

Marrying UML and Ontologies by Model Weaving

Fernando Silva Parreiras* and Steffen Staab

Institute for Computer Science, University of Koblenz-Landau
Universitaetsstrasse 1, Koblenz 56070, Germany
{parreiras,staab}@uni-koblenz.de

Abstract. It presents a weaving approach combining ontologies, rules and UML diagrams. Recent developments that supports this proposal are Model Driven Architecture (MDA), popularization of ontologies and the use of ontologies as domain object model. Code generation, model validation and reasoning support are some of the motivations. It describes the approach, including architecture, the weaving metamodel and illustrates the role picture with an example.

1 Introduction

The semantic web has changed the way of developing software and consequently the way of developing model driven software. Ontologies and rules are imported from internet and composed with models where classes, attributes, and operations, etc. are defined. There are basically three pillars that support this environment: Model Driven Architecture, the popularization of ontologies as more visual notations become available and the use of ontologies as Domain Object Model.

Model Driven Architecture (MDA) preaches that models should play a direct role in software production, being manipulated and transformed by machine [1]. The improvements brought with UML 2.0 specification [2] supports this path. These forces have also motivated the appearance of visual languages based on UML for ontology development [3] [4], including mappings to code generation [5]. At the same time, ontologies started being used as Domain Object Model [6], from which code are generated having the ontologies themselves as data repository [7].

From these trends, an environment where (1) ontologies can be designed using different visual notations and then formalized using an ontology language, (2) the development is oriented to models written in UML or any Domain Specific Language and, (3) enjoying the power of reasoning, ontologies are used as Domain Object Model, could bring advantages in both design time and runtime. In design time: crosscutting views of models improving the comprehensibility, model consistency checking, classification, reuse of elements of different models

* Financially supported by CAPES Brazil.

or ontologies in different computation models (declarative and imperative), composing new elements. In runtime: instance classification, rules verification and executable models.

In this scenario, the statement of this paper is the question whether an aspect oriented approach has the required features to succeed in realizing the advantages above. The objectives are present the separation of concerns as a candidate to achieve the aims listed above, describe the proposed metamodel and identify related works and synergy areas. The main contributions of this paper is the introduction of weaving for bidirectional mapping between software models and ontologies, and the proposal of a weaving metamodel.

This paper is organized as follows. Section 2 Presents the motivation, giving an example and justifying the problem. Section 3 presents the weaving approach listing requirements of architecture and language and the results of the application of both. After that, we present the related works on section 4 and concludes pointing future works on section 5.

2 Motivation Scenario

The growing availability of ontologies encourages the developers to find solutions that stimulate the use of ontologies in software systems. When already available, RDF and OWL ontologies [8] can be shared among applications, improving reuse and interoperability. These applications also use ontologies as object model, enabling the developer to check consistency quality and to do reasoning.

To illustrate our proposal, let's take as example an e-commerce application for EU customers. It is desirable first to check the existence of related ontologies. We here consider two hypothetical ontologies already pre-built: an e-commerce and an location ontology (Fig. 1). The e-commerce ontology formalize what a product is, its different charge kinds as well as the origin country. The location ontology formalize the addressing units and their relations. The yellow boxes represent OWL Classes in a UML-like graphical notation. Here the UML notation is merely used as an example, as the intention is not to capture the full semantics of OWL.

To be able to specify operations, arguments and actions, the development team may choose to design part of the application using UML. In this short version, the central class is PurchaseOrder, which has items that are related to products, a billing address and a shipping address. The class PurchaseOrder has also an operation called total, which adds the items. Naturally a customer can make many orders.

Based on trade agreements, some rules can be also formalized. One simple rule is that which states that a product originating from a country within Europa should be considered as duty free:

Rule 1 *Class (?pr, Product) & Class(?co, Country) & comesFrom(?pr, ?co) & partOfTradeZone(?co, 'EU') -> DutyFree(?pr)*

Starting from different sources of artifacts (2 ontologies, 1 UML model and 1 rule), it would be aimed to keep the compatibility with these artifacts. It would

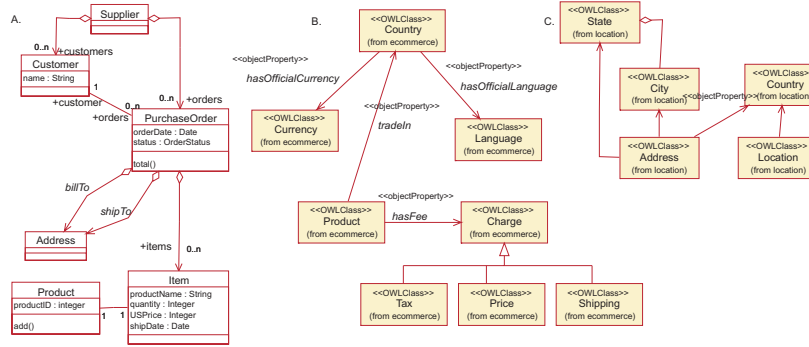


Fig. 1. A. e-commerce ontology B. location ontology C. UML model

an advantage in the future, keeping the interoperability among application. In this way, the ability to handle different view, or the separation of concerns is a requirement.

Some overlaps among elements from different artifacts are common and can be profitable. For example, we have the Product class present in two artifacts: the e-commerce ontology and the purchase order uml class diagram. the data property **name** is on the ontology whereas the attribute **productID** and the operation **add()** are on the class diagram.

Besides, due the weak support of UML tools for consistency checking, the ability to validate models, verifying for example the cardinality and the associations, act as a plus, increasing the reliability. This can be using mapping between UML and OWL and with a help from a reasoner.

From the runtime point of view, the rule can be uncoupled from the source code, designed using UML-like notations, and mapped into a knowledge representation formalism. In our example, on calculating the total sum of the items in a purchase order, it would be suitable to let the reasoner classify the instances instead of writing conditions to verify the charges of the items. OWL and extensions like SWRL can be used at runtime with a reasoner, driving the control logic of the program [9].

These examples justify enhancements proposals and we believe in a weaving approach of artifacts, explained in the next section, should succeed in the above issues.

3 The Model Weaving Approach

3.1 Aspect Language and Weaving Models

To define the compositional paradigm and the characteristics of the weaving approach, we based on [10] to consider the following requirements as essential ones:

- **Separation.** none of the concerns should have a dominant character, i.e. each concern is designed without reference to a base, and each is completely separated from the others;
- **Integration.** reflects The need to integrate separately developed software;
- **Reuse.** The concerns should be able to be combined and augmented, composing a new concern;
- Model Driven. In MDA the models are treated as first class citizens, being amenable to manipulation and transformation by machine. Following this rationale, the weaving between UML models and ontologies must be the form of a model as well;
- Extensibility. The language should be able to be extended according to a metalanguage, to accommodate future compositions techniques.

In response to these requirements and base on the evaluation on [10] we decided for a symmetrically organized paradigms for software composition where: concerns are composable elements that consist of declaratively complete class hierarchies; join points that are a points in artifacts, such as classes or attributes, at which composition can occur and; composition relationships specify the details of how composition is to occur at join points.

As an instance of such symmetrically organized paradigm and to accomplish the model driven and extensibility requirements we adopt the core weaving metamodel introduced in BDA05 and extend it based on the concepts of multi-dimensional separation of concerns [11] [12], and the definition of model composition operators from [13]. The Fig. 2 depicts just the fundamental classes, illustrating only one integration relationship: merge.

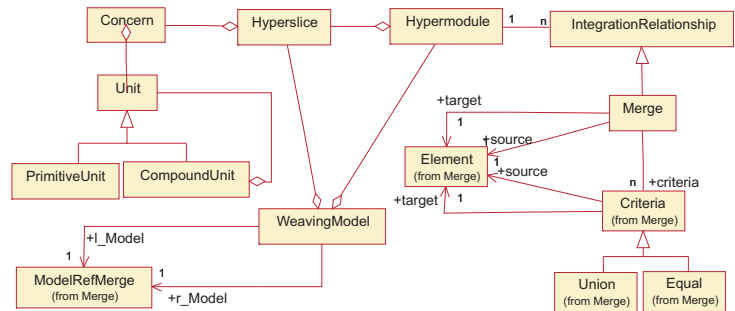


Fig. 2. Metamodel extension for concern weaving and composition

The WeavingModel class is the root element, which weaves two models: lModel and rModel. The weaving model has hyperslices and hypermodules. A hyperslice is a set of units of one or more concerns that are declaratively complete. A concern encompasses all units in some body of software. A unit is a syntactic construct in such a language. The integration between the overlaps among the hyperslices to be woven are defined by integration relationships, compound the

hypermodule. Merge is on kind of integration relationship between two elements, one of each model. this kind of relationship has some criteria which the merge should follow.

The example depicted on Fig. 3 is an instance of the metamodel, with a snapshot of the packages association. We represented here the SWRL rule and the OWL ontology using UML notations based on [14] and [15] respectively merely for the sake of illustration. Note that though this mappings can be useful they are not absolute necessary.

According to the example, we have one weaving model (po) with two hyperslices, each of them with one different concern: A product concern from the purchaseorder UML model and a Product concern from the ontology ecommerce. Each of these concerns have their respective Product Class. The set of both hyperslice form what we call hypermodule, because they overlap. The integration is handled by the merge relationship, which the criteria is an equal on the name field of each class.

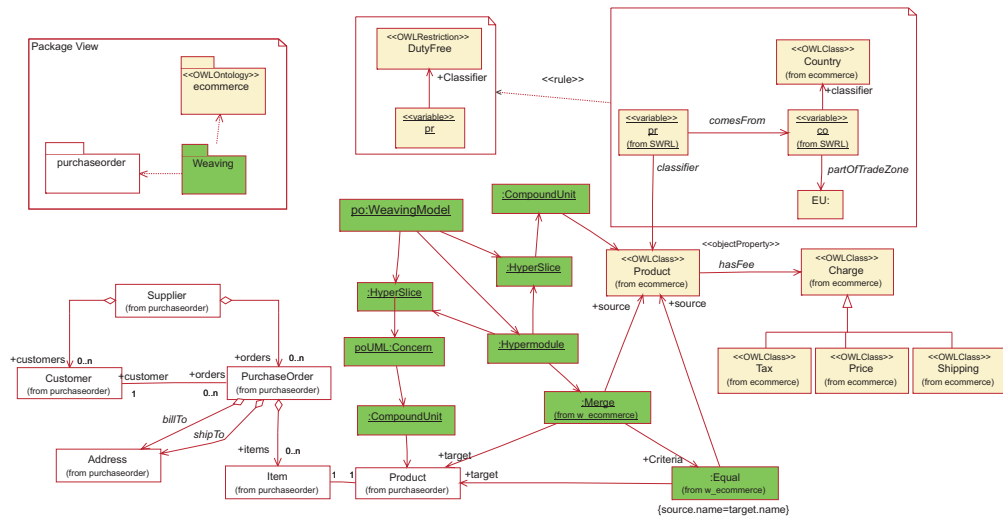


Fig. 3. Example with an instance of the weaving metamodel linking OWL and SRWL models

Realizing the benefits of this approach, the weaving model can be used to enhance reuse and comprehensibility. OWL classes, properties and rules are reused and composed with UML Classes, attributes and operations. Besides, graphical tools can use the weaving model, together with the UML models, rule and ontologies as input to generate software views. Hence, the concern can be graphically visualized allowing the traceability of the elements. The resulting weaving model can be also used for mappings between ontologies, rules or uml models, which is explained on the next section.

3.2 Woven Model

After designing the weaving model based on the weaving metamodel, both explained on the previous section, mappings can be written into both directions through a transformation script in languages like Atlas Transformation Language [16] or Meta Object Facility (MOF) 2.0 Query/View/Transformation [17].

The Fig. 4 shows a woven model, product of mappings from OWL and SWRL into UML with respective profiles. We choose to bring OWL and SWRL under the OMG Meta Object Facility umbrella, because there are already metamodels and mapping proposals written having MOF [18] as metamodelanguage. [19], [20] and [14] propose MOF metamodels for OWL with visual notation as well whereas [15] proposes a MOF metamodel for SWRL and a visual notation.

Now the woven class Product wears a OWLClass Stereotype and has the sum of properties, attributes and relations as well as the operations from UML class diagram. Tagging the classes with stereotypes can be useful for code generation. The generator identifies how to handle stereotyped classes, whether they will be persisted on a RDF Database, how to invoke a reasoner, etc.

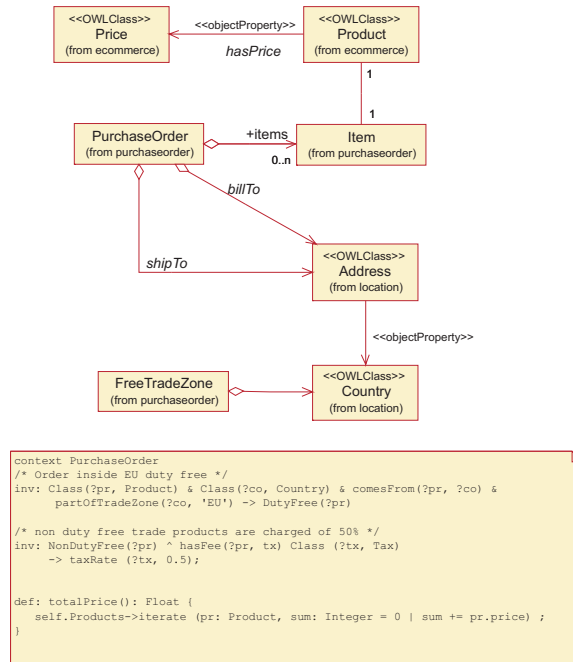


Fig. 4. Woven model

The box represented as a note tries to illustrate the impact of the declaration of two rules on the specification of the operation `totalPrice()` of the class

PurchaseOrder. Taking into consideration the two rules, it is only necessary to write one statement in the operation: to sum the value of the products.

The woven model plays also an important role into our objectives. Due the bidirectional mapping, ontologies designed using visual notations can be seen as woven models and using a transformation language and a transformation script we can end up with an ontology formalized in OWL, allowing visual design of ontologies using UML.

After consistency checking according to given directives using a reasoner, the woven model can be consumed for code generation, using a model driven engineering approach [21]. Having enough machinery, one can even think about executable models [22], at least for rapid development of prototypes.

The woven rules and restrictions, with respective stereotypes, can be used for generating code, taking as architecture an approach where reasoners can be invoked to do instance classification and rules verification [23].

3.3 Architecture

To support model weaving approach, we identify some requirements based on [21] that should be present in the architecture of the solution. They are the following:

- **Rich Client Interface.** A graphical user interface should allow the users to design the models using visual notations and the richness of an GUI environment;
- **Models Validation.** based on directives, these components should be able to interface with a reasoner to provide model consistency checking;
- **Model Translation.** The storage format should not be a problem when working with different kinds of language;
- **Reasoning Able.** Support of a OWL reasoner;
- **Code Generation.** Ability to execute transformations between Platform Independent Models (PIM) and Platform Specific Models (PSM);
- **Storage.** repositories for ontologies and models.

In response to these requirements we propose the architecture depicted on Fig. 5. At the upper part, the front end one, we have the Rich Client Platform which covers the three editors (ontology and rule, UML and OCL and Model Weaving) and part of the Model Driven Composer. This one has two components, the inference engine and the model broker that interface with the back end part, the storage.

The Concern Validator is responsible for verifying whether the concerns are encapsulated correctly, forming the hyperslices. The Model Checking is in charge of, based on directives previously formalized, verifying the consistency of UML models and weaving models as well. The Model Transformer is able to generate new models based on input models. It is used by the Code Generator to produce successive models towards code.

These elements should be necessary to implement the idea presented on the previous section.

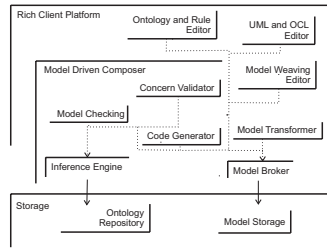


Fig. 5. Architecture components

4 Related Works

4.1 Model weaving approaches

[24] presents the language architecture and infrastructure of an Aspect Oriented Language for model weaving the Model Weaving Description Language (MWDL), but doesn't offers details enough for any kind of adoption.

[12] introduces the integration of the Hyperspace approach to UML for the feature driven decomposition and composition of similar software products. This approach is based on HyperJ [25]. Its scope is UML diagram and since we are working in a higher MOF level, from where we can work with OWL ontologies and SWRL rules, it is not extensible enough to support our proposal.

We have adopted the generic model weaver from [26] The Project Atlas has been applying model weaving to different areas, but not to UML and ontologies weaving. The single work into this directions seems to be [13], that implementes some merging operators. In our vision, the approach of this papers doesn't should be seen like just another application, because some particular improvements should take place. So, we prefer seeing it as a new branch of model weaving, that starts being developed.

4.2 MOF Metamodels for OWL and Rules

UML was proposed as an ontology modeling language in [4] but with the development of MDA and UML, MOF Metamodels and UML Profile for OWL Ontologies [27] [20] [14] started appearing, some of them with new adornments. Our approach should be able to give the developer the freedom to choose which profile or metamodel to use. To do this, just the transformation scripts should be changed.

MOF Metamodels and UML Profile for rule-extended OWL DL ontologies [15] [28] are also already available. citeRoW2006:Wagner presents an interchange format for rules integrating the Rule Markup Language (RuleML), the Semantic Web Rule Language (SWRL) and the Object Constraint Language (OCL).

From this interchange format(R2ML) [29] executes bidirectional transformations between OWL/SWRL and R2ML and between UML/OCL and R2ML.

these transformations could be useful for our approach, but under the weaving point of view.

4.3 Ontology as Object Model

The concept of generating Java code from an ontology can be found in OntoJava [30], a cross compiler that translates ontologies written with Protg and rules written in RuleML into a unified Java object database and rule engine, and in [31].

Implementations that use Ontology as Object model can be found in Protege UML plugin [7] and in RdfReactor [32]. Both allow programmatic data access to ontologies, and present software architectures driven by ontologies.

We believe all this approaches require deep analysis before adoption and we will come across them, when we start approaching the runtime environment.

5 Conclusion

This article presents an aspect oriented modeling approach, asymmetric and non-invasive, including a metamodel to create weaving models and illustrates how it can be used to bring the power of modeling languages and ontologies together.

Using Aspect Oriented Modeling techniques, we can realize some of the benefits of the imminent improvements in software development involving ontologies. The artifacts resulting of the weaving process can be used in design time and in runtime to: consistency checking, code generation, classification (subsumption testing), instance classification and rules verification.

Providing a weaving metamodel and a transformation script we enable the bidirectional mapping between visual notations and ontology languages. Bidirectional model transformation enables checking and validation in design time and instances classification and rules verification in runtime. Composing models we streamline comprehensibility preserving imported ontologies or rules. Importing and merging ontologies, rules and models already available we increase the reusability of artifacts as composing new ones.

This weaving process is part of an ambitious project which aims at providing an environment for requirements elicitation and prototypes generation relying on model driven development.

5.1 future works

For the future, we are planning target implementation issues, to have a more concrete idea of the shortcomings of the proposal. Some tools are already available but the bridges must be built.

Once we have a running environment, testing the approach against real world models will lead us to have material to realize quantitative proofs instead empirical findings.

This paper considers just UML class diagrams. We would extend the Idea not just to other behavioral UML diagrams, but to Domain Specific Languages which plays an increasing role on model driven strategies.

References

1. Frankel, D.: Model Driven Architecture: Applying MDA to Enterprise Computing. John Wiley & Sons, Inc., New York, NY, USA (2002)
2. OMG: Unified Modeling Language: Superstructure. Object Modeling Group. (2000)
3. Kogut, P., Cranefield, S., Hart, L., Dutra, M., Baclawski, K., Kokar, M., Smith, J.: Uml for ontology development. *Knowl. Eng. Rev.* **17**(1) (2002) 61–64
4. Cranefield, S., Purvis, M.K.: Uml as an ontology modelling language. In: *Intelligent Information Integration*. Volume 23 of CEUR Workshop Proceedings. (1999)
5. Cranefield, S.: Uml and the semantic web. In Cruz, I.F., Decker, S., Euzenat, J., McGuinness, D.L., eds.: *The Emerging Semantic Web*. Volume 75 of *Frontiers in Artificial Intelligence and Applications*., IOS press (2001)
6. Tetlow, P., Pan, J.Z., Oberle, D., Wallace, E., Uschold, M., Kendall, E.: *Ontology driven architectures and potential uses of the semantic web in systems and software engineering*. W3C Working Draft Working Group Note 2006/02/11, W3C (2006)
7. Knublauch, H.: *Ontology-driven software development in the context of the semantic web: An example scenario with protege/owl*. In Frankel, D.S., Kendall, E.F., McGuinness, D.L., eds.: *1st International Workshop on the Model-Driven Semantic Web (MDSW2004)*. (2004)
8. McGuinness, D.L., van Harmelen, F.: *Owl web ontology language overview*. Technical report (2004)
9. Knublauch, H.: *Ramblings on Agile Methodologies and Ontology-Driven Software Development*. In: *Workshop on Semantic Web Enabled Software Engineering (SWESE)*, Galway, Ireland (2005)
10. Harrison, W.H., Ossher, H.L., Tarr, P.L.: *Asymmetrically vs. symmetrically organized paradigms for software composition*. Research Report RC22685 (W0212-147), IBM (2002)
11. Ossher, H., Tarr, P.: *Multi-dimensional separation of concerns in hyperspace*. Research Report RC21452 (96717), IBM (1999)
12. Philippow, I., Riebisch, M., Boellert, K.: *The hyper/uml approach for feature based software design*. In Aldawud, O., Kand, M., Booch, G., Harrison, B., Stein, D., eds.: *Third International Workshop on Aspect-oriented modeling (AOM 2003)*. (2003)
13. Kurtev, I., Fabro, M.D.D.: *A dsl for definition of model composition operators*. In *2nd Workshop on Models and Aspects - Handling Crosscutting Concerns in MDSD, ECOOP 2006*, Nantes, France (2006)
14. OMG: *Ontology Definition Metamodel Third Revised Submission to OMG/ RFP ad/2003-03-40*. Object Modeling Group. (2005)
15. Brockmans, S., Haase, P., Hitzler, P., Studer, R.: *A metamodel and uml profile for rule-extended owl dl ontologies*. In Sure, Y., Domingue, J., eds.: *The Semantic Web: Research and Applications*. Volume 4011 of LNCS., Budva, Montenegro, Springer (2006) 303–316
16. Jouault, F., Kurtev, I.: *Transforming models with atl*. In: *Proceedings of the Model Transformations in Practice Workshop at MoDELS 2005*, Montego Bay, Jamaica (2005)

17. OMG: MOF QVT Final Adopted Specification. Object Modeling Group. (2005)
18. OMG: Meta Object Facility (MOF) Specification. Object Modeling Group. (2000)
19. Brockmans, S., Haase, P., Hitzler, P., Studer, R.: A metamodel and UML profile for rule-extended OWL DL ontologies. In Sure, Y., Domingue, J., eds.: ESWC. Volume 4011 of Lecture Notes in Computer Science., Springer (2006) 303–316
20. Djuric, D., Gasevic, D., Devedzic, V., Damjanovic, V.: A uml profile for owl ontologies. In: MDFAFA. (2004) 204–219
21. Kent, S.: Model Driven Engineering. In: Proceedings of IFM 2002. LNCS 2335, Springer-Verlag (2002) 286–298
22. Mellor, S.J., Balcer, M.: Executable UML: A Foundation for Model-Driven Architectures. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (2002)
23. Knublauch, H., Oberle, D., Tetlow, P., Wallace, E.: A semantic web primer for object-oriented software developers. W3C Working Group Note 9 March 2006, W3C (2006)
24. Milewski, M., Roberts, G.: The model weaving description language (MWDL)-Towards a formal aspect oriented language for MDA model transformations. In: Proceedings of the 1st Workshop on Models and Aspects - Handling Crosscutting Concerns in MDSO, in conjunction with the 19th European Conference on Object-Oriented Programming. (2005)
25. Ossher, H., Tarr, P.: Hyper/j: multi-dimensional separation of concerns for java. In: ICSE '01: Proceedings of the 23rd International Conference on Software Engineering, Washington, DC, USA, IEEE Computer Society (2001) 821–822
26. Fabro, M.D.D., Bzivin, J., Jouault, F., Breton, E., Gueltas, G.: Amw: a generic model weaver. In: Proceedings of the 1re Journe sur l'Ingnieurie Dirige par les Modles (IDM05). (2005)
27. Brockmans, S., Volz, R., Eberhart, A., Lffler, P.: Visual modeling of owl dl ontologies using uml. In: International Semantic Web Conference. (2004) 198–213
28. Wagner, G., Giurca, A., Lukichev, S.: A general markup framework for integrity and derivation rules. In Bry, F., Fages, F., Marchiori, M., Ohlbach, H.J., eds.: Principles and Practices of Semantic Web Reasoning. Dagstuhl Seminar Proceedings, Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), Schloss Dagstuhl, Germany (2006)
29. Milanovic, M., oevic, D.G., Giurca, A., Wagner, G., Devedic, V.: On interchanging between owl/swrl and uml/ocl. In Chiorean, D., Cluj-Napoca, Demuth, B., Gogolla, M., Warmer, J., eds.: 6th OCL for (Meta-)Models in Multiple Application Domains (OCLApps 2006). (2006)
30. Eberhart, A.: Automatic generation of java/sql based inference engines from rdf schema and ruleml. In: ISWC '02: Proceedings of the First International Semantic Web Conference on The Semantic Web, London, UK, Springer-Verlag (2002) 102–116
31. Kalyanpur, A., Pastor, D.J., Battle, S., Padget, J.A.: Automatic mapping of owl ontologies into java. In Maurer, F., Ruhe, G., eds.: SEKE. (2004) 98–103
32. Völkel, M., Sure, Y.: Rdfreactor – from ontologies to programmatic data access. In: Poster Proceedings of the Fourth International Semantic Web Conference. (2005)