Report of the 7th International Workshop on Aspect-Oriented Modeling

Jörg Kienzle[1], Jeff Gray[2], Dominik Stein[3]

[1]School of Computer Science, McGill University, Montreal, QC, Canada
[2]Department of Computer and Information Sciences, University of Alabama at Birmingham, Birmingham, AL, USA
[3]Institute for Computer Science and Business Information Systems, University of Duisburg-Essen, Essen, Germany
Joerg.Kienzle@mcgill.ca,gray@cis.uab.edu,dominik.stein@icb.uni-due.de

Abstract. This report summarizes the outcome of the 7th Workshop on Aspect-Oriented Modeling (AOM) held in conjunction with the 8th International Conference on Model Driven Engineering Languages and Systems – MoDELS 2005 – in Montego Bay, Jamaica, on the 2nd of October 2005. The workshop brought together researchers and practitioners from two communities: aspect-oriented software development (AOSD) and software model engineering. It provided a forum for discussing the state of the art in modeling crosscutting concerns at different stages of the software development process: requirements elicitation and analysis, software architecture, detailed design, and mapping to aspect-oriented programming constructs. This paper gives an overview of the accepted submissions, and summarizes the results of the different discussion groups.

1 Introduction

This paper summarizes the outcome of the 7th edition of the successful Aspect-Oriented Modeling Workshop series. An overview of what happened at previous editions of the workshop can be found at `http://dawis.informatik.uni-essen.de/events/AOM_MODELS2005/preveds.shtml`. The workshop took place at the Half Moon Resort in Montego Bay, Jamaica, on Sunday, October 2nd 2005, as part of the 8th International Conference on Model Driven Engineering Languages and Systems – MoDELS 2005[1], formerly known as the series of conferences on the Unified Modeling Language. Participation to the workshop was open to anyone attending the conference, and as a result there were approximately 40 participants. A total of 14 position papers were submitted and reviewed by the program committee, 12 of which were accepted to the workshop. In order to leave enough time for discussion, only the morning sessions were dedicated to presentations. Based on the reviews of the papers, six of the papers were allocated 10-minute presentation slots, five papers were chosen for 20-minute presentation slots with the intention to stimulate and provide provocative input to the afternoon discussions. Before the lunch break, the attendees were asked to submit a list of

questions for the afternoon discussion session. Based on these questions, the attendees split into two groups: a "model-transformation and aspect-oriented modeling" group, and a "core aspect-oriented modeling concepts and aspect-oriented software processes" group. The results of the discussion groups were collected at the end of the workshop, and presented and re-discussed with the entirety of the workshop participants.

The rest of this report is structured as follows: Section 2 gives an overview to the accepted papers. Section 3 summarizes the results of the discussion groups. Section 4 concludes the report and presents identified future research directions.

2 Overview of Accepted Position Papers

Robert France from Colorado State University presented a position paper in which the authors explore the relationship between (aspect-oriented) model composition and model transformation[2]. They compare two high-level architectures of model transformation engines that could achieve aspect-oriented composition of models. One architecture describes a very specialized/dedicated transformation engine that takes a primary model, an aspect model, composition directives, and signature definitions as input to finally produce the composed model. The second architecture is very generic and symmetric. It takes a primary model and an aspect model and bindings as an input to produce the composed model.

Wolfgang Grieskamp from Microsoft Research presented a framework for composing behavioral models [3]. In the framework, different aspects of the system behavior are described using action machines (state machines or scenarios). These models can then be symmetrically composed and transformed to yield integrated models that can be used for model checking, refinement checking, and testing purpose. The techniques described in the paper rely on symbolic representation of values and state.

Mark Mahoney from Carthage College described a technique that enables weaving crosscutting concerns expressed in Live Sequence Charts (LSC) [4]. He presented how one can use pattern matching techniques in the pre-charts of an LSC to define “cross-cutting triggers” (similar to pointcuts in standard AOP languages), that activate the behavior described in an associated main chart (comparable to an AOP advice).

Jaime Pavlich-Mariscal from the University of Connecticut presented how they modeled access control schemas using role slices, and how they used aspect-oriented programming techniques to implement their system [5]. They outlined their goals to extend role slices with dynamic facilities. Thereo, their future research directions include adding support for access control based on run-time elements, as well as relating role-slice hierarchies with class hierarchies. These changes at the modeling level might require new AOP language features to implement them.

Jean-Paul Bodeveix from the Paul Sabatier University in Toulouse showed how one can specify real-time constraints in EMITL (Event Metric Interval Temporal Logic), and then transform this description into timed automata, and fi-

nally into a B specification [6]. This timing information can then be combined with a B specification of the functional behavior of a system to result in a composed B specification that specifies the functional and the timing behavior.

Tihamér Levendovszky from Budapest University of Technology and Economics presented how they used aspect-oriented techniques for applying OCL constraints [7]. He presented the Visual Modeling and Transformation System (VMTS), a meta-modeling environment, together with its Visual Control Flow Language (VCFL), a language allowing to express model transformations using graph rewriting techniques. VCFL uses OCL constraints to define constraints on the nodes of the transformation steps and to choose between different control flow branches. Often, the same constraint has to be repetitively applied to many different places in a transformation. Aspect-orientation can help to modularize such crosscutting constraints.

Aswin Van Den Berg from Motorolla Labs presented a framework for modularizing crosscutting concerns in embedded software, and how this framework can automate the composition of concerns at different phases of the code generation process [8]. He presented AspectSDL, an aspect-oriented framework that makes it possible to compose SDL statecharts. The aspect weaver composes core models and aspect bean models according to a binding definition (connector) and weaving strategy definitions. The ultimate goal is to perform consistency checks on the composed model, as well as to use it for simulation purpose.

Ana Moreira from the University of Lisbon presented how to build a metadata repository that describes the content, quality, structure, and other important data of concerns during the early stages of software development [9]. Such a repository allows a developer to navigate over all stored information to facilitate reuse, version control, and traceability.

In his second presentation, Robert France showed how they extended their aspect-oriented modeling approach to use signatures when composing models [10]. In their approach, crosscutting functionality is described by aspect-models and the core application functionality is described by a primary model. When composing models, model elements are merged with one another if their signatures match. A signature in this case consists of some or all properties of a model element as defined in the UML metamodel.

Andrew Jackson from Trinity College in Dublin presented the high-level view of a generic aspect-oriented design process [11]. The paper defines the core requirements of an aspect-oriented process to include support for modularization, composition, conflict resolution, and internal and external traceability. In addition, a good process must be an open, customizable, platform independent process that integrates with existing software development methodologies. It should support quality assurance metrics, staged adoption, and product families. Based on these requirements, the paper defines a process architecture with the following phases: concern identification and classification, design tests, reuse design, concern module design, composition specification design, verification, and refinement.

Andrew also agreed to present [12], a paper that describes how model-driven software development and an aspect-oriented modeling technique called "Aspectual Collaborations" can be brought together by providing a graphical composition mechanism.

3 Summary of the Discussion Groups

The following section summarizes the results of the afternoon working group sessions. The participants split into two groups – a "Model Transformation" and a "Core Aspect-Oriented Concepts" group – to discuss the questions submitted by the attendees before the lunch break.

3.1 Model Transformation Group

The model transformation working group was charged with the task of discussing various issues related to the transformation mechanisms of model weaving. The questions discussed are highlighted below along with summary comments.

What is model transformation? Before getting into the details of aspect weaving at the modeling level, the working group began with a discussion of the meaning of model transformation in general. A distinction was made between a source model and the target model. In some cases, there may be multiple sources and targets. The participants agreed that model transformation can be summarized as graph transformation, where a model is a set of typed nodes in a hypergraph that are manipulated according to the goals of a transformation rule. The process of model transformation eventually reaches a fixed point after multiple iterations among a set of transformation rules.

What is model weaving? There are two essential characteristics that seem to be common among most model weavers: 1) a pattern matching engine, like a pointcut language that provides quantification among modeling elements, and 2) a composition mechanism that transforms source models according to a new concern. In general, there were three types of weaving that were discussed:

- Static weaving on static structure, such as weaving into class diagrams where the semantics are pre-existing
- Static weaving on dynamic behavior, such as weaving into state-charts
- Dynamic weaving on dynamic behavior, which is the current focus of dynamic AOP languages. The working group could not identify a typical usage scenario for this type of weaving at the modeling level. Furthermore, the issue of dynamic weaving on static structure was not very clear within the modeling context.

Is model transformation equivalent to weaving? The working group discussed the relationship between model transformation and model weaving. It was determined that all weaving is a model transformation, but not all model transformation is weaving (e.g., model refactoring can be a model transformation that is not crosscutting). This is similar to the relation of program transformation to aspect code weavers, where an aspect is a special type of program transformation that captures crosscutting concerns.

Are there generic patterns for model transformation and weaving? An interesting thread arose from the discussion that examined whether common patterns of transformation have emerged from the experience of model transformation experts. Two patterns that were mentioned are *Find a Leaf*, which can be used to flatten hierarchical structures, and *Transitive Closure*, which can be used to collect modeling attributes during stages of a model transformation. It was also observed that the UMLAUT tool uses visitors, abstract factory, and other well-known design patterns for transformations in an OO style, and can be made specific for different models of computation (e.g., stateflow).

Is model weaving fundamentally the same as applying rules in a rule-based engine? One of the working group participants asked if a rule-based engine could be used for model weaving. The consensus of the group was that a rule language could theoretically capture some categories of crosscutting in a model, but the pragmatic application was less clear. It was suggested by one member that a model engineer often desires a higher level of abstraction. Several comparisons were presented as analogies to using rules for aspect weaving, and why a more focused language would be more desirable. Some of those counterexamples include:

- C++ abstractions to support objects can be simulated as C function pointers, but a pointer approach lacks the level of abstraction provided by pure OO constructs.
- Database triggers can capture limited crosscutting concerns in stored procedures, but the same language cannot be used for general AOP.
- Metaobject protocols and reflection can also be used to address crosscutting concerns, but are not as easy to use as a pure aspect language.

The summary from this discussion is that there exists a tradeoff between naturalness of expression and power of language. A rule-based language could be used in some cases to describe modeling aspects, but the naturalness and applicability are not as evident when compared to a pure aspect modeling language.

How can properties of model weaving be proven? There was concern among the participants regarding the manner in which the resulting properties of the model weaving could be proven. This question was re-stated in terms of traditional verification (i.e., is the weaving itself performed correctly?) and

validation (i.e., is the result that which was in the mind of the designer?). This was cited as a strong need for future work, with little being done on the topic so far. The composition of modeling aspects and the resulting behavior is trivial if the concerns are orthogonal (i.e., no interference), but more challenging if non-orthogonal (e.g., composing two separate access control aspects, such as RBAC and mandatory access control).

What are the performance issues associated with aspect modeling? The notion of performance as it relates to aspect modeling can be broken down into three separate questions. The first issue relates to the actual performance of the model weaver itself (i.e., how long does it take to weave the models?). Of course, this will depend on the size of the source model and the speed of the model weaving tool. A second type of performance issue concerns the resulting size of the target model. An explosion of the size of the model after weaving may inhibit further analysis and generation. A third version of this question may apply to the performance of the actual modeled system. The desire to model performance using aspects may be motivated by different domain requirements, such as a model for real-time embedded systems. The ability to modularize crosscutting modeling concerns related to performance may enable a model engineer to change properties of a model in a rapid manner as compared to a tedious and error prone manual approach.

3.2 Core Aspect-Oriented Concepts Group

The core aspect-oriented concepts group looked at the issues that arise when applying aspect-oriented modeling techniques to real-world models.

How do existing aspect-oriented modeling techniques scale? Several participants with industrial background expressed their concerns about how existing aspect-oriented modeling techniques would scale to systems with hundreds of classes and aspects. Participants from Motorola mentioned that aspect-oriented techniques actually do work well in industrial settings, even in large scale systems, provided that the number of aspects is small. Orthogonal aspects such as logging work particularly well. The discussion group identified the lack of availability of real-world UML models, i.e. models with hundreds of classes, as one of the reasons why current AOM approaches have been applied to toy examples only. Also, functional crosscutting concerns are not trivial to identify and modularize. The problem of aspect dependencies and conflict detection was determined as one of the main scalability challenges.

Some argued that in order to achieve scalability we need a common core meta-model. The UML meta-model was deemed to be missing the power to express relationships among different models. Some suggested the definition of an AML – an Aspect Meta Language – that provides a unified type space linking UML and aspects together.

Does aspect-oriented modeling improve reuse? There has not been a lot of evidence that aspect-orientation improves reuse. The attendees of the workshop figured that the reason for this is the lack of a good aspect-oriented design process, and the fact that students and programmers in general are not educated to write reusable code, even for object-oriented systems. Different aspect-oriented approaches are also not compatible, which makes reuse very difficult. Again, the use of a common meta-model could improve this situation.

How good are aspect-oriented modeling tools? In aspect-oriented modeling, there is a big need for flexible tools. Unfortunately, tool vendors like to provide their home-grown extensions to UML, but do not allow (or make it very complicated for) users to extend the tool on their own. The only solution nowadays is to export models and then write "filter-like" mini-tools that parse the exported file and apply the desired transformations on it. Tools are also inflexible because they often present only one view of the system to the developer at a given time. For some approaches it would be nice to simultaneously display multiple models, e.g. the primary model and an aspect model. Since this is currently not supported, a lot of mental work has to be performed by the developer during design.

What decisions are human decisions and what choices can be automated? Many model transformations need additional human input in order to be applied to a given model. Sometimes the choice of the model transformation to be applied is done by a human. It is important to record these human decisions, and why these decisions have been made, in order to provide traceability and accountability during software development. It is not yet clear from which point on the generation of code can be completely automated.

Should it be allowed to "restrain" aspects, e.g. hide join points from other aspects in the system? The discussion started out from the question of why aspects are better than components. Components in the past have promised out-of-the-box reuse in the sense of "buy your component to take care of concern X". However, reuse of components only seems to work for very specific concerns. Some participants argued that aspects will deliver, because they are so flexible. Others argued that aspects are so flexible as to be useless.

This raised the question of whether it should be possible to restrict aspect configurations to not expose their join points to other parts of the system. The arguments in favor of restriction were the conservation of important software engineering properties such as encapsulation and information hiding. Controlling the scope of aspects can also help to distribute development of large systems among different teams without the danger of having one aspect in one part of the system unintentionally affecting other parts of a system. Finally, restriction would allow to ensure non-functional and run-time properties such as execution time, etc. On the other hand, the major argument against restriction is the fact

that unrestricted aspects make it possible to add functionality anywhere. This allows for rapid development and prototyping because developers can focus on a smaller part of the project without having to worry about designing for future extension. Also, unrestricted aspects can help tremendously in a world where requirements are likely to change.

What benefits does aspect-oriented modeling bring to industry? Aspect-orientation was identified as an elegant way to perform "functional decomposition" for large industrial projects. It provides a flexible framework in which concerns can be identified and resolved at different levels of refinement. It allows developers to postpone decisions and focus on important concerns first, maybe even implement a prototype, without having to worry about other concerns. Thanks to aspect-orientation, secondary concerns can be added to the system at a later phase.

4 Concluding Remarks

The workshop continued the tradition of having a very diverse representation of participants. The authors came from seven different countries (Argentina, France, Germany, Hungary, Ireland, Portugal, and USA), the organizing and programming committees represented nine countries (Canada, China, France, Germany, Ireland, Israel, Netherlands, Switzerland, and USA). In addition to the geographical diversity, the AOM workshop also attracted participants with wide research interests in aspects across the entire spectrum of the development lifecycle. As a result, this provided opportunities for a variety of opinions that were well-informed from the accumulated experience of the participants.

The growth of the workshop continued to increase, which indicates a strong interest in the area among researchers in aspect-oriented modeling. Many of the participants felt a new sense of maturity at the workshop that has not been evident in past editions. For example, the previous debates over definition of terms and mechanisms were replaced with deeper discussions focused on the core issues that need to be addressed to move the area into a common modeling practice. With this workshop report, we'd like to give researches who couldn't attend the workshop the opportunity to gain insights to these issues, and to point out a future research agenda in the field of aspect-oriented modeling.

Acknowledgements

The organizers spent a lot of time ensuring that the workshop was a success. The organizers for this edition of the workshop were Omar Aldawud, Tzilla Elrad, Jeff Gray, Mohamed Kandé, Jörg Kienzle, and Dominik Stein. An expert program committee provided assistance in reviewing the submitted papers. The members of the program committee were Mehmet Aksit, Elisa Baniassad, Jean Bézivin, Siobhán Clarke, Robert France, Sudipto Ghosh, Stefan Hanenberg,

Shmuel Katz, Raghu Reddy, Martin Robillard, and Christa Schwanninger. Last but not least, we'd like to thank all submitters and participants of the workshop who contributed with their papers and positions.

References

1. Briand, L., Williams, C., eds.: 8th International Conference on Model Driven Engineering Languages and Systems, Montego Bay, Jamaica, Oct. 2-7, 2005. Number 3713 in Lecture Notes in Computer Science, Springer Verlag (2005)
2. Baudry, B., Fleury, F., France, R., Reddy, R.: Exploring the relationship between model composition and model transformation. In: 7th International Workshop on Aspect-Oriented Modeling, Montego Bay, Jamaica, Oct. 2nd, 2005. (2005)
3. Grieskamp, W., Kicillof, N., Campbell, C.: Behavioral composition in symbolic domains. In: 7th International Workshop on Aspect-Oriented Modeling, Montego Bay, Jamaica, Oct. 2nd, 2005. (2005)
4. Mahoney, M., Elrad, T.: Weaving crosscutting concerns into live sequence charts using the play engine. In: 7th International Workshop on Aspect-Oriented Modeling, Montego Bay, Jamaica, Oct. 2nd, 2005. (2005)
5. Pavlich-Mariscal, J., Michel, L., Demurjian, S.A.: Role slices and runtime permissions: Improving an AOP-based access control schema. In: 7th International Workshop on Aspect-Oriented Modeling, Montego Bay, Jamaica, Oct. 2nd, 2005. (2005)
6. Rached, M., Bodeveix, J.P., Filali, M., Nasr, O.: Real-time aspects: Specification and composition in b. In: 7th International Workshop on Aspect-Oriented Modeling, Montego Bay, Jamaica, Oct. 2nd, 2005. (2005)
7. Lengyel, L., Levendovszky, T., Charaf, H.: Real-time aspects: Specification and composition in b. In: 7th International Workshop on Aspect-Oriented Modeling, Montego Bay, Jamaica, Oct. 2nd, 2005. (2005)
8. Cottenier, T., Van Den Berg, A., Elrad, T.: Modeling aspect-oriented compositions. In: 7th International Workshop on Aspect-Oriented Modeling, Montego Bay, Jamaica, Oct. 2nd, 2005. (2005)
9. Ferreira, R., Raminhos, R., Moreira, A.: Metadata driven aspect specification. In: 7th International Workshop on Aspect-Oriented Modeling, Montego Bay, Jamaica, Oct. 2nd, 2005. (2005)
10. Reddy, R., France, R., Gosh, S., Fleury, F., Baudry, B.: Model composition - a signature based approach. In: 7th International Workshop on Aspect-Oriented Modeling, Montego Bay, Jamaica, Oct. 2nd, 2005. (2005)
11. Jackson, A., Clarke, S.: Towards a generic aspect-oriented design process. In: 7th International Workshop on Aspect-Oriented Modeling, Montego Bay, Jamaica, Oct. 2nd, 2005. (2005)
12. Groher, I., Bleicher, S., Schwanninger, C.: Model-driven development for pluggable collaborations. In: 7th International Workshop on Aspect-Oriented Modeling, Montego Bay, Jamaica, Oct. 2nd, 2005. (2005)