Towards Continuous Reference Architecture Conformance Analysis

Georg Buchgeher[1] and Rainer Weinreich[2]

[1] Software Competence Center Hagenberg, Austria
georg.buchgeher@scch.at
[2] Johannes Kepler University Linz, Austria
rainer.weinreich@jku.at

Abstract. Reference architectures (RA) are reusable architectures for artifacts in a particular domain. They can serve as a basis for designing new architectures, but also as a means for quality control during system development. Quality control is performed through checking the conformance of systems in development to (company-wide) reference architectures. If performed manually, reference architecture conformance checking is a time- and resource-intensive process. In this paper we outline an approach for reference architecture conformance checking of application architectures in the banking domain. Reference architectures are defined on the basis of reusable rules, consisting of roles and of constraints on roles and role relationships. Conformance checking can be performed semi-automatically and continuously by automating important steps like the extraction of the actual application architecture, the binding of reference architecture roles to the elements of a specific application architecture, and the evaluation of the reference architecture rules for an application architecture.

Keywords: Software Architecture, Reference Architectures, Conformance Analysis.

1 Introduction

Multiple definitions have been provided for the term reference architecture both in the context of software architecture[1][2][3] and beyond [4][5]. While the perspective on what constitutes a reference architecture and how it is represented still varies quite a bit [5], there is a consensus on some defining characteristics. First, there is a focus on reuse [5][1]. Reference architectures are defined to be reused for the definition of concrete architectures of specific systems. This means they are more generic [2][4] and sometimes at a higher level of abstraction [2] than concrete architectures. Second, they encode important design decisions for applications in a particular domain. This means they are usually domain-specific [3][1], though this may depend on how one defines a domain [5]. Third, reference architectures can take different roles in software development [4]. They can take an instructive role for designing new application architectures, an informative role for sharing architectural knowledge, and a regulative role for restricting the design space of systems in development. A reference architecture can take all three roles, though it depends on how it is actually represented.

In this paper we focus on the regulative role of reference architectures and propose an approach for continuous reference architecture conformance checking. Reference architectures are specified as a set of rules that consist of roles and constraints among these roles and their relationships. Roles defined in a reference architecture specification are mapped onto elements of an existing architecture (model), which is then automatically checked against the defined reference architecture rules. Main contributions of the proposed approach are a way for defining and storing reference architectures as a reusable set of rules, and for automatically binding these rules to and evaluating them for specific application architectures. Since binding and evaluation activities are mostly automated, the approach can be used as a means for continuous quality control. The approach is currently being used for checking the architecture of service-oriented applications to company-wide reference architectures in the banking domain and is based on previous work on architecture extraction and review support [6] in this domain.

2 Approach

The main aim of the proposed approach is support for automatically analyzing the architecture of existing enterprise application systems for conformance to company-wide reference architectures. The approach consists of the two main activities shown in Figure 1: The definition of reusable reference architecture specifications, and the actual use of such a specification for checking the conformance of existing application architectures to the constraints of the reference architecture specification.

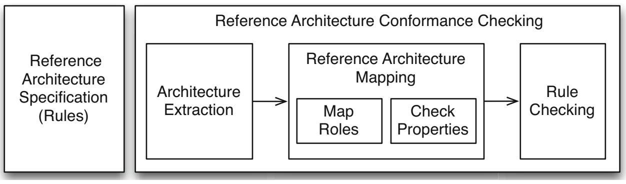
Reference Architecture Specification (Rules)
Reference Architecture Conformance Checking
Architecture Extraction
Reference Architecture Mapping
Map Roles
Check Properties
Rule Checking

Fig. 1. Approach Overview

A reference architecture is defined as a set of rules. Conceptually a rule consists of roles, required and/or forbidden relationships between these roles, and a set of constraints on both roles and relationships. During the checking process roles are mapped onto elements of the architecture of the checked system. These elements are then analyzed for conformance to relationships and constraints defined in the rules of the reference architecture. Roles and constraints can be specified using a structured editor, which works directly on the constraint model representing the reference architecture specification. The actual conformance checking is performed in three steps:

- In the first step the actually implemented architecture is automatically extracted from the system implementation. For a description of the extraction process we refer to [6].
- In the second step roles are assigned to elements of the extracted architecture model. Role assignment is performed semi-automatically. For each role it is possible to specify so-called role assignment rules. For example, a role assignment rule might specify to assign a role to all components based on a specific technology (e.g., EJB) or using a specific naming convention (e.g., to all components containing *Service* in their name). Roles may have additional properties, which are required for analysis. During role assignment we check whether elements with this role are able to provide the required properties. If properties cannot by provided by the elements themselves, they can be provided manually by the user or through extensions to the conformance checker.
- The final step of the checking process is the actual conformance checking. In this step the rules of the reference architecture are evaluated for an application architecture with assigned roles.

3 Related Work

Software architecture conformance checking is addressed by numerous approaches in both research and practice. Dependency analysis approaches like Lattix, Structure 101 and SonarJ support architecture/implementation conformance checking at the programming language abstraction level by analyzing source code dependencies. Analysis at higher abstraction levels, and analysis of information beyond static dependencies (e.g., communication protocols used) is not supported. Dependency analysis approaches also provide no support for defining reusable reference architectures.

Schmerl and Garlan [7] describe an approach for defining and automatically analyzing architectural styles based on the ACME/Armani architecture description language [8]. Their approach consists of two separate activities. (1) The definition of an architectural style and (2) the definition of architecture design models based on a previously defined architectural style. Their work is targeted at defining and checking new architectures based on architectural styles, and not at the continuous conformance checking of already implemented systems as supported in our approach.

Deiters et al. [9] present an approach with similar concepts to ours, which is based on so-called architectural building blocks (ABBs). Rules and constraints are Prolog-like fact bases which are defined textually in their approach, while we use a projectional editor on a constraint model. Their architecture model is simpler (entities and dependencies) and automatic role assignment is not supported in their approach. The differences are mainly because their approach is also more targeted at supporting composition during design than at continuous conformance checking during system evolution and development.

4 Conclusion

We have presented an approach for reference architecture conformance checking. The approach is currently being used by architects as a means for continuous quality control for enterprise applications in the banking domain. However, the developed concepts and tools are not limited to this domain. The basic concepts for defining reference architectures like roles, properties, relationships, and constraints are quite general and could also be used for checking the conformance to reference architectures in other domains, to patterns, and to architectural styles. We are currently investigating the use of the approach for automatically checking the correct application of security patterns. We are also working on strategies for eliminating human intervention during the checking process, which might currently still be required in some cases (like the provisioning of missing property values during role assignment).

References

1. Nakagawa, E.Y., Oliveira Antonino, P., Becker, M.: Reference Architecture and Product Line Architecture: A Subtle But Critical Difference. In: Crnkovic, I., Gruhn, V., Book, M. (eds.) ECSA 2011. LNCS, vol. 6903, pp. 207–211. Springer, Heidelberg (2011)
2. Angelov, S., Trienekens, J.J.M., Grefen, P.W.P.J.: Towards a method for the evaluation of reference architectures: Experiences from a case. In: Morrison, R., Balasubramaniam, D., Falkner, K. (eds.) ECSA 2008. LNCS, vol. 5292, pp. 225–240. Springer, Heidelberg (2008)
3. Hofmeister, C., Nord, R., Soni, D.: Applied Software Architecture. Addison-Wesley Professional (November 1999)
4. Greefhorst, D., Proper, E.: Architecture Principles: The Cornerstones of Enterprise Architecture, vol. 4. Springer (2011)
5. Cloutier, R., Muller, G., Verma, D., Nilchiani, R., Hole, E., Bone, M.: The concept of reference architectures. Systems Engineering, 14–27 (2009)
6. Weinreich, R., Miesbauer, C., Buchgeher, G., Kriechbaum, T.: Extracting and facilitating architecture in service-oriented software systems. In: 2012 Joint 10th IEEE/IFIP Working Conference on Software Architecture & 6th European Conference on Software Architecture (WICSA-ECSA 2012). IEEE (2012)
7. Schmerl, B., Garlan, D.: Acmestudio: supporting style-centered architecture development. In: Proceedings of 26th International Conference on Software Engineering, ICSE 2004, pp. 704–705 (2004)
8. Monroe, R.T.: Capturing Software Architecture Design Expertise with Armani. Carnegie-mellon univ pittsburgh pa school of computer Science (October 2001)
9. Deiters, C., Rausch, A.: A constructive approach to compositional architecture design. In: Crnkovic, I., Gruhn, V., Book, M. (eds.) ECSA 2011. LNCS, vol. 6903, pp. 75–82. Springer, Heidelberg (2011)