

Major Influences on the Design of ODM

Daniel T. Chang
IBM Silicon Valley Lab
555 Bailey Ave., San Jose, CA 95141
dtchang@us.ibm.com

Elisa Kendall
Sandpiper Software, Inc.
2053 Grant Road, #162, Los Altos, CA 94024
ekendall@sandsoft.com

ABSTRACT

The purpose of this paper is to discuss the major influences on the design of ODM. We first discuss the need for an ontology definition metamodel, provide an overview of the ODM architecture and present basic considerations that were taken in developing ODM. We then discuss in detail several “watershed issues” that were debated by the co-submission team and whose resolutions guided the design and evolution of ODM.

INTRODUCTION

In March 2003 the OMG issued an RFP for an Ontology Definition Metamodel (ODM) [1], which seeks a specification of a MOF 2.0 (MOF2) compliant metamodel, a UML 2.0 (UML2) profile, and any additional information needed to support:

- Development of ontologies using UML modeling tools.
- Implementation of ontologies in the W3C Web Ontology Language (OWL).
- Forward and reverse engineering of ontologies.

Four companies made initial submissions to the OMG in response to the RFP in August that year. These included DSTC, Genteware/AT&T, IBM, and Sandpiper Software. As is common in the OMG standardization process, the four initial submitters agreed to work together to develop a joint revised submission, which is due for delivery to the OMG in October 2004.

Early in the submission development process, the team realized that there was little consensus in the broader OMG community on what comprised a valid ontology or on how ODM-based ontologies might be applied. While there were some common goals among team members, there were also diverse opinions, particularly with respect to usage goals. Establishing consensus based on a comprehensive analysis is essential in order to reach a common mindset and vision within any diverse team beginning a collaborative research effort. The first draft of such an analysis, reflecting usage scenarios and goals for the ODM, was completed in January 2004 and was posted on the OMG Ontology PSIG Web page for public comment. A paper based on the results of this analysis will be presented at the forthcoming Fifth International Conference on Web Information Systems Engineering (WISE '04) [2].

Subsequent to the completion of the usage scenarios and goals analysis, the team embarked on the development of ODM. Rather than trying to merge the four initial

submissions, we elected to design ODM essentially from scratch, leveraging the initial submissions where appropriate. The purpose of this paper is to present some of the major influences on the resultant design. We first discuss the motivation for development of an ontology definition metamodel, provide an overview of the ODM architecture, and present basic design considerations. We then discuss in detail several “watershed issues” that were debated by the team and whose resolution has influenced the design and evolution of ODM. This paper reflects the most recent draft ODM specification, posted on the OMG Ontology PSIG Web page for public comment in August 2004. ODM is a work in progress. The design choices discussed here are subject to change until ODM has been approved by the OMG as an adopted specification.

THE NEED FOR AN ONTOLOGY DEFINITION METAMODEL

With the advent of the semantic web movement [3] and the consequent development of XML-based ontology representation languages such as OWL, ontologies are becoming increasingly common in the broader business community. The usage scenarios we identified are likely representative of emerging applications that will leverage ODM and knowledge-based capabilities within the next few years, ranging from business applications to analytic applications to engineering applications [2].

Ontology development can be a difficult and time-consuming process. In general, it involves formalization of one or more subject matter expert's knowledge of a given domain, which in and of itself can be challenging. Further, since it is unlikely that a single expert possesses all relevant expertise in a given domain for a particular application or role, it frequently requires consensus building among a number of experts, which increases the level of complexity and time required to achieving the desired result.

In addition, ontology development must be accomplished within the context of a business need, and should be grounded in requirements relevant to a particular software development activity, or set of activities, to be of practical use in most business settings. That is, the ontologies that an enterprise develops must form an integral part of that enterprise's information and application infrastructure.

One of the current trends in enterprise-scale, commercial software engineering is to apply a model driven development and management methodology, in part as a result of the MDA related standards effort by the OMG and the availability of EMF (Eclipse Modeling Framework), an open source model-driven software development platform [8]. To

leverage recent advances in these technologies and methodologies for ontology development, what is needed is a MOF2 metamodel for ontology definition.

Such a metamodel will enable model-driven ontology development in a number of ways:

- Forward engineering: development of ontologies using MOF based (e.g., UML) modeling tools.
- Reverse engineering: leveraging existing ontologies (e.g., represented in OWL) for ontology modeling.
- Integrated ontology/software development: making ontology development an integral part of software development.

THE ODM ARCHITECTURE

ODM consists of three major components: metamodels, metamodel mappings, and one or more UML profiles for ontology, as shown in Figure 1.

ODM is not a single metamodel, but consists of six metamodels. These are grouped logically together according to the nature of the representation formalism that each models: formal first order and description logics, structural and subsumption / descriptive representations, and traditional conceptual or object-oriented software modeling.

At the core of the ODM are two metamodels that represent formal logic languages: DL (Description Logics [4]) and SCL (Simplified Common Logic [5]). While the heritage of these languages is distinct, together they cover a broad range of

representations that lie on a continuum ranging from higher order, modal, probabilistic and intentional logics to very simple taxonomic expression.

There are three metamodels that represent more structural or descriptive formalisms that are somewhat less expressive in nature than DL and SCL. These include metamodels of the abstract syntax for: RDFS (RDF Schema) [9], OWL [10,11], and TM (Topic Maps) [12]. RDFS, OWL and TM are commonly used in the semantic web community for describing vocabularies, ontologies and topics, respectively.

Two additional metamodels considered essential to the ODM represent more traditional, software engineering approaches to conceptual modeling: UML2 [6] and ER (Entity Relationship) diagramming. UML and ER methodologies are the two most widely used modeling languages in software engineering today, particularly for conceptual or logical modeling. Interoperability with and use of intellectual capital developed in these languages as a basis for ontology development and further refinement is a key goal of the ODM. Since UML2 is an adopted OMG standard, we simply reference it in the ODM.

Three UML profiles have been identified to date for potential use with the ODM: UML4RDFS, UML4OWL and UML4TM, with the likely addition of a profile for the DL core metamodel. These enable the use of UML notations (and tools) for ontology modeling and facilitate “code generation” for corresponding ontology descriptions in RDFS, OWL and TM, respectively.

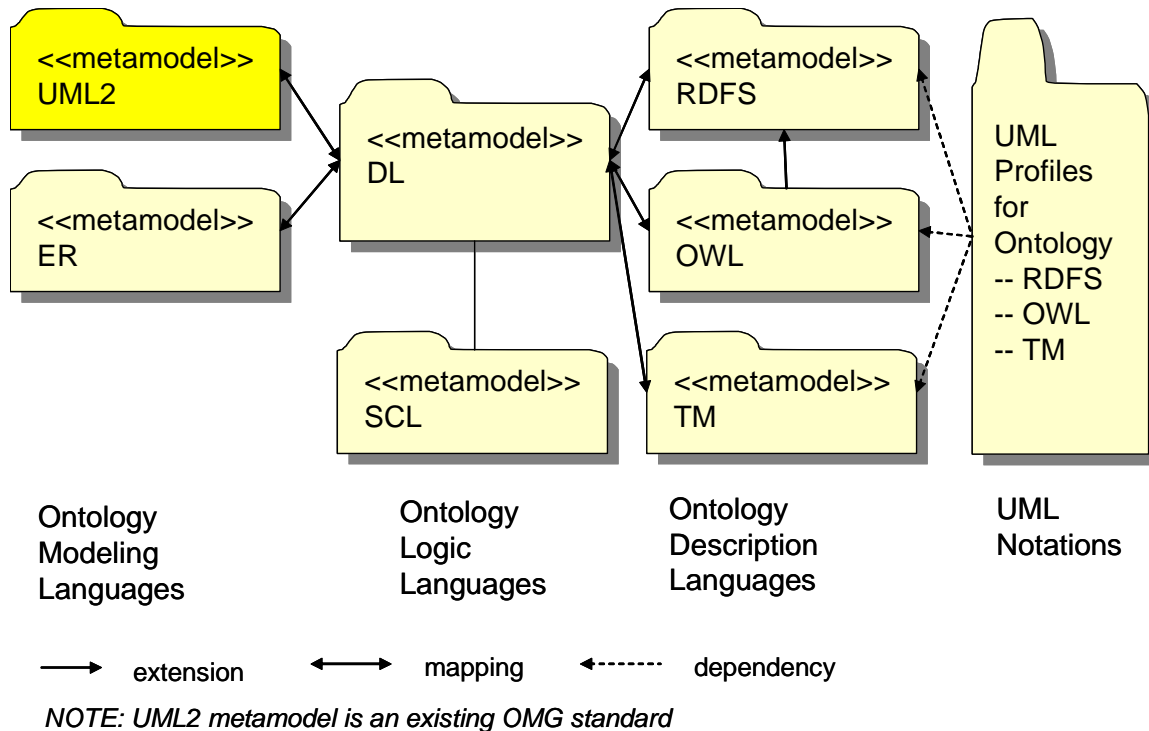


Figure 1 ODM Architecture

BASIC CONSIDERATIONS

The most important consideration in developing ODM was the requirement (from the original RFP) that it must be based on MOF2; our primary goals include leveraging MDA and EMF technologies, both of which depend on MOF. That is, MOF2 is used as the single, common language for metamodeling. Another important consideration is that we want ODM to directly support forward and reverse engineering of source material as well as ontologies. Therefore, the ODM team decided from the very beginning against developing an “abstract” metamodel that represents either the least or the most common denominator of the selected set of ontology definition languages. Rather, we elected to develop as many specific metamodels as required to maximize interoperability between those languages and other MOF-based applications and technologies.

WATERSHED ISSUES

Beyond the basic considerations, there are a number of major issues that were debated by the ODM team and whose resolutions have since been highly influential in the design and evolution of ODM. We call these “watershed” issues and we discuss them in detail in the following sections.

Objectives of ODM

As mentioned previously, the initial work product developed by the joint submission team, after some discussion on how to proceed, was the usage analysis. The ODM usage scenarios and goals document [2] identifies 22 requirements to support various usage scenarios for ontologies as well as knowledge-based applications that use them: 7 on structural features, 5 on generic concepts, 2 on runtime tools, and 8 on design-time tools. Among these, the following eight were identified as key objectives of ODM:

- Support ontologies expressed in existing description logics (e.g. OWL DL) and a limited set of higher order logic languages (e.g. OWL Full, SCL).
- Represent complex objects as aggregations of parts.
- Support multiple inheritance of complex types.
- Provide capabilities required to support physical world concepts, including time, space, bulk or mass nouns like ‘water’, and things that do not have identifiable instances.
- Support object concepts that have multiple facets of representations, e.g., conceptual versus representational classes.
- Provide a basis for information systems process descriptions to support interoperability, including such concepts as player, role, action, and object.
- Provide support for other generic concepts relevant to particular domains or universes of discourse.
- Support interoperation with other MOF and UML-based tools, including, for example, forward and reverse engineering of ontologies.

The first three relate to structural features, the next four to generic conceptual representation, and the last one to tool support.

As might be expected, the focus of ODM is on representation and not on applications or tooling, with the clear exception that an important consideration of ODM is to support forward engineering and reverse engineering.

What is Ontology

Ontology is a notion based in philosophy, introduced by the Artificial Intelligence community in the 1980s as a means for unambiguously describing real world concepts independently from their realization in software. Over the past two decades, knowledge representation methodologies and technologies have subsequently been used in other computing disciplines where there is a need to represent and share contextual knowledge or content independently from the applications that produce or consume them.

The following definition was adopted by the submission team from the ODM RFP [1]:

An ontology defines the common terms and concepts (meaning) used to describe and represent an area of knowledge. An ontology can range from a Taxonomy (knowledge with minimal hierarchy or a parent/child structure) to a Thesaurus (words and synonyms) to a Conceptual Model (with more complex knowledge) to a Logical Theory (with very rich, complex, consistent and meaningful knowledge).

This definition, and the subsequent usage scenarios related analysis, led to the determination that the ODM would ultimately include six metamodels (seven with UML2). One relates to Taxonomy: TM; four relate to Conceptual Modeling: UML, ER, RDFS, and OWL; and two relate to Logical Theory: DL and SCL.

UML Profiles vs. Metamodels

The UML2 profiling mechanism provides a means for extending UML through the use of stereotypes and tagged values. Early in the development of ODM, a number of people who are experts in UML but not necessarily well versed in knowledge representation questioned an approach that required development of new metamodel(s), if perhaps a UML profile for ontology could be considered sufficient. Again, once the usage scenarios and goals analysis was completed and as the team became more aware of the structural as well as semantic distinctions between UML and some of the other representation formalisms we were attempting to capture, we found it necessary to provide a combination of metamodels, mappings, and one or more UML profiles.

The metamodels are needed to precisely represent the abstract syntax of the selected set of ontology definition and conceptual modeling languages. The UML profiles facilitate the use of UML notation (and tools) for ontology modeling and for adequate generation of corresponding ontology descriptions.

UML Extensions vs. New Metamodels

Upon learning that the ODM team had decided to develop new metamodel(s) for ontology, some of the same interested parties suggested that these new metamodel(s) should be related to one another and to UML by extension (or subclassing) from the UML2 metamodel. The ODM team studied both the UML2 Infrastructure and UML2 Superstructure specifications to determine whether extensions from UML2 metamodel were possible or appropriate. We found that the UML2 metamodel in its entirety is extremely complex, and that using extension mechanisms to relate the ODM metamodels to UML2 would only serve to complicate the ODM itself. At the time, the UML2 metamodel was still evolving, which also weighed on our decision. In the end, we decided to develop the set of metamodels discussed here, none extending from the UML2 metamodel. This approach simplified the metamodels considerably and enabled us to avoid unnecessary dependencies on the UML2 metamodel.

Subclassing vs. Mapping

Once the decision was made to develop new metamodels, a major issue surfaced and lingered for some time: should there be a core metamodel such that all other metamodels extend (subclass from) this core metamodel? Since there was no obvious common core metamodel to start with, other than a trivial one that contained very simple metaclasses such as Element and NamedElement, the team decided to defer resolution until much later in the development process.

Ultimately, the six metamodels identified above were developed. Among these, the OWL metamodel naturally extends the RDFS metamodel, but none of the others are related by extension due to lack of straightforward alignment in underlying abstract syntax. Thus, the August 2004 draft of the ODM consists of four unrelated metamodels and one metamodel set: DL, SCL, RDFS/OWL, TM, and ER. To leave the situation as is would be undesirable, since most of these do share similar constructs. Prior to completing a final submission document, the ODM team will, in fact, develop mappings between the metamodels as shown in Figure 1.

Namespaces and Names

Finally, RDF Schema and OWL use URI references for naming purposes. MOF2 does not support URI references for naming at this juncture. Instead, it supports package scoped naming conventions. In particular, associations in MOF2 are locally named. To overcome this impedance mismatch, three prefixes are currently used in the RDFS and OWL metamodels to represent RDF, RDF Schema and OWL namespaces, respectively: RDF, RDFS, and OWL. For example, RDFSResource represents `rdfs:Resource` and RDFSsubClassOf represents `rdfs:subClassOf`.

ACKNOWLEDGMENT

The authors would like to thank the rest of the ODM co-submission team for ideas and contributions that led to the design of ODM: Robert Colomb, Dave Frankel, Patrick Emery, and Lewis Hart. This paper represents the personal perspectives of the authors and does not necessarily represent the common perspectives of the whole team.

REFERENCES

- [1] Ontology Definition Metamodel Request for Proposal, OMG Document ad/2003-03-40.
- [2] Lewis Hart, Patrick Emery, Robert Colomb, Kerry Raymond, Dan Chang, Yiming Ye, Elisa Kendall, and Mark Dutra, *Usage Scenarios and Goals for Ontology Definition Metamodel*, The Fifth International Conference on Web Information Systems Engineering (November 22 – 24, 2004).
- [3] Tim Berners-Lee and Mark Fischetti, *Weaving the Web: the Past, Present and Future of the World Wide Web by Its Inventor*, Orion Business, London, 1999.
- [4] F. Baader, D. Calvanese, D.L. McGuinness, D. Nardi, P.F. Patel-Schneider (editors), *The Description Logic Handbook: Theory, Implementation and Applications*, Cambridge University Press, January 2003.
- [5] Harry Delugach, *ISO/IEC WD 24707 Information technology – Common Logic (CL) – A Framework for a Family of Logic-Based Languages*, Pacific Northwest National Laboratory, Chantilly, VA, 7 June 2004.
- [6] See http://www.omg.org/techprocess/meetings/schedule/UML_2.0_Superstructure_FTF.html
- [7] T. R. Gruber and G. R. Olsen, An Ontology for Engineering Mathematics, in Jon Doyle, Piero Torasso and Erik Sandewell, eds. Fourth International Conference on Principles of Knowledge Representation and Reasoning, Morgan Kaufmann, 1994.
- [8] See EMF (Eclipse Modeling Framework), <http://www.eclipse.org/emf/>.
- [9] RDF Vocabulary Description Language 1.0: RDF Schema. W3C Recommendation 10 February 2004. Latest version is available at <http://www.w3.org/TR/rdf-schema/>.
- [10] OWL Web Ontology Language Reference. W3C Recommendation 10 February 2004. Latest version is available at <http://www.w3.org/TR/owl-ref/>.
- [11] OWL Web Ontology Language Semantics and Abstract Syntax. W3C Recommendation 10 February 2004. Latest version is available at <http://www.w3.org/TR/owl-semantics/>.
- [12] See ISO 13250-2 Data Model, <http://www.isotopicmaps.org/sam/sam-model/>.