Aspect-Oriented Modeling to Support Dynamic Adaptation

Brice Morin, Franck Fleurey, Olivier Barais, Jean-Marc Jézéquel

Aspect-Oriented Modeling to Support Dynamic Adaptation*

Brice Morin[1], Franck Fleurey[2], Olivier Barais[1,3], and Jean-Marc Jézéquel[1,3]

[1] INRIA, Centre Rennes - Bretagne Atlantique
Brice.Morin@inria.fr
[2] SINTEF, Oslo, Norway
Franck.Fleurey@sintef.no
[3] IRISA, Université Rennes1
barais@irisa.fr | jezequel@irisa.fr

Abstract. Since software systems need to be continuously available under varying conditions, their ability to evolve at runtime is increasingly seen as one key issue. Modern programming frameworks already provide support for dynamic adaptations. However the high-variability of features in Dynamically Adaptive Systems (DAS) introduces an explosion of possible runtime system configurations (often called modes) and mode transitions. Designing these configurations and their transitions is tedious and error-prone, making the system feature evolution difficult. This demo presents a tool-chain developed by the DiVA project, which combines AOM and Model-Driven Engineering to tame the combinatorial explosion of DAS modes. Using AOM techniques, we derive a wide range of modes by weaving aspects into an explicit model reflecting the runtime system. We use these generated modes to automatically adapt the system using MDE techniques.

1 A Brief Introduction of DiVA

DiVA is a European project that aims at providing a new tool-supported methodology with an integrated framework for managing dynamic variability in adaptive systems. DiVA addresses this goal by combining Aspect-Oriented and Model-Driven techniques in an innovative way.

DiVA mobilizes leading players within the domains of model-driven engineering, aspect-oriented analysis and modeling, variability modeling and partners that provide advanced end user systems and applications where the demand for adaptation is a major concern. There are 3 academic partners in DiVA: SINTEF (Norway), Lancaster (UK) and INRIA (France), and 3 industrial partners: pure-systems GmbH (Germany), Thales (France) and CAS Software AG (Germany).

2 An Overview of the Demo

The demonstration illustrates the model-based and aspect-oriented DiVA tool-chain on a simple case study. It basically presents the tools associated with previous publications [1,2,3].

* The research leading to these results has received funding from the European Community's Seventh Framework Program FP7 under grant agreements 215412 (DiVA, http://www.ict-diva.eu/)

2.1 DiVA Metamodels

In the context of the DiVA European project, we have modeled four dimensions/aspects of an adaptive system [2]:

- **Variability Model**: The Variability model is a feature model, describing the variability of the system. Feature models are commonly used in the SPL community [4]. It allows to describe hierarchies of features with mandatory features, options, alternatives, n among p choices, etc, as well as constraints (require, exclude) among features. The Variability model is a regular feature diagram model, with a naming convention to refer to architectural fragments (aspect models) refining the features. This way, we can exploit any existing feature model tools (graphical editors, checkers, etc) with no modification.
- **Context Model**: The context model specifies the environment of the system. A set of context variables specifies the aspects of the environment which are relevant to adaptation. At runtime, the values of the variables are provided by context sensors and these may trigger a reconfiguration of the system.
- **Reasoning Model**: The reasoning model describes how the features of the DSPL are selected according to the context. There exist several formalisms such as Event-Condition-Action (ECA) rules [5], or Goal-Based Optimization rules [2]. An ECA model will typically describe, for particular contexts, which features to select: **when** *context* **choose** *features*. A Goal-Based model will typically describe how features impact QoS properties, using for example *help* and *hurt* relationships [6], and specify when QoS properties should be optimized (*e.g.*, when a property is too low).
- **Architecture Model**: This model allows the designer to describe component-based architectures. We can use any metamodel, such as the UML 2 component diagrams or SCA (Software Component Architecture), or any Architecture Description Language (ADL), to describe architecture. Since our dynamic adaptations are (currently) reconfigurations, we are concerned with components and bindings. These concepts are present in the above mentioned metamodels. In practice, we have defined our own minimal metamodel [7] to reduce the memory overhead at runtime. Other metamodels are simply mapped to our metamodel via model transformations written in Kermeta [8]. Each (leaf) feature of the DSPL model is refined into a fragment of architecture. We use Aspect-Oriented Modeling (AOM) techniques to design and compose features into a core model containing the mandatory elements. We leverage these models at design-time to perform validation and simulation tasks, as well as runtime to fully automate the dynamic adaptation process.

These models are built at design-time and will be leveraged at runtime to drive the dynamic adaptation process. The quality and the correctness of these models are crucial and have to be checked as early as possible. Tools are provided so that the designer can execute scenarios (sequences of contexts) in order to simulate the adaptation logic of the system [9] and check the produced configurations [3]. These configurations have to ensure the constraints (cardinalities, requires/excludes) defined in the DSPL model, and will be composed of suitable features selected according to the reasoning model.

2.2 Refining Features as Aspect Models

In DiVA, we leverage Aspect-Oriented Modeling techniques to refine features and automatically build complete configurations, before the actual adaptation. A base model refines the commonalities (elements present in all the configurations) of the system as an architecture made of components and bindings (connections between components). Aspect models refine the variants of the system by specifying their precise architectures. More specifically, each aspect model is an architecture fragment that contains all the information needed so that it can easily be plugged into the base architecture. An aspect model is composed of three parts, as illustrated in Figure 1:

- **Advice model**. An architecture fragment specifying **what** is needed to realize the associated variant. A graft model does not need to be fully consistent. The elements that are missing to make the graft model consistent should be brought by the base model, when weaving the aspect.
- **Pointcut model**. An architecture fragment specifying the components and bindings that the aspect expects from the base in order to be woven *i.e.*, **where** the aspect should be woven. The most precisely defined is the pointcut model, the smallest is the set of potential places where the aspect can be woven, and conversely. For example, if the type of a component is not specified in the pointcut model, this component would be matched by any component of the base architecture, irrespectively of its real type.
- **Composition protocol**. It describes **how** to integrate the *advice model* into the *pointcut model*. When weaving the aspect, the *composition protocol* is automatically contextualized in order to actually weave the aspect into the base model at all the places matching the *pointcut model*.

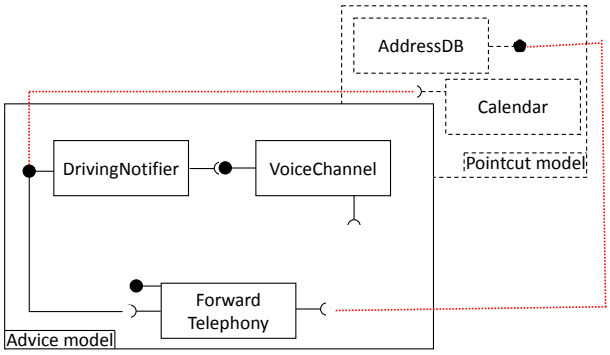

Fig. 1. An Aspect Model

2.3 MDE to automate the Dynamic Reconfiguration

Modern adaptive execution platforms like OSGi [10] propose low-level APIs to reconfigure a system at runtime. It is possible to dynamically reconfigure applications running on these platforms by executing platform-specific reconfiguration scripts specifying which components have to be stopped, which components

and/or bindings should be added and/or removed. These scripts have to be carefully written in order to avoid life-cycle exceptions (*e.g*, when a component is removed while still active), dangling bindings (*e.g.*, when a component is removed while it is still connected to other (client) components).

To prevent errors in writing such error-prone scripts, we leverage MDE techniques to generate safe reconfiguration scripts [3,11,1], as illustrated in Figure 2.

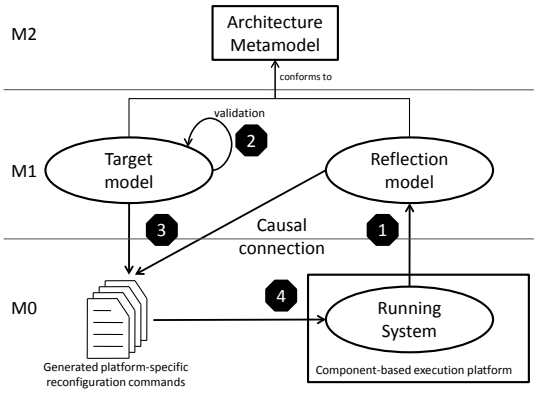

Fig. 2. Leveraging MDE to generate safe reconfiguration scripts

The key idea is to keep an architectural model synchronized with the running system [3]. This reflection model, which conforms to the architecture metamodel, is updated (Figure 2, 1) when significant changes appears in the running system (addition/removal of components/bindings). It is important to note that the reflection model can only be modified according to runtime events. Still, it is possible to work on a copy of this reflection model and modify it: model transformation, aspect model weaving, manipulation by hand in a graphical editor, etc.

When a target architectural model is defined, it is first validated (Figure 2, 2) using classic design-time validation techniques, such as invariant checking [3] or simulation. This new model, if valid, represents the target configuration the running system should reach. We automatically generate the reconfiguration script, which allows to switch the system from its current configuration to the new target. We first perform a model comparison between the source configuration (the reflection model) and the target configuration (Figure 2, 3), which produce an ordered set of reconfiguration commands. This safe sequence of commands is then submitted (Figure 2, 4) to the running system in order to actually reconfigure it. Finally, the reflection model is automatically updated and becomes equivalent to the target model (Figure 2, 1).

3 A Video of the Demo

A preliminary video of the demo can be found at:
`https://transfert.inria.fr/fichiers/6af7a145966fa5ba154a569a448177c8/video.zip`

References

1. Morin, B., Barais, O., Jézéquel, J.M., Fleurey, F., Solberg, A.: Models at runtime to support dynamic adaptation. Computer **42**(10) (October 2009) 46–53
2. Fleurey, F., Solberg, A.: A Domain Specific Modeling Language supporting Specification, Simulation and Execution of Dynamic Adaptive Systems. In: MODELS'09: ACM/IEEE 12th International Conference on Model-Driven Engineering Languages and Systems, Denver, Colorado, USA (oct 2009)
3. Morin, B., Barais, O., Nain, G., Jézéquel, J.M.: Taming Dynamically Adaptive Systems with Models and Aspects. In: ICSE'09: 31st International Conference on Software Engineering, Vancouver, Canada (May 2009)
4. Clements, P., Northrop, L.: Software product lines: practices and patterns. Volume 0201703327. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (2001)
5. David, P., Ledoux, T.: An Aspect-Oriented Approach for Developing Self-Adaptive Fractal Components. In: SC'06: 5th Int. Symposium on Software Composition. Volume 4089 of Lecture Notes in Computer Science., Vienna, Austria (2006) 82–97
6. Goldsby, H., Sawyer, P., Bencomo, N., Cheng, B., Hughes, D.: Goal-Based Modeling of Dynamically Adaptive System Requirements. In: ECBS'08: 15th IEEE International Conference on the Engineering of Computer Based Systems, Belfast, Northern Ireland, IEEE Computer Society (2008) 36–45
7. Morin, B., Barais, O., Jézéquel, J.M.: K@rt: An aspect-oriented and model-oriented framework for dynamic software product lines. In: 3rd International Workshop on Models@Runtime, at MoDELS'08, Toulouse, France (oct 2008)
8. Muller, P.A., Fleurey, F., Jézéquel, J.M.: Weaving executability into object-oriented meta-languages. In L. Briand, S.K., ed.: Proceedings of MODELS/UML'2005. Volume 3713 of LNCS., Montego Bay, Jamaica, Springer (October 2005) 264–278
9. Fleurey, F., Dehlen, V., Bencomo, N., Morin, B., Jézéquel, J.M.: Modeling and Validating Dynamic Adaptation. In: 3rd International Workshop on Models@Runtime, at MODELS'08, Toulouse, France (oct 2008)
10. The OSGi Alliance: OSGi Service Platform Core Specification, Release 4.1 (May 2007) http://www.osgi.org/Specifications/.
11. Morin, B., Fleurey, F., Bencomo, N., Jézéquel, J.M., Solberg, A., Dehlen, V., Blair, G.: An aspect-oriented and model-driven approach for managing dynamic variability. In: MODELS'08: ACM/IEEE 11th International Conference on Model Driven Engineering Languages and Systems, Toulouse, France (October 2008)