

Semantic-enabled software engineering and development

Bernhard Bauer, Stephan Roser

Angaben zur Veröffentlichung / Publication details:

Bauer, Bernhard, and Stephan Roser. 2006. "Semantic-enabled software engineering and development." In *Informatik 2006: Informatik für Menschen; Beiträge der 36. Jahrestagung der Gesellschaft für Informatik e.V., Band 2*, edited by Christian Hochberger and Rüdiger Liskowsky, 293–96. Bonn: Gesellschaft für Informatik. <https://dl.gi.de/handle/20.500.12116/23528>.

Semantic-enabled Software Engineering and Development

Bernhard Bauer, Stephan Roser

Programming of Distributed Systems, University of Augsburg, 86135 Augsburg
[bauer|roser]@informatik.uni-augsburg.de

Abstract: Like in automotive and other production oriented domains industrialization was the key issue for reducing production costs and being competitive. However software is still developed mainly from scratch in a labour intensive way. In order to be competitive with e.g. low-wage countries the software development process has to be industrialized and automated. The model-driven software development approach as well as Semantic Web technologies can help to support such software industrialization. Therefore in this position paper we will set up a vision and give concrete examples how semantic enabled software development can look like and give a first architecture of a development environment supporting this vision.

1 Introduction

In the last years software development has been subject to fundamental changes. The key for success is now named *software industrialization*. In contrast to software development by hand, industrial production processes are considered as an alternative paradigm for solving the problems of today's software industry. With OMG's model-driven architecture (MDATM [MDA]), i.e. automated model transformations and code generation [Fr03], a first step towards software industrialization is taken. Moreover production lines support this idea for specific application domains. The key issues in industrial production environments are reuse of standardized components and processes, global cooperation and little vertical integration. To overcome the heterogeneity and making things working smoothly together Semantic Web technologies and ontologies (see e.g. [Bo02], [Ob05]) can be applied. In our opinion the combination of model-driven software development and ontological concepts from the Semantic Web area can therefore support the industrialization of software development. The goal of the presented work is to combine the MDS approach with ontological concepts through the use of ontologies in the development process, semantic-annotated models, semantic-enabled model transformations, attribution of meta-models as well as semantic-enabled methodologies for customizing and document generation. After discussing shortcomings of today's software development in section 2, in section 3 we present how semantic-enabled software engineering and development can solve the shortcomings. In our work we focus on build-time of a systems, however as we will show in the outlook the run-time environment is also an area for further research.

2 Shortcomings of Today's Software Development

Methodologies: Methodologies, like the Rational Unified Process, Agile Development methodologies and V-model XT are usually customized to the needs of an enterprise.

Though specific guidelines for using the methodology and document templates are set-up, these aspects are not covered in the necessary detail by current approaches. **Interoperability issues:** Today interoperability issues are mainly taken into consideration at run-time, i.e. XML-standards are defined (like ebXML or RosettaNet). But at build-time model-driven integration of systems is widely neglected, i.e. taking the different models into consideration to get a consistent “big picture” of the different systems. **Search and composition of components:** Nowadays the (re)use of components of the shelf and services are done by a manual search and looking up textual descriptions. The few solutions supporting developers in searching efficiently for such components in distributed environments operate essentially at code level. The lack of semantic rich descriptions of e.g. functionality and as well as non-functional properties like quality attributes, which can be used throughout the development process, also prohibits from supporting developers by semi-automatic orchestration of components to value-added services. **Composition of models:** In composing models current approaches lack two dimensions, namely the automatic composition of models, e.g. in the context of aspect oriented modelling, and elaborated consistency checks to ensure unambiguousness of the merged models’ elements, i.e. checks on e.g. UML models which cannot be performed on the basis of languages like OCL. Though formal approaches exist, they are not sufficiently integrated in real-world development environments. **Syntactic-based model transformations:** Model transformations are specified on meta-models. Since their descriptions do not take into account meta-models’ semantics in a machine-understandable way, there are shortcomings in the reuse of model transformations and adjusting model transformations to new versions of meta-models.

3 Semantic-enabled Software Engineering and Development

With semantic-enabled Software Engineering and Development (seSED) we employ the technological spaces (TS) approach presented by Kurtev et al. [KBA02] to improve efficiency of work by using the best possibilities of different technologies, in our case MDA TS and Ontology TS.

UML is the standard for developing industrial software projects and a key in the MDA approach. For a selected set UML diagrams we show exemplarily how ontologies can be applied to add more semantics to UML diagrams. **Class diagrams** can be used for visualizing and developing domain and application specific ontologies in MDA TS-centric software development. By applying these ontologies during modelling the requirement specification errors or ambiguities can be found and retracted already in early phases of the development process automatically. Also the integration of existing domain standards will be facilitated. During the analysis phase **use case diagrams** are often applied for modelling. Actors define roles interacting with the system. The use of ontologies would add more semantics, especially in describing the interrelationships between the actors. One could also imagine annotating «includes»- and «extends»-relationships enabling an automatic checking. Moreover requirements in the terms of non-functional properties like performance, modifiability or security can be specified using ontologies. Ontologies representing domain or application data models can be used

in various UML diagrams. In **sequence diagrams** for example, parameters' types and invariants could be defined, taking into consideration the data model of those ontologies. The new UML2 **activity diagram** is also applied to business process modelling. In [LB06] we have shown how IOPEs (input, output, pre-conditions and effects) of processes can be annotated by ontology concepts, facilitating the automated analysis and synthesis of business processes. Using Semantic Web's logics to fully describe the IOPEs can offer increased expressivity and reasoning capabilities. **Composite structure diagrams** can be combined with ontological API and functionality descriptions, like it is done in the Semantic Web Service area with OWL-S and WSDL-S. Integrating those semantic descriptions of components in models will allow automatic search and composition of components. The development environment will provide some kind of directory service, supporting search for components during the specification of systems, and matchmaking mechanisms for checking, whether existing components can solve a certain problem. Though approaches try to close the gap between the world of MDA and the world of Semantic Web technology, a first result based on the work of Cranefield [CP99] is now standardized by the OMG through the ODM standard or others like [Knu06]. Having a look at the above mentioned shortcomings semantic-enabled software engineering can help to overcome them: **Methodologies:** A *software methodology* is typically characterized by a modelling language and a software process. In particular, the software process defines phases for process and project management as well as quality assurance. Each activity results in one or more deliverables – such as specification documents, analysis models, code, testing reports, etc. – serving as input for subsequent activities. Those processes can be described by models (see e.g. SPEM [SPEM]). Ontological annotations of meta-models can be used for automatic document generation during the development process. Modelling guidelines, realized as an attribution of meta-models by ontologies, for customized methodologies can be enforced and checked, while help is given when modelling guidelines are violated. **Interoperability issues:** Interoperability at build-time will be achieved by model-driven integration of systems. Models can be exchanged between various enterprises despite of different modelling styles, and models semantics is specified by ontologies in a machine-understandable way. Enabling collaborative modelling amongst various enterprises will also facilitate semi-automatic composition of processes. By using ontologies for describing meta-models and expressing interrelationships between meta-models, deduction on those ontologies can be used to compare meta-models and to automatically derive mappings from one meta-model to another. Even if the meta-models used in diverse systems are completely different, interoperability will be achieved much easier at ontological than at syntactical level. **Search and composition of components:** Semantic descriptions of e.g. existing components or services will be stored in knowledge bases where ontological reasoning and matchmaking mechanisms can be applied. API descriptions could similar to WSDL-S enable automatic consistency checks on the components and services, whilst the model elements of such components are used in the system model. An ontological API description can also be used for automatic search of missing components during system development and offers the possibility for automatic composition of existing components and services through reasoning. **Composition of models:** Composition can be applied to models e.g. for merging models of sub-systems, composing different diagrams to e.g. interaction overview diagrams or generating platform specific models out of platform independent and platform models. Ontologically annotated meta-models

will extend capabilities of aspect oriented modelling approaches. Moreover sophisticated consistency checks can be conducted when merging models. **Syntactic meta-model-based transformations:** Taking into account meta-models semantics for model transformations like described in [RB05], it will be possible to automatically generate mappings between meta-models of different tools and enterprises. Despite different model representation formats and modelling styles, model transformations, which are e.g. developed for a certain architectural style or methodology, can be reused in and adjusted to various development scenarios.

4 Related Work and Outlook

It has already been shown that Semantic Web technologies can successfully be applied at run-time. For example Paolucci et al. [PS+05] have developed a broker for OWL-S Web Services. Oberle et al. presented the vision of a Semantic Middleware for Web Application Development in [Ob05], which aims to complement MDA with an ontology infrastructure enabling reasoning and querying at runtime. In this work we presented a vision of how to introduce Semantic Web technology in software engineering and development, i.e. in the model-driven software development process. Starting from shortcomings of currently used technology we explained, how introducing ontologies in modelling at build-time of software systems improves those shortcomings. Finally we outlined how the gap between MDA and Ontology TS can be closed. Up to now we had no deeper look at the run-time environment. Here also we see a potential coming from a semantic-enabled software engineering and development. Moreover as stated in the introduction, we see a high potential that ontological concepts – as pointed out in this paper – can improve the industrialization process of software development.

References

- [Bo02] S. Borgo et al.: *OntologyRoadMap*. WonderWeb Deliverable D15, 2002.
- [CP99] S. Cranefield, M. Purvis: *UML as an Ontology Modelling Language*, Workshop on Intelligent Information Integration held during 16th IJCAI, Stockholm, Sweden, 1999.
- [Fr03] D. S. Frankel: *Model Driven Architecture – Applying MDA™ to Enterprise Computing*, Wiley, 2003.
- [Knu06] H. Knublauch *Ramblings on Agile Methodologies and Ontology-Driven Software Development*", SWESE 2006
- [KBA02] I. Kurtev, J. Bézivin, M. Aksit: *Technological Spaces: An Initial Appraisal*, Int. Federated Conference (DOA, ODBASE, CoopIS), Industrial Track, Irvine, 2002.
- [LB06] F. Lautenbacher, B. Bauer: *Semantic Reference and Business Process Modeling Enables an Automatic Synthesis*, Workshop on Semantics for Business Process Management, Budva, Montenegro, 2006.
- [Ob05] D. Oberle: *Semantic Management of Middleware*, Springer, 2005.
- [PS+05] M. Paolucci, J. Soudry, N. Srinivasan, K. Sycara, *A Broker for OWL-S Web Services, in Extending Web Services Technologies: the use of Multi-Agent Approaches*, 2005
- [RB05] S. Roser, B. Bauer: *Ontology-based Model Transformation*, Proceedings of Satellite Events at the MoDELS 2005 Conference, Montego Bay, Jamaica, October 2-7, 2005.