Bridging the Concrete and Logical Domains for Software Architecture Reconstruction

Claudio Riva
Nokia Research Center
P.O. Box 407, FIN-00045
Helsinki, Finland
{claudio.riva}@nokia.com

Abstract

Software architecture reconstruction is essentially the process of creating a set of architectural views with the information recovered from the actual implementation of a software system. While recovering the basic facts about the implementation can be automated with source code analyzers, mapping them to logical concepts is mainly a conceptual activity. We investigate the problem of how to bridge the logical and concrete domains for the purpose of creating architectural views. Our conclusion is that there is still a considerable gap between the theoretical framework and the practical experiences. This hinders the possibility of developing a general, rigorous and effective architecture reconstruction method.

1 Introduction

Software architecture reconstruction is the process of obtaining a documented architecture for an existing system. It is a reverse engineering process where the available evidence (such as source code, existing documentation, interviews with the experts) is analyzed in order to create a description of the high-level logical dependencies of the system (i.e. its software architecture). In our previous work [6], we have proposed a view-based architecture reconstruction process where we explicitly specify the views to recover. We distinguish between the target views (i.e. the goals of the reconstruction) and the source views (i.e. the sources of information).

While the architectural views are focused on the *logical* aspects of the system, the source views model the *concrete* facts about the implementation. This gap often causes a mismatch between the results of the reverse engineering methods (mainly focused on the code-level aspects) and the needs of the software architecture audience (mainly interested in the logical and runtime dependencies). In this paper, we investigate the research question of how to bridge the logical and concrete domains for the purpose of creating architectural views. Any architecture reconstruction method must pay a careful attention to this aspect. Our approach is the following: (1) we outline a reference framework of viewpoints based on the current state of art in the fields of software architecture and reverse engineering, (2) we conduct one case study where the viewpoints have been explicitly defined and, (3) we draw several observations by comparing the viewpoints used in practice against the reference framework.

2 Overview of the reference viewpoints

Although a number of architectural viewpoints has been proposed by various authors, there is no general consensus on what viewpoints are needed for forward architecting. Some viewpoints from different authors are actually very similar while others are focused on very specific domains. Moreover software practitioners typically develop their own set of viewpoints, architectural styles, rules and policies that suit their own domain. In their classical paper [4], D. Perry and A. Wolf noted that even if the system was originally designed according to standard architectural styles, the ubiquitous customization of architectural elements turns the system into an unique creation where different architectural styles are overlaid.

We propose a reference framework that consists of three distinct layers: code, design and architecture. At the top of the framework, the architecture viewpoints are concerned with the logical aspects of the software system and provide a comprehensive and abstract understanding of the important design decisions. They are focused on the *logical* aspects and not on the *concrete* details of the implementation that are addressed by the design and code viewpoints. The book of P. Clements et al. [1] proposes a unified set of archi-

tectural viewpoints for documenting software architectures that we take as a reference for this layer.

The design viewpoint describes the essence of a software program at an abstract level. By providing an authentic representation of the source code (but with less details than the code viewpoint), it documents the design decisions at the class and function level about the implementation of the logical design with concrete elements. We can establish a one-to-one mapping between elements in the source code and the elements in the design view. As a reference, we take the FAMIX specification that models the design aspects of object oriented software in a language-independent way [2].

3 Observations on the case study

We have applied the view-based reconstruction process to one software system developed by Nokia. The goal was to recover an architectural model of the overall structure of the system and to check the conformance of the implementation against certain architectural rules. We created an ad hoc reconstruction process that produces the information needed for the conformance checking as we have presented in our previous work [5]. The case study demonstrates how the viewpoints have been explicitly defined with the stakeholders in order to address their specific needs. We can make several general observations about our initial research question:

- The gap between the design and architecture viewpoints is considerable. There is no simple guideline for mapping the abstract and generic architectural concepts of the architecture layer to the elements of the underlying design layer. The case study shows that such mapping has been carefully recovered by the implementation by examining the existing documentation and interviewing the experts. Without their help, we would have risked of missing or misinterpreting the important aspects of the system.

- The concepts of the architecture viewpoints are too generic and open to various interpretations. This hinders the development of a general and precise reconstruction process. This also strengthes the importance of concept determination activity for tailoring the reconstruction to the architectural style.

- The design viewpoint fails to formalize those concepts that are not strictly part of the design domain. The source viewpoint of the case study contains elements extracted from the build process, from the symbol definitions and from the organizational structure. In the present form, the design viewpoint is incomplete as a source viewpoint but it typically needs to be augmented with additional concepts.

- The architecture viewpoints are not suitable to represent the target viewpoints of the reconstruction process. While there are some similarities, we can arduously map the target viewpoints of the case study to the reference viewpoints of the architecture layer.

- The reference framework succeeds in providing the basic material and understanding for creating the source and target viewpoints.

4 Conclusions and Future Work

In this article we investigate the problem of bridging the logical and concrete domains to support architecture reconstruction. We can conclude that there is still a considerable gap between the theoretical framework based of viewpoints and the practical reconstruction of software architectures. In the practical case, linking the logical and concrete domains is a manual and peculiar process that has been achieved by reasoning on the architectural concepts with the help of the experts. Moreover, the experience with the particular system still plays a key role for delivering useful architectural views. Hence, the difficulties of developing a general, rigorous and effective architecture reconstruction method.

In the future, we will evaluate how the recent UML 2.0 specification suits our needs. We also invite the community to conduct similar experiences by comparing their reconstruction experiences against the reference framework that we have proposed in this article.

References

[1] P. Clements, F. Bachmann, L. Bass, D. Garlan, J. Ivers, R. Little, R. Nord, and J. Stafford. *Documenting Software Architectures: Views and Beyond*. Addison Wesley, 2003.

[2] S. Demeyer, S. Tichelaar, and S. Ducasse. FAMIX 2.1 – the FAMOOS information exchange model. Technical report, University of Bern, 2001.

[3] D. Garlan, R. Allen, and J. Ockerbloom. Architectural mismatch, or, why it's hard to build systems out of existing parts. *IEEE Software*, 12(6):17–26, November 1995.

[4] D. E. Perry and A. L. Wolf. Foundations for the study of software architecture. *ACM SIGSOFT Software Engineering Notes*, 17(4):40–52, October 1992.

[5] C. Riva. *View-based Software Architecture Reconstruction*. PhD thesis, Vienna University of Technology, October 2004.

[6] A. van Deursen, C. Hofmeister, R. Koschke, L. Moonen, and C. Riva. Symphony: View-driven software architecture reconstruction. In *Proc. of 4th Working IEEE / IFIP Conference on Software Architecture (WICSA 2004), 12-15 June 2004, Oslo, Norway*, pages 122–132. IEEE Computer Society, 2004.