View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Dagstuhl Research Online Publication Server

# Graph Transformation Based Models of Dynamic Software Architectures and Architectural Styles

## Abstract

Sebastian Thöne (seb@upb.de) *

International Graduate School of Dynamic Intelligent Systems
University of Paderborn, Germany

Software architectures play an important role in software development. As abstract models of the run-time structure they help to bridge the gap between user requirements and implementation. In the context of e-business, self-healing, or mobile systems, *dynamic architectures* gain more and more importance. They represent systems that do not simply consist of a fixed, static structure, but can react to certain requirements or events by run-time reconfiguration of its components and connections. The availability of those reconfiguration operations depends on the chosen run-time platform which has to support the desired modifications.

The development of such dynamic architectures is a complex task which is usually driven by a stepwise modeling and refinement approach. The software architect derives a first abstract model of the architecture from the user requirements. This model mainly covers the functional aspects and business-related components. Later in the design process, more and more non-functional requirements like security concepts and implementation-specific aspects are integrated into the core functionality. This leads to a sequence of refined architectures down to the real system design for implementation.

A recent example of this general modeling principle is the *Model-Driven Architecture (MDA)* put forward by the OMG. Here, platform-specific details are initially ignored at the model-level to allow for maximum portability. Then, these platform-independent models are refined by adding details required to map to a given target platform. Thus, at each refinement level, one imposes more assumptions on the resources, constraints, and services of the chosen platform.

In software architecture research, *architectural styles* are used to describe families of architectures by common resource types, configuration patterns and constraints. We propose in [1] to consider the restrictions imposed by a certain choice of platform as an architectural style. Moreover, to account for component interactions and platforms that support dynamic reconfigurations, we extend the classical notion of architectural style, which is restricted to structural constraints, by also describing platform-specific communication and reconfiguration mechanisms.

We formalize the architectural styles as graph transformation systems including architectural types, constraints, and graph transformation rules. Based on

* in collaboration with Luciano Baresi (Milano), Reiko Heckel (Paderborn), and Dániel Varró (Budapest)

Dagstuhl Seminar Proceedings 04101
http://drops.dagstuhl.de/opus/volltexte/2005/17

that, a notion of *refinement* is defined in [2], which preserves both semantic correctness and platform consistency. This means that a concrete architecture must satisfy the same requirements as the abstract architecture, and that it must be consistent with constraints and mechanisms imposed by the chosen target platform.

For this purpose, we define refinement relations between abstract and concrete styles which enable us to check for correct refinement of two given architectures. We do not only consider *structural* refinements of fixed configurations but also *behavioral* refinement, which means refining abstract scenarios of component interactions and reconfigurations into platform-specific scenarios.

Since refinements are often tedious and error-prone, a further goal of our work is to automate the derivation of refined models. Indeed, the maximum gain of reusing platform-independent models is achieved if the mapping to various target platforms can be automated. For this purpose, we propose a formulation of the behavioral refinement problem as a reachability problem which can be solved by classical graph transformation and model checking tools.

## References

[1] L. Baresi, R. Heckel, S. Thöne, and D. Varró. Modeling and validation of service-oriented architectures: Application vs. style. In *Proc. ESEC/FSE 03 European Software Engineering Conference and ACM SIGSOFT Symposium on the Foundations of Software Engineering*, pages 68–77. ACM Press, 2003. `wwwcs.upb.de/cs/ag-engels/Papers/2003/BaresiHeckelThoeneVarro_ESEC03.pdf`.

[2] L. Baresi, R. Heckel, S. Thöne, and D. Varró. Style-based refinement of dynamic software architectures. In *Proc. WICSA4 – 4$^{th}$ Working IEEE/IFIP Conference on Software Architecture*, 2004. to appear. `wwwcs.upb.de/cs/ag-engels/Papers/2004/WICSA4_Baresi-Heckel-Thoene-Varro.pdf`.