

## Workshop on Modeling in Software Engineering at ICSE 2009

Robert Baillargeon

Panasonic Automotive Systems

[rbaillargeon@acm.org](mailto:rbaillargeon@acm.org)

Robert France

Colorado State University

[france@cs.colostate.edu](mailto:france@cs.colostate.edu)

Steffen Zschaler

Lancaster University

[szschaler@acm.org](mailto:szschaler@acm.org)

Bernhard Rumpe

RWTH Aachen

[rumpe@se.rwth-aachen.de](mailto:rumpe@se.rwth-aachen.de)

Steven Völkel

RWTH Aachen

[voelkel@se-rwth.de](mailto:voelkel@se-rwth.de)

Geri Georg

Colorado State University

[georg@cs.colostate.edu](mailto:georg@cs.colostate.edu)

DOI: 10.1145/1543405.1543432

<http://doi.acm.org/10.1145/1543405.1543432>

### Abstract

The Modeling in Software Engineering (MiSE) workshop series provides a forum for discussing the challenges associated with modeling software and with incorporating modeling practices into the software development process. The main goal of the series is to further promote cross-fertilization between the modeling communities (e.g., MODELS) and software-engineering communities.

In particular, the workshop provides a medium to exchange innovative technical ideas and experiences related to modeling. The 2009 MiSE workshop provided a venue for presentation and discussion of eleven papers in the five areas of model evolution, domain specific languages, verification and validation, model transformation and state-of-the-art modeling usage in software development. These papers represent a 44% acceptance rate to the workshop. Three posters were also accepted. This report summarizes the discussions and conclusions of the workshop.

**Keywords:** MiSE workshop, software modeling

### Introduction

The 2009 MiSE workshop at ICSE represents the third in this series of workshops. It provides a venue for interactive presentation and discussion of ideas and experiences in the innovative uses of modeling in software engineering. The topics of interest in these workshops include technical challenges associated with modeling software, incorporating modeling practices into existing software development practices, and even the use of models during software execution. This year we received twenty-five submissions, and accepted eleven papers and

three posters. The papers were grouped into the following categories: Using models in software evolution, Domain-specific modeling languages, Using models in software system verification and validation, Model transformation, and State-of-the-art use of models.

A present-respond-discuss format was used to stimulate discussions during the workshop. Each presentation was followed by a reaction to the topics raised in the paper by one of the workshop PC members. This response was then followed by a short discussion on the presentation topic and reaction points. Poster presentations used a similar format, shortened to accommodate the topics.

The workshop also included discussion of a proposed model repository to be used by researchers and practitioners, and a final discussion bringing together the threads of the workshop into a summary and direction for future research on modeling in software engineering.

### Summary of Papers and Topics

The workshop papers are summarized according to the session in which they were presented.

#### *Using Models in Software Evolution*

There were two papers presented during this session. In the first, “Towards Engineered Architecture Evolution” [3], Chaki, et al. argue that close-ended architectural evolution, where the starting and ending design points are known, can be modeled as a sequence of steps. The authors present a framework to describe such evolution trajectories. The framework enables exploration of potential evolutionary paths and their evaluation.

“Relationship-Based Change Propagation: A Case Study”, by Chechik et al. [5], describes an automated algorithm that propagates changes between requirements and design models. Propagation is driven by defined relationships between these two types of models. Conditions are defined to allow syntactic and semantic relationship validity checking. The algorithm uses the relationships to minimize portions of the models that must be changed. The authors demonstrate their algorithm using a case study.

#### *Domain-Specific Languages*

This session contained two papers and the poster session. The first paper, “Raising the Level of Abstraction in the Development of GMF-based Graphical Model Editors”, by Kolovos, et al. [7], tackles the well-known complexity of creating a visual editor based on GMF. The paper describes a method that raises the level of abstraction a developer uses to specify the visual editor. Annotations to the meta-model are used to produce low-level GMF models. These models are used to create the visual editor via model transformations.

The second paper in this session “Tailoring a Model-Driven Quality-of-Service DSL for Various Stakeholders” [10], by Oberortner et al addresses the problem of service-oriented contract and agreement specification. These agreements are often specified in technical models or even the implementations of systems, which makes it difficult for non-technical stakeholders to provide input. The authors present a domain-specific language for specifying these contracts, including sub-languages tailored to specific stakeholder sets. The approach is demonstrated through

development of a quality-of-service DSL to specify contracts and agreements.

#### *Verification and Validation*

Three papers were presented and discussed in this session. Spanfelner et al. presented “Formal Specification of System Functions” [11], proposing an approach to provide informal, intuitive validation and efficient formal verification of system functions, specified as services. The approach uses algebraic techniques to specify relations among services, which were initially described as use cases. The common approach of defining use cases early in system development, then translating them into formats suitable for formal verification is inefficient compared to the algebraic technique.

In “Finding Inconsistency for UML-Based Composition at Program Level [4]”, Chavez and Shen describe a method to detect inconsistency between a class model that uses the concept of composition of constituent parts and its associated implementation. The method generates all valid object diagrams systematically, then applies an owner-object destructor, and validates that all external links to the owned objects are also destroyed. The approach prunes the large search space for efficiency.

Herrmannsdoerfer et al. describe a simulator for the Component Language in “Model-Level Simulation for COLA” [6]. COLA is a language used in avionic and automotive embedded systems. The simulator guarantees the same behavior as specified in associated models. The architecture and capabilities of the simulator are described using a case study.

#### *Model Transformation*

Acher et al. present an approach to generate an application from a specification, taking advantage of variations in both the specification and the component implementations. Two feature models are used, one for the specification and one for the components. Variability and constraint modeling then drive transformations. Their paper, “Tackling High Variability in Video Surveillance Systems through a Model Transformation Approach” [1], describes the method in the context of video surveillance applications.

The second paper in this session was “Model Transformation of Dependability-Focused Requirements Models” [8], by Mustafiz et al. This work extends the process of use case elicitation and specification to model exception behavior of a system. Well-formed use cases are then transformed into activity diagrams using transformation rules. Partial or degraded outcomes and exception-handling activities are documented with stereotypes.

#### *State-of-the-Art Model Usage*

There were two papers in this session. In the first, “Non-Functional Requirements Analysis Modeling for Software Product Lines” [9], Nguyen surveys existing techniques to design and analyze non-functional requirements in the context of software product line commonality and variability. An automated methodology, called Product Line UML-Based Software Engineering, is introduced to deal with the problem.

The second paper in this session was “On the Use of Software Models during Software Execution” by Bencomo [2]. This paper describes software systems that must run and evolve, with little human intervention. These systems require first-class

representations of themselves to support change. Different types of changes and analysis dictate using different types of models, and in all cases reflection is needed to create and maintain the models. Bencomo summarizes on-going research to support runtime models, and presents related issues and questions.

### **Workshop Discussion Summary**

The workshop discussion centered around two areas: software development modeling concepts that are widely understood and accepted by practitioners and aspects of software modeling that are still active areas of research and discussion.

#### *Current Understanding of Modeling*

##### **Theory of Modeling**

The discussion noted that while abstraction is necessary to create models, every diagrammatic description of software is not necessarily an abstraction over code level descriptions. Rather, there is a need for some “distance” between the abstractions present in a model and the concrete implementation it represents. Abstractions have specific meaning for their creator, which, if not properly communicated in models, may not coincide with the meaning gleaned by a reader. This results in problems when the model is used as a communication medium. An abstraction may also be described from many viewpoints, and thus there may be multiple valid models that describe the same abstractions.

##### **Application of Modeling**

There are several paradigms that are generally accepted for modeling various software applications, for example, relational, state-based, and object-oriented paradigms. Each paradigm uses particular notations and concepts. Where agreement and common use has occurred, industrial/organizational standards are possible. Such standards are useful in that they provide a foundation for the interchange of model information across automations. Automations, in the form of tools, also provide lower entry points to model usage and at the same time provide more robust environments in which to use modeling techniques.

##### **Practice of Modeling**

Once modeling has been put into practice, several other observations are apparent. First, workshop participants noted that unchanging models are not particularly valuable, since the context in which they are used is constantly changing. Thus, the models must be manipulated to meet new needs. Secondly, more and more users have perceived the value of domain specialization. Practitioners are attempting to leverage models by using their own domain-specific languages for model development and use. Finally, participants noted the use of models to abstract variations in domain applications. Applications must often deal with concerns that have pervasive implications, and models, in the form of aspects, can be used to specify the cross-cutting elements that address these concerns.

#### *On-going Areas of Research and Discussion*

##### **Theory of Modeling**

Abstraction continues to be an on-going issue, in particular the question of how to find the proper level of abstraction of any given application of modeling. The purpose of an abstraction is clearly an important factor, but the drivers for deciding the appropriate level of abstraction are still unclear in many cases. This has led to “abstraction” being viewed as a skill that cannot be taught other

than by a process of trial and error through practical application, which appears unsatisfactory for an engineering discipline. In addition, the process of modeling—the order in which activities occur and the relationship between the resulting artifacts are an important issue. It was noted that the value of a particular modeling technique can often only be assessed in the context of a methodology in which such models are to be used. Guidance is available from many sources, but concrete application is open to interpretation. MDE attempts to address this to some degree, but it was noted that more work is needed. In summary, what are the base modeling methods, that, when applied in the large, constitute the transition of modeling to an engineering practice.

##### **Application of Modeling**

The discussion of future research in this area focused on domain-specific modeling. Several points were brought up, including the question of deciding when domain specialization should be pursued, and how to transition from general modeling concepts and notation to ones that are domain-specific. A more basic problem is defining what is meant by a domain-specific language; possibilities range from light weight approaches such as profiles to complete specification such as using ECORE with a domain-specific front-end. In either case, there are questions that should be used in designing a DSL, but these are not well-defined. Such questions should include precise notions of what is being abstracted, for which stakeholders, and from which domain concepts the abstractions are consequently derived. DSLs also have relationships among themselves in many instances, and these relations must be defined. Finally, creators of DSLs currently represent a mixture of roles, from developers to architects. The question arises whether this broad range of activities actually requires the creation of a new role of domain-language engineer.

##### **Practice of Modeling**

Open areas of research in practice include the on-going issue of keeping models consistent, or at least correlated. This problem is related to that of defining and maintaining relationships between models, in particular where these models are expressed in different modeling languages. Finally, the issue remains of continuing to enhance the links between various modeling communities, such as ICSE, MODELS, and Eclipse, for example. The problems and concerns of each of these communities appear to be related, but additional work is needed to bridge the differences to promote communication and exchange of ideas.

### **Conclusions**

Finally, the workshop participants proposed topics for the next installment of MiSE. A potential organizing principle is to focus on the challenges of modeling—a case study or set of case studies was proposed as an opportunity to highlight the challenges and solicitation of tools and techniques that can be applied directly to the case studies. However, it was unclear whether the community is sufficiently homogeneous to be covered by a single case study proposed by the workshop organizers. As an alternative, a panel on challenge topics was recommended, along with the presentation of a keynote address.

We would like to thank the ICSE workshop organizers, the MiSE program committee, and all the MiSE workshop participants for another outstanding workshop event.

## References

- [1] Acher, M., P. Lahire, S. Moisan, and J.-P. Rigault (2009): Tackling High Variability in Video Surveillance Systems through a Model Transformation Approach. In *MiSE'09 – Workshop on Modeling in Software Engineering at ICSE 2009, Vancouver, Canada, May 17-18, 2009*, IEEE.
- [2] Bencomo, N. (2009): On the Use of Software Models during Software Execution. In *MiSE'09 – Workshop on Modeling in Software Engineering at ICSE 2009, Vancouver, Canada, May 17-18, 2009*, IEEE.
- [3] Chaki, S., A. Diaz-Pace, D. Garlan, A. Gurfinkel, and I. Ozkaya (2009): Toward Engineered Architecture Evolution. In *MiSE'09 – Workshop on Modeling in Software Engineering at ICSE 2009, Vancouver, Canada, May 17-18, 2009*, IEEE.
- [4] Chavez, H., and W. Shen (2009): Finding Inconsistency for UML-Based Composition at Program Level. In *MiSE'09 – Workshop on Modeling in Software Engineering at ICSE 2009, Vancouver, Canada, May 17-18, 2009*, IEEE.
- [5] Chechik, M., W. Lai, S. Nejati, J. Cabot, Z. Diskin, S. Easterbrook, M. Sabetzadeh, and R. Salay (2009): Relationship-Based Change Propagation: A Case Study. In *MiSE'09 – Workshop on Modeling in Software Engineering at ICSE 2009, Vancouver, Canada, May 17-18, 2009*, IEEE.
- [6] Hermansdoerfer, M., W. Haberl, and U. Baumgarten (2009): Model-Level Simulation for COLA. In *MiSE'09 – Workshop on Modeling in Software Engineering at ICSE 2009, Vancouver, Canada, May 17-18, 2009*, IEEE.
- [7] Kolovos, D., L. Rose, R. Paige, and F. Pollack (2009): Raising the Level of Abstraction in the Development of GMF-Based Graphical Model Editors. In *MiSE'09 – Workshop on Modeling in Software Engineering at ICSE 2009, Vancouver, Canada, May 17-18, 2009*, IEEE.
- [8] Mustafiz, S., J. Kienzle, and H. Vangheluwe (2009): Model Transformation of Dependability-Focused Requirements Models. In *MiSE'09 – Workshop on Modeling in Software Engineering at ICSE 2009, Vancouver, Canada, May 17-18, 2009*, IEEE.
- [9] Nguyen, Q. (2009): Non-Functional Requirements Analysis Modeling for Software Product Lines. In *MiSE'09 – Workshop on Modeling in Software Engineering at ICSE 2009, Vancouver, Canada, May 17-18, 2009*, IEEE.
- [10] Oberortner, E., U. Zdun, and S. Dustdar (2009): Tailoring a Model-Driven Quality-of-Service DSL for Various Stakeholders. In *MiSE'09 – Workshop on Modeling in Software Engineering at ICSE 2009, Vancouver, Canada, May 17-18, 2009*, IEEE.
- [11] Spanfelner, B., C. Leuxner, and W. Sitou (2009): Formal specification of system functions. In *MiSE'09 – Workshop on Modeling in Software Engineering at ICSE 2009, Vancouver, Canada, May 17-18, 2009*, IEEE.