9th International Workshop on Aspect-Oriented Modeling

Jörg Kienzle[1], Dominik Stein[2], Walter Cazzola[3],
Jeff Gray[4], Omar Aldawud[5], Tzilla Elrad[6]

[1]McGill University, Canada; [2]University of Duisburg-Essen, Germany
[3]University of Milano, Italy; [4]University of Alabama at Birmingham, USA
[5]Lucent Technologies, USA; [6]Illinois Institute of Technology, USA
joerg.kienzle@mcgill.ca, dominik.stein@icb.uni-due.de, cazzola@dico.unimi.it,
gray@cis.uab.edu, oaldawud@lucent.com, elrad@iit.edu

Abstract. This report summarizes the outcomes of the 9th Workshop on Aspect-Oriented Modeling (AOM) held in conjunction with the 9th International Conference on Model Driven Engineering Languages and Systems – MoDELS 2006 – in Genoa, Italy, on the 1st of October 2006. The workshop brought together approximately 25 researchers and practitioners from two communities: aspect-oriented software development 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. Papers and presentation slides of the workshop are available at http://www.aspect-modeling.org/.

1 Introduction

This report summarizes the outcomes of the 9th edition of the successful Aspect-Oriented Modeling Workshop series. The workshop took place at the Bristol Hotel in Genoa, Italy, on Sunday, October 1st 2006, as part of the 9th International Conference on Model Driven Engineering Languages and Systems – MoDELS 2006. A total of 11 position papers were submitted and reviewed by the program committee, 9 of which were accepted to the workshop. Approximately 25 participants attended the presentation session and took part in lively discussions. Papers, presentation slides, and further information can be found at http://www.aspect-modeling.org/.

2 Overview of Accepted Position Papers

Marcelo Sande from the Military Institute of Engineering in Rio de Janeiro, Brazil, described how he and his colleagues mapped AspectualACME, an architectural description language, to UML 2.0 [8]. He presented why the base UML 2.0 modeling

abstractions of component diagrams are not strong enough. One reason to this is that standard UML only allows to connect provided interfaces of components to required interfaces of other components. He explained how they made connectors first-order elements, and how they defined a special aspectual connector that can be used to connect the provided interface of a (crosscutting) component to both the provided (!) and the required interfaces of another (base) component.

Natsuko Noda from the Japan Advanced Institute of Sciences and Technology in Nomi, Japan, presented a symmetric aspect-oriented modeling technique for aspect-oriented design [6]. In Noda's presentation, each concern of the system is modeled with aspects that are composed of class diagrams, object diagrams and state diagrams. Each aspect is self-contained. The connections between aspects are defined in aspect relation rules, which define how a transition change in one aspect can affect other aspects (i.e., by introducing events into other aspects).

Asif Iqbal from the Honeywell Technology Solutions Lab in Bangalore, India, works in the context of modeling of safety-critical systems. He talked about the issue of modeling temporal behavior, which usually crosscuts the functional model of a system [4]. In order to reason about concepts such as Worst Case Execution Time, time-depending behavior has to be explicitly represented in models. As an example, Asif mentioned the synchronization of local clocks with a global clock. He showed how this concern can be modeled with timed state diagrams, and how the crosscutting can be modeled using the AOSF framework with time extensions. However, state diagrams that are created using orthogonal composition run on a single clock, which is a problem that still needs to be addressed.

Thomas Cottenier from Motorola Labs in Chicago, USA, argued that reactive functionality of a system should be modeled using a reactive modeling formalism such as state diagrams [2]. He showed a small demonstration of the Motorola Aspect WEAVR, a tool for aspect-oriented composition of state diagrams. The Motorola models are executable (or transformable into executable code). Thomas argued that aspect-oriented modeling is more powerful than aspect-oriented programming: The join point model of state diagrams is better suited to express crosscutting reactive concerns than the classic join point model of aspect-oriented programming languages.

Sonia Pini from the University of Genoa, Italy, argued that current pointcut definitions require global knowledge of the base program by the developer in order to write meaningful pointcuts [1]. Hence, current join point selection mechanisms are fragile, because they fail to provide reusability and evolvability. In order to reason about the semantics of join points, she proposed a technique in which the join points are expressed at a higher level of abstraction (i.e., at the modeling level with sequence and activity diagrams). Furthermore, she presented a mechanism to map these high-level join point selections to program code.

Arnor Solberg from SINTEF/the University of Oslo, Norway, presented an aspect-oriented modeling technique based on sequence diagrams [7]. In this approach, aspect sequence diagrams are defined that represent a template of crosscutting behavior. To instantiate the aspects, the base model is annotated with tags that define where the aspects should be applied (i.e., instantiated). Simple aspects are inserted into the base sequence diagram at one specific point, whereas composite aspects are applied to regions within the base diagram (annotated with a tagged fragment). In order to allow fine-grained application of crosscutting behavior within this tagged fragment, a

composite aspect defines several parts: begin/end parts that execute when the fragment is entered/exited, before/after parts that execute before or after every message invocation, and a body part that can alter the actual message sending.

Andrea Sindico from ELT Elettronica in Rome, Italy, presented an aspect-oriented modeling approach in which concerns are specified in an aspect diagram [3], which defines static crosscutting in the form of an inter-type declaration diagram, and dynamic crosscutting in the form of advice diagrams. Inter-type declaration diagrams are composed of two class diagrams. Advice diagrams are composed of pointcut diagrams and behavioral diagrams. In both cases, one diagram explains the context of the base program that is of interest, while the other shows what has to be added to the base context. Pointcut diagrams (comprised in advice diagrams), for example, define the set of join points to which an aspect is to be applied. In his work, Andrea suggests to specify them in the form of a UML activity diagram.

Thomas Cottenier from the Motorola Labs in Chicago, USA, also presented work on aspect interference at the modeling level [9]. He showed a demo of the Telelogic TAU tool, in which they implemented different dependencies in their aspect deployment diagrams: A «follows» B, which specifies that aspect A's behavior has lower precedence than B; A «hidden_by» B, which specifies that the behavior of aspect A is not activated when A and B apply to the same join point; and A «depends_on» B, which specifies that aspect A's behavior can only be applied where aspect B's behavior is also applied.

Roberto Lopez-Herrejon from Oxford University, UK, related Feature-Oriented Programming (FOP) to the approach of Aspect-Oriented Software Development with Use Cases (AOSD w/UC) [5]. He demonstrated how features can crosscut other features and how aspects can help to resolve this crosscutting. Roberto referred to the existing approach of AOSD w/UC and pointed out its limitations with respect to a well-defined composition mechanism. After that, he introduced the algebraic approach of FOP, which contains a formal composition model, but lacks an "intuitive" notation. Roberto proposed to combine FOP with AOSD w/UC to achieve mutual benefit.

3 Overview of Discussion Topics

Due to space limitations, this section offers a summary of the most interesting and significant issues that were addressed during the discussion sessions. These issues also emerged during the questions and comments in the presentation sessions.

Is AOM about visual representation? During the workshop, the participants expressed several opinions about the essence of AOM. The general idea of modeling is to make something simpler (i.e., more comprehensible). Very often, this goal is achieved by providing a visual notation. However, most of the participants agreed that a visual notation is not essential. Once the semantics of an abstraction are well-defined, finding a suitable graphical representation for it is only syntactic sugar. The discussion did not go into further details, unfortunately, about what precisely AOM should make simpler or more comprehensible other than "visual communication."

Is there a need to look at woven models? There has been a disagreement on whether developers need to have a look at woven models. Although some participants argued that this is necessary for comprehending the execution of an aspect-oriented program (or model) and for debugging, others claimed that, once the semantics of a given weaving mechanism is clear, developers do not care about (and do not need to look up) how these semantics are actually accomplished.

Does AOM meet its goals? One of the participants questioned if AOM actually meets its goals, such as an improved readability, comprehensibility, extensibility, and reusability of software (artifacts). The participant reported on a case study that was conducted in which aspect-oriented modeling techniques were used throughout the entire software development lifecycle. That is, each concern was separated all the way down from requirements elucidation to the pre-coding phase. In the end, the participant obtained a nicely separated set of concern specifications. However, this results in a full load of very complex composition specifications determining how those nicely separated concerns are supposed to work together. These composition specifications were not readable, comprehensible, extendible, or reusable.

What is the role of model composition specifications (join point selections, composition rules, etc.) in the software development process? Various participants were concerned about the relevance of model composition specifications (such as join point selections, pointcuts, composition rules, composition directives, or other kinds of dependency relationships between concern models) in the software development lifecycle. It has been stated that the gap between join points[1] in requirement specifications and join points in the corresponding code is huge. Consequently, the mapping of join point selections (composition rules) between different levels of abstraction is often problematic. One solution to this might be to introduce notions of join points at various levels of abstraction, such as architectural join points for architectural system descriptions, and map the join point notion of one abstraction layer to the join point notion of the layer beneath.

A statement from an industry participant suggested that AOM may help to keep concepts separated and consistent throughout the development process. However, AOM should also provide a means to indicate explicitly how those separated concepts interact with each other in order to document design decisions and tradeoffs. Furthermore, AOM should provide support for documenting the application of a particular policy in the general case and at the same time outlining under which circumstances a more specialized policy is used (e.g., in general, use password authentication, but in these and those special cases, use biometric authentication).

What is the target application context of AOM? Another question concerned the target application context of AOM and how AOM should support it. Industry mentioned that software projects rarely attack problems from scratch. Usually, existing software needs to be extended. Therefore, AOM should provide a means to support extensibility. Another point was that introducing aspects into industry should start with simple cases. Such simple aspects should be implemented by a small group of developers, which facilitates support for the larger group of "base program developers." Once the simple aspects are adopted, more elaborate aspects could be introduced. One problem that has to be solved concerns the fact that even simple

[1] or, more generally speaking, some kind of concern interaction points

aspects can add an enormous amount of new possible states to a base program. Perhaps AOM can help in estimating the effects of an aspect onto a given base system. Another scenario mentioned concerns how AOM could be used to document design decisions and tradeoffs (see previous question).

4 Concluding Remarks

The 9th International Workshop on Aspect-Oriented Modeling in Genoa provided evidence that the AOM community has reached a state of maturity. Most participants were well aware of the fundamental ideas and key concepts of AOSD. Consequently, the focus of discussions shifted from "what are the right abstractions to use in AOM in general?" towards "how to use these abstractions and AOM in order to reach certain goals?" The participants from academia critically evaluated the existing modeling approaches with respect to certain claims and specific problems. Participants from industry expressed clear expectations of what they anticipate from AOM. These problems and expectations outline cardinal and substantial topics for future research on AOM.

Acknowledgements

We would like to thank the program committee members who have helped to assure the quality of this workshop: Mehmet Aksit, Aswin van den Berg, Thomas Cottenier, Robert France, Sudipto Ghosh, Stefan Hanenberg, Andrew Jackson, Jean-Marc Jézéquel, Kim Mens, Alfonso Pierantonio, Raghu Reddy, and Markus Völter. We also thank all submitters and workshop participants who helped to make this workshop a success.

List of Accepted Position Papers

[1] Cazzola, W., Pini, S., *Join Point Patterns: A High-Level Join Point Selection Mechanism*
[2] Cottenier, T., van den Berg, A., Elrad, T., *Model Weaving: Bridging the Divide between Elaborationists and Translationists*
[3] Grassi, V., Sindico, A., *UML Modeling of Static and Dynamic Aspects*
[4] Iqbal, A., Elrad, T., *Modeling Timing Constraints of Real-Time Systems as Crosscutting Concerns*
[5] Lopez-Herrejon, R., Batory, D., *Modeling Features in Aspect-Based Product Lines with Use Case Slices: An Exploratory Case Study*
[6] Noda, N., Kishi, T., *An Aspect-Oriented Modeling Mechanism Based on State Diagrams*
[7] Reddy, R., Solberg, A., France, R., Ghosh, S., *Composing Sequence Models using Tags*
[8] Sande, M., Choren, R., Chavez, C., *Mapping AspectualACME into UML 2.0*
[9] Zhang, J., Cottenier, T., van den Berg, A., Gray, J., *Aspect Interference and Composition in the Motorola Aspect-Oriented Modeling Weaver*