Autom Software Eng (2006) 13: 199–200
DOI 10.1007/s10515-006-7735-7

Introduction

Software architecture is now broadly recognized as a critical element in the successful development and evolution of large-scale software-intensive systems. A software architecture is a "blueprint" for a software development effort. It is both a technical and a social/managerial document that is used to guide and control all aspects of software development. An architecture description can be analyzed to determine whether all functional and quality requirements can be fulfilled, and it is then implemented. The architecture description helps to communicate with all of the system's stakeholders, including customers, management, and programmers.

But once the system is built, new requirements will arise and the system will evolve. In most cases the system will evolve in ways that were not anticipated by the original architects. And the architecture will, over time, inevitably erode. Along with the divergence of the "as-built" architecture from the "as-designed" architecture goes—in most cases—the erosion of the architectural documentation, The benefits of having an architecture description in the first place are lost. The software architecture description no longer serves as a blueprint for construction, an artifact that can be analyzed, a means of measuring and managing progress, or a communication vehicle. And this erosion is, sadly, the norm, not the exception.

Software architecture recovery aims at reconstructing views of the as-built architecture, so that the benefits of having an architecture description in the first place can be restored to a project. But architecture recovery is a complex task, and is exacerbated by the sheer size and complexity of the artifacts involved (for example, millions of lines of code with complex interrelationships, and dozens of commercial components), by the fact that whatever documentation exists is typically out of date, and by the fact that the system's original stakeholders may have moved on to other projects or even other lives.

In the past 10 years the fields of software architecture and software architecture recovery have grown up. Now there are many conferences, workshops, professional organizations, academic and industrial courses, and professional standards associated with software architecture. Methods and notations exist to model and analyze architectures during design and development. Techniques have also been developed to reconstruct architectural views of existing legacy systems in the post-delivery phase of a system's life cycle. However, despite these achievements, there are still many open research questions. In this special issue of ASE, we concentrate on the relationships between modelling an architecture of a new system and reconstructing and evolving the architecture of an existing system.

Existing systems form the starting point for all three papers accepted for this special issue. Koning and Van Vliet consider the architectural documentation of four different systems, comprising over 450 pages. They search for the best way to organize such descriptions, so that stakeholders can answer their architectural questions most easily. Medvidovic takes the source code of existing systems as starting point, and proposes an architecture reconstruction method that is driven by the changes that are to be made. A key characteristic of his method is that it aims on recovering not only a system's components, but also its connectors and the architectural styles used. Gang et al. focus on the run time architecture of a system, covering such concerns as the actual number of threads, the number of network connections, number of component instances, etc. Their approach allows engineers to monitor systems at run time so that they can make appropriate architectural changes later on, an approach they illustrate by means of a J2EE compliant application server.

The papers included in this journal issue are an outgrowth from a week-long seminar that was held in Dagstuhl, Germany in February of 2003. This seminar was convened because we, the editors of this special issue, perceived a communication and awareness gap between the two sub-fields of software architecture. The fact is that, to a large extent, these two different research topics have been examined separately in two barely-overlapping research communities of forward and reverse engineering. That seminar, and this special issue, are an attempt to examine and encourage relationships between these two critically important sub-fields.

After the Dagstuhl seminar, we invited all participants in particular and the whole architecture modelling and reconstruction community in general in an open call for participation to submit a paper to this special issue. As a result, the paper selection in this special issue is authored by people from both the Dagstuhl seminar and the wider community. We would like to thank all the authors who submitted a paper to this special issue, as well as the many anonymous reviewers who helped make this special issue possible.

Guest Editors
Rick Kazman
Arie van Deursen
Rainer Koschke