Chapter 9

UNDERSTANDING ARCHITECTURE THROUGH STRUCTURE AND BEHAVIOR VISUALIZATION

Dirk Heuzeroth
University of Karlsruhe, IPD, Innovative Programming
Adenauerring 20a, D-76128 Karlsruhe, Germany

Welf Löwe
University of Växjö, MSI, Software Technology Group
Vejdes plats 7, SE-351 95 Växjö, Sweden

1. INTRODUCTION

Understanding the architecture of a legacy system is essential for the further development, the maintenance and the re-engineering of this system. Unfortunately, architecture is hardly documented in such systems. The only trustworthy source of information is the system implementation. Hence, the architecture has to be retrieved from this source. As real world legacy systems tend to be large, the source code cannot be read directly. Instead, we propose to use (semi-)automatic program analyses to extract the information.

As these analyses are not unique by nature, system engineers have to be involved to accept or reject certain results proposed by the automatic analyses. Hence, the result of such analyses ought to be presented in a form that is intuitive to the system engineer. Therefore, the program analysis must go hand in hand with interactive software visualizations.
The combination of program analyses and software visualization techniques is crucial to succeed in comprehending legacy systems' architectures: plain program analysis results are hard to capture by the software engineers. Moreover, the analyses have to be controlled by the software engineers interactively, making it necessary to assess intermediate results. Such an assessment is preferably based on software visualizations. However, plain structure and behavior visualizations of software cannot provide the right abstractions to gain a system's architecture. Software engineers would drown in the flood of information. For large legacy systems, these abstractions can only be achieved by program analyses.

The major task in software architecture comprehension is the identification of components and the essential communications between them. Given the source of a legacy system, program analyses ought to be able to identify components and communications.

Components are larger units of "coherent" modules or classes. The notion of coherency usually mixes static and dynamic system properties: it includes structural connection among the modules or classes in the call or inheritance graphs (static information). Additionally, it requires strong interactions between the modules or classes by actual calls (dynamic information).

The essential communications between the components define the transfer of data independently of their implementations by simple calls, shared memory accesses, events, or callbacks. Many analyses only picture this implementation of communication. Instead, we strive to analyze the essential pattern instead of its implementation. Again, static structure analysis alone is insufficient. It often comes up with misleading results regarding source and target of the communication: Assume, e.g., communication is implemented by an event-listener pattern. The source of the communication provides a method called by the target to add itself as an event listener. Note that the direction of this call is opposed to the direction of the essential communication. Moreover, the event source captures the listeners in a container of abstract listener objects. There is usually no static type information pointing back to the communication target. This connection is only visible via the object identifiers captured in the communication source – such information is runtime information.

We propose to provide multiple graphical views combining different aspects of the software to understand. As sketched above, both, components and essential communication among them, are defined by (static) structure and (dynamic) behavior and, hence, require an understanding of static and dynamic system properties. We therefore offer static and dynamic views: Static program information captures the program structure, but even elaborated static analysis techniques obtain only little information on the