Bridging UML Profile Based Models and OWL Ontologies in Model-driven Development – Industrial Control Application

David Hästbacka and Seppo Kuikka

Department of Automation Science and Engineering, Tampere University of Technology
P.O. Box 692, FI-33101 Tampere; Korkeakoulunkatu 3, Tampere, Finland

Abstract. Model-driven development is considered to improve productivity and quality in software application development. The increasing complexity in models and the number of modeling methods used requires new approaches for knowledge management to make the handling of models easier both during design and run-time. Modeling in MDD shares characteristics with ontology development. This paper discusses UML based models used in MDD and their relationship to OWL ontologies. A concept is proposed how to create ontologies corresponding to these models and how they can be used concurrently in supporting the application development. The main principle of the approach is the distinct separation of knowledge in the domain model and model instances. As a result the instance model transformations can be kept simple and corresponding ontology representations of application models can be used to support the development. Applications of the approach to model-driven development and engineering of industrial control applications are also discussed.

1 Introduction

The pervasive uses of computers and software in various application domains and the advances in networking technologies have created a demand for new methods for developing complex software applications. Model-driven engineering (MDE) and model-driven development (MDD) have been proposed as methods that promote the use of models on different levels of abstraction to narrow the gap between the problem domain and implementation technologies.

Models are abstractions of some aspects of a system and they are used for developing new and describing existing systems. Expertise and important aspects of the domain can be used and taken into account when models and modeling concepts on an appropriate level of abstraction are being used. In MDD the models on different levels are gradually refined and transformed finally towards the executable application.

From a technical point of view, MDD can be carried out with the use of standard UML and its extension profile mechanism, e.g. SysML or custom profiles, or with the use of domain-specific languages (DSL). In order to cater domain-specific needs it is often required to implement the modeling concepts either using an extension profile

Hästbacka D. and Kuikka S..
Bridging UML Profile Based Models and OWL Ontologies in Model-driven Development – Industrial Control Application.
DOI: 10.5220/0003561900130023
In *Proceedings of the International Joint Workshop on Information Value Management, Future Trends of Model-Driven Development, Recent Trends in SOA Based Information Systems and Modelling and Simulation, Verification and Validation* (FTMDD-2011), pages 13-23
ISBN: 978-989-8425-60-7

or a domain-specific modeling language. The use of these new abstractions, i.e. modeling concepts, causes new challenges such as supporting and using them in models, overlapping viewpoints, as well as concerns related to defining and using model transformations and maintaining traceability in models [3].

As the diversity and complexity of modeling methods and associated modeling elements is increasing, new approaches are required to ease the handling of models both during development and at run-time. For example, modeling constructs used in development can be enriched with semantics that are better described with mechanisms other than the metamodel, e.g. for domain knowledge representation and later analysis. The use of domain-specific elements often results in hierarchical, pattern-like structures that may be out of the scope of the metamodel. A large variety in modeling conventions also prevents the interpretation of models used, for example, at run-time in order to provide information on the capabilities and the structure of a system.

The vision of the Semantic Web [1] is a world where knowledge is shared in an open environment with machine-readable metadata to enable automated agents and software applications intelligent access to resources. The emergence of ideas related to the Semantic Web has increased the interest in associated technologies, i.e. those recommended by the W3C such as the Resource Description Framework (RDF), RDF Schema (RDFS) and the Web Ontology Language (OWL).

In the Semantic Web and artificial intelligence research, ontologies are used to specify taxonomies for defining classes of objects with associated relationships and properties. The intent of the aforementioned technologies is to provide a formal description of concepts and their relationships within a specific domain of knowledge. These machine interpretable descriptions enable software applications to access and manipulate information, and further infer new knowledge by application of inference rules.

The MDD paradigm shares characteristics with the aims of the Semantic Web and its technologies. In a way, from an application development point of view both strive to provide abstractions of the things being described. The knowledge of the domain is especially important in ontology development but domain-specific aspects are typical also in MDD system modeling. The interesting features provided by ontologies, and the main justifications for using ontologies in MDD, are the semantically enriched descriptions - unrestricted by the metamodel. The descriptions, in combination with rule-based inference can be used, for instance, to support development and ease the use of modeling elements, and for different kinds of examination purposes.

In this paper MDD and metamodeling, as defined by OMG in its Meta-object Facility (MOF) specification (2006), is discussed from the point of view of combining UML based models with OWL ontologies. The paper is based on experiences using a domain specific profile for modeling and development of industrial control applications. Section 2 presents the background and some of the challenges related to combining UML based models with OWL ontologies. A conceptual approach dividing the problem into domain knowledge transformation and instance model transformation is presented in section 3. Results from experiments as well as possible use cases and applications of the approach in engineering are also discussed. Related work is presented in section 4 before concluding the paper in section 5.

2 Background

Models used in model-driven development typically adhere to some modeling language, e.g. a metamodel, that provides the rules and building blocks for constructing the models. In MDD different metamodels can be used and the aim is usually also in defining automatic transformations that can be executed to ease transformation from one level of abstraction to another. Computer interpretability and formality of the models is required in order to facilitate automatic transformations. This may require the use of a relatively small amount of fixed domain-specific modeling concepts. As a result, more than one modeling method may be needed to express all aspects which, in turn, could present new challenges combining different and possibly overlapping viewpoints of concurrent models.

Technologies for defining ontologies have a different approach. The specification of an ontology basically starts from nothing and knowledge about the domain is added on concept by concept or by linking to existing knowledge in previously created ontologies. The ontology is expressing all the knowledge in the domain whereas the metamodel and its defined modeling concepts typically have agreed semantics in the field of application. For ontology development there are, fortunately, basic concepts in core and domain ontologies that can be used as a basis when defining new concepts.

In ontology development, RDF is used for making statements about resources on the web in the form of triples. The triples are constructed in a subject-predicate-object manner and enable representation of information in the form of graphs. RDF Schema is a basic vocabulary for RDF that can be used to create hierarchies of classes and properties. In the stack of Semantic Web technologies OWL provides an additional layer of constructs for describing semantics of RDF statements. OWL is based on description logic bringing reasoning to the approach and allows, for example, stating constraints on cardinality, restrictions of values and characteristics of properties.

2.1 Major Differences Modeling Objects in MDD and Ontologies

There are challenges unifying models and ontologies due to the different nature and point of focus of the approaches. In comparison, a descriptive modeling approach based on ontologies is, in general, more flexible and less restrictive allowing expressivity beyond typical object-oriented modeling. Oren, Heitmann and Decker [11] compared object-oriented programming languages and the Semantic Web and state that objects in typical object-oriented languages must be a member of exactly one class, inherit only one super class, and conform exactly to the structure of the class definition. Compared to RDF(S) the resources can have multiple types and super classes, and differ from their original definitions or not have any definitions at all.

The philosophy interpreting ontologies also differs in a constitutive way. For example, when resources are being identified they are matched against the ontology descriptions and class memberships are resolved based on properties and relationship statements. The relationship between objects and OWL/RDF resources has been considered by Hillairet, Bertrand and Lafaye (2008) focusing on attributes vs. properties, structural inheritance, and object conformance. In the object-oriented model attributes are defined inside a class whereas OWL and RDF properties are entities which can be used by any resource in the absence of domain and range declarations. Hillairet et al. (2008) continue that because properties in OWL are not inherited the property domains are instead propagated upwards indicating the membership of the resources using the property.

As a result of using ontologies for describing the nature and behavior of systems, the interpretability may suffer as the knowledge base may get excessively large unless planned and constructed carefully. This happens, for example, when many aspects of a system are described in an ontology and the concepts have to be expanded with detailing semantics in the absence of previous knowledge. Modeling objects in MDD strives for good interpretability and a model in a diagram, for example, is typically easily understood by a human. This is also an issue of using tailored or generic modeling concepts which is heavily reflected on the intended usage scenarios of the models.

2.2 Example MDD Environment: UML Based Profile

The modeling concepts used as reference and discussion in this paper are from the UML Automation Profile (UML AP) aiming to provide domain-specific concepts for modeling of industrial control applications [13]. The profile is extended from suitable elements of the UML Real-Time Profile (UML Profile for Schedulability, Performance and Time), SysML and UML Profile for Quality of Service and Fault Tolerance. The profile is based on a first-class extension mechanism extended from UML and SysML metamodels, which enables the use of domain concepts concurrently with UML and SysML. Featured are subprofiles for modeling of requirements, domain-specific and platform independent functionality, distribution of components on a system level, and devices and resources of the platform. The metamodel and the tool support are implemented using Eclipse (EMF) among other tools.

The MDD approach [7] is based on OMG MDA and utilizes the aforementioned modeling constructs as its metamodel. The development process includes three main phases: requirement modeling and unification of source data, functional modeling, and platform specific modeling on the execution platform. Various transformations have been specified and implemented to support and automate the development as much as possible. The process aims to increase efficiency, quality and reusability of solutions while allowing domain expertise to be taken into account during development.

3 Approach Unifying UML Based Models with OWL Ontologies

Defining a modeling language is in effect specification of building blocks, their interrelations, and rules that dictate how systems are to be modeled. Considering the nature of the process it can be argued that defining a modeling language also involves describing the domain in a similar manner to creating domain ontologies. Real-world objects that are modeled either using MDD models or ontologies are then related to the metamodel elements or the domain ontology concepts, respectively.

3.1 Outline

A plausible approach combining models in MDD with ontologies is therefore divided into two separate tasks: the metamodel to domain ontology transformation and the model instance to ontology individual transformation, as illustrated in Figure 1. The aim is to keep knowledge about the domain static with regard to the modeling language. Transformation of model instances to ontology individuals is delimited to mapping of structures and data when individuals created can be related to previously created ontology concepts.

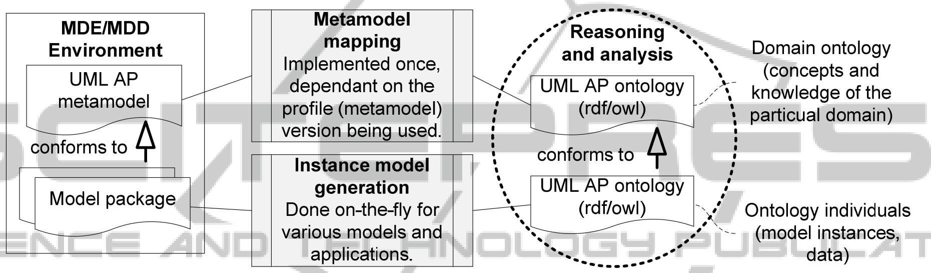
MDE/MDD Environment
UML AP metamodel
conforms to
Model package
Metamodel mapping
Implemented once, dependant on the profile (metamodel) version being used.
Instance model generation
Done on-the-fly for various models and applications.
Reasoning and analysis
UML AP ontology (rdf/owl)
conforms to
UML AP ontology (rdf/owl)
Domain ontology (concepts and knowledge of the particual domain)
Ontology individuals (model instances, data)

Fig. 1. The transformation from UML based models to OWL is divided into domain knowledge transformation and instance model transformation.

At this point transformations are considered only one-way and round-trip transformations back from OWL to UML are not examined due to issues that would require a different transformation approach. For example, converting a graph structure in OWL into a tree based structure in UML based models, and more importantly the possibility of unforeseen descriptions in ontologies that cannot be transformed automatically to keep the parallel models synchronized. It is an interesting thought, however, to model simultaneously in both MDD and ontologies.

3.2 Metamodeling in MDD and Domain Ontologies

From a MOF based MDD perspective the mapping of a metamodel to a domain ontology can be seen as a metametamodel transformation. The transformation includes creating a domain ontology corresponding to the elements of the UML based modeling constructs on the M2 level (as seen on the left in Figure 2). Because the modeling elements of the modeling language (metamodel, M2) are defined using higher-level elements of the metametamodel (M3) the transformation is then defined using M3 elements. For example, a metametamodel transformation could define that a MOF Class is transformed to an RDF Class.

Metametamodels are usually implemented using platform specific notations, such as Eclipse EMF in the case for UML AP. Therefore, a metametamodel transformation is considered best handled using tools and technologies of the platform, e.g. QVT or some other transformation language supporting EMF based models. As the level of abstraction increases there are typically fewer transformation mappings to define. On the other hand, transformations easily become complex so that automatic transformations are almost impossible to implement.

The benefits of automatic transformations on a metameta level depend on how frequently the modeling language changes and how the corresponding domain ontologies are going to be used. For a standardized modeling language the transformation is done only once and manually while automatic generation is preferred for rapidly evolving DSLs. If automatic metametamodel transformations, such as the EMF Triple Eclipse plug-in (2010), are used and only partial solutions that produce basic classes and hierarchies are available, the generated ontologies can, nevertheless, serve as an excellent basis for manual completion.

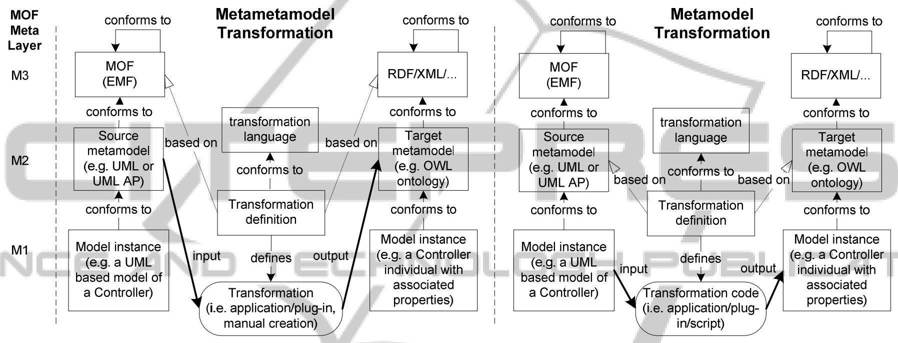
MOF Meta Layer
M3
M2
M1
Metametamodel Transformation
conforms to
MOF (EMF)
Source metamodel (e.g. UML or UML AP)
Model instance (e.g. a UML based model of a Controller)
transformation language
Transformation definition
Transformation (i.e. application/plug-in, manual creation)
based on
input
defines
output
RDF/XML/...
Target metamodel (e.g. OWL ontology)
Model instance (e.g. a Controller individual with associated properties)
Metamodel Transformation
Transformation code (i.e. application/plug-in/script)

Fig. 2. Transformation of UML based metamodels into OWL domain ontologies and the transformation of UML based model instances to OWL individuals.

3.3 Model Instances and Ontology Individuals

The correspondence of model instances and ontology individuals in relation to transformations in this approach is presented on the right in Figure 2. Similar to the metametamodel transformations discussed in the previous section, the transformation from model instances to ontology individuals can be defined using higher-level elements, i.e. the metamodel elements defining the modelling language. For example, a transformation could specify that an UML AP Controller instance is to be mapped as a Controller individual of the domain ontology previously created.

The key idea in the approach is to separate the transformation of domain knowledge from transformation of instance models. In this way, the instance transformations can be kept simple and straightforward to implement as transformations can concentrate on mapping of serialized structures between the notations. This ensures that model transformations can be automated and ontology representations for further utilization can be created without additional effort.

3.4 Experiences from Model Instance Transformation Based on XMI

To reduce the dependence of the MDD environment, an XML Metadata Interchange

(XMI, 2007) based approach was used in order to provide flexibility better suited for a distributed engineering environment. XMI is interesting because standard XML transformation tools can be used to transform models into OWL/XML. In principle, it is straightforward to generate OWL individuals corresponding to UML AP model elements because both source and target models have a previously defined metamodel and semantics, and only the serialized structure of the model is of relevance.

The Eclipse (EMF) environment allows exporting UML based models in XMI. XMI, however, only defines a metaformat for a further specified transfer format for serializing models. The XMI exports of UML AP, for instance, have resource dependencies to EMF, UML2 and SysML, and the metamodels with their schema definitions are required in external tools to perform the transformations. Finally, it is also worth noting that as the models are XMI based there is no real metamodel available that could be utilized in typed mappings between the constructs of the different structures.

The XMI transformations become easily large and complex making it challenging to maintain compared to e.g. metamodel transformation tools available for the Eclipse environment. The lack of a strongly typed metamodel and object inheritance also increases redundancy when same kinds of transformations have to be defined for different objects with similar nature. Other minor problems were also encountered such as when triplets were added it was not unambiguous to which individual the new assertions actually belonged. However, with reasoning on properties and the structure, i.e. hasPart/isPartOf relations, some of these problems can be resolved. A universal solution is to rely on unique identifiers available in XMI elements when naming and referring to specific individuals. Consequently, the naming practice must be addressed in assertions that reflect relations between individuals as the unique identifiers are not present in all references from one UML based element to another.

For XMI serialized model elements it is not evident which attributes are of value as there are also attributes related to the tool environment that do not reflect the metamodel semantics that the element is an instance of. Manual specification of required attribute transformations may be challenging as the amount of attributes can be substantial. An approach using input parameters for the transformation could therefore be the most efficient and yet still maintainable solution to control what attributes should be transformed using generic assertions.

The proposed concept was tested with an XSLT template that takes the UML AP model instances from the XMI serialization as its input. As a result, a new OWL/XML document with OWL individuals is created along with a domain ontology import to which the newly created individuals conform to.

3.5 Results and Future Applications in Engineering

Figure 3 illustrates a subset of an industrial process control application model and the resulting ontology representation of the individuals with class relationships to the domain ontology. The platform independent application model consists of two measurement inputs for monitoring temperature and level, one control function without any specified algorithm, one on/off type actuator output for controlling a heater, and a safety interlocking to prevent the heater from being on when the level is too low. The resulting ontology may then be used concurrently to facilitate the use of complex models in MDD and support knowledge management.

The information gathered in the domain ontology and the ontology individuals does not necessarily provide additional benefits unless further knowledge is inferred

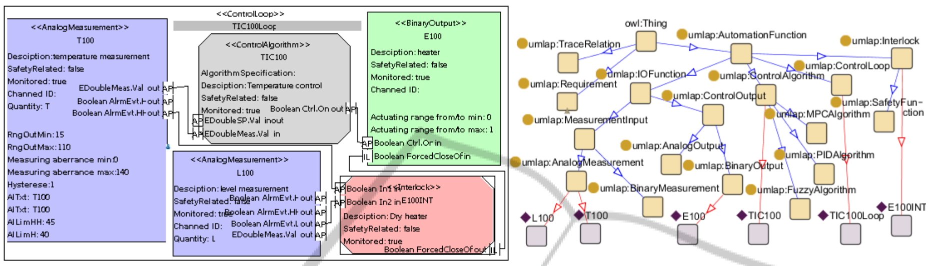

Fig. 3. Subset of a control application model and its simplified ontology representation.

from it or it is combined with other existing knowledge, or used, for example, with query languages such as SPARQL Query Language for RDF. Supplementary information such as company or project specific practices, previously captured tacit knowledge and other points of interest can be presented in special ontologies and used in the synthesis of a knowledge base to support engineering.

The most apparent applications in the near future are different types of services supporting MDD design tasks. In the MDD environment interactive aids can be provided to help choosing and using the modeling concepts. An interactive guide, for example, could simultaneously analyze the application being modeled and suggest appropriate elements. The approach could be applied to a previously developed work support tool [6] in order to support the transfer and usage of tacit knowledge in MDD of control applications. For example, during the platform specific phase it could be beneficial to have additional knowledge automatically presented as there can be hundreds of design constructs available in typical distributed control system platforms.

Structural analysis of models could also be performed that based on rules can reason whether required and correct elements are used, element connections are complete and that common identifiable human design errors, for instance, are avoided. Industrial control applications are complex and can contain thousands of objects that need to be managed and manual checking of which is challenging. Automatically executed model examinations, for example, could be provided as external services in versioning to plant information models used in distributed engineering. As the models also include modeling elements from requirements and functions to platform specific constructs, similar analysis could also be done to support the MDD process by examining traceability soundness between model elements in the different phases.

Using the approach with mappings of similar modeling concepts, knowledge could be mined even from instance models developed with different modeling languages in project databases of previous solutions. Frequently occurring control or safety interlocking structures as well as best practice solutions, e.g. for specific types of devices, could be extracted as new knowledge with suitable mining techniques. There are, of course, many uses of the approach outside of the MDD environment. For run-time operation of the system, a platform independent ontology-based description of the structure and the capabilities could be used to facilitate integration of networked dynamic system configurations, e.g. in association with agent technologies.

4 Related Work

One of the most important advantages of using ontologies is flexibility in information integration when combining information from various sources and inferring new facts on this. The use of ontologies in the software engineering lifecycle in general has been analyzed by Happel and Seedorf (2006) and they argue that within software engineering the specific advantages are the formal definitions of a domain that encourages a broader use of ontologies throughout the whole engineering lifecycle. Merging model-driven and ontology driven system development has been studied by Soylu and De Causmaecker (2009) for pervasive computing applications. They argue that when the context space is expanded the applications need to be more intelligent which also reflects on the development methodology employing formalized concepts.

Na, Choi and Jung (2006) have presented a method for transforming standard UML models into OWL ontologies. In their approach an XSLT transformation was implemented to map generic UML constructs to OWL concepts. Walter, Parreiras and Staab (2009) have proposed an approach for using ontologies to describe domain-specific languages. The approach constitutes an ontology-based framework for defining DSLs enriched by formal class descriptions. The framework then provides tools for checking the consistency of models and reasoning for dynamic classification. TwoUse (Parreiras, Staab, 2010) is a framework for integrated use of UML class-based models and OWL ontologies. In the case study presented, TwoUse features have been analyzed for non-functional requirements and it is stated to achieve improvements on maintainability, reusability and extensibility.

5 Conclusions

As a result of increasing productivity and quality requirements, the continuing shifting of design towards a higher level of abstraction is expected by utilization of advanced modeling techniques. In addition, the size and complexity of systems is increasing and new methods are needed for model management. Ontologies typically demand additional modeling effort that must pay off in order to be established. One way of promoting the use of ontologies is in a higher reuse of ontological knowledge and the use of it not only during the development but throughout the whole application lifecycle.

The challenges of combining UML based models with OWL ontologies were discussed related to metamodeling and use in MDD. In the presented approach, a domain ontology is first developed separately to which corresponding model instances are later appended as ontology individuals. Novel for the approach is the partition of the domain knowledge and the instance models making the implementation of model instance transformations straightforward. The concept has been successfully tested in transforming a subset of model instances used for developing industrial control applications. The approach opens up new possibilities to apply reasoning and analysis of models to support MDD of industrial control applications. Automatic transformation of UML based metamodels to domain ontologies was considered less important due to platform dependencies of the metamodel implementation and the not so evident benefits in the case of stable metamodels evolving in a controlled way.

The information in the generated ontologies along with other knowledge, i.e. presented as separate ontologies, forms a knowledge base that can be used for reasoning in various services supporting MDD and structural analysis of models, for example. In the future, research will be continued on how knowledge in ontologies can be applied to engineering processes to support design. There is also interest to study the lifecycle of MDD models as a part of the plant model and the plant lifecycle.

References

1. Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web, Scientific American (2001)
2. EMF Triple Eclipse Plugin. URL: http://code.google.com/p/emftriple/ (April, 2010)
3. France, R., Rumpe, B.: Model-driven Development of Complex Software: A Research Roadmap. In: 2007 Future of Software Engineering (FOSE '07) (23-25 May 2007) 37-54
4. Happel, H-J., Seedorf, S.: Applications of Ontologies in Software Engineering. In: Proceedings of the 2nd International Workshop on Semantic Web Enabled Software Engineering (SWESE 2006) (November 6th 2006) Athens, USA.
5. Hillairet, G., Bertrand, F., Lafaye, J.-Y.: Bridging EMF applications and RDF Data Sources. In: 4th International Workshop on Semantic Web Enabled Software Engineering (SWESE) at ISWC'08, Karlsruhe, Germany.
6. Hästbacka, D., Laitinen, O., Tommila, T., Kuikka, S.: Implementing a Work Support and Training Tool for Control Engineers. In: 4th IEEE International Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS'2007), Dortmund, Germany (6-8 September 2007) 512-517
7. Hästbacka, D., Vepsäläinen, T., Kuikka, S.: Model-driven Development of Industrial Process Control Applications. Journal of Systems and Software (2011). In Press, Accepted Manuscript. doi:10.1016/j.jss.2011.01.063
8. Na, H.-S., Choi, O-H, Lim, J.-E.: A Method for Building Domain Ontologies based on the Transformation of UML Models. In: Proceedings of the Fourth International Conference on Software Engineering Research, Management and Applications (SERA '06) 332-338 doi:10.1109/SERA.2006.4
9. Object Management Group: Meta Object Facility (MOF) Core Specification, Version 2.0 formal/06-01-01 (January 2006)
10. Object Management Group: MOF 2.0/XMI Mapping, Version 2.1.1 (December 2007)
11. Oren, E., Heitmann, B., Decker, S.: ActiveRDF: Embedding Semantic Web data into object-oriented languages. Web Semantics: Science, Services and Agents on the World Wide Web 6 (3), World Wide Web Conference 2007, Semantic Web Track (September 2008) 191-202
12. Parreiras, F., S, Steffen Staab, S.: Using ontologies with UML class-based modeling: The TwoUse approach. Data & Knowledge Engineering, Special issue on contribution of ontologies in designing advanced information systems 69 (11) (November 2010) 1194-1207
13. Ritala, T., Kuikka, S.: UML Automation Profile: Enhancing the Efficiency of Software

Development in the Automation Industry. In: 5th Int. IEEE Conf. on Industrial Informatics (INDIN 2007), Vienna, Austria, (23-26 July 2007) 885-890

14. Soylu, A., De Causmaecker, P.: Merging model driven and ontology driven system development approaches pervasive computing perspective. In: 24th International Symposium on Computer and Information Sciences (ISCIS 2009) (14-16 September 2009) 730-735
15. Walter, T., Parreiras, F. S., Staab, S.: OntoDSL: An Ontology-Based Framework for Domain-Specific Languages. In: Model Driven Engineering Languages and Systems, 12th International Conference (MODELS 2009) 5795 (2009) 408-422