On the Relationship of Ontologies and Models

Colin Atkinson, Matthias Gutheil and Kilian Kiko

Chair of Software Technology
University of Mannheim
A5, 6
68161 Mannheim
atkinson@informatik.uni-mannheim.de
gutheil@informatik.uni-mannheim.de
kilian.kiko@gmx.net

Abstract: As models and ontologies assume an increasingly central role in enterprise systems engineering the question of how they compare and can be used together assumes growing importance. On the one hand, the semantic web community is starting to propose a central role for "ontologies" in software engineering, while on the other hand the software engineering community has over the last few years been enthusiastically embracing "models" as the core artefact in software development. Superficially, however, ontologies and models are very similar, and in fact are sometimes visualized using the same language (e.g. UML). This has given rise to a variety of different interpretations of the roles of the two technologies, and had generated a certain degree of confusion about the relationship between them. So what exactly is the difference between ontologies and models and why are both concepts needed? Are they basically the same thing viewed from different angles or is there some fundamental difference between them beyond the idiosyncrasies of current tools and languages? This paper discusses this question. After considering common informal distinctions between ontologies and models, and analyzing their fundamental definitions, we propose how they might usefully be distinguished in the future.

1 Introduction

For many years the disciplines of software engineering and artificial intelligence have been polite neighbors, occasionally exchanging high level ideas and techniques, but retaining distinctly disjoint communities, languages and foci. However, with the trend towards embedding intelligent devices in almost all parts of our environment (i.e. ubiquitous computing) and increasing interconnection across the enterprise and the Internet, this separation is rapidly disappearing. As evidenced by the growing number of papers and workshops dealing with the overlap of the two disciplines (e.g. SEKE [SE05], VORTE [VO05], MDSW [MD04], SWESE [SW05], ONTOSE [ON05], WoMM [Wo05]), the integration of software engineering and artificial intelligence technologies has already generated considerable academic interest, and developers are starting to face the question of how to use these technologies together in practical software engineering projects.

The central question is how to represent information. Traditionally, software engineers have taken a very pragmatic approach to data representation, encoding only the information needed to solve the problem in hand, usually in the form of language data structures or database tables. However, over the last ten to fifteen years, the notion of "modeling" has risen to prominence in software engineering practice, and the representation of "meta" information and metamodels has become an increasingly important technique for creating flexible and adaptive applications. These ideas are today wrapped up in the notions of Model Driven Development (MDD) or Model Driven Architecture (MDA).

Artificial Intelligence, on the other hand, has traditionally focused on the representation of "knowledge" to support "intelligent" behavior and the discovery of new information using various kinds of inference techniques. Over the last ten to fifteen years the notion of "ontology" has risen to prominence in Artificial Intelligence research, and the representation of type hierarchies and taxonomies has become an increasingly important technique for influencing the behavior of applications. These ideas are closely intertwined with the notion of the Semantic Web.

In the literature dealing with the integration or co-use of these two technologies the accepted wisdom seems to be that "models" and "ontologies" are fundamentally "different", and must somehow be jointly accommodated in modern software engineering practices. This is most strikingly evidenced by some of the recent initiatives of major standards definition bodies in the software and web domains. For example, the OMG recently released a proposal to standardize an Ontology Definition Metamodel (ODM) as a "bridge" between the semantic web and MDA "technology spaces" while the W3C released a vision document on software engineering [W3C06] that proposes an "Ontology Driven Architecture" as a "supplement" to MDA. Also a paper [FHK04] by a team of MDA and semantic web experts proposes the notion of the "Model Driven Semantic Web". Without the assumption that modeling and ontology development are different technologies, there would be no need for any of these proposals.

Our goal in this paper is to explore the underlying reasons for this accepted wisdom and to clarify the relationship between models and ontologies. Ultimately we aim to propose practical criteria for classifying information representations as ontologies or models. There is a need for this as several approaches currently use the terms interchangeably. In the next section we start by reviewing some of the main informal distinctions currently found explicitly or implicitly in the literature. Then, in section 3 we go back to first principles and compare models and ontologies based on their fundamental definitions. Section 4 presents our proposal for a useful distinction between ontologies and models in the future. Finally, section 5 concludes with a summary of the lessons learned from the exercise.

2 Common Informal Distinctions

Although formal definitions for ontologies and models are available (see section 3), several informal distinctions have taken hold and have become "accepted wisdom" in certain communities. Generally reflecting the different histories and foci of different communities, these fall into two groups - those distinctions focusing on the purpose of a model/ontology and those based on the properties of a model/ontology. In order to discuss them and avoid writing "model/ontology" whenever we wish to be unspecific about whether the subject is a model or an ontology, we will use the neutral term "information representation", abbreviated IR. The referenced articles do not necessarily state (all of) the mentioned assumptions explicitly, but they can be easily inferred by "reading between the lines"

2.1 Purpose-based Distinctions

Purpose-based distinctions explain or motivate the need for a distinction between ontologies and models based on the goal that they are trying to achieve or the role they are trying to fulfill.

IF1. Models focus on Realization (Ontologies do not)

One of the first papers that articulated a difference between modelling and ontology development was the highly influential paper by Noy and McGuinness that proposed a method for developing ontologies [NM01]. The basic argument is that models focus on realization issues while ontologies focus on capturing abstract domain concepts and their relationships. Their viewpoint is concisely expressed in the following paragraph from section 1 of their paper:

> *"Some ontology-design ideas in this guide originated from the literature on object-oriented design (Rumbaugh et al. 1991; Booch et al. 1997). However, ontology development is different from designing classes and relations in object-oriented programming. Object-oriented programming centers primarily around methods on classes□ a programmer makes design decisions based on the operational properties of a class, whereas an ontology designer makes these decisions based on the structural properties of a class. As a result, a class structure and relations among classes in an ontology are different from the structure for a similar domain in an object-oriented program."*

Issues: While the last statement is correct it suggests that all object-oriented models are focused on realization. This is because it mixes object-oriented programming (OOP) with object-oriented design (OOD) and omits object-oriented analysis (OOA). In fact, there is a long and successful tradition of using models at all levels of abstraction in software engineering [La01], and the "modeling" works cited by Noy and McGuiness emphasize that object-orientation can be used seamlessly across all development phases. The first [Ru91] actually focuses much more on analysis level "domain" modelling than program modelling, and it is mainly the object-oriented analysis techniques from this book which were adopted by Noy and McGuiness in their guide. In the intervening years, the UML and MDA literature has become even more explicit about the fact that models can exist at all levels of abstraction and can be used exclusively for describing domains. The Computation Independent Model (CIM) in the MDA paradigm plays exactly this role.

IF 2. Ontologies are for run-time Knowledge Exploitation (Models are not)

One of the common roles assigned to ontologies in software engineering is realize "intelligent databases" that can offer various kinds of reasoning services on data at run-time. This is the essential idea behind the "Ontology Driven Architecture" proposal in the W3C's software engineering vision document [W3C06] describing software engineering uses for ontologies. However, the notion that it is necessary to have ontologies or ontology based systems to build such applications is based on the assumption that "model-based" IRs are not intended to contain instance data or be accessible at run-time. For example, the W3C's paper [W3C06] states that

> *"[…], currently MDA has not been applied for run-time relevant characteristics of component management, such as which version of an application interface requires which versions of libraries. MDA requires a compilation step preventing changes at runtime which are characteristic for component management. Besides, an MDA itself cannot be queried […]. Hence, there is no way to ask the system whether some configuration is valid or whether further elements are needed."*

Issues: Since UML style modeling originally evolved to support the creation of applications in traditional technologies, it is true that instance data (instances of classes) has traditionally not been stored at run-time in a UML-based (i.e. model-based) form. Instead, instance data has traditionally been stored and manipulated either in databases or as objects in a programming language such as C++ or Java. However, this does not mean that models are somehow incompatible with or incapable of storing instance data at run-time. On the contrary, recent developments in MDA technology have been designed to do just this. MOF repositories are designed to provide run-time access to all forms of model data [HS04], [MHM05].

IF 3. Ontologies are for Representing Web Based Information (Models are not)

Because of the recent association of ontologies and high profile ontology representation languages with the web, there is an inference by some that ontologies are mainly for representing web based data while models, being intended for software engineering, are not. This is why some people have suggested enriching the semantic web with “models” and model based technologies [FHK04].

Issues: The semantic web standards focus on web-related issues in their language definitions. However, this is not an aspect of ontologies per se. There are numerous examples of ontologies based on languages like FLogic [KLW95] that are not "web-related".

2.2 Property-based Distinctions

Property-based distinctions explain or motivate the need for a distinction between ontologies and models based on the fundamental characteristics of an IP.

IF 4. Ontologies are Formal (Models are not)

An often cited shortcoming of the UML from the perspective of its ability to support the description of ontologies is its lack of formal semantics. “Formal” in this context in contrast to section 3 does not mean machine-readable but “explicit and precise”, and by extension, subject to mathematical or logical reasoning. Many papers [Dr05, Br04, Ba02, Ha04] therefore distinguish “ontologies” from “models” because they expect the former to be represented with well defined semantics in a language like OWL, and the latter to be represented in a less precise manner in a language like the UML.

Issues: Although the UML 2.0 standard [OMG03b, OMG04] still lacks a precise (i.e., model-theoretic [Ta56]) semantics definition, the concepts specified in the UML are described and used so clearly and explicitly in a common standard way that they can easily be mapped into a language like OWL [Ha04, Br04], DAML [Ba02], DLR [Ca02] or FOL [AK06]. In addition, the axiom language OCL has a precise semantics [Ri01] that makes OCL-enhanced UML models just as formal as OWL ontologies [AK06]. Note, that formality does not imply expressiveness. Because of their different “world-views” the expressiveness of both languages is hard to be compared [AK06].

IF 4. Ontologies can support Reasoning (Models can not)

Following on from the previous two reasons, a common argument is that ontologies support “reasoning” while models cannot (or do not).

Issues: Although the UML was not initially devised for reasoning, there is nothing to stop reasoning steps from being applied to UML models as demonstrated by Cali et al. [Cali et al.]. They investigated the expressiveness of the class diagram subset of UML in comparison with a description logic. As long as models are given enough precision (using e.g. OCL) they can support the same reasoning algorithms as ontologies. However, it is not possible to use the algorithms used on OWL (e.g. subsumption) on UML models as the two languages have different semantics (UML has closed world semantics, while OWL has open-world semantics). A common corollary of this assumption is that models are exclusively intended for transformation (i.e. for serving as the source and target of transformations) while ontologies are not. This again is a reflection of their respective histories rather than their inherent properties.

IF 5. Models use the Close World Assumption (Ontologies use the Open World Assumption)

Another grounds for distinction, which is only really of interest to people with detailed knowledge of modelling and ontology representation languages is that models use the close world assumption while ontologies use the open world assumption. The different assumptions have an impact on the interpretation of IR concepts. This means that concepts that are apparently the same have slightly different denotations if viewed from different perspectives.

Issues: This distinction is correct for UML models and OWL ontologies. However, it is not possible to generalize from this to other types of ontologies (e.g. those in the language Prolog) since they can be explicitly based on a closed-world assumption. That all types of UML models are perceived as having closed-world semantics is historical rather than inherent. OOA models and CIMs have no relation to the realization of the system they describe and could easily be written from an open-world perspective.

2.3 Consequences of the Diversity of Criteria

Since there is no widely accepted definition of what distinguishes a model from an ontology (i.e. when an IR deserves to be called an ontology and when it deserves to be called a model), none of these distinguishing criteria is wrong per se. They are simply more or less widely held reasons for labeling IRs as ontologies and/or models depending on the specific context in which they are being used. We also recognize that different members of the software engineering and AI communities will take issue with some of these distinctions and that the strength and influence of each is subject to debate. However, the overriding (and we believe undeniable point), is that the shear number and diversity of these criteria causes great confusion in academia and industry and is an increasing hindrance to their effective co-usage and integration.

This problem is exacerbated by differing definitions of the notions of models and ontologies within sub communities of the respective communities (for example, different sub communities have different definitions for ontology) and confusion about the relationship of the abstract notion (of model and ontology) to the de facto standard languages in each community (UML and OWL respectively). Both the software engineering and the semantic web/AI communities have a powerful flagship language which is widely used for the respective purpose of modeling and ontology representation. However, the strength of these languages blurs the boundary between the idiosyncrasies of the respective language and underlying notion.

The consequences of this lack of consensus, and the pitfalls of trying to define new technologies within this vast array of possible assumptions and interpretations is highlighted by recent high profile attempts to bring the two worlds together. Moreover, these questions are not just academic. Because of the current separation of the technologies for modeling and ontology representation, companies have to make decisions about whether their particular IR needs should be represented as one or the other. This decision usually has to be taken early (because modeling at least is usually done early in a project lifecycle) and once taken is difficult to change. Worse still, if a company wishes to do model-like things with their IR at some stages in the lifecycle and ontology-like things at other stages they have no option but to duplicate their IRs, even though the concepts used in each have a very high overlap.

3 Towards a Precise Characterization of Models and Ontologies

Having outlined some of the commonly found informal distinctions in this section we try to shed light on the relationship between ontologies and models based on their fundamental definition. From these we try to identify more precise and explicit distinctions between the two.

3.1 Models

The term model has a long history of use in other disciplines (e.g. engineering), but it is used for a variety of different purposes and is defined in a variety of different ways. In software engineering you can find several different types of models, each possessing its own distinct definition and interpretation. Models can be classified (inter alia) according to the development phase they are used in (e.g. analysis models, design models), the purpose they are used for (problem domain, solution domain) the degree of abstraction they represent (e.g. platform independent models, platform specific models), the system aspect they describe (e.g. data models, behavioural models) or the language they are written in (e.g. UML models, CWM models). However, all of these models share a common core purpose, definition and interpretation that is also true for models in other disciplines. A general and generic definition adapted from Herbert Stachowiak's general model theory of 1973 [Wi66] is the following:

DM1. “A model is a homomorphic (or isomorphic) mapping of a subject matter into a system of symbols.”

The MDA guide offers the following definition [OMG03a]:

DM2. “A model of a system is a description or specification of that system and its environment for some certain purpose”

The core ideas are the mapping of a universe of discourse, the abstraction of detail and the pragmatic usage for a certain purpose. Usually these models are linguistic models, i.e., they are linguistically verbalized. The system of symbols (vocabulary) is then viewed as a language.

3.2 Ontologies

The most famous definition for ontologies is given by Gruber [Gr93]:

DO1. "An Ontology is an explicit specification of a conceptualization."

Because this is a very broad definition, the ontology community generally uses it to highlight the different character of an ontology with respect to other kinds of information representations. Studer and colleagues [SBF98] extended it as follows:

DO2. "An ontology is a formal, explicit specification of a shared conceptualization. Conceptualization refers to an abstract model of some phenomenon in the world by having identified the relevant concepts of that phenomenon. Explicit means that the type of concepts used, and the constraints on their use are explicitly defined. Formal refers to the fact that the ontology should be machine-readable. Shared reflects the notion that an ontology captures consensual knowledge, that is, it is not private of some individual, but accepted by a group."

This definition introduces the aspects of formality and commitment. A representation that is not formal is useless for machine to machine communication and reasoning. This definition also regards ontologies as referring to consensual knowledge, so that the effect of ambiguity is minimized. These two aspects are at the core of many applications of knowledge representations (e.g. the Semantic Web).

A recent, implementation oriented definition [Ma02] describes the current use of the term in ontology engineering using further technical terms:

DO3. "[…] an ontology refers to an engineering artefact, constituted by a specific vocabulary used to describe a certain reality, plus a set of explicit assumptions regarding the intended meaning of the vocabulary. Usually a form of first-order-logic theory is used to represent these assumptions, vocabulary appear as unary and binary predicates, called concepts and relations, respectively."

This last definition subdivides the notion of ontology into a vocabulary, which concretizes the abstract model from the second definition, and a set of statements referring to the meaning of the vocabulary.

3.3 Observations

From these definitions we can make the following important observations

O1. Any IR that fulfils the conditions DO1, DO2 and DO3 for being an ontology also fulfils the requirements DM1 and DM2 for being a model. It follows from their core definitions, therefore, that all ontologies are models, but not all models are ontologies.

O2. None of the purpose-oriented characteristics identified in section 2, are mentioned in the core definitions. Support for reasoning, intelligent databases and the description of web accessible information are not therefore required for conformance to DO1, DO2 and DO3.

O3. Of the property-oriented characteristics identified in section 2, only IF4, formality, is mentioned in the core definition DO3. There is no requirement for open or closed word interpretations for either models or ontologies.

O4. There is no mention of the intended scope of a model in either of the core definitions DM1 and DM2 (i.e. there is no reference to whether models are or are not intended for representing shared information). However, there is no requirement that models are restricted to the representation of private (i.e. non shared) data.

O5. When supported by OCL, IRs in the MOF/UML can be created that satisfy DO1, DO2 and DO3 and thus can be considered ontologies based on these core definitions.

O6. IRs created with OWL that have no shared understanding contradict DO2. It is therefore possible to create IRs in OWL that are no ontologies.

4 Proposed Criteria

From the above discussion we can conclude that in fact none of the informal distinctions given in section 2 is actually justified based on the core definitions of ontologies and models in section 3. Moreover, we can also conclude that all IRs are in fact to be regarded as models. The critical question, therefore, is whether it is possible and useful to distinguish ontology models from non-ontology models in some way, and on what characteristics should such a distinction be based? To try to answer the question we start with five characteristics that can be found in the definitions in section 3.

1. Conceptualization
2. Explicit
3. Machine Readable
4. Based on First-order Logic
5. Shared

Since they are in the definitions we can safely say that these are necessary characteristics of an ontology. This means that any model that does not possess all of these properties is not an ontology. However, are they sufficient?

Of course we could define them to be so, but this would force us to accept certain kinds of models routinely used in software engineering today as ontologies. These are the so called "domain models" or "computation independent models" as they are known in MDA, which are usually the starting point for software engineering projects. Their main purpose is to explicitly and clearly describe the concepts in a particular domain of interest, and thus represent a conceptualization of the domain. All computer generated models (e.g. UML models) are machine readable and if supported by OCL can make precise statements in first order logic. They are also definitely intended to support the sharing of information, at least amongst the development team, but more usually among all stakeholders.

If the above characteristics were regarded as sufficient, therefore, one would be forced to regard software engineering domain models and indeed any other precise (i.e. OCL enhanced) model which is intended to be shared as an ontology. However, since almost all models are intended to be shared this would exclude few models from the subset of ontologies and thus would not be a very helpful distinction,

Neither the informal characteristics in section 2 nor the core definitions in section 3 provide a basis for a useful way of distinguishing ontologies from normal models. We therefore believe that the most practical approach is to revert back to the original meaning of "ontology" from philosophy and other domains of research and to adopt a purpose-based distinction which captures the fact that a model is intended to have universal scope. More specifically, we propose to use the term "ontology" to characterize models which are intended to capture some standard or universally applicable information of the kind that might be found in an encyclopaedia or widely accepted authority. Denying the label ontology to a model would thus indicate that it was of limited scope of interest to, or sanctioned by, a limited group of people. Examples of IRs that deserve the term "ontology" according to this definition are the so called "general ontologies" [vH97] like the Mereology Ontology [Bo97], "upper-level" or "top-level ontologies" [vH97] like the Suggested Upper Merged Ontology [NP01] and "domain ontologies" [vH97] that represent a standardized model of the respective domain.

5 Conclusion

In this paper we have attempted to bring together and summarize the various informal criteria (both implicit and explicit) that are used in different sub communities of the AI and software engineering disciplines to distinguish between models and ontologies, and by referring to widely accepted core definitions of these concepts, we have attempted to filter out which of these criteria are justified and useful in practical engineering projects. From this exercise we learned three main things.

Firstly, and perhaps not surprisingly, we learned that from the core definitions of models and ontologies, the latter are to be regarded as a subset of the former. In other words, all ontologies satisfy the criteria for being models and thus should be regarded as such.

Secondly, and perhaps more surprisingly, we learned that only one of the currently used informal criteria is actually justifiable from the core definitions of models and ontologies widely accepted in the IT literature. The most surprising aspect of this is that the features which are usually used to motivate the use of OWL (the flagship ontology language) instead of UML (the flagship modelling language) are not actually required in the core definitions. This includes features such as the use of the open world rather than closed word assumptions and the associated support for the discovery of new knowledge based on such reasoning processes as subsumption.

Thirdly, if one views the five characteristics that are given in the core definitions of ontologies as being sufficient as well as necessary conditions, then almost all (OCL-enhanced) models used in modern software engineering qualify as ontologies. The one debatable characteristic is the criteria that the representing information be shared, but since they are usually intended to support communication almost all models arguably satisfy this criterion. Certainly the subset of models known as domain or computation independent models satisfy this criterion since their prime goal is to facilitate communication between all stakeholders in a software development process.

These lessons have some important consequences when it comes to the future integration and co-use of the two technology spaces. One important consequence is that most of the recently proposed visions for the integration or joint use of the two technologies, such as the Ontology Driven Architectures, Ontology Definition Metamodel and Model Driven Semantic Web ideas are left with little of meaning or value. For example, if ontologies are already models, what does an ontology driven architecture add to the notion of Model Driven Architecture. If most models are ontologies, why do we need a general Ontology Definition Metamodel? Isn't the UML metamodel not already an Ontology Definition Metamodel and if so would not language specific metamodel, such as an OWL metamodel, therefore not be sufficient to accommodate other representation formats with the MOF framework? And if ontologies are already models, what is the meaning of a model-driven semantic web? We do of course understand and see the value in the ideas that the authors of these proposals are trying to convey. What we are questioning here is the uses of the labels "ontology driven", "model driven" etc to characterize these ideas.

Another important consequence is that without changes to the currently used ways of distinguishing between models and ontologies the confusion highlighted in this paper will only grow and become more widespread, at least in the short term, with non trivial consequences for industrial users of these technologies. Under such circumstances, in the long term it can only be hoped that one of the communities grows to accept the terminology of the other group, so that either the AI community starts to refer to their IRs as models or the software engineering community starts to refer to their IRs as ontologies. But this will be a long and painful process.

The only other approach is for both communities to unite around a single unified definition of what criteria are sufficient for a model to be regarded as ontology along the lines of the proposal that we have suggested in this paper. This will not only reduce the level of confusion but will significantly smooth and accelerate the integration of the two technology spaces.

References

[AK06] Atkinson, C. and Kiko, K.: A Detailed Comparison of UML and OWL, submitted to Transaction on Software Engineering, 2006.

[Atk04] Colin Atkinson: Unifying MDA and Knowledge Representation Technologies, The Model-Driven Semantic Web Workshop (MDSW 2004), September, Monterey CA 2004.

[Ba02] Baclawski, K, Kokar, MM, Kogut P., Hart, L., Smith, JE, Letkowski, J. and Emery, P.: Extending the Unified Modeling Language for ontology development, Software and System Modeling, vol. 1 , pages 142-156, 2002.

[BHM05]Butler, J., Hubbly, R. and Melo, W., An MOF-based Repository for Enterprise Architecture Models, IBM developerWorks Document, March 2005

[Bo97] Borst, WN: Construction of Engineering Ontologies, PhD thesis, Centre for Telematica and Information Technology, University of Tweenty. Enschede, 1997.

[Br04] Brockmans, S., Volz, R., Eberhart, A. and Löffler, P.: Visual Modeling of OWL DL Ontologies Using UML, Int'l Semantic Web Conference, 2004, pp. 198-213.

[Ca02] Cali, A., Calvanese, D., De Giacomo, G. and Lenzerini, M.: A Formal Framework for Reasoning on UML Class Diagrams, Lecture Notes in Computer Science, Vol. 2366, p. 503-512, 2002.

[CP99] Stephen Cranefield and Martin Purvis: UML as an Ontology Modelling Language, in Proceedings of the IJCAI-99 Workshop on Intelligent Information Integration, held on July 31, 1999 in conjunction with the Sixteenth International Joint Conference on Artificial Intelligence City Conference Center, Stockholm, Sweden, 1999.

[Dr05] Djuric, D., Gasevic, D. and Devedzic, V.: Ontology Modeling and MDA, Journal of Object Technology, vol. 4, no. 1, pages 109-128, 2005.

[FHK04] Dave Frankel, Pat Hayes, Elisa Kendall and Deborah McGuinness: The Model Driven Semantic Web, The Model-Driven Semantic Web Workshop (MDSW 2004), September, Monterey CA 2004.

[Go01] Gove PM: Webster's Third New International Dictionary. Of the Englisch Language. Unabridged, Könemann, 2001.

[Gr93] Thomas R. Gruber: A Translation Approach to Portable Ontology Specifications, Knowledge Acquisitation, 5(2) pages 199–220, 1993.

[Ha04] Hart, L., Emery, P., Comb, B., Raymond, K., Taraporewalla, S., Chang, D., Ye, Y., Kendall, E., Dutra, M.: OWL Full and UML 2.0 Compared, OMG TFC Report, 2004.

[HS04] Hoessler, J., and Soden, M., OCL Support in MOF Repositories, First European Workshop on Model Driven Architecture with Emphasis on Industrial Application, Enschede, The Netherlands. March, 2004

[Ke02] Kendall EF, Dutra ME, McGuiness DL: Towards A Commercial Ontology Development Environment (poster), In: Horrocks I, Hendler JA (eds) First International Semantic Web Conference (ISWC 2002), Sardinia, Italy, June 2002.

[KLW95]Kifer, M., Lausen, G., Wu, J.: Logical Foundations of Object Oriented and Frame Based Languages, Journal of the Association for Computing Machinery, May 1995.

[Kn04] Holger Knublauch: Ontology-Driven Software Development in the Context of the Semantic Web: An Example Scenario with Protégé/OWL, International Workshop on the Model-Driven Semantic Web, Monterey, Canada, September 2004.

[Ko02] Kogout P., Cranefield S., Hart L., Dutra M., Baclwaski K., Kokar M. Smith J.: UML for Ontology Development, The Knowledge Engineering Review 17(1), pages 61-64, 2002.

[La01] Larman C.: Applying UML and Patterns, Prentice Hall, 2001.

[Ma02] Maedche, A: Ontology Learning for the Semantic Web, Kluwer Academic Publishers, Boston, Massachusetts, 2002.

[MD04] The Model-Driven Semantic Web Workshop (MDSW 2004), Monterey, Canada, September 2004.

[NM01] Natalya F. Noy and Deborah L. McGuinness, Ontology Development 101: A Guide to Creating Your First Ontology. Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880, March 2001.

[NP01] Niles, I., and Pease, A. 2001. Towards a Standard Upper Ontology. In Proceedings of the 2nd International Conference on Formal Ontology in Information Systems (FOIS-2001), Chris Welty and Barry Smith, eds, Ogunquit, Maine, October 17-19, 2001.

[OMG03a] Object Management Group: MDA Guide V1.0 .1, OMG Specification, June 2003.

[OMG03b] Object Management Group: UML 2.0 Infrastructure Final Adopted Specification, OMG Specification, Dec. 2003.

[OMG03c] Object Management Group: Meta Object Facility 2.0 Core Final Adopted Specification, OMG Specification, Oct. 2003.

[OMG03d] Object Management Group: Ontology Definition Metamodel Request for Proposal 2003.

[OMG04] Object Management Group: UML 2.0 Superstructure Revised Final Adopted Specification, OMG Specification, Oct. 2004.

[OMG05a] Object Management Group: OCL 2.0 Specification, OMG Specification, June 2005.

[OMG05b] Object Management Group, DSTC, IBM, Sandpiper Software Inc.: Ontology Definition Metamodel Second Revised Submission, OMG Specification, May 2005.

[OMG06] Object Management Group: MOF to RDF Mapping, Request for Proposal, June, 2006.

[ON05] Workshop on Ontology, Conceptualizations and Epistemology of Software and Systems Engineering (ONTOSE 2005), Alcal□, Spain, June 2005.

[Ri01] Richters, M.: A Precise Approach to Validating UML Models and OCL Constraints, Logos Verlag Berlin, 2001.

[Ru91] Rumbaugh et al.: Object-oriented Modeling and Design, Prentice Hall, 1991.

[SBF98] Studer, R., Benjamins, VR., Fensel, D.: Knowledge Engineering: Principles and Methods. IEEE Transactions on Data and Knowledge Engineering 25(1-2):161-197, 1998.

[SE05] International Conference on Software Engineering (SEKE 2005), Taipei, Taiwan, July 2005.

[SW05] Workshop on Semantic Web Enabled Software Engineering (SWESE 2005), Galway, Ireland, November 2005.

[Ta56] Tarski, A.: Logic, Semantics, Mathematics: Papers from 1923 to 1938, Oxford University Press, 1956.

[vH97] van Heijst, G., Schreiber, A., Wielinga, BJ.: Using explicit ontologies in KBS development. International Journal of Human-Computer Studies 45:183-292, 1997.
[VO05] International Workshop on Vocabularies, Ontologies and Rules for The Enterprise (VORTE 2005), Enschede, Netherlands, September 2005.
[Wi66] Wild, J.: Grundlagen und Probleme der betriebswirtschaftlichen Organisationslehre, 1966.
[WK03] Warmer, J., Kleppe, A.: The Object Constraint Language, Second Edition. Getting your Model ready for MDA, Addison Wesley, 2003.
[Wo05] Workshop on Meta-Modeling and Corresponding Tools (WoMM 2005), Essen, Germany, March 2005.
[W3C06] Ontology Driven Architectures and Potential Uses of the Semantic Web in Systems and Software Engineering, http://www.w3.org/2001/sw/BestPractices/SE/ODA/060211.