

# Unifying MDA and Knowledge Representation Technologies

Colin Atkinson

University of Mannheim  
atkinson@informatik.uni-mannheim.de

## ABSTRACT

As model-driven approaches to software engineering and ontology-based approaches to knowledge representation have continued to expand their influence and importance in recent years they have inevitably started to encroach on each others' territories. There is therefore growing interest in understanding how precisely the two technologies relate to one another and how they can best be used together. Several papers have been published describing strategies for interchanging model and ontology information, and dedicated workshops on this topic have started to appear. This short position paper discusses the issues involved in integrating MDA and ontology representation languages and then argues the case for a core-level unification of the two technologies.

## INTRODUCTION

Model Driven Development as embodied by the MDA, and ontology-based knowledge representation as embodied by the semantic web, are the IT industry's primary visions for the future of software engineering and the World Wide Web respectively. However, as software applications become increasingly web-based, and web applications require increasing "intelligence", it is clear that these technologies will need to work more and more closely together in the future. There is consequently a growing interest in finding practical ways of using them effectively together.

Over the last couple of years researchers have put forward a number of different proposals for integrating Ontology Representation Languages (RDLs), exemplified by languages such as OWL [Bechhofer et. al. 2004] and DAML-OIL [McGuinness et. al. 2002], and MDA [CraneField 2001, Baclawski et. al. 2002, Djuric et. al. 2003, Falkovych et. al. 2003]. Last year (2003) the OMG also issued an official RFP for an "Ontology Definition Metamodel" to define an ontology definition language within the framework of the MDA [OMG 2003]. The drive to integrate the two technologies is definitely picking up momentum therefore. However, all the existing initiatives to date share two fundamental premises:-

Premise 1. MDA and ontology description technologies are inherently distinct technology spaces with at least some fundamentally distinct concepts and mechanisms.

Premise 2. the two technologies should be integrated by using "official" OMG extension mechanisms to define an ontology representation language within the OMG's MDA framework.

The term "official" OMG extension mechanism is used here to refer to the extension mechanisms explicitly recognized by the OMG as ways of introducing new language capabilities into the overall MDA framework, namely:-

1. the definition of a new "metamodel" or profile as an instance of (or effective instance of) the MOF,
2. direct extension of the UML metamodel through metaclass specialization,
3. indirect extension of the UML using the profile mechanism (i.e. stereotypes, tags etc.).

The basic idea behind current proposals is that the key features of the leading ontology representation languages such as OWL and DAML-OIL should be added to the MDA suite of standards by extending the UML and/or the predefined set of metamodels in one of these three ways. We therefore group these existing proposals together under the label of "extension-based" approaches.

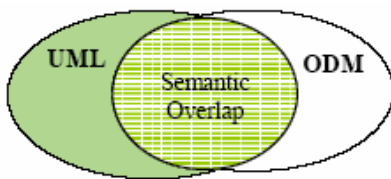
To the extent that MDA and ORLs have been developed by different communities, and employ a superficially distinct set of concepts, it is undeniably the case that they represent distinct technology spaces. However, it has never been unequivocally established whether these differences are truly fundamental consequences of the requirements of model and ontology representation, or whether they are merely reflections of their different development histories and backgrounds. If the latter, the case for extension-based integration approaches is significantly weakened, and the case for a more fundamental merging of the two technologies at

their core is strengthened. As a counterpart to the extension-based strategy for combining the technologies we therefore identify an alternative “unification-based” approach in which the two technologies spaces are merged from the ground up into a single, unified information representation framework.

The remainder of this paper first discusses the extent to which the first premise is true, and then based on the conclusions drawn, makes the case that the unification-based integration strategy holds the best prospect of effectively bringing the MDA and semantic web worlds together.

## ONTOLOGY ENGINEERING VERSUS MODEL ENGINEERING

The relative merits of the two basic strategies for integrating ORLs and MDA to a large degree depends on the extent to which they constitute distinct technology spaces [Kurtev et. al. 2002]. The accepted wisdom today is that ontology representation and model representation are two distinct problems and thus require languages which have at least some fundamentally distinct mechanisms. Figure 1, taken from the Ontology Definition Metamodel RFP [OMG 2003], gives perhaps the best graphical depiction of the prevailing thinking.



**Figure 1 Overlap of Modeling and Ontology Representation Concepts**

This diagram is actually intended to depict the overlap between the UML and the envisaged Ontology Definition Metamodel, but it reflects the commonly held view that ontology definition requires some fundamental capabilities which are not required for modeling, and vice versa.

As mentioned above, to the extent that it reflects the current relationship between the UML and ontology representation languages, Figure 1 is an accurate diagram. However, the premise that this is a natural consequence of the inherent properties of models and ontologies does not bear up to close scrutiny. Since there are no precise or universally accepted definitions of the terms “model” or “ontology” one must instead compare the mainstream practical interpretations or meanings of these terms in their relative domains.

According to the OMG’s MDA Guide –

“a model of a system is a description or specification of that system and its environment for some certain purpose”.

The ODM RFP, on the other hand, states that -

“a model is a representation of a part of the function, structure and/or behaviour of an application or system.... In MDA, a representation that is not formal ..... is not a model. Thus, a diagram with boxes and lines and arrows that is not supported by a definition of the meaning of a box, and the meaning of a line and of an arrow is not a model—it is just an informal diagram”

The ODM RFP’s definition of ontology, in contrast, is as follows -

“An “ontology” ..... defines the common terms and concepts (meaning) used to describe and represent an area of knowledge....A well-formed ontology is one that is expressed in a well-defined syntax that has a well-defined machine interpretation consistent with the above definition.”

The most obvious difference between these definitions is in the focus or scope of models versus ontologies. The definition of models emphasizes a “systems” focus whereas the definition of ontology emphasizes a “knowledge” focus. However, there is nothing to stop the “certain purpose” for which a model is used from being knowledge representation, and the “area of knowledge” for which an ontology is created from being the data for a computing system. In fact, this is what often happens in practice. A UML model of a particular body of knowledge is generally referred to as a conceptual model or domain model, while an ontology of the “knowledge” needed to drive a system is traditionally called the data model for that system. Both definitions above identify the need for a model and ontology to have well-defined-machine interpretable semantics.

In practice, the core part of an ontology as currently understood in the semantic web community is a taxonomy of the concepts in the domain of discourse. If an ontology only contains taxonomical information it is sometimes referred to as “lightweight”, whereas if it adds additional information about properties and relationships it is sometimes referred to as “heavyweight”. In both cases, however, a UML practitioner would instantly recognise an ontology as a conceptual or domain model expressed as a UML class diagram. The taxonomical part would be embodied as an inheritance hierarchy, class features as attributes and/or methods and relationships as various forms of association and/or dependency.

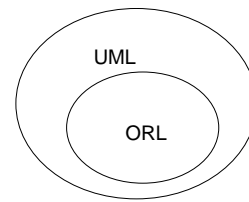
The close correspondence between ontologies and conceptual models captured as class diagrams is highlighted by the very similar processes used to create them. Consider, for example, the well known guide to developing ontologies by Noy and McGuinness [2004]. This development process is almost identical to the processes that one finds in software engineering text books for developing conceptual models using UML class

diagrams [Larman 2002]. In fact, replacing the term “ontology” with “conceptual model” or “domain model” in [Noy and McGuinness 2004] leads to perfectly acceptable set of guidelines for developing class diagrams in the UML.

The very fact that so many ontology practitioners use UML class diagrams to visualize their ontologies also strongly suggests that the UML is rich enough to capture the information that is conveyed in mainstream RDLs. There may be differences in the specific mechanisms used in the UML to capture certain characteristics of the domain of discourse [Guizzardi et. Al. 2002], but these do not diminish the language’s ability to fully represent the knowledge in an ontology. The situation is analogous to the differences between programming languages. Although C++ and Java have quite a number of distinct features (e.g. templates, multiple inheritance, interfaces etc.) no one would suggest that there are (sequential)<sup>1</sup> programs that can be written in C++ that cannot be written in Java and vice versa. The value of these programming language mechanisms manifests itself not in the range of programs that can be written but in the non-functional properties of programs (e.g. efficiency etc.). In contrast, the differences between ORLs manifest themselves not in the range of knowledge that they can describe but in the precision by which this knowledge is represented.

When taken in its entirety it is clear that the UML has far more features than is needed to define ontologies for the semantic web context. Thus, the UML has a set (quite a large set) of features which are not found in ORLs like OWL and DAML-OIL. The crucial issue, however, is whether there are any fundamental features needed to support ontologies which are not found in the UML. In other words, should the set of required ORL features be considered a proper subset of the UML or as depicted in Figure 1, an overlapping set? Some authors, such as Baclawski et. al. [2002] identify concepts in the leading ORLs such as “properties” which are not directly supported in UML. But as mentioned above, although C++ has mechanisms that Java does not have, it does not stop Java programmers from being able to write all the programs they need to write. They simply have to “program around” features that are not directly supported.

The feature differences between the UML and OWL, for example, are not significantly larger than those between the leading ORLs, or between the concepts represented in the various proposals for upper level ontologies [Guizzardi et. al. 2002]. Similarly, they are no larger than the differences that existed between the various modeling languages that were unified to create the UML such as OMT, Booch and the like.



**Figure 2 True relationship between UML and ORLs**

The bottom line is that there seems to be no well founded reason for labelling some languages, such as OWL and DAML-OIL, as “ontology representation languages” whilst denying this label to the UML. All the available evidence suggests that the UML is capable of capturing the same kind of knowledge that can be captured in ORLs. In other words, the UML should already be considered an ontology representation language. This means that Figure 2 is a much better depiction of the true relationship between ORLs and the UML than Figure 1.

### THE CASE FOR UNIFICATION

If the motivation for defining bridges between MDA and ORLs were simply to facilitate data interchange the extension-based approaches identified above would be more than adequate. In fact, defining different metamodels for different platforms (e.g. Java, C++ etc.) is one of the cornerstones of the MDA approach. The problem is that the proposed ODM does not correspond to a platform in this usual sense because it does correspond directly to any of the established ontology representation “platforms” like OWL. It is rather an attempt to add supposedly “missing” ontology representation features to the MDA framework in order to support the definition of platform independent ontologies. Moreover by issuing an RFP for a totally new metamodel for ontology definition, the OMG is implying that the UML should not be used for (or is not adequate for) defining ontologies. The message is that since the ODM is “for” creating ontologies the UML should be used just for “modeling”.

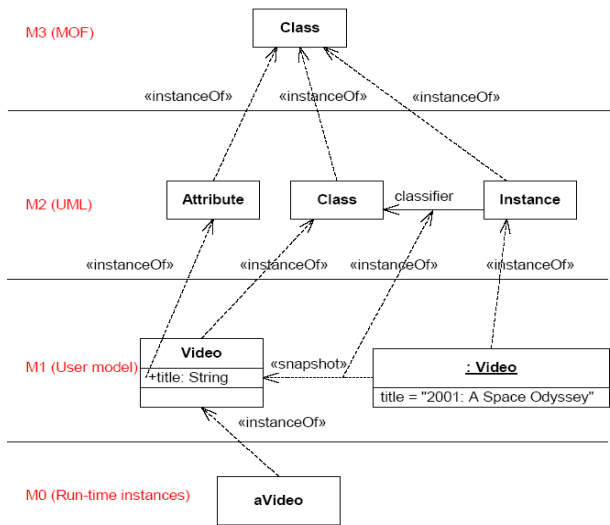
This message is unfortunate for several reasons. First, rather than integrating the two technologies it cements into place the idea that “modeling” and “ontology definition” are distinct activities. Second, it adds another language to the already over populated set of platform independent languages within the MDA framework. There are already two basic information representation languages under the MDA umbrella (UML and CWM) and adding a third will confuse things further. Once the ODM has been added to the collection, to select which of the three languages to use developers will be faced with the daunting task of working out whether they need to represent “data”, “knowledge” or “a model”.

The issue is further complicated by the fact that the MDA offers not one fundamental modeling language but two - the MOF and the UML - which are built upon (or share) a common core set of features. Since this common

<sup>1</sup> Ignoring concurrency mechanisms

core contains the basic features needed to represent ontologies, the end user has a further dilemma to wrestle with - when seeking to represent the information in a real word domain of discourse, should concepts in that domain be represented at the meta (i.e. M2) level as instances of the MOF, or at the regular model (i.e. M1) level using the UML (or the CWM or ODM). This question is much more difficult to answer in the context of ontologies than in the context of regular software development because the kind of information represented in ontologies is traditionally captured in metamodels in the MDA. So called “upper ontologies” from the semantic web community capture vertical “domain” information which in the MDA community have traditionally been embodied in metamodels or profiles (e.g. SPEM etc).

Adding another way of capturing predefined domain knowledge (i.e. via the ODM) to the MDA further complicates what is already the source of a great deal of confusion in MDA development - namely the location of information with the OMG four level infrastructure. As illustrated in Figure 3, “data” has traditionally been “housed” in the bottom “M0” layer of the four-level hierarchy. However, in the latest (UML 2.0) interpretation of this infrastructure, the bottom level is now inhabited by “real world” objects while models of these objects (so called instance specifications) occupy the M1 level alongside classes. While this solves some problems, it leads to the somewhat counterintuitive situation that the M1 level of the infrastructure contains both type and instance information.



**Figure 3 MDA Model Hierarchy**

The fact that these problems are only evident in the MDA should not be taken as an indication that they do not exist (or have been solved) in the ontology community. On the contrary the vast majority of ontologies are constructed according to the simple “two level” framework which the “modeling” community

moved on from several years ago. Concepts in the domain of discourse are viewed as types or “universals”, while specific “knowledge” is viewed as a set of instances or “individuals”. Only one variant of a mainstream ORL-OWL Full [Bechhofer et. al. 2004] allows a class to be treated simultaneously as a collection of individuals and as an individual in its own right. However, there is no discussion of how this ability to define “meta” or “higher-order” classes should be used, and none of the problematic issues associated with multiple meta levels in MDA have been addressed. This includes the issue of how to simultaneously represent both the type and instance fact of higher order elements [Atkinson and Kühne 2002], and how to manage the different dimensions of metaness - the one dealing with “form” or linguistic classification and the other dealing with “content” or domain classification [Atkinson and Kühne 2003].

These issues go to the very heart of what it means to “model” and to “define ontologies”, and any attempt to unify modeling and ontology representation technologies at a fundamental level should clearly take them into account. The implication is that a unification of the two currently separate technology spaces should occur not just at the level of the UML but should occur at the very core of the MDA infrastructure - that is, simultaneously at the MOF, the UML and the four level model hierarchy. Only then will a clean unification be possible that truly integrates the two technology spaces and addresses the issues discussed above.

Another strong argument for unifying the two technologies at this level is that it would go along way toward solving one of the fundamental obstacles currently standing in the way of the MDA vision - the lack of a well-defined, machine-comprehensible semantics for the UML. As discussed in the previous section, although the UML is rich enough to be able to represent taxonomic information the semantics of the resulting “knowledge” is not as sharp as in specialized ORLs such as OWL. Because of their heritage, specialized ORLs usually have precise semantics built on some well founded formal system such as predicate logic. Unifying MDA and ORLs at a core level could therefore place MDA on a sounder semantic footing and thus facilitate the intelligent model-to-model transformation which is the very point of MDA technology.

### 1.1 The Role of Extension-Based Integration

The arguments in favour of fundamental unification of MDA and ORL technologies does not mean that extension-based integration as defined above does not have an important role to play. However, it does mean that the role of ontology-oriented extensions needs to be reinterpreted (or presented differently). Instead of serving to integrate the two technologies for the purposes of supporting platform independent knowledge represent-

tation, new metamodels and/or profiles should be used to support platform-specific knowledge representation in a particular ORL such as OWL.

Even if a fundamental unification of MDA and ORL technologies of the kind described above is attained there will also be a need for this kind of platform specific extension to facilitate interchange with existing knowledge representation technologies. Such profiles or metamodels would have names like OWL lite profile or DAML-OIL metamodel. There would be no need for a generic platform independent metamodel such as the proposed ODM since this would define mechanisms already supported in the unified core. Neither would there be a need for extensions to the UML, either direct or indirect.

## CONCLUSION

In the first part of this paper we identified two distinct strategies for bringing the ontology-based knowledge representation world and the MDA world together – one based on the use of “official” MDA extension mechanisms to add ontology representation features to the MDA infrastructure, the other based on a fundamental unification of the two technologies from the ground up. We also made explicit the underlying premise that leads most existing integration approaches to follow the extension-based strategy – namely the premise that modeling and ontology representation are two inherently distinct technology spaces.

In the second section, we challenged this accepted wisdom and argued that there really are no fundamental differences between modeling and ontology representation, and that the only sensible integration approach is to unify the two “technologies” into one single technology space. In fact, one may ask why two separate communities or technologies arose in the first place, since the difference between “a model”, “data” and “knowledge” is at best highly philosophical, and at worst vacuous. Thus, any continued dichotomy between modeling and knowledge representation in the software and web communities would be totally artificial.

The advantage of bringing two communities together that have essentially tackled the same problem from a different perspective is that they bring different strengths to the table. The ontology community brings to the table languages which have a much firmer grounding in formal representation technologies such as predicate logic and set theory, and thus provide a base for a formal semantics of a unified language. On the other hand, the MDA community brings to the table a richer set of mechanisms, and a more sophisticated treatment of advanced representation issues such as those arising from multiple meta levels and dimensions. Thus, any resulting unified language will be the richer because of this more diverse experience base.

The final main point made in the paper is that even if

such a unified language is developed as the core for the MDA, the extension-based approach still has a valuable role to play. However, this role is to support the mapping of platform independent “knowledge” into platform specific “knowledge” represented in an alternative ontology representation “platform” such as OWL or DAML-OIL.

## REFERENCES

- Atkinson, C. and Kühne, T. (2002). Profiles in a Strict Metamodeling Framework, *Science of Computer Programming*, 44(1).
- Atkinson, C. and Kühne, T. (2003). Model-Driven Development: A Metamodeling Foundation. *IEEE Software* 20(5).
- Baclawski, K., Kokar, M.K., Kogut, P., Hart, L., Smith, J.E., Letkowski, J. and Emery, P. (2002). Extending the Unified Modeling Language for ontology development. *International Journal Software and Systems Modeling (SoSyM)*, 1(2).
- Bechhofer, S., van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D.L. Patel-Schneider, P.F. and Stein, L.A. (2004). *OWL Web Ontology Language Reference*. W3C Recommendation.
- Cranefield, S. (2001).. Networked Knowledge Representation and Exchange using UML and RDF. *Journal of Digital information*, 1(8).
- Djuric, D., Gasevic, D., and Devedzic V. (2003). A MDA-based Approach to the Ontology Definition Metamodel, 4<sup>th</sup> *International Workshop on Computational Intelligence and Information Technologies*, Faculty of Electronics, Niš, Serbia.
- Falkovych, K., Sabou, M. and Stuckenschmidt, H. (2003). UML for the Semantic Web: Transformation- Based Approaches, in Omelayenko, B. and Klein, M. eds. *Knowledge Transformation for the Semantic Web, Frontiers in Artificial Intelligence and Applications*, Vol. 95, IOS Press.
- Guizzardi, G., Herre, H., Wagner, G. (2002). Towards Ontological Foundations for UML Conceptual Models, *1st Int. Conf. on Ontologies, Databases and Applications of Semantics (ODBASE 2002)*, Springer LNCS 2519.
- Kurtev, I., Bézivin, J. and Aksit, M. (2002). Technological Spaces: An Initial Appraisal, *Confederated International Conferences CoopIS, DOA, and ODBASE*.
- Larman, C. (2002). *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and the Unified Process (2nd Edition)*, Prentice Hall.
- McGuinness, D.L., Fikes, R., Hendler, J. and Stein, L.A. (2002). DAML+OIL: An Ontology Language for the Semantic Web, *IEEE Intelligent Systems*, 17(5).
- Noy, N.F. and McGuinness, D.L. (2004). *A Guide to Creating Your First Ontology*, Knowledge Systems Laboratory, Stanford University.
- OMG (2003). *Ontology Definition Metamodel Request for Proposals*, OMG Document: ad/2003-03-40.