Architecture Analysis needs an Open Source Process

Auke Jilderda
Philips Research Laboratories
Prof. Holstlaan 4
5656 AA Eindhoven, The Netherlands
auke.jilderda@philips.com

Tobias Rötschke
Philips Research Laboratories
Prof. Holstlaan 4
5656 AA Eindhoven, The Netherlands
tobias.roetschke@philips.com

April 6, 2001

Abstract

According to our experience within Philips, industrial embedded systems are increasingly large and software intensive. While hardware design used to require the highest effort only a decade ago, software has taken over this role right now, and its share is growing fast. Considering the high value of such systems (e.g. Medical Imaging, Business Communication), their typical lifetime is 10 years or even longer. During this time, clients require system updates to add or modify features, while the system should keep running continuously to avoid financial losses. Thus, proper management of the software architecture is essential.

Software Architects take the responsiblity for this task, and they continuously need static as well as dynamic information about various structural and behavioural aspects of the system. To cope with large amounts of available facts, automated analysis tools are needed to extract, process, visualize and verify architectural information. While versatile reverse engineering tools are quite common in programming, this is not yet true for architecture analysis. Programming languages are usually well-standardized, but the definition and usage of architectural concepts varies between different product development departments. To solve a concrete analysis problem in such a department, different tools must be glued together and modified or extended with additional functionality. Although tool interoperability of reengineering tools will become easier in future, as a standard exchange format has been introduced, the resulting analysis tools sets are likely to be very diverse, having insufficient features and are difficult to reuse, especially as soon as the tool developer leaves the team.

The authors are convinced that the combination of two approaches will provide the missing infrastructure to improve the current situation:

- Decomposition of analysis tools into a shared architecture of self-contained services.
- Introduction of an open source process for the realization of an architecture analysis framework.

Like other products, analysis tools make use of various technologies which differ in added value and innovation for the developing company. A development team should spend most effort on the most business-relevant technology. For the rest, it should co-operate with others or even try to outsource or buy the required technology . As indicated above, analysis functionality related to specific architectural concepts, concrete rules to be checked, is critical for a producer of industrial embedded systems. On the other hand, a large part of analysis tools is generic base technology, like parsers, graph browsers, etc. Isolating these different kinds of services and publishing them on a central repository, will allow to create solutions for concrete analysis problems more efficiently, by just adding missing services and combining them with already existing ones. Because the new services should be added to the repository, the amount of available services will grow continuously and soon all effort can be spent on the services adding the highest value and being most innovative.

To make this approach work, the following steps are necessary:

- Identify appropriate services in existing tools.
- Define and realise an initial framework to prove the concept.
- Determine the value and innovation of different technologies.
- Decide on the correct scope to produce and offer each service (team, company, world-wide).

To make this idea a success, the solution must be approached in small steps from two sides, i.e. designing the framework top-down and integrating present as well as future solutions bottom-up, so that they fit in the growing framework. This evolutionary process will finally help to better manage large software architectures and to improve the product development processes.