Software architecture reconstruction
Krikhaar, R.

Citation for published version (APA):
Krikhaar, R. (1999). Software architecture reconstruction

General rights
It is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), other than for strictly personal, individual use, unless the work is under an open content license (like Creative Commons).

Disclaimer/Complaints regulations
If you believe that digital publication of certain material infringes any of your rights or (privacy) interests, please let the Library know, stating your reasons. In case of a legitimate complaint, the Library will make the material inaccessible and/or remove it from the website. Please Ask the Library: http://uba.uva.nl/en/contact, or a letter to: Library of the University of Amsterdam, Secretariat, Singel 425, 1012 WP Amsterdam, The Netherlands. You will be contacted as soon as possible.
Summary

This thesis concerns Software Architecture Reconstruction of large embedded systems of the kind developed at Philips (MRI scanners, telephony switching systems, etc.). These systems typically consist of millions of lines of code\(^1\), from which the first lines of code may have been developed more than fifteen years ago. A complete crew of software developers, typically sixty persons, continuously maintains and extends the system with new functionality.

Chapter 1 discusses the term software architecture. It concerns the design of simple and clear structures to be able to share software code among different products. A well designed architecture can be reused in different products. We must note that the term software architecture is somewhat ambiguous. Therefore, we discuss a number of definitions and views on software architecture. Finally, a number of product’s aims from a business perspective are discussed. For example, a product should be available in the market as soon as possible. The business goals determine the objectives of an architecture, for example possibilities to reuse parts of the software in other systems. On its turn these architectural objectives can be translated into architectural patterns.

In Chapter 2 we give an overview of the SAR (Software Architecture Reconstruction) method. First, we introduce terminology like software architecting (the construction of a new software architecture), reverse architecting (the process of making explicit the software architecture of an existing system) and architecture improvement. Next, the basic notions of the SAR method are discussed: InfoPacks, ArchiSpets and software architecture reconstruction levels. InfoPacks describe information extraction from software and ArchiSpets describe aspects of software architecture. A number

\(^{1}\)To print a system of two million lines of code, we need a pile of 300 books of the size of this thesis.
of InfoPacks and ArchiSpects can be defined for each architectural view. The following software architecture reconstruction levels are identified: initial, described, redefined, managed and optimized. A framework (with a focus on the module view and code view) of the SAR method is given in Table S.1. The cells of the table contain various InfoPacks and ArchiSpects discussed in this thesis.

In Chapter 3 Relation Partition Algebra (RPA) is discussed. Relation Partition Algebra is an algebra based on sets, binary relations and operations on them. Partitions play a special role in the algebra, which can be expressed in so-called part-of relations. In particular, dedicated operations
upon relations and part-of relations make the algebra very useful for software architecture. Multi-relations are an extension of binary relations, which were found to be very useful for architecture analysis. In the SAR method, we consequently apply RPA to describe the InfoPacks and ArchiSpects.

In Chapter 4 we show how the software architecture of an existing system can be described by defining a number of InfoPacks and ArchiSpects. The ArchiSpects Source Code Organisation and Build Process belong to the code view and the ArchiSpects Software Concepts Model, Component Dependency and Using and Used Interface belong to the module view of software architecture. The InfoPacks that are required to reconstruct these ArchiSpects are also discussed.

In Chapter 5 we describe a number of ArchiSpects that support the improvement (or redefinition) of an existing software architecture. The ArchiSpects Component Coupling, Cohesion and Coupling and Aspect Coupling are discussed. These ArchiSpects can be used by an architect to analyse the impact of the introduction of certain architectural changes.

In Chapter 6 we discuss architecture verification, i.e. the process of checking whether the implementation agrees with the defined software architecture. Therefore, we discuss the ArchiSpects (Layering Conformance, Usage Control, Aspect Conformance, and Generics and Specifics) that can help to manage a software architecture.

Chapter 7 contains some concluding remarks and recommendations.

The appendices contain the extraction, abstraction and presentation tools used to reconstruct and present ArchiSpects of a number of Philips systems. The last appendix contains an overview of the RPA operators.