An Architecture of Ontology-aware Metamodelling Platforms for Advanced Enterprise Repositories

Srdjan Zivkovic, Harald Kühn and Marion Murzek

BOC Information Systems GmbH, Wipplingerstrasse 1, 1010 Vienna, Austria

Abstract. Enterprise repositories have become core information assets of today's enterprises. They store and manage models of different aspects of an enterprise, such as business strategy, business processes, organizational structures, and IT infrastructure in an integrative way based on a bundle of domain-specific modelling languages. To produce such models for enterprise repository and profit from their usage, number of mechanisms such as model querying, validation, simulation, model transformation, versioning and traceability are needed. Metamodelling platforms provide flexible and extensible environment for realizing such advanced enterprise repositories, where various mechanisms working on repository may be integrated, thus providing a superior modelling solution. Going one step further, the question is, can such platforms profit from semantic technologies? How architecture of ontology-aware metamodelling platforms can be designed? In this paper, we give an in-depth view of such platform architecture by discussing its main building blocks and their dependencies.

1 Introduction

Comprehensive enterprise modelling tools are for enterprise repositories as database management systems are for raw data. Raising the abstraction level of data management, modelling repository-enabled tools offer sophisticated means for storage, querying, visualising and manipulating enterprise information, thus leading to a model management. Such tools enable a holistic approach to IT-based, model-aware strategy, business process and IT management of the enterprise. Metamodelling platforms provide flexible environment for realizing enterprise repository based modelling tools. Such platforms offer: (1) modelling and metamodelling capabilities (2) set of mechanisms to work on models and metamodels (3) guidance on how to apply particular modelling method using corresponding mechanisms in order to achieve method specific goals [2]. Here, a model as a data abstraction represents a first-order entity for capturing enterprise information. Going one data abstraction step further, the question is, can enterprise modelling tools become ontology-aware? How can ontology and related semantic technology leverage models & metamodels, mechanisms and guidance of the modelling process? We analysed some scenarios on how ontologies and automatic reasoning may bring benefit to metamodelling platforms [2]. We concluded that modelling languages and models enriched by machine-

Zivkovic S., Kühn H. and Murzek M. (2009).
An Architecture of Ontology-aware Metamodelling Platforms for Advanced Enterprise Repositories.
In *Proceedings of the Joint Workshop on Advanced Technologies and Techniques for Enterprise Information Systems*, pages 95-104
DOI: 10.5220/0002202900950104

readable semantics enhance quality of modelling solutions. Furthermore, configuration of mechanisms may be semi-automated and their quality of service may be enhanced by relying on ontologies. Finally, the guidance gains benefits from ontologies by having process models extended by formalized rules, conditions and actions, thus leading to more flexible guidance. Considering previous findings, we extended the metamodelling architecture to be ontology-aware.

Consequently, in this paper, our main goal is to give an in-depth view of such platform architecture by discussing its main building blocks and their dependencies from the logical viewpoint. We also summarize our current results of the work-in-progress and give outlook for the future work.

2 Logical Architecture

The ontology-aware platform architecture for advanced model repositories extends the generic metamodelling platform architecture [1]. It introduces the ontology aspect, such that the platform core features may profit from semantic technologies [2].

The root core architectural element is the *meta-metamodel* which contains the concepts available for the definition of modelling languages. Based on it, the *metamodel library* contains metamodels of defined modelling languages. The metamodel library conforms to the meta-metamodel and, in turn, forms the foundation of the *model repository*, where all models are stored. As an extension to models on different levels, the *ontology repository* serves as storage of the domain, upper-level and process ontologies. These four elements form the *ontology-aware model management building block* (see section 2.1). All mechanisms used for evaluating and processing of models, metamodels and ontologies are part of the extensible *ontology-aware, model-aware mechanisms building block* (see section 2.2). *Persistency services* abstract from concrete data persistence systems and offer support for the durable storage of models, metamodels, and ontologies. *Access services* serve two main tasks (section 2.3). On the one hand they enable the open, bi-directional exchange of all metamodel, model and ontology information. On the other hand they cover all aspects concerning security such as access rights, authorization, and en-/decryption. User interface components such as *graphical and textual based editors* are used for the definition, representation and maintenance of models, metamodels, ontologies and mechanisms (see section 2.4). A workbench layer serves as a common environment for integrating different editors in a single modelling solution.

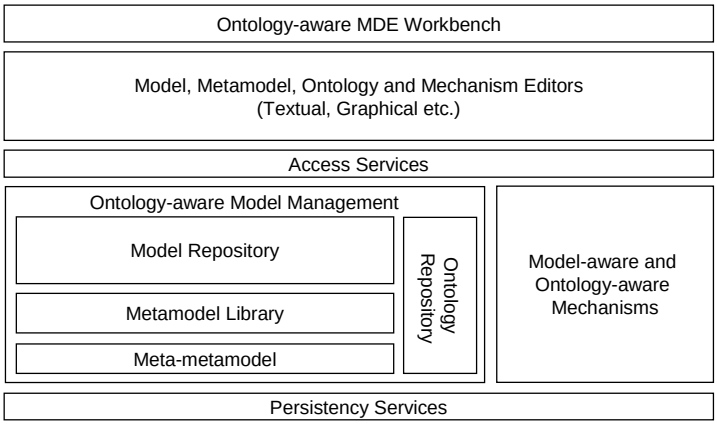

Fig. 1. Platform Architecture for Advanced Enterprise Model Repositories.

2.1 Ontology-aware Model Management Building Block

Meta-metamodel. Given the 4-layer metamodelling architecture, the meta-metamodel defines the constructs available for the definition of modelling languages. Typical metamodelling constructs are class, relation, endpoint, attribute, model type, package etc. There are different possible implementations of the M3 model for different modelling frameworks. The Meta Object Facility (MOF) is a standard meta-metamodel to build the OMG-based family of modelling languages like UML and BPMN [3]. Another implementation of the M3 model is the Ecore model [4], the meta-metamodel of the Eclipse Modelling Framework (EMF) which is, from its coverage, comparable to the subset of MOF, essential MOF (EMOF). Ecore with its features is customized to support seamless projection to Java. A further implementation of the M3 model is the KM3 [5], the meta-metamodel of the ATLAS Model Management Platform [6]. The KM3 is the lightweight textual metamodel definition language which allows easy creation and modification of metamodels. It uses just few concepts like Class, Attribute and Reference. It is structurally close to EMOF and Ecore. Comparably, the ADOxx Metamodelling Platform[1] implements on M3 level the ADOxx Meta^2-Model, which is optimized for the rapid definition of the visual modelling languages for enterprise modelling.

Metamodel Library. A metamodel library represents a collection of metamodels of defined modelling languages. A metamodel describes basic constructs of a modelling language. A metamodel is defined using meta-metamodel constructs. Metamodel forms the foundation for the model repository, where all models conforming to the corresponding set of metamodels are stored. For example, the BPMN metamodel alone or a family of domain specific languages (DSL-s) which cover different aspects of an enterprise are all defined on the M2 layer. Using a translational semantics

[1] ADOxx® is the extensible, multi-lingual, multi-os, repository based platform for the development of modelling tools of the BOC Group. ADOxx® is a registered trademark of BOC Group. http://www.boc-group.com.

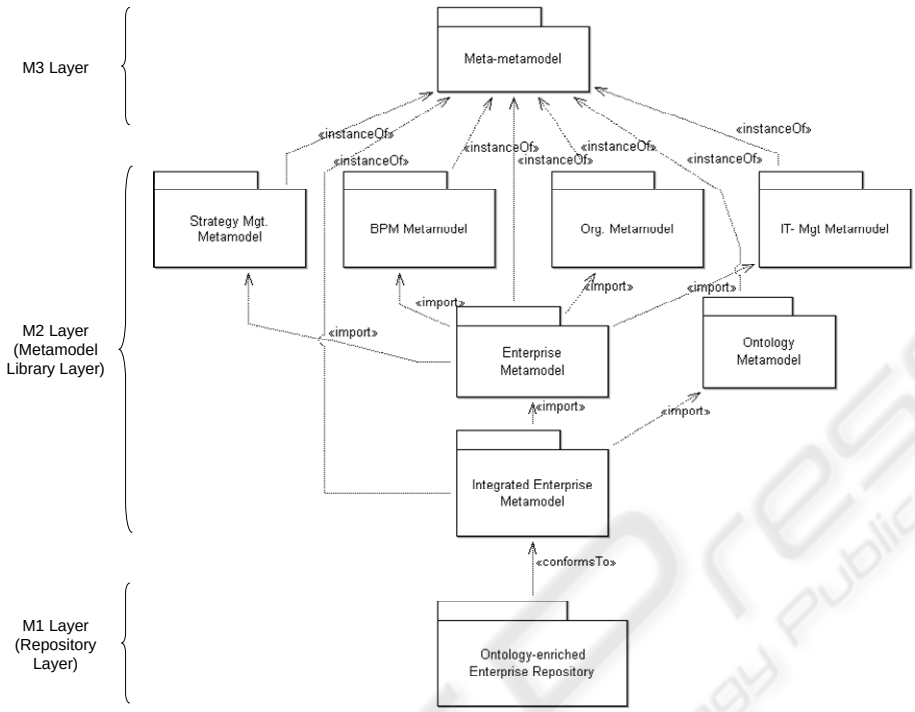

Fig. 2. Ontology-aware Model Management Building Block.

approach, the semantics of some language L can be translated to some other language L1 with known semantics. Thus, models can be transformed to other languages whose semantics are formally defined, such as ontologies. Therefore, an ontology language like ODM [7] is defined on M2 level as well. Ontology languages and target modelling languages are then combined on the M2 level, which all results in an integrated modelling approach on M1 level, where semantic constraints on models may be expressed using ontology concepts (see Fig. 2). The TWOUSE approach addresses the integration of UML-like languages and OWL, which results in an integrated modelling language for software engineers [8]. The same approach may be applied to an enterprise metamodel, thus resulting in an ontology-enriched modelling language of the enterprise repository.

Model Repository. A model repository is the central architectural component of the platform responsible for storage and management of models. A repository is a generic model storage system, always configured by a specific metamodel library (see Fig. 2, M1 Layer). The main core elements of the model repository are: repository, repository instance, model, modelling instance, attribute instance, relation instance, endpoint instance etc. Each repository concept has its corresponding concept in the metamodel library so that the “conformsTo" relationship between M1 and M2 instances is fulfilled. Unlike conventional repositories, having the integrated ontology-enriched metamodel, the model repository is able to manage ontologies as enrichment to models. Usually, only semantics of the models not expressible in conventional

languages are stored in ontologies such as different kinds of constraints.

Ontology Repository. An ontology repository serves as a global storage of ontologies. Various domain ontologies and upper-level ontologies are managed within this repository. For example, reference domain ontologies are stored here, which represent formally described semantics of a particular domain, against which the corresponding models from the model repository on M1 level are validated. The similar is true for the modelling languages defined on the M2 layer in the metamodel library. Here, modelling languages are validated against their corresponding reference, upper-level ontologies.

2.2 Ontology-aware Model Mechanisms Building Block

The mechanisms building block represents an extensible set of various platform features all working on the ontology-aware model management building block. Typically, mechanisms use meta-information stored either on M3 or on M2 level, in order to manipulate or work on instances on M2 or on M1 level, respectively. In the following, the main platform mechanisms are described in more detail.

Integrated Querying. Querying a model repository is a basic capability which makes comprehensive model information available. In an ontology-enriched modelling environment, both model query languages like GReQL [9] or OCL[10] as well as ontology query languages like SPARQL[11] are needed, to support querying in an integrated way. A proposed platform support for integrated querying may include an evaluator of the integrated query language which transforms part of the queries to either model QL or ontology QL (see Fig. 3). Such queries are then executed against the model i.e. ontology repository. Query language transformation engine also performs translation of the query results back to the integrated query result. An existing approach for integrating querying is the TWOUSE OCL [8]. The TWOUSE OCL is an extension of the OCL with support to built-in operations which calls reasoning services to reason over models after they are translated into OWL.

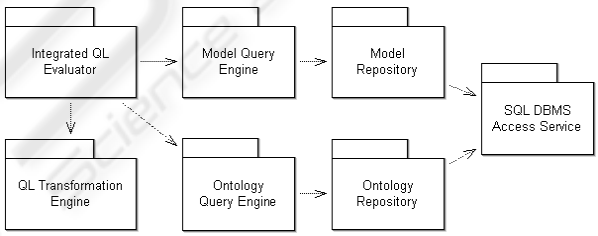

Fig. 3. Integrated Querying Mechanism.

Metamodel Mapping. Metamodel mappings have an important role in the metamodelling environment by adding knowledge about integrative usage of different modelling languages, still leaving their metamodel independent. A metamodel mapping represents a structural and semantic correspondence between concepts of two meta-

models. One mapping takes at least one metamodel element (e.g. class, attribute or relationship) from each of the source metamodels and relates them, in order to denote the nature of the appearing structural and semantic conflicts. Bridging of metamodelling and ontology languages is done via mappings [12]. Mapping may build basis for MDA-based model transformation rules [13] or even for metamodel composition rules [14]. The role of ontologies here is to facilitate identification of mappings. Metamodels are translated to ontologies in order to infer possible concept mappings.

Model Transformation. Model transformation is a backbone mechanism in model-driven approaches, since transformation engines enable MDA model chains (CIM-PIM-PSM). Business process model refinement is a typical scenario for model transformations within enterprise repositories. Manipulating a model repository, transformations are also used for model migration scenarios. Model transformation is a function that receives a source model, a source metamodel, a target metamodel and a set of rules as input and produces a target model which conforms to a target metamodel. The model transformation approach reuses the MDA architecture and defines a transformation rule language as a metamodel on the M2 layer of the architecture. The concrete transformation rules conform to this grammar and define how models are to be translated between different modelling languages [15]. Ontologies may improve the generation of model transformation rules [16].

Model Merge, Diff, Copy and Global Change. Model management mechanisms like merge, diff and copy are generic operators executed on models [17] in a model repository. The merge operator takes two or more models as input and produces as output the union of models as a new model. The diff operator computes the difference between two models. The difference between models represents the set of model elements (also as a model) in one model that do not correspond to any model elements in the other model. The operator copy takes one model as input and returns a copy of that model. The returned model includes all of the relationships of the input model, including those that connect its objects to objects outside the model. The global change operator applies global changes to objects of a model repository or of a particular fragment of a repository. The global change operator changes a property value or a set of property values which represent common properties/attributes of the selected set of the objects. For example, a global change operator may change the names of all model elements in one model or in the whole model repository, so that each model element name begins with the certain prefix. The target set of models the global change operator should work on can be determined by querying the model repository.

Model Versioning. The versioning mechanism allows management of different model versions in model repository over time. The unit of versioning can be either model or a definable set of model elements. The versioning mechanism utilizes operators like merge, copy and diff.

Model Validation. Validation mechanisms on models and metamodels are used, in order to support correct modelling. Basically, one can perform syntax and semantic checks on models. Syntax checks validate a model against its grammar, i.e. metamodel, to find out whether the models satisfy all the relevant well-formedness rules. On the other side, models can be also checked for semantic consistency. Semantic

checks validate models against defined semantic constraints. The prerequisite for automated semantic consistency checking are formally defined model semantics. A model validator requires as input a model to validate which resides in the model repository with the reference to its metamodel. A syntax checker validates a model for its conformance to its metamodel. Semantic checks are done by a semantic checker. The semantic checker translates the model part to the ontology and triggers a reasoner to perform a semantic check.

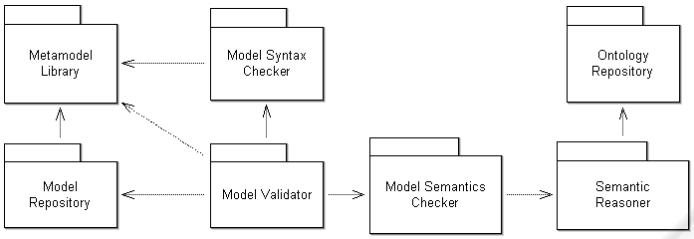

Fig. 4. Model Validation.

Traceability Management. One of the challenges in modelling environments is tracing of interdependencies between models and model elements created during the course of modelling or during the execution of various mechanisms like, model merge, transformation etc. For example, change impact analyses of enterprise models which trace inter-level and intra-level dependencies of model elements may be realized using traceability links. The solution may base on the similar approach which analyzes impact changes in the code back to their requirements [18]. Traceability relationships are created and stored within the model repository to track different kinds of such dependencies. Each TR conforms to particular TR type which is part of the metamodel and defines a set of possible relations between entities of different metamodels. Traceability manager as a software component offers API-s to create and query traceability links. Different platform mechanisms use then the traceability manager to trace their operations. Traceability manager may also profit from a semantic reasoner, for example, in inferring traceability links which are not explicitly modelled in the repository.

Guidance. A guidance engine is a platform mechanism which guides the modeller in achieving his goal within the particular modelling methodology. It is driven by the guidance knowledge base. The guidance knowledge base consists of several information sources. This may include formally described modelling process information, but may also consist of existing modelling artefacts and/or interpretation of traceability links. The architecture of the envisioned guidance mechanism is illustrated in Fig. 5.

Semantic Reasoning. The core mechanism which enables any ontology-aware approach is a semantic reasoner. Semantic reasoner offers a set of mechanisms which enable inferencing over inference rules represented using e.g. a formal ontology language. It is thus able to process queries on ontologies. A core building block is an inference engine which receives, for instance, an OWL ontology as input, applies inference production rules, and gives a query answer as output. A prominent implementation of the semantic reasoner is KAON2 [19].

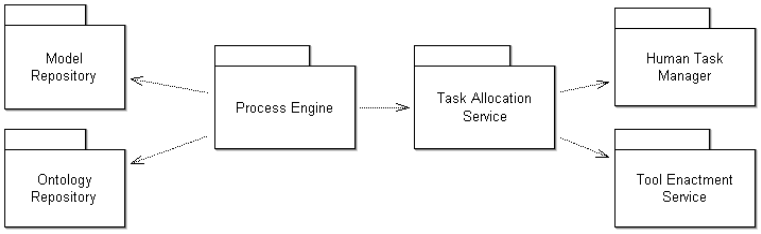

Fig. 5. The Main Building Block of Guidance Mechanism.

2.3 Platform Access Services Building Block

Access services represent a building block which acts as a facade to the middleware building blocks like ontology-aware model management and ontology-aware, model-aware mechanisms (see Fig 1). On the one hand, the access services building block enables the open, bi-directional exchange of all metamodel, model and ontology information. This is done by using various import/export services on model and meta-model level. For example, depending on the core meta-metamodel implementation of the platform, import and export interfaces to other meta-metamodels can be provided such as KM3 [4], Ecore [3] or XMI. Further, access services building block covers all aspects concerning security such as access rights, authorization, and en-/decryption. API-s for scripting languages and other programming languages are also part of this building block, which allows internal middleware components or UI building blocks as well as external systems to access the metamodel library, the model repository and mechanisms functionalities. Similarly, web service facade offers some of the platform functionality as modular web services to other systems.

2.4 Model, Ontology and Mechanism Editors Building Block

Model, metamodel, ontology and mechanism editors form the building block repre senting the presentation layer of the overall system (see Fig 1). Each of the editors provides user interfaces for corresponding building block of the middleware layer and is built according to a Model-View-Controller (MVC) architectural pattern. Since an ontology-ware modelling environment is dedicated to ontology-aware model-intensive tasks, a rich graphical editor for ontology-enriched model design is a mandatory feature. The design of such editor therefore leverages advantages of the integrated modelling language by supporting creation, editing, querying of combined models. Additionally, it also allows for decoupled, independent views on models on the one side and ontologies on the other side.

3 Summary and Future Work

In this research-in-progress paper we elaborated on the architecture of the ontology-aware metamodelling platforms as environments for realizing advanced enterprise

repositories. The main contribution of the work is a generic architectural blueprint for building such systems. An in-depth view of the main building blocks was given, focusing on how ontologies and semantic technology may be used to enhance the platform. At the same time, this paper presents the preliminary results of the work done within the EU funded FP7 research project MOST. The project MOST will improve software engineering by leveraging ontology and reasoning technology. To reach this goal, it aims at developing a seamless integration technology for ontologies into model-driven software development, resulting in ontology-driven software development (ODSD). Compared to the conventional approaches, ODSD should provide better understanding of the software artefacts, in particular of models, by applying the reasoning mechanism. Furthermore, it should provide better understanding of the relationships between artefacts in different phases of the software development process via sophisticated traceability techniques. Finally, ODSD should lead to improved understanding of the software development process itself by applying ontology-driven guidance. Within the project, we use the ADOxx metamodelling platform as a modelling and a metamodelling environment which should offer advanced ontology-aware model repository and guidance capability.

We are currently working on the first prototype evaluation of this architecture by implementing the integrated metamodel and by integrating various ontology-aware and model-aware mechanisms into the platform.

Acknowledgements

This research has been co-funded by the European Commission and by the Swiss Federal Office for Education and Science within the 7th Framework Programme project MOST N° 216691, http://most-project.eu.

References

1. Karagiannis, D., Kühn, H.: Metamodelling platforms, Invited paper.: In Proceedings of the Third International Conference EC-Web 2002 – Dexa 2002. Springer-Verlag, Berlin, Heidelberg (2002)
2. Zivkovic, S., Murzek, M., and Kuehn, H.: Bringing Ontology Awareness into Model Driven Engineering Platforms. In Proceedings of the 1st International Workshop on Transforming and Weaving Ontologies in Model Driven Engineering (TWOMDE 2008), pages 47-54. http://sunsite.informatik.rwthaachen.de/Publications/CEUR-WS/Vol-395/. (2008)
3. Meta Object Facility (MOF) Specification-V2.0. http://www.omg.org/docs/ptc/03-10-04.pdf
4. Steinberg, D., Budinsky, F., Paternostro, M., and Merks, E..: EMF: Eclipse Modeling Framework 2nd Edition, Addison-Wesley Professional, Eclipse Series. (2008)
5. Jouault, F. and Bezivin, J.. KM3: A DSL for Metamodel Specification: In Proceedings of 8th IFIP International Conference on Formal Methods for Open Object-Based Distributed Systems, pages 171-185. LNCS. (2006)
6. ATLAS. Atlas Megamodel Management (AM3). http://www.eclipse.org/gmt/am3/ (2006)

7. Ontology Definition Metamodel (ODM) – Version 1.0, Beta 3, http://www.omg.org/spec/ODM/1.0/Beta3/PDF/ (2008)
8. Silva Parreiras, F., Staab, S., Winter, A.: TwoUse: Integrating UML Models and OWL Ontologies. Universität Koblenz-Landau, Fachbereich Informatik. Nr. 16/2007. Arbeitsberichte aus dem Fachbereich Informatik. Koblenz (2007)
9. Kullbach, B., Winter, A.: Querying as an Enabling Technology in Software Reengineering. In: Verhoef, C. (Hrsg.); Nesi, P. (Hrsg.): Proc. of the 3rd Euromicro Conference on Software Maintenance & Reengineering. Los Alamitos: IEEE Computer Society, (1999), S. 42–50
10. Object Constraint Language (OCL) Specification, Version 2.0, http://www.omg.org/cgi-bin/doc?formal/2006-05-01 (2006)
11. SPARQL Query Language for RDF, W3C Reccomendation, http://www.w3.org/TR/rdf-sparql-query/ (2008)
12. Silva Parreiras. F., Staab, S., and Winter, A.: On marrying ontological and metamodeling technical spaces. In Proceedings of the 6th joint meeting of the European Software Engineering Conference and the ACM SIGSOFT International Symposium on Foundations of Software Engineering, 2007, Dubrovnik, Croatia, September 3-7. (2007)
13. Lopes, D., Hammoudi, S., Bezivin, J., and Jouault, F.: Mapping specification in MDA: From theory to practice. In: Konstantas, D., Bourrieres, J.-P., Leonard, M., Boudjlida, N. (Eds). Interoperability of Enterprise Software and Applications - INTEROP-ESA, p.253-264. (2006)
14. Zivkovic, S., Kuehn, H., and Karagiannis, D.: Facilitate modelling using method integration: An approach using mappings and integration rules. In: Oesterle, H.; Schelp, J.; Winter, R. (Eds.): Proceedings of the 15th European Conference on Information Systems (ECIS2007) - "Relevant rigour - Rigorous relevance", St.Gallen, Switzerland, (2007) pages 2038-2050. http://is2.lse.ac.uk/asp/aspecis/20070196.pdf
15. Atlas Transformation Language (ATL). http://www.eclipse.org/m2m/atl/ (2008)
16. Roser, S. and Bauer, B.: An approach to automatically generated model transformations using ontology engineering space. In 2nd International Workshop on Semantic Web Enabled Software Engineering (SWESE), (2006)
17. Bernstein, P. A.: Applying model management to classical meta data problems. In Proceedings of CIDR Conference 2003, Asilomar, CA, pages 209-220. (2003)
18. Schwarz, H., Ebert, J., Riediger, V., Winter, A.: Towards Querying of Traceability Information in the Context of Software Evolution. In: GI Proceedings Bd. 126, (2008)
19. KAON Tool Suite Home Page, http://kaon.semanticweb.org/ (2009)