Essentials of the 3rd UML Workshop in Software Model Engineering (WiSME'2004)

Martin Gogolla[(A)], Paul Sammut[(B)], and Jon Whittle[(C)] (Eds.)

University of Bremen, Germany[(A)]
Xactium, Great Britain[(B)]
QSS/NASA Ames, USA[(C)]

Abstract. This paper reports on a workshop held at the 7th UML conference. It describes motivation and aims, organisational issues, abstracts of the accepted papers, and questions raised during discussion.

1 Motivation and Aims

Model Driven Architecture (MDA) is an OMG initiative that attempts to separate business functionality specification from the implementation of that functionality on specific middleware technological platforms (e.g., CORBA, C#/DotNet, Java/EJB, XML/SOAP). 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 tools interoperation based on the separation of platform specific models (PSMs) from platform independent models (PIMs). Models of low 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, implemented by commercial tools that will allow to build, transform, merge or verify these different models. Key standards in the MDA will be based on OMG recommendations like UML, MOF, XMI, CWM, QVT.

In fact, MDA can be considered an implementation of a more general trend that has been gathering momentum in recent years called Model Driven Development (MDD). This aims to make models the primary driving assets in all aspects of software development, including system design, platform and language definition and mappings as in MDA, but also design data integration, design analysis, tool specification and product family development.

The stage is set but the effort to move from the present situation to the idyllic automatic generation of executable models for various platforms and other applications of MDD remains huge. We need to mobilize the creative energies of a very broad category of contributors, from tool builders to theoretical specialists in fields like language compilers, graph rewriting, model checking, metamodelling, and ontology engineering. We need to bring together young researchers planning to invest in this emerging new area as well as more experienced professional having previous experience in areas related to automatic code generation, transformational and generative approaches or model checking

2 Organisational Issues

This workshop is the third in a series of workshops started at UML'2002 [1] and continued at UML'2003 [2]. The paper selection process was carefully supported by an international programme committee and additional referees. All accepted papers can be found in [3].

Programme Committee

- Jean Bezivin, University of Nantes, France
- Krysztof Czarnecki, University of Waterloo, Canada
- Phillipe Desfray, Softteam, USA
- Johannes Ernst, NetMesh, USA
- Jacky Estublier, University of Grenoble, France
- Andy Evans, Xactium, Great Britain
- Martin Gogolla, University of Bremen, Germany
- Stuart Kent, Microsoft Research, Great Britain
- Steve Mellor, ProjTech, USA
- Paul Sammut, Xactium, Great Britain
- Laurie Tratt, Kings College London, Great Britain
- Jos Warmer, De Nederlandsche Bank, Netherlands
- Jon Whittle, QSS/NASA Ames, USA
- James Willans, Xactium, Great Britain

The workshop was structured into 4 thematic sessions:

- Principles, Theories and Methodologies (Papers 3.1-3.3)
- Tools and Industrial Experience (Papers 3.4-3.6)
- Model Testing and Compliance (Papers 3.7-3.9)
- Applying MDA to Ontologies (Papers 3.10-3.11)

3 Presented Papers

3.1 Towards a Basic Theory to Model Driven Engineering

Author: Jean-Marie Favre

Abstract: What is a model? What is a metamodel? What is a language? What is a transformation? How are these concepts related? It is striking to see that, though MDE is supposed to be about precise modelling, MDE core concepts are usually described in natural language or at best, using sketchy a-la UML diagrams. These diagrams are very often inconsistent and too vague to reason about. Most of the time, they are neither validated, not even used by their authors. When precise descriptions are provided, it is only to describe a specific technology. But since Model Driven Engineering is a about supporting multiple Technological Spaces (TS), the concepts of model, metamodel, and transformation should be declined, not only in the MDA TS, but also in the Grammarware TS, Documentware TS, Dataware TS, etc. To cope with these problems, we decided to start research on defining a megamodel of MDE. This paper shown how the set theory and language theory could help in understanding MDE concepts and their relationships. The megamodel could also be seen as a first version of a rudimentary theory for reasoning about MDE concepts.

3.2 A Tool Architecture Supporting Change Propagation

Authors: Marcus Alanen, Ivan Porres

This article discusses a tool architecture that supports the construction of a Model-Driven Development (MDD) tool as a collection of loosely coupled components. Each component performs a highly specialized task under the coordination of an active model repository. The key issue in this architecture is to decide how to notify changes produced by a component to the other components and when.

3.3 Are Models the DNA of Software Construction?

Authors: Friedrich Steimann, Thomas Kühne

Abstract: MDA is advocated as the next step in software construction. It builds on the availability of a powerful modelling language and model compilers that translate models into executable code. In the following Dr. Con-known as a harsh critic of UML and as a sceptic of the feasibility of MDA- and Dr. Pro-a believer in the MDA vision-are discussing whether MDA is fundamentally flawed from the beginning or represents the most promising new development paradigm we work on today.

3.4 A Case Study on a Transformation Focused Industrial MDA Realization

Authors: Miroslaw Staron, Ludwik Kuzniarz, Ludwik Wallin

Abstract: Model Driven Architecture (MDA) is a proposal to the realization of a vision of model driven software development defined by the Object Management Group (OMG). Its adoption, nevertheless, varies in different endeavors at different companies. The presented industrial case study performed at Volvo Information Technology (Volvo IT) contributes with the evaluation of how the studied team values profiles, constraints, transformations and other elements used in their realization of MDA. In addition it provides an insight into the practical MDA based framework development process followed at that company. The findings of the study may provide a tangible basis to the forthcoming endeavors of similar kind together with the experiences (the process) of the MDA realization, which are presented in this paper.

3.5 The TopModL Initiative

Authors: Pierre-Alain Muller, Cedric Dumoulin, Frederic Fondement, Michel Hassenforder

Abstract: We believe that there is a very strong need for an environment to support research and experiments on model-driven engineering. Therefore we have started the TopModL project, an open-source initiative, with the goal of building a development community to provide an executable environment for quick and easy experimentation, a set of source files and a compilation tool chain, and a web portal to share artefacts developed by the community. The aim of TopModL is to help the model-engineering research community by providing the quickest path between a research idea and a running prototype. In addition, we also want to identify all the possible contributions, understand how to make it easy to integrate existing components, while maintaining architectural integrity. At the time of writing we have almost completed the bootstrap phase (known as Blackhole), which means that we can model TopModL and generate TopModL with TopModL. Beyond this first phase, it is now of paramount importance to gather the best possible description of the requirements of the community involved in model-driven engineering to further develop TopModL, and also to make sure that we are able to reuse or federate existing efforts or goodwill. This paper is more intended to set up a basis for a constructive discussion than to offer definitive answers and closed solutions.

3.6 Traceability Across Refinement Steps in UML Modeling

Authors: Claudia Pons, Ralf-Detlef Kutsche

Abstract: Documenting the refinement relationship between layers allows developers to verify whether the code meets its specification or not, trace the impact of changes in the business goals and execute test assertions written in terms of abstract models vocabulary by translating them to the concrete models vocabulary. Refinement has been studied in many formal notations such as Z and B and in different contexts, but there is still a lack of formal definitions of refinement in semi-formal languages, such as the UML. The contribution of this article is to clarify the abstraction/refinement relationship between UML models, providing basis for tools supporting the refinement driven modeling process. We formally describe a number of refinement patterns and present PAMPERO, a tool integrated in the Eclipse environment, based on the formal definition of model refinement.

3.7 Specification-Driven Development of an Executable Metamodel in Eiffel

Authors: Richard Paige, Phillip J. Brooke, Jonathan S. Ostroff

Abstract: Metamodels precisely define the constructs and underlying well-formedness rules for modelling languages. They are vital for tool vendors, who aim to provide support so that concrete models can be checked formally and automatically against a metamodel for conformance. This paper describes how an executable metamodel - which supports fully automated conformance checking - was developed using a model-driven extension of test-driven development. The advantages and disadvantages of this approach to building metamodels are discussed.

3.8 Challenges and Possible Solutions to Model Compliance in the Context of Model Driven Development

Authors: Juan Pablo Zamora Zapata, Francis Bordeleau, Jean-Pierre Corriveau, Toby McClean

Abstract: Specifications define requirements. Numerous authors address the validation of implementations against such requirements. But there generally exists a large semantic gap between implementations and specifications, thus significantly limiting such validation approaches. Model driven development promotes the creation of a series of models to go from specification to implementation. Consequently, compliance can now be tackled between such models. In this paper, we specifically explore the challenges faced in the task of verifying the compliance of the structure of a design model against a specification model.

3.9 Running and Debugging UML Models

Authors: Miguel Pinto Luz, Alberto Rodrigues da Silva

Abstract: Software development evolution is a history of permanent seeks for raising the abstraction level to new limits overcoming new frontiers. Executable UML (xUML) comes this way as the expectation to achieve the next level in abstraction, offering the capability of deploying a xUML model in a variety of software environments and platforms without any changes. This paper comes as

a first expedition inside xUML, exploring the main aspects of its specification including the action languages support and the fundamental MDA compliance. We also explore the model debugging capabilities as a premature means of conceptual fail discovery. In this paper is presented a new xUML tool called XIS-xModels that gives Microsoft Visio new capabilities of running and debugging xUML models. Keywords: UML, executable UML, Model Debugging, Action Language.

3.10 Approaching OWL and MDA Through Technological Spaces

Authors: Dragan Gasevic, Dragan Djuric, Vladan Devedzic, Violeta Damjanovic

Abstract: Web Ontology Language (OWL) and Model-Driven Architectures (MDA) are two technologies being developed in parallel, but by different communities. They have common points and issues and can be brought closer together. Many authors have so far stressed this problem and have proposed several solutions. The result of these efforts is the recent OMGs initiative for defining an ontology development platform. However, the problem of transformation between an ontology and MDA-based languages has been solved using rather partial and ad hoc solutions, most often by XSLT. In this paper we analyze OWL and MDA-compliant languages as separate technological spaces. In order to achieve a synergy between these technological spaces we define ontology languages in terms of MDA standards, recognize relations between OWL and MDA-based ontology languages, and propose mapping techniques. In order to illustrate the approach, we use an MDA-defined ontology architecture that includes ontology metamodel and ontology UML Profile. Based on this approach, we have implemented a transformation of the ontology UML Profile into OWL representation.

3.11 Augmenting Domain Specific UML Models with RDF

Authors: Jörn Guy Süß, Andreas Leicher

Abstract: Models are created and maintained in the context of their problem domain. Inclusion of domain rules and background knowledge by means of profiles can be complicated and demanding. This paper presents an alternative approach based on the Resource Description Framework, which attaches background knowledge and rules as present in ontologies at the fringes of the model, rather than to include them within the model. Logical reasoning and queries are thus possible in the context of ontologies and background imported from different sources into RDF. Further, the paper discusses representation of complete UML and MOF Models using RDF Schema.

4 Additional Accepted Papers

4.1 A Formal Model of Component Behaviour

Authors: Remi Bastide, Eric Barboni, Amelie Schyn

Abstract: This paper presents a component model inspired by the CORBA Component Model, and an associated formal notation based on Petri nets and dedicated to the modelling of concurrent and distributed components. The model is illustrated by a case study that illustrates its hierarchical features, and shows how the main features of components can be mapped to the constructs of the Petri net.

4.2 Modeling and Transforming the Behavioural Aspects of Web Services

Authors: Behzad Bordbar, Athanasios Staikopoulos

Abstract: This paper introduces the modeling, mapping and transformation of behavioural aspects of interacting Web services, within the context of MDA. There are certain systems, such as Web services, where the dynamic aspects are of high importance and need to be considered during the modeling and transformation process, so to create accurate representations in their target domains. To demonstrate the approach, a realistic example is presented, involving a number of Web services, participating in a business process expressed as choreography of exchanged messages.

4.3 Defining Model Driven Engineering Processes

Authors: Frederic Fondement, Raul Silaghi

Abstract: Software engineering techniques made it possible for developers to build larger, and more accurate, reliable, and maintainable software-intensive systems. This was essentially possible by introducing techniques for raising the level of abstraction for describing both the problem and its solution, and by clearly establishing a methodology to define the problem and how to move to its solution. Model Driven Engineering (MDE) targets precisely at organizing such levels of abstraction and methodologies. It encourages developers to use models to describe both the problem and its solution at different levels of abstraction, and provides a framework for methodologists to define what model to use at a given moment (i.e., at a given level of abstraction), and how to lower the level of abstraction by defining the relationship between the participating models. Such an MDE process is supposed to be defined by means of assets and methodologists have the duty to provide such assets. However, it is not yet clear

what exactly these assets are, despite the fact that techniques to express them have already been widely studied. This position paper addresses this issue by identifying some of the MDE assets that have to be provided, and shows how they should be defined in order to enable them to participate in different MDE process definitions.

4.4 Systematic Validation of Model Transformations

Author: Jochen M. Küster

Abstract: Like any piece of software, model transformations must be validated to ensure their usefulness for the intended application. Properties to be validated include syntactic correctness as well as general requirements such as termination and confluence (i.e., the existence of a unique result of the transformation for every valid input). This paper introduces the idea of systematic validation and then focusses on validation of syntactic correctness for rule-based model transformations.

4.5 Towards a Language for Querying OMG MOF-based Repository Systems

Authors: Ilia Petrov, Stefan Jablonski

Abstract: This paper introduces a SQL-aligned declarative query language called mSQL (meta-SQL) for querying OMG-MOF based repository systems. Querying repository systems may be related to querying multi-database systems having a powerful data dictionary. The problem of schematic heterogeneity in multi-database, which is elegantly solved through higher order queries and the advantages they bring to various filed have been discussed in. Systems of this kind find extensive application in data-intensive Web applications, information, application and heterogeneous data source integration. Some of the key features of mSQL are: support for higher order queries and schema independent querying, unified handling of repository data and metadata, quantification over repository model elements. Additional areas to which mSQL may be applied are: querying schematically disparate models e.g. abstract schemata of components; information (schema) discovery; generic browsing of complex data collections and scientific repositories.

4.6 Mapping UML Class Diagrams to Object-Oriented Logic Programs for Formal Model-Driven Development

Authors: Franklin Ramalho, Jacques Robin

Abstract: MODELOG aims at automatically mapping UML class, object, statechart, activity and collaboration diagrams adorned with Object-Constraint Language expressions to non-monotonic, dynamic, object-oriented logic programs in Transaction Frame Logic (TFL). Coupled with the Flora-2 inference engine for TFL, MODELOG will fill five gaps in the current UML-based infrastructure for the Common Warehouse Meta-model, Model-Driven Architecture and Semantic Web visions: (1) automated data transformation transactions specified using the Meta-Object Facility for data warehousing and mining, (2) automated UML model transformations for refinement and refactoring, (3) formal verification of UML models, (4) complete UML model compiling into running code and (5) deductive and abductive inference in intelligent agents leveraging UML semantic web ontologies. In this paper, we present the MODELOG mapping of UML class diagrams to structural TFL clauses.

4.7 Improving SoC Design Flow by Means of MDA and UML Profiles

Authors: Elvinia Riccobene, Alberto Rosti, Patrizia Scandurra

Abstract: We tackle the problem of improving the SoC (System on a Chip) design flow in order to provide a modeling framework which allows exchange, reuse and integration of IP (Intellectual Property) models. In this paper, we present a UML profile of the SystemC language exploiting the MDA capabilities of defining modeling languages, platform independent and reducible to platform dependent languages. Furthermore, we discuss the advantages of high-level modeling SoC components in the style of UML using the SystemC design primitives, rather than designing at a lower level by means of coding.

4.8 An Object Oriented Model Transformer Framework Based on Stereotypes

Authors: Weerasak Witthawaskul, Ralph Johnson

Abstract: MDA modelers, like programmers in general, will develop and reuse libraries. Some of these libraries will hide details of the platforms, so the mapping from a PIM to a PSM will have to transform libraries as well. Some libraries provide common object services while others provide domain specific functionalities. These libraries will not just be class libraries, but also profiles containing stereotypes. Mercator is an extensible tool for transforming a PIM that uses platform independent libraries to a PSM. It allows model compiler developers to specify stereotype-based transformation both for transforming new libraries and for transforming libraries to new platforms. Using the Mercator framework and platform independent libraries, it becomes possible to extend the tool to migrate from one technology to another by creating new mappings for the new technology and reusing the same PIM.

5 Questions Raised during Discussion

- What do notions like modeling-in-the-small and modeling-in-the-large refer to? What is the relationship between modeling-in-the-small and programming-in-the-large?
- Will the field 'Software Engineering' emerge sometime in the future to 'Model Engineering'? Is 'Software Model Engineering' something in between?
- Are 'Models' the basic building blocks for software construction? What is the difference between modelling, specifying and programming?
- What kind of languages do we need for modeling: One big language or many small languages? In case we have many languages, how do the different languages cooperate?
- A question coming up on many events discussing 'models': How do we integrate structural and behavioral models?
- What about the importance of the UML? Will other modeling languages like Microsoft's Whitehorse initiative decrease the influence of UML? On the other hand, will it strengthen the importance of modeling languages in general?
- What is the relationship between domain specific languages and profiles for modeling languages?
- What is beyond the model? ... the metamodel? ... the megamodel? ... the megametamodel? ... the metamegamodel?

Acknowledgement

The organizers of this 3rd workshop gratefully acknowledge the initiative of Jean Bezivin and Robert France in establishing this series of workshops.

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.