business data models and elevate the level of collaboration. The result is a consistent data model, validated and approved by all team members, which leads to a more accurate ontology script.

In general, there is no one correct methodology for developing ontologies, since there is no one correct way to model a domain [2]. Ontology itself is a data model based on formal logic and greatly overlaps with a UML object model, as both share many basic concepts. While a UML object model has the concepts of classes, properties, associations, constraints, and instances, ontology has the same concepts named classes, datatype properties, object properties, restrictions, and individuals, respectively. Providing a single data model for all parties of project team will increasingly eliminate design ambiguity, reduce the complexity of the enterprise data model, and speed up the overall development.

Therefore, a UML object model can be seen as a common model for ontologists and software architects, as it enhances communication between both camps and brings other parties to the table. It also aligns the effort of building a consistent data model that is accessible and usable not only by ontologists, but also by other team members.

Model Driven Ontology Methodology

In this section, we will discuss in detail the Model Driven Ontology approach with a simple, yet complete, example [1]. The following diagram (Fig. 1) shows a UML class diagram of a purchase order example in terms of classes, attributes, enumerations, and relationships including inheritance, composition, aggregation and associations with constraints represented as cardinalities.

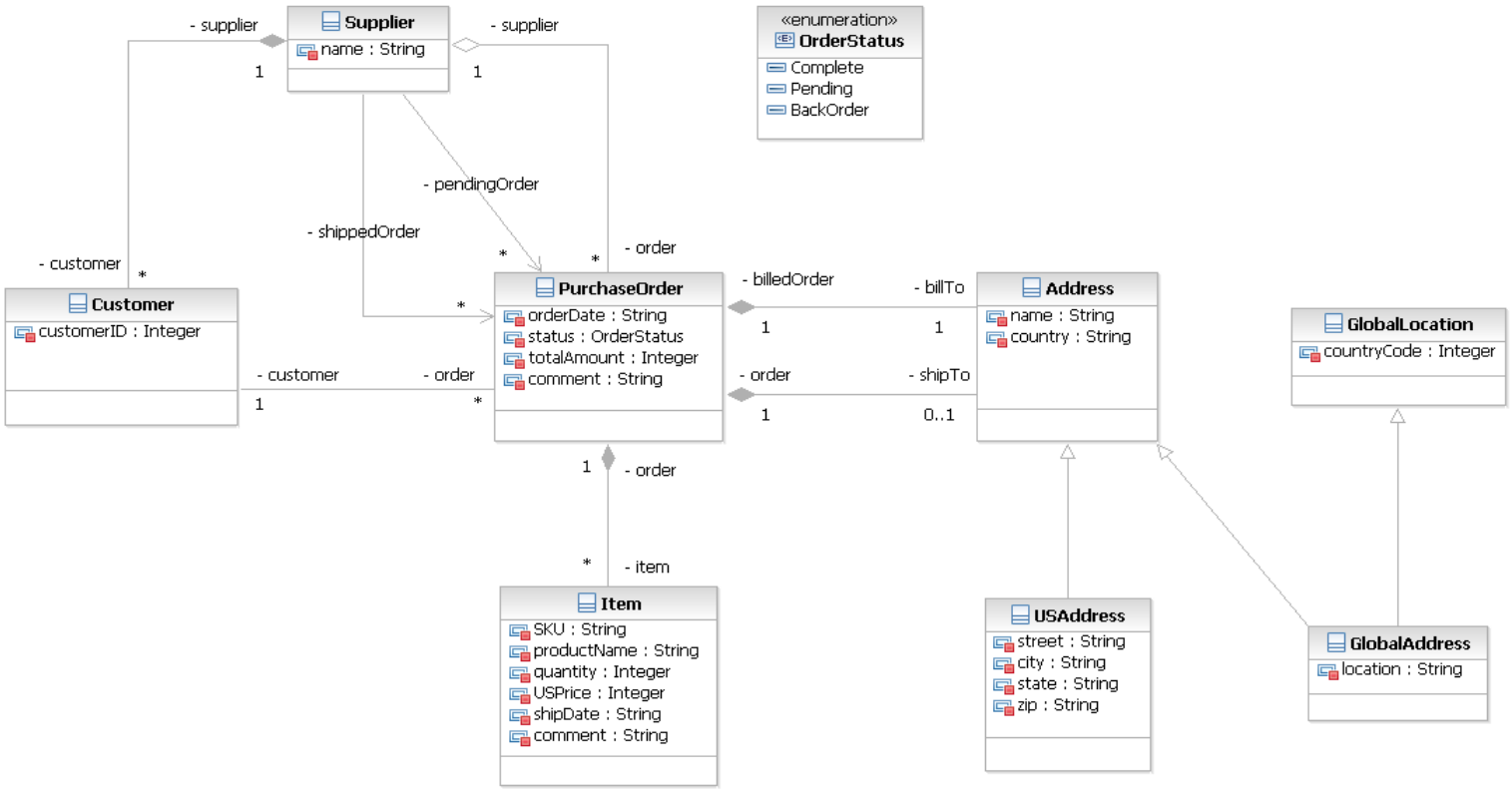

Fig-1

Our approach starts with the UML class diagram which represents the data model of a particular domain. Once the team comes up with the correct UML data model, validated and approved by all parties, we generate a UML version that is encoded in XMI by exporting the model using a standard UML tool such as RSA. We then apply transformation rules by parsing the XMI file into ontology script.

This parsing is done through Eclipse Modeling Framework (EMF) API provided by UML2 plugin [3], which is an EMF-based implementation of the UML 2.x metamodel for the Eclipse platform. The objective of this plugin is to provide a useable implementation of the UML metamodel to support the development of model processing tools, a common XMI schema to facilitate interchange of semantic models, test cases as a means of validating the specification, and validation rules as a means of defining and enforcing levels of compliance [3].

Our transformation platform is EMF which is part of the Model Driven Architecture (MDA) and is the implementation of a subset of the MDA in Eclipse platform [1]. An EMF model is essentially the class diagram subset of UML. EMF is originally based on MOF (Meta Object Facility) by OMG (Object Management Group). EMF uses XMI (XML Metadata Interchange) as its canonical form of a model definition. EMF has its own meta-metamodel called Ecore. Ecore is considered the metamodel for UML in addition to some other metamodels, such as XSD, WSDL, BPEL, etc. Ecore is located at the M3 layer of MDA paradigm and defines all kinds of metamodels located at M2, including UML. Ecore, itself, is very similar to EMOF (Essential MOF), but has Eclipse as a runtime environment.

EMF lets you define a data model in one of three formats: Java interface, XML schema, or UML class diagram, then

allows you to generate the other two formats. The most-likely scenario is to start with a UML model and generate the corresponding Java interfaces and XML schema. Our approach extends this capability by generating RDF/OWL script from the same UML model.

Building transformation rules is a joint effort between the architecture team, the ontology team and business domain experts. The expertise of these teams helps generate the correct script corresponding to the data model. For the purpose of illustrating the transformation mechanism, we have isolated a subset of the diagram (Fig. 2). The complete generated OWL script is too lengthy to include in this paper.

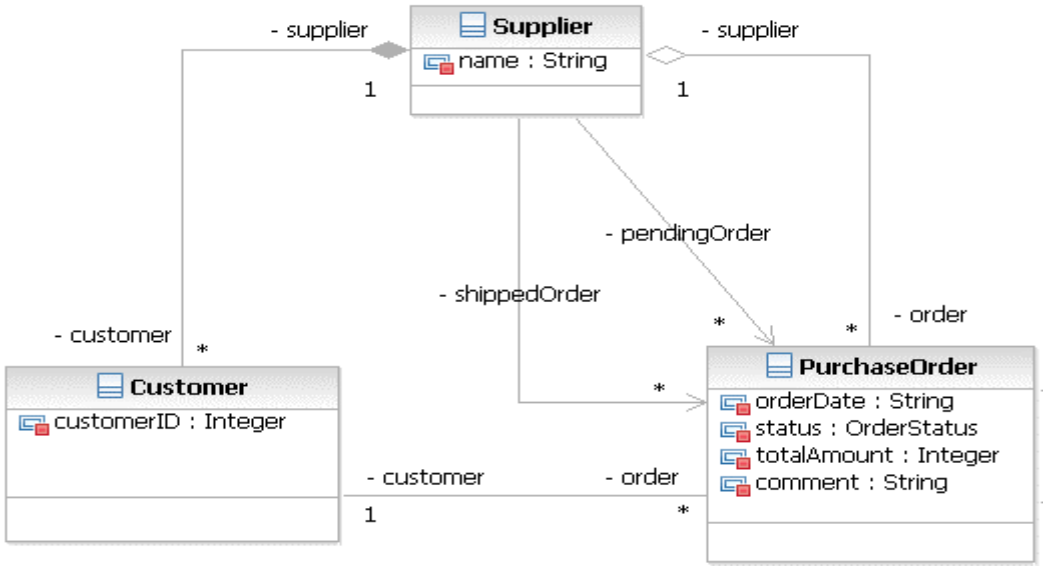

Fig-2

The following is XMI script for "Supplier" class:

```
<packagedElement xmi:type="uml:Class" xmi:id="_maCsFE3GEd2Y9cy9X2GvMA" name="Supplier">
 <ownedAttribute xmi:id="_maCsFU3GEd2Y9cy9X2GvMA" name="name" visibility="private">
  <type xmi:type="uml:PrimitiveType" href="pathmap://UML_LIBRARIES/UMLPrimitiveTypes.library.uml#String"/>
 </ownedAttribute>
 <ownedAttribute xmi:id="_maCsFk3GEd2Y9cy9X2GvMA" name="customer" visibility="private" type="_maCsDk3GEd2Y9cy9X2GvMA"
     aggregation="composite" association="_maCsKU3GEd2Y9cy9X2GvMA">
  <upperValue xmi:type="uml:LiteralUnlimitedNatural" xmi:id="_maCsF03GEd2Y9cy9X2GvMA" value="*"/>
  <lowerValue xmi:type="uml:LiteralInteger" xmi:id="_maCsGE3GEd2Y9cy9X2GvMA"/>
 </ownedAttribute>
 <ownedAttribute xmi:id="_maCsGU3GEd2Y9cy9X2GvMA" name="pendingOrder" visibility="private"
     type="_maCr5U3GEd2Y9cy9X2GvMA" association="_maCsKk3GEd2Y9cy9X2GvMA">
  <upperValue xmi:type="uml:LiteralUnlimitedNatural" xmi:id="_maCsGk3GEd2Y9cy9X2GvMA" value="*"/>
  <lowerValue xmi:type="uml:LiteralInteger" xmi:id="_maCsG03GEd2Y9cy9X2GvMA"/>
 </ownedAttribute>
 <ownedAttribute xmi:id="_maCsHE3GEd2Y9cy9X2GvMA" name="shippedOrder" visibility="private"
     type="_maCr5U3GEd2Y9cy9X2GvMA" association="_maCsLk3GEd2Y9cy9X2GvMA">
  <upperValue xmi:type="uml:LiteralUnlimitedNatural" xmi:id="_maCsHU3GEd2Y9cy9X2GvMA" value="*"/>
  <lowerValue xmi:type="uml:LiteralInteger" xmi:id="_maCsHk3GEd2Y9cy9X2GvMA"/>
 </ownedAttribute>
 <ownedAttribute xmi:id="_maCsH03GEd2Y9cy9X2GvMA" name="order" visibility="private" type="_maCr5U3GEd2Y9cy9X2GvMA"
     aggregation="shared" association="_maCsM03GEd2Y9cy9X2GvMA">
```

```
    <upperValue xmi:type="uml:LiteralUnlimitedNatural" xmi:id="_maCsIE3GEd2Y9cy9X2GvMA" value="*"/>
    <lowerValue xmi:type="uml:LiteralInteger" xmi:id="_maCsIU3GEd2Y9cy9X2GvMA"/>
  </ownedAttribute>
</packagedElement>
```

And its corresponding OWL script is the following:

```
<owl:Class rdf:about="&PO;Supplier">
  <rdfs:label>Supplier</rdfs:label>
</owl:Class>

<owl:DatatypeProperty rdf:about="&Supplier;name">
  <rdfs:domain rdf:resource="&PO;Supplier"/>
  <rdfs:range rdf:resource="&xsd;string"/>
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
</owl:DatatypeProperty>

<owl:ObjectProperty rdf:about="&Supplier;order">
  <rdfs:domain rdf:resource="&PO;Supplier"/>
  <rdfs:range rdf:resource="&PO;PurchaseOrder"/>
  <owl:inverseOf rdf:resource="&PurchaseOrder;supplier"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="&Supplier;customer">
  <rdfs:domain rdf:resource="&PO;Supplier"/>
  <rdfs:range rdf:resource="&PO;Customer"/>
  <owl:inverseOf rdf:resource="&Customer;supplier"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="&Supplier;shippedOrder">
  <rdfs:domain rdf:resource="&PO;Supplier"/>
  <rdfs:range rdf:resource="&PO;PurchaseOrder"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="&Supplier;pendingOrder">
  <rdfs:domain rdf:resource="&PO;Supplier"/>
  <rdfs:range rdf:resource="&PO;PurchaseOrder"/>
</owl:ObjectProperty>
```

Conclusion

This paper explains the benefits and values that modeling practice can offer for ontology-based applications, by treating modeling as a first class artifact, rather than documentation. In addition to presenting a single common data model that all team members can share, a mechanism is presented to generate the ontology script directly from the UML model once it is validated and proofed. In this case, the model is used not only as a diagram or a blueprint, but also as a primary artifact from which efficient script is generated by applying transformation rules.

We argue that the use of Model Driven Ontology would increasingly boost productivity, eliminate mistakes due to human misunderstanding, break the monopoly of ontologists over ontology development, and save a significant amount of development effort.

References

[1] Frank Budinsky. Dave Steinberg, Ed Merks, Ray Ellersick, and Timothy J. Grose, "Eclipse Modeling Framework", Addison-Wesley Professional, August 2003.

[2] Dragan Gasevic, Dragan Djuric, and Vladan Devedzic, "Model Driven Architecture and Ontology Development", Springer, 1st edition, July 2006.

[3] Eclipse UML2 project, http://www.eclipse.org/modeling/mdt/?project=uml2

[4] D. S. Frankel, Model Driven architecture: Applying MDA to Enterprise Computing, OMG Press, ISBN: 0471319201, January 2003.

[5] Nešić, S., Jazayeri, M., Jovanović, J., Gašević, D., "Ontology-based content model for scalable content reuse", In Proceedings of the 4th ACM International Conference on Knowledge Capture, Whistler, BC, Canada, 2007, pp. 195-196.

[6] "National Information Exchange Model - NIEM", http://www.niem.gov/

[7] dos Santos, E.S., Ralha, C.G., Carvalho, H.S., Gašević, D., "MDA-based Ontology Development: A Study Case," In Proceedings of the 19th International Conference on Software Engineering and Knowledge Engineering, Boston, USA, 2007.

[8] Stephen Cranefield, Jin Pan, "Bridging the Gap Between the Model-Driven Architecture and Ontology Engineering", The Information Science Discussion Paper Series, Number 2005/12, December 2005, ISSN 1172-6024.