Representation Language-Neutral Modeling of Ontologies

A. Mädche, H.-P. Schnurr, S. Staab & R. Studer

Abstract

In this paper we present a new approach for language-neutral modeling of large-scale ontologies. The gist of our approach lies in the way we treat the majority of axioms. Instead of capturing axiom semantics in some specific representation language, we categorize axioms into different types and specify them as complex objects that refer to concepts and relations. A separate layer that is language-specific, in fact it may even vary for different inference engines working on the same language, describes how these objects are translated into a target representation. In addition to its far reaching independence with regard to specific representation languages, this approach benefits engineering since the semantics of important types of axioms may be much more elucidated in our ontology engineering tool, OntoEdit, than in comparable tools. Furthermore, our approach is principled in a way that allows for comparably easy adaptation of our tool to requirements for modeling axioms in specific domains.

The apparition of these faces in the crowd;
Petals on a wet black bough.
Ezra Pound, 1913.

1 Introduction

Ontologies have shown their usefulness in application areas such as intelligent information integration, information brokering, or knowledge-based systems. The role of ontologies is to capture domain knowledge in a generic way and provide a commonly agreed upon understanding of a domain. The common vocabulary of an ontology, defining the meaning of terms and their relations, is usually organized in a taxonomy and contains modeling primitives such as classes, relations, functions, and axioms.

A couple of representation languages (e.g., Ontolingua, KIF, CycL, F-Logic, LOOM, OML, OXL) and ontology modeling tools (cf. Section 4) have been developed that allow for the representation and engineering of ontologies. In fact, these languages and tools have matured considerably over the last few years. Nevertheless, a major drawback remains with them, namely their tight linkage to particular languages – often to particular knowledge representation and reasoning systems. Thus, the ontology engineer must oblige himself to a particular environment too early in his software development process. If need arises to switch from one ontology language and engineering tool to another one, many ontology engineering efforts are lost forever. Though there are a few approaches that translate between representation languages (e.g., OKBC [FFR 1997], ODE [BFG+ 1998]), these approaches typically fail to produce the desired results, when they are applied to *axiom specifications*. This, however, is a major pitfall, since the semantics of ontology definitions are mostly void without an adequate specification of axioms.

With our approach of ontology engineering we pursue the modeling of ontologies that are rather language-independent and that include axiom specifications. The motivation of our approach is manifold. *(i),* though one may translate between different ontology languages, translations usually incur a loss of conciseness that aggravates engineering (modeling, debugging, etc.), *(ii),* there is currently no general inference engine that takes advantage of all the features one may be interested in during modeling time and is still applicable in realistic applications, *(iii)* for the purpose of scalability there may even be the need to turn inferencing mechanisms on or off within one single application. For instance, one might use our F-Logic inference engine offering services on the conceptual level, but one may need to employ a database management system for efficient and safe querying of instances. *(iv),* with RDF, RDF schema and XML schema a new family of representation languages is developed and world wide web-applicable tools will be built that will probably become the cornerstones of ontology applications and, hence, a major field for explicit modeling methodology and tools. However, at their current stage of gestation these techniques are too immature to be relied upon for comprehensive usage. Thus, it is important that we provide sophisticated modeling techniques early on, which, however, must be rather independent from the language standards lest one risks to model in a way that is not compatible with stipulations put forth by sophisticated WWW techniques of tomorrow. *(v),* we do not know of any good user interface for modeling ontology axioms and we want to improve on that. Finally, and most important, *(vi),* the translation of an axiom from one representation language into another usually requires to abstract from its syntactic realization – just like Pound's haiku that must not be translated literally lest one looses what it is all about. Rather, it is necessary to capture the *meaning* of the axiom and to translate its meaning, i.e. the consequences it leads up to in the given ontology.

We have collected these experiences in various ontology engineering projects for purposes like information integration, information extraction, information brokering or knowledge management. Besides the problems mentioned above, we have collected further requirements that are derived from these application: An ontology engineering environment has to support different views on a given ontology. Such views support the construction and management of large ontologies. Ontologies must include axioms for supporting inference processes. Axioms need to be specified on a conceptual level which abstracts from details of the underlying specification language. The internal representation of ontologies must be kept separate from the external presentation. Thus, multi-lingual external presentations may be provided. Ontologies have to be well documented for real-life applications. Ontologies must be linked to other information repositories like (object)-relational databases or domain lexicons. Thus, the database design process or linguistic analyses may profit from the knowledge structures provided by ontologies.

In this paper we present our approach and our tool, OntoEdit, for modeling large-scale, language-neutral ontologies. In Section 2, we explain our approach for ontology modeling, which includes concepts, relations and axioms. Our tool OntoEdit for specifying language-neutral ontologies is presented in Section 3. Section 4 gives an short overview over related work. Section 5 outlines further work and concludes.

2 A Principled Approach to Ontology Modeling

The requirements that we have derived so far, have shown that any ontology engineering approach should provide comprehensive support for modeling concepts, relations, axioms, and metadata. In this section we abstract from any actual implementation, and consider what mechanisms lay the principle foundations for our ontology engineering environment (described in Section 3). Thereby, we proceed from the (straightforward) modeling of concepts and relations[1], over a new approach for language-neutral modeling of axioms. The importance of metadata appears at a different level of the ontology engineering task since metadata may be attached to the ontology itself as well as to concepts, relations and axioms. This is reflected in Section 3.

2.1 Concepts and Relations

Considering the level of concepts and relations, we provide very much the same object-oriented model that is well-known from tools like ODE [BFG+ 1998] and Protegé [GEF+ 1999] (cf. [BDS+1999] for a survey). Multiple taxonomies of concepts, i.e. `subconceptOf` relations with multiple inheritance, provide the backbone of our approach. Relations are considered as first-order entities that come with several properties of their own, e.g. names. Concepts are linked by relations to other concepts or to types, i.e. simple, system-defined, concepts. While one may conceive of extending this principle approach, e.g., to account for finer distinctions like the ones between `subconceptOf` and `properSubconceptOf` (or `:is` and `:is-primitive` in a system like LOOM), the overall approach for modeling concepts and relations remains stable over many representation languages and systems.

2.2 Axioms are Objects, too

Ontology modeling is much more critical when one pursues a language-neutral approach that includes axiom specifications. The reason is that usually some kind of first-order predicate logic is involved that deals with quantifiers, quantifier scope, negation and scope of negation. Axioms are difficult to grasp in a language-independent way, since the representation of quantifier scope and its likes is usually what a particular syntax is about. However, a closer look at the bread and butter issues of ontology modeling problems reveals that the majority of axioms that need to be formulated aim at much simpler purposes than arbitrary logic structures. Indeed, we have found that many axioms, most often the majority of axioms, belong to one of the following categories:

1. Axioms for a relational algebra[2]
    a) Reflexivity of relations

[1] We here consider the notion of *relation* to subsume the more specific notion of *attribute*.
[2] These axioms may be found in any algebra textbook. A very nice, compact survey is given at `http://www.bestweb.net/~sowa/misc/mathw.htm`

   b) Irreflexivity of relations
   c) Symmetry of relations
   d) Asymmetry of relations
   e) Antisymmetry of relations
   f) Transitivity of relations
   g) Inverse relations
2. Composition of relations
3. (Exhaustive) Partitions
4. Axioms for stating that a relation is a subrelation of another relation
5. Axioms for part-whole reasoning
6. Exceptions of superordinate axioms

We received the impression that in practical ontology engineering many, though not all, axioms deal with structures that appear again and again. Also, we have noted that these structures need to be realized differently in various representation languages. In fact, axiom specifications even turned out to differ for different inference machines working on the basically same representation language. Hence, our approach distinguishes the structures that are repeatedly found in axiom specifications from the corresponding description in a particular language. In our approach, axioms are described as complex objects that refer to concepts and relations. An additional layer then provides the transformation step that realizes axioms in a particular target representation language.

The motivation underlying this two-layer approach is that we may thus abstract from a large set of representation languages and we may get a better grasp at the *meaning* of an axiom (or a set of axioms) rather than its syntactic realization in a particular language. Our claim is that it becomes much easier to realize these implications in a particular target representation, if we categorize axioms according to what we here loosely term their meanings, i.e. the implications they enforce in their respective frameworks. In contrast, related approaches like ODE specify axioms in a first-order representation and then *translate directly* into the target language. Since literal translations usually do not work (just like a literal translation of Ezra Pound's haiku would not achieve its effects in German or Italian), the ODE approach would need to *recognize* the meaning of an axiom - sometimes even the meaning of a set of axioms, before it could translate successfully. As our approach does not require the recognition of meanings, but instead lets the engineer categorize axiom specifications, we facilitate translation into target languages *and* give more sophisticated means to organize the large set of axioms that appear in common ontologies such that ontology engineering is faciliated.

In order to elucitate our approach, we proceed through a few examples of our categorization of axiom specifications shown above. We give examples that explain the difference between two prominent ontology representation languages, viz. F-Logic ([KLW 1995], [Deck 1998]) and

Description Logics ([WoSc 1992], [MaBa 1987]) and, in addition, specify the transformation step from an object description into an executable specification with a well-defined semantics. For the latter issue, we exploit the expressiveness of F-Logic, which allows the direct definition of transformation rules. Hence, the executable specification is simply given through the object descriptions of our axioms and their corresponding transformation rules. Thus, the axiom structures we propose receive a proper semantic interpretation. We indicate with our examples that a similar step may be specified easily for other languages, like versions of description logics, and that the intuitive meanings of axiom specifications are rather well-preserved.

Axioms for a relational algebra

The axiom types that we have shown above are listed such that the "simple" axioms come first and the harder ones appear further down in the list. Axiom specifications that are referred to as "axioms for a relational algebra" rank among the simplest ones. They describe axioms with rather local effects, because their implications only affect one relation (with the exception of axioms for inverse relations that affect two relations). We here list just one example, since the principle approach remains unchanged and the meanings of the axioms is very easy to understand.

Let us consider an example for symmetry. A common denotation for the symmetry of a relation `isRelated` (such as used for "Homer Simpson is related to Bart Simpson") in first-order predicate logic boils down to:

```
FORALL X,Y isRelated(X,Y) <- isRelated(Y,X).
```

In F-Logics, this would be a valid axiom specification, too. Most often, however, modeling in F-Logics uses an object-oriented view, i.e.

```
FORALL X,Y Y[isRelated->>X] <- X[isRelated->>Y].
```

In contrast, a description logics language like LOOM provides a modeling primitive for specifying symmetry:

```
(define-relation isRelated :symmetric)
```

In our approach, we denote symmetry as a predicate that holds for particular relations:

```
Symmetric(isRelated)
```

For a particular language like F-Logic, one may then derive the implications of symmetry by a general rule and, thus, ground the meaning of the predicate `Symmetric` in a particular target language. The corresponding transformation rule (here in F-Logic) states that if for all relations R and object instances X and Y it holds that X is related to Y via R, then Y is also related to X via R.

```
FORALL R,X,Y  Y[R->>X]<- symmetric(R) and X[R->>Y].
```

This small example already shows three advantages. First, the axiom specification is rather language independent. Second, our approach for denoting symmetry is much sparser than its language specific counterparts. And, third, symmetry now constitutes a class of its own and one may easily give an editor view that lists all relations that are symmetric or that are not. In order to allow for a better overview over these different steps, we summarize them in Table 1 (with the exception of the first-order predicate logic denotation which is almost the same as its F-Logic counterpart).

`Object`	`Symmetric(isRelated).`
`F-Logic`	`FORALL X,Y X[isRelated->>Y] <- Y[isRelated->>X].`
`DL`	`(define-relation isRelated :symmetric)`
`Transf.`	`FORALL R,X,Y  Y[R->>X]<-symmetric(R) and X[R->>Y].`

Table 1: Symmetry

For easier understanding, we will reuse this table layout for all subsequent examples.

Composition of relations

The next example concerns composition of relations. For instance, if a person works in a project and the project is for a particular company then one may assert that the person also knows the company. Again different representation languages require completely different realizations of such an implication (cf. Table 2). This time, F-Logic (or predicate logics) allows for a rather concise description, while description logics languages do not support composition of relations in general, since it makes subsumption undecidable [ScSm 1989]. In Loom, e.g., one may ressort to capabilities outside of the logic kernel, viz. production rules that achieve the intended effects[3]. Both ways may be easily "summarized" by the object description

```
Composition(worksInProject,projectForCompany,knowsCompany).
```

The transformation rule works very similarly as the transformation rule for symmetry.

`Object`	`Composition(worksInProject,projectForCompany,knowsCompany).`
`F-Logic`	`FORALL X,Y,Z X[knowsCompany->>Z] <- X[worksInProject->>Y] and Y[projectForCompany->>Z].`

[3] Of course, the disadvantage of extra-logical capabilities is that their semantics is not tightly integra-ted into the system. If one of the premisses is retracted, a fact that has been asserted by a production rule is not retracted automatically, too.

`DL`	`(defproduction ComposeKnowsCompany :when (:detects (:and (worksInProject ?person ?project) (projectForCompany ?project ?company))) :perform (AssertMethod knowsCompany ?person ?company))`
`Transf.`	`FORALL R,Q,S,X,Y,Z  X[S->>Z] <- Composition(R,Q,S) and X[R->>Y] and Y[Q->>Z].`

Table 2: Composition

Axioms for stating that a relation is a subrelation of another relation

A major requirement for ontologies is the ability to structure relations into hierarchies. Natural language applications (but also other areas like medical domains) rely on very specific relations that denote background knowledge, like a hotel `hasDoubleRoom` double room. In order to bridge from a conceptually high-level linguistic expressions like "*has*", which closely resembles the general `hasPart` relation, to the more specific `hasDoubleRoom`, via other relations like `hasRoom` and `hasSubarea`, information about the relation hierarchy must be retrievable from the ontology modeling framework (cf. [HSR 1999]). An object representation of corresponding axioms, which closely resembles its description logics counterpart, provides the structural information about relations, allows for a corresponding visualization, and may be easily translated by a general axiom into a target representation language (cf. Table 3).

`Object`	`subrelationOf(hasDoubleRoom,hasRoom).`
`F-Logic`	`FORALL X,Y X[hasRoom->>Y] <- X[hasDoubleRoom->>Y].`
`DL`	`(define-relation hasDoubleRoom (:is-primitive hasRoom))`
`Trans`	`FORALL R,Q,X,Y X[Q->>Y] <- subrelationOf(R,Q) and X[R->>Y].`

Table 3: Subrelation

Axioms for part-whole reasoning

Two aspects of reasoning on part-whole relations have received special attention in the ontology engineering community. The first issue that has been debated is whether transitivity should be considered a general property of partonomies. We here only refer to recent work of [HSR 1999], [Arta1996] and [LaPa 2000] who survey this issue. Our approach provides the modeler with the flexibility to turn transitivity constraints of part-whole (sub-)relations on or off as she prefers (cf. the section on axioms for a relational algebra).

The second issue in the debate is about partonomic reasoning (cf. [Rect 1997]). Rector distinguishes between role propagation and concept specialization. As for the first, properties of parts may sometimes be propagated to become properties of their wholes. Assume that the `ColorOfCarBody` is defined as the `colorOf` the `CarBody` and the `CarBody` is defined as a `physicalPartOf` the `Car`. While the color of the car body is also the color of the car, the same does not hold for the color of the seats - in spite of the fact that there is no structural difference found in these two examples. Hence, it is necessary to specify axioms that do propagate *some* roles from parts to wholes - sometimes over several `hasPart` relations (cf. [HSR 1999]). In our approach, we directly encode that a role may be propagated over `hasPart` relations in an object representation that is straightforward and comparatively easy to understand:

```
PartonomicRolePropagationOn(ColorOfCarBody,ColorOf,CarBody).
```

In a typical application there will be dozens of definitions like this. It is easy to imagine that their language-specific counterparts are much more cumbersome to understand. In fact, PL1 style denotation needs to specify at least two axioms for each role that needs to be propagated, viz. one at the conceptual level and one at the instance level. Description logics provides no direct support, but the corresponding implications may be achieved by a - rather complicated - encoding via new concept nodes (cf. [HSR 1999]) or via user-coded functions [LaPa 2000].

`Object`	`PartonomicRolePropagationOn(ColorOfCarBody,ColorOf,CarBody)`

`F-Logic`	`Instance level: FORALL X,Y,Z,S,R X[R->>Z] <- X[R->>Y] and subrelationOf(S,part-of) and Y[S->>Z].`
	`Concept level: FORALL X,Y,Z,S,R X[R=>>Z] <- X[R=>>Y] and subrelationOf(S,part-of) and Y[S=>>Z].`

`DL`	`Artificial concept triples may encode partonomic reasoning into taxonomic reasoning (cf. [HSR 1999]), or a piece of lisp code may act to perform propagation in an approach by [LaPa 2000].`

`Trans`	`Instance level:`
	`FORALL C,D,X,Y,Z,S,R X[R->>Z] <-`
	`PartonomicRolePropagationOn(C,R,D) and X:C and Y:D and X[R->>Y] and subrelationOf(S,partOf) and Y[S->>Z].`
	`Concept level: analogous to the instance level`

Table 4a: Part-Whole Reasoning - (i) role propagation

The second aspect of partonomic reasoning that is reflected upon by Rector et al. is concept specification through partonomic reasoing. To cut it short, we only mention here that this type of axiom specification also relies on the `PartonomicRolePropagation` specified in Table 4a and show a corresponding F-Logic representation in Table 4b. This type of reasoning is provided for by mechanisms analogous to the ones in Table 4a.

`F-Logic`	`FORALL X,Y,Z,S,R,W X::W <- X[R=>>Y] and subrelationOf(S,partOf) and Y[S=>>Z] and W[R=>>Z].`

Table 4b: Part-Whole Reasoning - (ii) concept specification

Exceptions of superordinate axioms

To give a further outlook on modeling "hard problems", we have investigated how exceptions may be captured in our approach. By "exceptions" we refer to axiom combinations that, e.g., allow to conclude that `hotelGuests` must pay in advance, unless they are `regularHotelGuests`. We have found that it is indeed very elegant and helpful to model axioms and refine them with more specific exceptions. The semantics of the latter may then affect the semantics of the superordinate axioms. While the general structure that holds between premises and implications of general and refining axioms has been easy to cover in our approach, we have encountered a problem in formulating the premises and implications themselves in a language-independent manner. The reason is that one may use rather arbitrary logical formulas to specify premises or implications. Here, we reach a point where our approach still helps to organize axiom specifications, e.g. through views like "show me all exceptions of the axiom about payments", but where it must (partially) fall back onto a more general, but less concise way for modeling general axioms.

General axioms

The approach we have described so far is not suited to cover every single axiom specification one may think of. Hence, we still must allow for axioms that are specified in a particular representation language like first-order predicate logic and we must try to translate these axioms, possibly with human help, into their target representation language, e.g., description logics. For such general axiom specifications, we do not gain anything compared to related approaches, like ODE [BFG+ 1998], however, we do not fare any worse either.

Naturally, we have restricted our attention to axiom types that we have found of particular interest in our applications. We do not claim that we cover all axiom types that may be of interest. Rather, our goal is the provision of a general methodology that may be employed for the definition of new axiom types for new domains and applications and for the specification of transformation rules that we have not thought of yet.

3 OntoEdit - A Tool for Ontology Engineering

OntoEdit is an ontology engineering environment[4] based on the principles described in the last section. Modeling in OntoEdit along theses principles is supported by an ontology engineering process model depicted in Figure 1. In our approach we mainly distinguish between the meta descriptions, concepts, relations and axioms. First of all, if a new ontology is created inital meta descriptions have to be defined. This includes, e.g., the name of the ontology, the information about the author(s) and some inital documentation. Meta descriptions about the ontology are extended during the entire modeling process (e.g. documenting incoming requirements).

The modeling process proper is initiated with the creation of new concepts (such as `Person`, `Organization`, ...) to the ontology. The concepts are positioned into the is-a hierarchy, multiple inheritance being allowed. Subsequently relations are defined. Both, concepts and relations can be

[4] The OntoEdit environment is fully implemented in Java (JDK 1.2) using the JFC Swing packages to create the GUI (cf. Figure 2).

further described using metadata descriptions, such as documentation, multlingual interpretation or mapping information. On the basis of existing concepts and relations, axioms are defined.

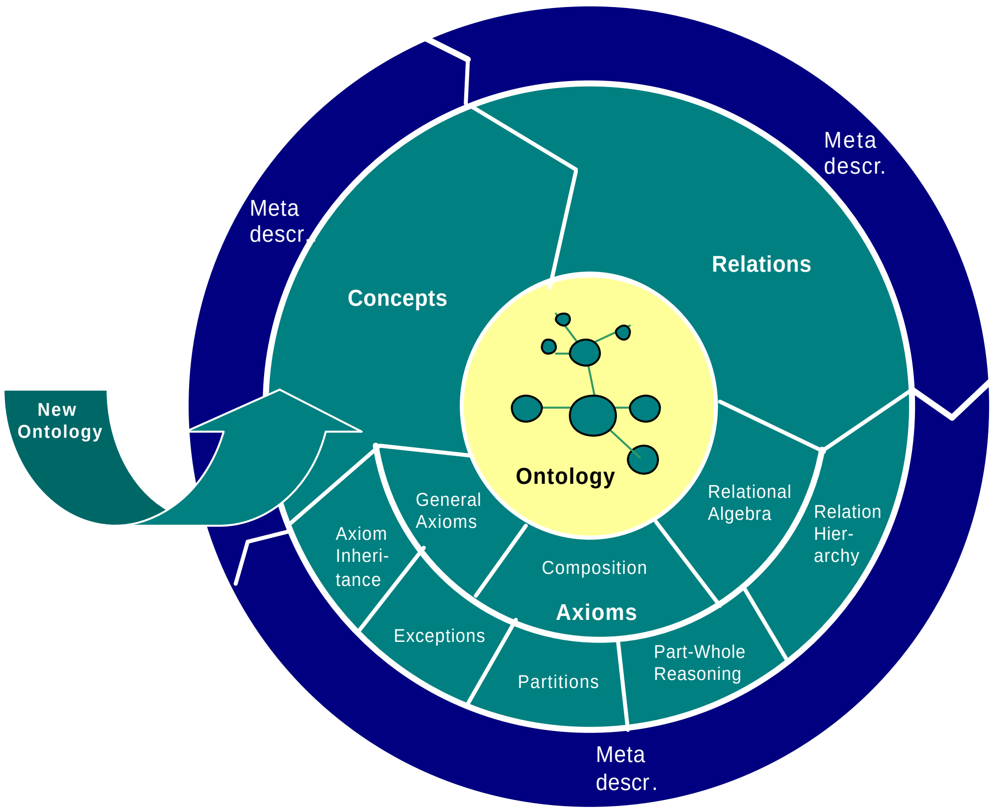
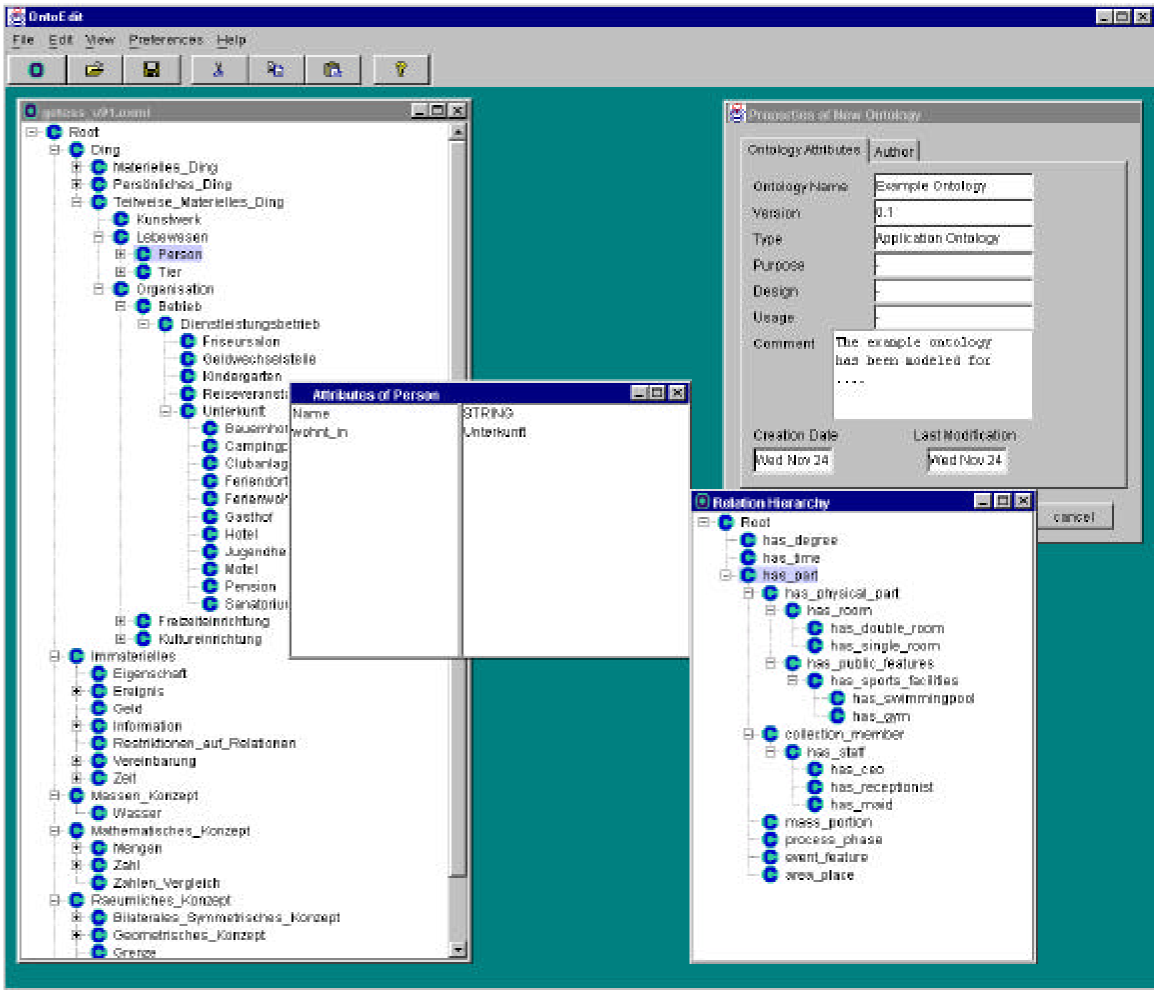


Figure 1: Ontology Modeling process using OntoEdit

The axiom classification introduced in the last section is used to facilitate the modeling process. The six different types of axioms introduced in the last section and depicted in Figure 1 are offered to the user in separate modeling views. Our general strategy here is to model the "easy", local axioms first and the harder axioms a later stage. Thereby, OntoEdit offers various views onto axioms classified into the different types. Axiom types which belong to "Relational Algebra" and "Composition" are used for all types of applications. Depending on the domain for which the ontology is built, several axiom types may play a major or minor role (for example, part-whole reasoning plays a important role in medical applications (cf. [HSR 1999])). Axiom modeling may necessitate a revision of the already modeled concepts and relations. For instance, new sub-concepts of existing concepts may have to be introduced, relations may have to become concepts.



Name	STRING
wohnt_in	Unterkunft

Ontology Name	Example Ontology
Version	0.1
Type	Application Ontology
Purpose	
Design	
Usage	
Comment	The example ontology has been modeled for ....
Creation Date	Wed Nov 24
Last Modification	Wed Nov 24

Figure 2: OntoEdit – Ontology Engineering Environment

View on general metadata descriptions

Metadata for ontology engineering has to be subdivided into metadata for the entire ontology and metadata for the knowledge structures (concepts, relations and axioms) contained in the ontology. If we talk about metadata for the entire ontology, this includes identifying and descriptive characteristics of an ontology. This schema is taken from [AGT+ 1999], who developed a framework for finding and retrieving ontologies. Identifying attributes of an ontology enclose the name of the ontology, the creation date, last modification date, the version and information about the authors. Descriptive attributes store information and statistics about the ontology. The general information about the ontology is decomposed into characteristics like type of the ontology, purpose, design techniques and application goal (cf. right part of Figure 2). Statistics about the ontology are computed automatically by OntoEdit (number of concepts, number of relations, number of axioms, the highest and the average depth level).

Views on concepts and relations

OntoEdit uses the object-oriented modeling paradigm to structure new concepts in a general way. The concept browser is depicted in the left part of Figure 2. Using this "concept browser" developers can edit concept hierarchies via drag & drop. Introducing a new concept means positioning it in the taxonomy. Concepts are defined using abstract, natural language-independent identifiers. As already mentioned, natural language-dependent external representations of concepts (e.g. for visualizing the ontology) are realized via metadata descriptions for the identifiers.

Having defined a core taxonomy, relations of and between concepts are modeled in an extra view. This view depends on the concept selected in the taxonomy and is updated each time another concept is selected. In the middle part of Figure 2 the relation view of the concept `Person` is depicted. A person in our small example ontology is characterized by the attributes `Name` and `wohnt_in`. Similar to concepts, relations are defined using abstract, language-independent identifiers. The external representation and documentation of attributes is attached through metadata descriptions.

Views on axioms

As already explained in Section 2.2 modeling axioms in a language-neutral way is crucial. The axiom view of OntoEdit consists of multiple views that correspond to the classification of axioms.

In Figure 2 we give an example for an axiom view, that realizes the modeling of a relation hierarchy. Modeling subrelation relationships into the hierarchy using OntoEdit is a very simple task, because the relation hierarchy view is very similar to the taxonomy view on concepts. Introducing a new subrelation axiom boils down to the selection of one of the existing relations and its addition as a subrelation.

Translating for inferencing

As described in section 2, a transformation step from the object description produced by OntoEdit into an executable specification with a well-defined semantics has to be performed. This transformation step has been illustrated in the examples in Section 2 using F-Logic. It has to be mentioned that this transformation is not only dependent on the representation language, but also on the capabilitities of the underlying inference engine. Our examples have been executed on a running inference engine for F-Logic developed in the context of the OntoBroker project (cf. [Deck 1998]).

A transformation module for Description Logics inference engines (cf. [WoSc 1992], [Rect 1995], [BBM+ 1989]) may also be developed based on the object descriptions which have been created using OntoEdit.

4 Related Work

Like ODE (Ontology Design Environment) [BFG+ 1998], we choose a modelling approach interacting with the user at the conceptual level. This approach is in contrast with tools like the Ontolingua system and OntoSaurus (cf. [FFR 1997]), that interact with the user at the symbol level (e.g. the representation language Ontolingua). However, axioms are specified in ODE in a first-order representation and then translated directly into the target language. Since literal translations usually do not work, the ODE approach would need to recognize the meaning of an axiom - sometimes even the meaning of a set of axioms, before it could successfully translate.

As already mentioned, the main difference in our approach to existing approaches such as ODE is that the most important axioms are considered as objects and defined according to the presented axiom classification. As our approach does not require the recognition of meaning, but instead lets the engineer categorize axiom specification, we facilitate translation into target languages *and* give more sophisticated means to organize the large set of axioms that appear in common ontologies such that ontology engineering is faciliated.

5 Conclusion

We have presented a new approach towards representation language-neutral modeling of ontologies in the engineering environment OntoEdit. First, our approach incorporates a new concept for modeling axioms that is easy to read and easy to organize in a number of different views. Second, the scheme is generally applicable with regard to its language scope - in fact it is language-neutral to a large extent as we have illustrated by various examples for F-Logic and Description Logics. Third, our approach is even adaptable to particular inferencing mechanisms. One may easily implement means for tuning the computation of particular axioms, e.g., efficient incremental computation of transitive closures. Inferencing becomes scalable as particular *axiom types* may be turned on or off depending on what exactly is required at a particular point in time. Fourth, our scheme for axiom representation may be easily incorporated into common user interfaces for the visualization of tree structures, lists, or records. Finally, and most important, our approach preserves the *meaning* of axiom specifications to a large extent over a wide range of representation languages and systems.

For the future we will have to investigate further what type of axioms other than those mentioned here exist and how one may best get a hold on their structures. As for the implementation side, we plan to integrate our ontology engineering environment, OntoEdit, into a larger client-server framework, OntoServer, that offers ontology services through a range of various client applications. Important features that we consider now are, e.g., versioning and merging of ontologies and the interaction of ontologies with new web standards (XML, RDF, RDF Schema).

6 References

[AGT+ 1999] J. C. Arpirez Vega, A. Gomez-Perez, A. L. Tello, H. Pinto. How to Find suitable Ontologies Using an Ontology-based WWW Broker. LNCS 1607. 1999.

[Arta 1996] Artale et al.: Part-Whole Relations in Object-Centered Systems. Part-Whole Relations in Object-Centered Systems: An Overview (special issue on Modeling Parts and Wholes). Data and Knowledge Engineering, Vol. 20, No. 3. 1996, pp. 347-383.

[Bala 1995] M. Balaban.The F-Logic Approach for Description Languages. Annals of Mathematics and Artificial Intelligence, 15, pp. 19-60, 1995.

[BBM+ 1989] A. Borgida, R. J. Brachman, D. L. McGuinness, L. A. Resnick. CLASSIC: A Structural Data Model for Objects. In: Proceedings of the 1989 ACM SIGMOD International Conference on Management of Data, 1989.

[BDS+ 1999] R. Benjamins, A. J. Duineveld, R. Stoter, M. R. Weiden, B. Kenepa Wondertools? A comparative study of ontological engi-neering tools. Proceedings of the 12th International Workshop on Knowledge Acquisition, Modeling and Mangement (KAW'99), Banff, Canada, October 1999.

[BFD+ 1999] R. Benjamins, D. Fensel, S. Decker, A. Gomez Perez: KA2. Building Ontologies for the Internet: A Midterm Report. In: International Journal of Human Computer Studies, 51(3), 1999. pp. 687-712.

[BFG+ 1998] M. Blázquez, M. Fernández, J. M. García-Pinar, A. Gómez-Pérez. Building Ontologies at the Knowledge Level using the Ontology Design Environment. Proceedings of the 11th International Workshop on Knowledge Acquisition, Modeling and Mangement (KAW'98), Banff, Canada, October 1998.

[CFF+1998] V. K. Chaudhri, A. Farquhar, R. Fikes, P. D. Karp, J. P. Rice. OKBC: A Programmatic Foundation for Knowledge Base Interoperability. Proceedings of the 15th National Conference on Artificial Intelligence (AAAI-98) and of the 10th Conference on Innovative Applications of Artificial Intelligence (IAAI-98), pp. 600-607. 1998.

[DBS+1998] S. Decker, D. Brickley, J. Saarela, J. Angele: A Query and Inference Service for RDF. QL'98 - The Query Languages Workshop. 1998.

[Deck 1998] S. Decker: On Domain-Specific Declarative Knowledge Representation and Database Languages In: Proceedings of the 5th Knowledge Representation meets Databases Workshop (KRDB98). 1998.

[DEF+ 1999] S. Decker, M. Erdmann, D. Fensel, and R.Studer: Ontobroker: Ontology Based Access to Distributed and Semi-Structured Information. In: R. Meersman et al. (eds.), Database Semantics: Semantic Issues in Multimedia Systems, Proceedings TC2/WG 2.6 8th Working Conference on Database Semantics (DS-8), Rotorua, New Zealand, Kluwer Academic Publishers, Boston, 1999.

[ErSt 1999] M. Erdmann and R. Studer: Ontologies as Conceptual Models for XML Documents. Proceedings of the 12th International Workshop on Knowledge Acquisition, Modeling and Mangement (KAW'99), Banff, Canada, October 1999.

[FFR 1997] R. Fikes, A. Farquhar, J. Rice: Tools for Assembling Modular Ontologies in Ontolingua. Proceedings of the Fourteenth National Conference on Artificial Intelligence and Ninth Innovative Applications of Artificial Intelligence Conference, AAAI 97, IAAI 97, July 27-31, 1997, Providence, Rhode Island. AAAI Press / The MIT Press, 1997.

[GEF+ 1999] W. E. Grosso, H. Eriksson, R. W. Fergerson, J. H. Gennari, S. W. Tu, M. M. Musen. Knowledge Modeling at the Millennium – The Design and Evolution of Protege-2000. Proceedings of the 12th International Workshop on Knowledge Acquisition, Modeling and Mangement (KAW'99), Banff, Canada, October 1999.

[HSR 1999] U. Hahn, S. Schulz, M. Romacker. Part-Whole Reasoning: A Case Study in Medical Ontology Engineering. IEEE Intelligent Systems, Vol. 14, No. 5, 1999, pp. 59-67.

[KLW 1995] M. Kifer, G. Lausen, & J. Wu. Logical Foundations of Object-Oriented and Frame-Based Languages. Journal of the ACM, 42, 1995.

[LaPa 2000] P. Lambrix, L. Padgham. Conceptual Modeling in a Document Management Environment using Part-of Reasoning in Description Logics. Data & Knowledge Engineering, Vol. 32, No. 1, 2000, pp. 51-86.

[MaBa 1987] R. MacGregor, R. Bates. The LOOM Knowledge Representation Language Proceedings of the Knowledge-Based Systems Workshop, 1987.

[RDF 1999] RDF Schema Specification. http://www.w3.org/TR/PR-rdf-schema/, 1999.

[Rect 1997] L. Rector et al. The Grail Concept Modelling Language for Medical Terminology. Artificial Intelligence Medicine, Vol. 9, No. 2, 1997, pp. 139-171.

[SBD+ 1999] S. Staab, C. Braun, A. Düsterhöft, A. Heuer, M. Klettke, S. Melzig, G. Neumann, B. Prager, J. Pretzel, H.-P. Schnurr, R. Studer, H. Uszkoreit, B. Wrenger: GETESS - Searching the Web Exploiting German Texts. In: CIA'99 - Proceedings of the 3rd Workshop on Cooperative Information Agents, 1999.

[ScSm 1989] M. Schmidt-Schauß and Gerd Smolka. Subsumption in KL-ONE is undecidable. KR-89 - Proceedings of the Conference on Knowledge Representation and Reasoning, Morgan Kaufmann, 1989, pp. 421-43.

[WCH 1987] M. Winston, R. Chaffin, and D.J. Herrmann. A Taxonomy of Part-Whole Relationships. Cognitive Science, Vol. 11, 1987, pp. 17-444.

[WoSc 1992] W.A. Woods and J.G. Schmolze. The KL-One Family. Computers and Mathematics with Applications, Vol. 23, No. 2 and 5, 1992, pp. 133-177.