Essentials of the 4th UML/MoDELS Workshop in Software Model Engineering (WiSME'2005)

Krzysztof Czarnecki$^{(A)}$, Jean-Marie Favre$^{(B)}$, Martin Gogolla$^{(C)}$, and Tom Mens$^{(D)}$ (Eds.)

University of Waterloo, Canada$^{(A)}$
University of Grenoble, France$^{(C)}$
University of Bremen, Germany$^{(C)}$
University of Mons-Hainaut, Belgium$^{(D)}$

Abstract. Model-Driven Engineering is a form of generative engineering, by which all or at least central parts of a software application are generated from models. Model Driven Engineering should be seen as an integrative approach combining existing software engineering techniques (e.g., testing and refinement) and technical spaces (e.g., 'ModelWare', 'XmlWare') that have usually been studied in separation. The goal of the workshop is to improve common understanding of these techniques across technical spaces and create bridges and increase the synergies among the spaces. This year's WiSME workshop will concentrate on two complementing themes: Bridging Technical Spaces and Model-Driven Evolution. This paper reports on a workshop held at the 8th UML/MoDELS conference. It describes motivation and aims, organisational issues, and abstracts of the accepted papers.

1 Motivation and Aims

The OMG initiative Model Driven Architecture (MDA) attempts to separate business functionality specification from the implementations of that functionality on specific technology platforms. This approach is intended to play a key role in the fields of information system and software engineering. MDA is supposed to provide a basic technical framework for information integration and tool interoperability based on the separation of platform specific models (PSMs) from platform independent models (PIMs). Models of coarse granularity and high abstraction will represent the various functional and non-functional aspects of computer systems. In the long term, there will be well-defined operations on models, implemented by well-defined commercial tools, that will allow us to build, transform, merge, verify, and evolve models. Key standards in the MDA are based on OMG recommendations such as UML, MOF, XMI, CWM, QVT.

MDA can be considered an implementation of a more general trend that has been gathering momentum in recent years called Model Driven Engineering

(MDE). Model-Driven Engineering is a form of generative engineering, by which all or at least central parts of a software application is generated from models. The basic ideas of this approach are close to those of other disciplines of software engineering, such as Generative Programming, Domain Specific Languages (DSLs), Model Integrated Computing (MIC), and Software Factories. In this new perspective, models, metamodels, and transformations take a primary position amongst the artifacts of development. MDE aims at making models the primary driving assets in all aspects of software engineering, including system design, platform and language definition, definition of mappings among artifacts, but also design data integration, design analysis, tool specification, and product family development. Model Driven Engineering should be seen as an integrative approach combining existing techniques and technical spaces that have usually been studied in separation. The goal of the workshop is to improve common understanding of these techniques, to increase the benefits of combining approaches such as grammars, schemas, or metamodels, and to create bridges between such various technical spaces.

The workshop will concentrate this year on two complementing themes: Bridging Technical Spaces and Model-Driven Evolution. When dealing with models residing in different technical spaces (e.g., UML/MOF and XML), one encounters the problem of co-evolution or model synchronisation across technical spaces. For example, whenever a change is made to a UML model, which is also represented in another format (e.g., XMI), both models need to be synchronised. The fact that this synchronisation occurs in different technical spaces makes the problem difficult and interesting. Another topic within these themes is the problem of language evolution: How can one deal with the fact that the languages in which models are developed (e.g., UML, MOF, XML, XMI) evolve at a rapid pace? The descriptions developed in these languages need to co-evolve as well. This form co-evolution has been known in the technology space of databases as schema evolution.

The non-exhaustive list of topics included:

- Transformation systems for models; Types of transformations; Merging of a platform model and a business model
- Transformation frameworks; Transformation libraries; Transformations processes; Transformations as reuseable assets
- Operations on models: Merging, alignment, verification, validation, refactoring, refinement, abstraction, reduction, normalization, generalization
- Models, Schemas, Grammars, Meta-models and Ontologies; Reflection in meta-model architectures
- Meta-modeling frameworks and tool integration
- Modularity aspects for model organization: profiles, packages, name spaces, viewpoints, contexts; Issues of granularity for models
- Meta-data and repositories; Middleware-supported model engineering
- Requirements modeling; Know-how modeling; Architecture modeling; Service modeling; Rule modeling

– Co-evolution of and synchronization between models (including business models, requirements, analysis and design models, and source code)
– MDA and MDE industrial cases, best practices and empirical studies
– Co-evolution of meta-models and models, schema and data, meta-grammar and grammar
– Co-transformation (e.g., co-transforming database schemas and SQL expressions and applications)
– Megamodeling (i.e., defining basic MDE concepts and structure of MDE component repository structures)
– Technology spaces (e.g., MDA, GrammarWare, XmlWare) and bridges between them (e.g., the relationships between 'Grammar' and 'Class Diagram')

2 Organisational Issues

This workshop is the fourth in a series of workshops started at UML'2002 [1] and continued at UML'2003 [2] and UML'2004 [3]. The paper selection process was carefully supported by an international programme committee and additional referees. All accepted papers can be found in [4]. Revised versions of the papers will be published in Electronic Notes in Theoretical Computer Science (ENTCS), Elsevier.

Programme Committee

– Jean Bezivin, INRIA & University of Nantes, France
– Mireille Blay, University of Nice, France
– Krysztof Czarnecki, University of Waterloo, Canada
– Johannes Ernst, NetMesh, USA
– Jack Greenfield, Microsoft, USA
– Jean-Marie Favre, University of Grenoble, France
– Robert France, University of Boulder, USA
– Martin Gogolla, University of Bremen, Germany
– Pieter Van Gorp, University of Antwerp, Belgium
– Reiko Heckel, University of Manchester, Great Britain
– Gabor Karsai, Vanderbilt University, USA
– Ivan Kurtev, University of Twente, The Netherlands
– Ralf Laemmel, Microsoft, USA
– Steve Mellor, ProjTech, USA
– Tom Mens, University of Mons-Hainaut, Belgium
– Alfonso Pierantonio, University of L'Aquila, Italy
– Paul Sammut, Xactium, Great Britain
– Gabi Taentzer, Technical University of Berlin, Germany
– German Vega, University of Grenoble, France
– Jon Whittle, George Mason University, USA
– Andreas Winter, University of Koblenz, Germany

The workshop was structured into 4 thematic sessions:

- Bridging Approaches for Technical Spaces (Papers 3.1-3.3)
- Bridging Textware and Modelware (Papers 3.4-3.6)
- Modeling Approaches (Papers 3.7-3.9)
- Modeling Engineering Environments (Papers 3.10-3.11)

3 Presented Papers

3.1 Model Engineering Support for Tool Interoperability

Jean Bezivin, Hugo Bruneliere, Frederic Jouault, Ivan Kurtev

ATLAS group - INRIA - LINA, Universite de Nantes, France

Abstract: In this paper we want to show how MDE (Model Driven Engineering) approaches may help solving some practical engineering problems. Our view of MDE is not based on the usage of huge and rather monolithic modeling lan- guages like UML 2.0 but instead on small DSLs (Domain Specific Languages) defined by well focused metamodels. As a consequence we use a rather "agile" view of MDE where each tool is characterized by one of several metamodels and where the interoperability between tools is implemented by specific model transformation operations. We base our discussion on a practical illustrative problem of bug-tracking in a collaborative project involving different partners using tools at different maturity levels. We conclude by discussing the help that MDE approaches may bring to solving these kinds of situations.

3.2 A Metamodel Refinement Approach for Bridging Technical Spaces - A Case Study

A. Staikopoulos, B. Bordbar

School of Computer Science, University of Birmingham, UK

Abstract: To benefit from positive aspects of an existing diverse set of Technical Spaces, it is important to develop methods of automated transformation of models between such domains. Sometimes it is possible to describe Technical Spaces via metamodels. In such cases, the Model Driven Engineering and Architecture pose as a natural candidate for dealing with such transformations between Technical Spaces. This paper deals with the case where the metamodel of the source Technical Space is more complex than the metamodel of the destination. Thus, the gap between the two Technical Spaces is highly non-trivial. The method presented in this paper is based on successive metamodel refinements to bridge this gap. Finally, the method is applied to the transformation from Business Process Execution Language to Petri nets.

3.3 Bridging Persistency Impacts: Towards Automatic Integration of Domain-Data Layers

Mira Balaban and Lior Limonad

Computer Science Department, Ben-Gurion University of the Negev, ISRAEL

Abstract: Domain and data layers are two different technical spaces that need to co-exist and co-evolve in integrated complex systems. The combined operation of Domain and Data modules usually suffers from model mismatch problems, and raises problems of consistency. Existing methods and technologies offer partial solutions that do not support full automation of persistency decisions. In this paper we introduce a method for automatic construction of a Data Access Layer that bridges between a Domain and a Data layers. The method is based on analysis of class navigation patterns that involve persistency stereotyping in the domain layer. Such patterns are termed Data Access Patterns. For each pattern we suggest a data access layer implementation that can be automated, and leaves the domain layer classes intact. The combined analysis of all patterns provides a complete solution for bridging persistency impacts. The current implementation handles the static aspects, i.e., schema transformation.

3.4 Bridging Grammarware and Modelware

Manuel Wimmer, Gerhard Kramler

Business Informatics Group, Vienna University of Technology, Austria

Abstract: In Software Engineering many text-based languages and sup- porting tools are used, forming the grammarware technical space. Cur- rently model driven engineering is the new emerging paradigm for soft- ware engineering, which uses modelling languages and tools, forming the modelware technical space. Transition to the new technical space and interoperability between these two technical spaces is needed in

many development scenarios. Building a bridge between these two technical spaces is a tedious task, that has to be repeated for each language to be transformed. Therefore, we propose a generic bridge between grammarware and modelware technical spaces, that can generate a specfic bridge based on the EBNF of a given language semi-automatically. The generation comprises of two steps, (1) automatic generation of metamodel corresponding to the EBNF and (2) annotations to provide the additional semantics not captured by the EBNF. The generated bridge is capable of bi-directional transformations between sentences (programs) and corresponding models and can be used in re-engineering applications and for integration of text-based and model-based tools. The benefits of this approach are rapid development and correctness.

3.5 HUTN as a Bridge between ModelWare and GrammarWare: An Experience Report

Pierre-Alain Muller, Michel Hassenforder

IRISA/INRIA, Rennes, France

Abstract: In this paper we report on our experience using HUTN as a bridge between ModelWare and GrammarWare, to generate parsers and editors for DSLs defined under the shape of metamodels. We describe the problems that we have encountered with the ambiguities of the current HUTN specification and discuss how this specification may be fixed to be usable with grammar- driven tools.

3.6 Specifications for Mapping UML Models to XML Schemas

Krish Narayanan, Shreya Ramaswamy

Department of Computer Science, Eastern Michigan University, General Motors Corporation

Abstract: An imminent activity in software design is to map models, either from one level of detail to another or from different perspectives of an application. It is extremely important that these models are consistent with each other in the semantics they carry for a successful software implementation. With the introduction of a number of CASE tools, this activity has been partially automated with little or no human intervention. Yet, one of the major drawbacks of these tools is that the mapping process sometimes tends to loose the inherent semantics of an

application. Also, for some applications, there are no such commercially available tools. In this paper, we focus on XML applications that use a standard structure specified by their corresponding schemas. The output of the design activity for such applications is typically, a generic model that represents the schema in an abstract way. UML being the language of choice for modeling software systems, is often used in modeling XML applications. But, the mapping from UML models to XML schemas is far from perfect. We consider the intricacies involved in both the forward (UML to XML) and reverse (XML to UML) engineering processes and ensure that the application's structural semantics are maintained in the mapping process. In this paper, we define the complete specifications for forward engineering UML model constructs to their corresponding XML counter parts. We also share our experiences from a tool we have developed that automates the mapping process.

3.7 Navigating the MetaMuddle

Arnor Solberg, Robert France, Raghu Reddy

SINTEF/University of Oslo, Norway, Colorado State University, USA

Abstract: Developers of model transformations and other model-driven development (MDD) mechanisms that manipulate models often work at the metamodel level. The complexity of the UML metamodel can hinder the development of UML-based MDD technologies. In this paper, we identify some of the current barriers to understanding, using and evolving the UML 2.0 metamodel and present ideas on how to simplify these tasks using query based views and aspect oriented techniques.

3.8 A UML precise specification of design patterns using decoupling constraints

Samir Ammour, Mikal Ziane, Xavier Blanc and Salima Chantit

LIP6, Laboratoire d.Informatique de Paris 6, Paris, France

Abstract: UML collaboration templates do not capture the problem part of design patterns. It is then possible to apply a pattern solution in contexts that are not consistent with this solution. Moreover, when the context evolves, it must stay consistent with the patterns that were previously applied. Since design problems can be quite varied, we focused on one of the most frequent design goals of design patterns: avoiding

unnecessary coupling. In this paper, we adapt the notion of decoupling constraints to UML models and we express it using the Object Constraint Language. Adding decoupling constraints to collaboration templates allows checking if a context is consistent with the solution of a pattern and eases the adaption of this solution to an evolving context. These constraints must be checked before applying a pattern but also when the context evolves, to guarantee that the solution of the pattern is consistent and complete.

3.9 A Meta-model for the Problem Frames Approach

Maria Lencastre1, Juliane Boetlho1, Pedro Clericuzzi1, Joao Araojo

Pernambuco University, Brazil, Universidade Nova de Lisboa, Portugal

Abstract: Michael Jackson's Problem Frames are a well-known software engi- neering approach for requirements analysis and problem domain specification. This paper describes this technique through the definition of a meta-model, whose aim is to clarify the involved concepts to help the devel- opment and implementation of tool support and also to compare, transform, merge different requirement techniques.

3.10 A Presentation Framework for Metamodeling Environments

Gergely Mezei, Tihamor Levendovszky, Hassan Charaf

Budapest University of Technology and Economics, Budapest, Hungary

Abstract: Although the flexibility provided by metamodeling systems is proven to be enough in practical applications, the presentation is not supported to the same extent. The goal of this paper is to present a presentation framework which exploits the advantages of the underlying metamodeling system and provides enough flexibility to present the concrete syntax of feature models, UML 2.0 diagrams, and resource diagram for mobile telephones. The static structure as well as the behavioral aspect are described to illustrate the solutions and the design decisions related to the framework.

3.11 Some Lessons Learnt in the Building of a Model Engineering Platform

Jean Bezivin

Atlas Group, INRIA and LINA, University of Nantes, France

Abstract: As we are currently improving AMMA (ATLAS Model Management Architecture), a second generation model engineering platform installed on top of the Eclipse Modeling Framework (EMF), we borrow inspiration from a previous work undertaken fifteen years ago at the University of Nantes. This initial model engineering platform named sNets (Semantic Networks) included several functional blocks like the sMachine, the sBrowser, the sQuery, the sAction system, etc. Several parts of these tools are still in use today. At a time when we are defining the main architectural style of the new platform, it may be helpful to come back on the initial learning of this project. This paper describes the sNets project and summarizes some lessons learnt in the course of the design and utilization of this first generation modelling framework.

References

1. Jean Bezivin, Robert France: Proc. 1st UML Workshop in Software Model Engineering (WiSME'2002). www.metamodel.com/wisme-2002.
2. Jean Bezivin, Martin Gogolla: Proc. 2nd UML Workshop in Software Model Engineering (WiSME'2003). www.metamodel.com/wisme-2003.
3. Martin Gogolla, Paul Sammut, Jon Whittle: Proc. 3rd UML Workshop in Software Model Engineering (WiSME'2004). www.metamodel.com/wisme-2004.
4. Krzysztof Czarnecki, Jean-Marie Favre, Martin Gogolla, Tom Mens: Proc. 4th UML Workshop in Software Model Engineering (WiSME'2005). www.metamodel.com/wisme-2005.