Architectural Compliance in Component-Based Systems
Foundations, Specification, and Checking of Architectural Rules

Sebastian Herold

The intended software architecture of a software system manifests the earliest and most fundamental design decisions. To ensure that the final software product is consistent with those design decisions and the requirements realized that way, the software architecture has to be refined correctly. This means that the artefacts of detailed design and implementation have to comply with the intended software architecture.

A basic task to ensure architectural compliance is the checking of architectural rules. These rules are constraints resulting from the application of architectural principles, like patterns, reference architectures or guidelines, and restricting the way an architecture can be refined. However, checking those rules is difficult. Manual checks are in general not possible due to the size and complexity of modern software systems. Architectural rules are often described only implicitly and informally, partially due to missing description techniques; thus, realizing powerful tool support is challenging. Moreover, compliance checking tools have to be very flexible. The great variety of architectural rules requires expressive formalisms; in realistic scenarios, furthermore, rules have to be checked in many different artefacts, like code of different programming languages.

This thesis develops an approach to flexible architecture compliance checking in model-based development approaches for component-based systems. It describes a conceptual framework representing component-based systems as relational structures. Models are interpreted as first-order logic statements about those structures; architectural rules, which are considered being fundamental part of architectural models, are logical statements, too. Meta model-specific transformation definitions specify, how instances of the meta model are transformed into logical statements, and which architectural rules are generated if applicable. Compliance between models can be expressed this way as semantically-founded relations between logical formulae.

The developed concepts are evaluated by a case study system following several different architectural principles. Architectural rules are developed for those principles and are checked for compliance. Furthermore, the concepts are implemented by a prototypical compliance checking tool basing upon a logic-based knowledge representation and reasoning system.

Both, the conceptual framework as well as the prototypical implementation, allow very flexible architectural compliance checking. Due to the developed formalization, a broad range of architectural rules can be specified, largely meta model-independently. Integrating new meta models to check compliance between their instances and a software architecture is easy. Aside from the significant improvement of tool support compared with the state of the art, the approach fosters furthermore a new understanding of the role of software architecture; as blue-print for design and implementation, the need for explicitly modelled architectural rules is emphasized.