Editorial for the Special Issue on Aspects and Model-Driven Engineering

Robert France[1] and Jean-Marc Jézéquel[2]

[1] Colorado State University, Fort Collins, Colorado, USA,
france@cs.colostate.edu,
[2] IRISA-Université de Rennes 1, France,
jezequel@irisa.fr

1 Introduction

Model-Driven Engineering (MDE) is an approach to software development in which models are used to drive the development of all software artifacts, from code to documentation to tests. MDE is gaining acceptance in several software domains with demonstrated benefits such as cost reduction and quality improvement.

Modeling is not just about expressing a solution at a higher abstraction level than code. This limited view on modeling has been useful in the past (e.g., assembly languages abstracting away from machine code, 3GL abstracting over assembly languages) and it is still useful today, but much more can be accomplished using modeling techniques.

A model can be an abstraction of an aspect of a system (existing or under development) that handles a given concern. Complex systems typically give rise to more than one model because many aspects must be considered when addressing all relevant software development concerns. These models may be expressed with a general purpose modeling language such as the UML, or with Domain Specific Languages when they are deemed more appropriate.

From a modeling point of view, the terms aspect and model can be considered synonymous. This notion of aspect goes beyond the usual meaning found in the Aspect Oriented Programming community where an aspect is often narrowly defined as the modularization of a cross-cutting concern. Given a "main" decomposition paradigm (such as object orientation), there are many classes of concerns (e.g., security, mobility, availability, distribution) for which clear allocation into modules is not possible (i.e., they are "cross-cutting" concerns).

However, the growing uptake of the term aspect outside of the programming world, has resulted in a growing acceptance of a broader definition in which an aspect is a concern that can be modularized. Work on aspect-oriented techniques above the code level is concerned with the systematic identification, modularization, representation, and composition of concerns. The goal of work in this area is to improve our ability to reason about the problem domain and the corresponding solution, thereby reducing the size of software models and application code, development costs, and maintenance time.

From the above, an important software development activity is the separation of concerns in problem domains. This activity is called *analysis*. If solutions to these concerns can be described as aspects, the design process can then be characterized as a weaving of these aspects into a base design model. This is not new: designers have been doing this for some time. However, the various aspects are often not *explicitly* defined, and when they are, it is done informally. Currently, designers do the weaving mentally (i.e., in their heads), and then produce the resulting detailed design as a tangled structure of design elements. This may work for small problems, but it introduces significant accidental complexities when tackling larger problems.

Note that the real challenge here is not how to design the system to take a particular aspect into account: there is significant design know-how in industry on this and it is often captured in the form of design patterns. Taking into account more than one aspect can be a little harder, but many large scale successful projects in industry provide some evidence that engineers know how different concerns should be handled. The real challenge is reducing the effort that the engineer has to expend when grappling with many inter-dependent concerns. For example, in a product-line context, when an engineer wants to replace a variant of an aspect used in a system, she should be able to do this cheaply, quickly and safely. Manually weaving every aspect is not an option.

Unlike many models used in the sciences, models in software and in linguistics have the same nature as the things they model. In software, this provides an opportunity to automatically derive software from its model, that is, to automate the weaving process. This requires models to be formal, and the weaving process be described as a program (i.e., an executable meta-model) manipulating models to produce a detailed design. The detailed design produced by the weaving process can ultimately be transformed to code or at least test suites.

In the above, we make the case that aspects are at the core of Model Driven Engineering. From this perspective, work on aspect-oriented approaches to modeling is important because it can yield significant insights into how the MDE vision of software development can be realized. There is thus a growing community interested in the convergence of Aspect-Oriented Software Development (AOSD) and MDE ideas. In this issue, we present papers that provide good examples of how AOSD and MDE ideas can be integrated to produce techniques that manage software complexity.

2 Content of this Special Issue

The papers in this issue cover a number of issues including the following:

- Methods and techniques supporting separation, composition, and evolution of aspects identified in different development phases (e.g., requirements, architecture, detailed design, deployment).
- Simulating runtime weaving of aspects using aspect-oriented models.
- Techniques for verifying and validating aspect-oriented models.

– AOM case studies that provide significant insights into how aspect-oriented modeling techniques can be applied across the development life-cycle.
– Providing tool support for use of integrated AOSD and MDE techniques.
– Providing language support for aspect-oriented modeling.

Submissions

Dynamic Weaving of Aspect-Oriented Executable UML models In this paper, the authors, *Lidia Fuentes and Pablo Sanchez*, present a model weaver that can be used to simulate runtime weaving of aspects at design time. This allows designers to identify and correct errors that can arise as a result of dynamic weaving before expending significant effort and cost on implementing the design. The ideas are illustrated using a location-aware intelligent transportation system.

On Language-Independent Model Modularisation In this paper, the authors, *Florian Heidenreich, Jakob Henriksson, Jendrik Johannes, and Steffen Zschaler*, present a generic approach to modularizing and composing models. The approach can be adapted to construct language- and purpose-specific composition techniques for specific modelling languages. The authors claim that the approach can be used as (1) a tool for developing specific model modularisation and composition techniques, and (2) a research instrument for studying properties and concepts of model modularisation.

Aspects across Software Life Cycle: A Goal-Driven Approach In this paper, the authors, *Nan Niu, Yijun Yu, Bruno Gonzalez-Baixauli, Neil Ernst, Julio Cesar Sampaio do Prado Leite, and John Mylopoulos*, propose a model-driven framework for tracing aspects from requirements to testing and implementation. In the framework, goal models are engineering assets and model-to-code transformations are used to bridge the gap between domain concepts and implementation technologies. The frameworks applicability and usefulness is evaluated using an open-source e-commerce platform case study.

Aspect-Oriented Model-Driven Software Product Line Engineering In this paper, the authors, *Iris Groher and Markus Voelter*, present an integrated AOSD and MDE approach to variability implementation, management, and tracing in product-line development of software. Features are modeled separately and the models are composed using aspect-oriented composition techniques. Model transformations are used to transform problem models to solution models. The ideas presented in the paper are illustrated using a home automation system case study.

Constraint-based ModelWeaving In this paper, the authors, *Jules White, Jeff Gray, and Douglas C. Schmidt*, present a constraint-based weaving technique that reduces model weaving to a constraint satisfaction problem (CSP). A constraint solver is used to deduce an appropriate weaving strategy. The paper also presents the results of a case study in which the constraint-based weaving technique is applied to an enterprise Java application. The evaluation showed that use of the technique resulted in a reduction of manual effort.

MATA: A Unified Approach for Composing UML Aspect Models based on Graph Transformation In this paper, the authors, *Jon Whittle, Praveen Jayaraman, Ahmed Elkhodary, Ana Moreira and Joo Arajo*, describe an aspect-oriented modeling technique called MATA (Modeling Aspects Using a Transformation Approach). MATA uses graph transformations to specify and compose aspects. In MATA, any model element can be a join point and composition is a special case of model transformation. MATA has been applied to a number of realistic case studies and is supported by a tool built on top of IBM Rational Software Modeler.

Model Driven Theme/UML In this paper, the authors, *Andrew Carton, Cormac Driver, Andrew Jackson and Siobhan Clarke*, describe how the Theme/UML approach to modularizing and composing concerns can be integrated with MDE techniques. The resulting method includes a tool-supported technique for transforming platform-independent models to platform-specific models. The transformation tool utilizes standards defined in the Object Management Group's Model Driven Architecture. The paper also describes a process that guides the use of the MDE/AOSD techniques. The utility of the approach is demonstrated through a case study.

2.1 Biographies

Robert France: Professor Robert France is a full professor in the Department of Computer Science at Colorado State University. He is actively engaged in research on object-oriented (OO) modeling, aspect-oriented modeling, model transformations, and formal description techniques. He is an editor-in-chief for the journal on Software and System Modeling (SoSyM) and is an Software Area Editor for the IEEE Computer. He was organizing chair for the Second Conference on the UML, past chair of the UML Conference steering committee and member of the MoDELS Conference steering committee.

Jean-Marc Jézéquel: Prof. Jean-Marc Jezequel received an engineering degree in Telecommunications from the ENSTB in 1986, and a Ph.D. degree in Computer Science from the University of Rennes, France, in 1989. He first worked in Telecom industry (at Transpac) before joining the CNRS (Centre National de la Recherche Scientifique) in 1991. Since October 2000, he is a Professor at the University of Rennes, leading an INRIA research team called Triskell. His interests

include model driven software engineering based on object oriented technologies for telecommunications and distributed systems. He is the author of the books "Object-Oriented Software Engineering with Eiffel" and "Design Patterns and Contracts" (Addison-Wesley 1996 and 1999), and of more than 100 publications in international journals and conferences. He is a member of the steering committees of the AOSD and the MODELS/UML conference series. He also served on the editorial boards of IEEE Transactions on Software Engineering and on the Journal on Software and System Modeling: SoSyM and the Journal of Object Technology: JOT.

For more information please visit http://www.irisa.fr/prive/jezequel