6 E-CARES research project: Interactive, stakeholder-tailored re-engineering

Simon Giesecke
Dept. of Computer Science III, RWTH Aachen University, 52056 Aachen, Germany
giesecke@i3.informatik.rwth-aachen.de

André Marburger
Dept. of Computer Science III, RWTH Aachen University, 52056 Aachen, Germany
marand@cs.rwth-aachen.de

6.1 Introduction

The E-CARES project, which was initiated in 1999, is committed to researching reverse and re-engineering approaches of telecommunications software systems based on a graph transformation infrastructure. Ericsson's AXE10 mobile-services switching center serves as a case study during the project. The progress of the E-CARES research project has been presented to this forum in the previous years. In this paper, we first briefly present the current status of the project (Section 6.2), in particular the developments in the last two years. Afterwards, we point to several ideas for future directions of the project in Section 6.3, particularly the use of reverse engineering repositories for communication, explorative re-engineering, and improved interoperability of reverse and re-engineering tool suites.

6.2 Current state of the project

First results from state machine extraction, which was proposed in [5], have been obtained by analyzing more than two million lines of PLEX code spread over more than a hundred compilation units (blocks). These results show that certain assumptions regarding the structure of the state machines are often violated, so that we had to partly revise our extraction algorithm.

Further progress has been achieved concerning static link chain analysis exploiting combinations of static and dynamic analyses, as well as visualizing signal traces [6].

In the beginning of the E-CARES project, only the PLEX programming language was supported. In the meantime, support for the C programming language was added. As part of the process of adding C support, the whole prototype was restructured, so that it is now possible to support a variety of languages with little effort. If one accepts that language-specific features will be neglected, it is possible to add support for a new language without changing the underlying graph schema, as long as it adheres to an imperative paradigm. It is then only necessary to provide a parser generating compatible output.

In the near future, Ericsson's experts themselves will be able to use and evaluate the tool at their site. We expect improved feedback on the adequacy of the features provided by the E-CARES prototype.

6.3 Future directions

It would be desirable to complete the existing reverse engineering tool suite towards a re-engineering tool suite, as already planned in the starting phase of E-CARES. However, in order to gain new insights into the problems of re-engineering, an approach is taken that is in some way substantially different from the numerous previous approaches. Being built by means of the high-level graph transformation language PROGRES [7], the existing prototype stands out from the majority of reverse engineering tools in the dimension of the underlying specification language. It is interesting in itself to investigate the advantages and disadvantages of this approach when extending the tools towards re-engineering support. Other specifics of the intended approach are presented below.

Communication through reverse engineering tools. There are approaches to software architecture focusing on the use of architectural descriptions as a means of communication between different stake-holders [2]. Different stake-holders have differing background knowledge, and different interests in the software system and its description documents. Similarly, the descriptions of a software system regained by reverse engineering techniques can be used for communication. Reverse engineered descriptions

are commonly not at a genuine architectural level. In many organizational contexts, however, they might be the only descriptions that are apt to be used, since original architectural descriptions might not be available at all or too outdated to be useful. Communication through these descriptions might be realized with less intrusion to an organization's processes than that caused by direct migration to an architecture-centered process. Thus, this approach might have more potential for real impact on industrial practice.

One possible usage scenario of communication through reverse engineering tools involves testers and system architects as stake-holders. Testers are able to provide typical signal traces, and may use the reverse engineering system themselves to visualize the traces in the context of the actual system structure. System architects may use these traces for identifying communication hot spots, which may help in deciding where to split a subsystem, for example. If this kind of information is placed in the reverse engineering repository, it may be easily used by all relevant stake-holders.

Interactive, explorative re-engineering. Based on our reverse engineering tool prototype, we plan to investigate methods for interactive and explorative re-engineering in the domain of telecommunications systems. It is difficult and not sustainable to provide a tool pursuing a fixed re-engineering method, which can be applied to a software system once. Instead, small re-engineering steps may be performed throughout time. These steps may correspond to recurring tasks, but probably there will be substantial variations in the experts' requirements. Therefore, we want to give domain experts a tool at hand, using which they can create re-engineering methods tailored for specific situations. These may not apply to the system as a whole, but to multiple parts of the system. Thus, it is desirable that a re-engineering method may be recorded using one example, reworked into a general set of transformation rules and replayed at other parts of the system. It will have to be examined how a specialization of the PROGRES language, a specific method of use of PROGRES, or a language built on top of PROGRES will be helpful in guiding this process. In the latter case, atomic PROGRES transformations must be identified. It is obvious that the less effort is put into manual reworking, the less general the resulting method will be. Here, it is interesting to find a balance which makes it possible for domain engineers to use the tool after an acceptably short learning period, and which is expressive enough to be of sustainable utility.

An explorative approach to re-engineering has been taken by Jahnke et al. [4]. However, their approach is concerned with database re-engineering, which is related to, but substantially different from re-engineering software, let alone process-centered software systems. They concentrate on the integration of external changes in the subject system. On the other hand, we want to consider the exploration of different re-engineering approaches to a given subject system.

Re-engineering frameworks interoperability. In order to reduce the tendency of re-implementation of functionally equivalent components of re-engineering frameworks, interoperability between different frameworks should be improved [1]. Such efforts have already been pursued in the re-engineering community (e.g. [3]). The use of a common repository interchange format, like GXL [8], provides a starting point for such work, but is not sufficient for most uses. For example, a query language like GReQL or Grok [3] would be a useful facility in the context of E-CARES. The PROGRES language could be seen as a query language as well, but we consider it too generic and complex to be used by a re-engineer. On the other hand, other re-engineering frameworks (e.g. the GUPRO project) could benefit from the interactive visualization and processing framework provided by the E-CARES prototype. It should be possible to exchange parsers between the frameworks, which requires not only a common interchange format, but also a common ground of its semantics and pragmatics. Integration of a language like C++ into the E-CARES graph schema would be feasible, but writing an adequate parser would require a major effort.

Acknowledgments

The work presented here has been generously supported in part by Ericsson Eurolab Deutschland GmbH (EED) and the Deutsche Forschungsgemeinschaft (GK 643).

Literaturverzeichnis

[1] S. Ducasse and S. Tichelaar. Dimensions of reengineering environment infrastructures. *Journal of Software Maintenance*, 15(5):345–373, 2003.

[2] Hasso-Plattner-Institut Potsdam. Initiative "Kommunikation in der Software-Entwicklung". `http://fmc.hpi.uni-potsdam.de/index.php?cat=research&subcat=KommSE/overview-german`. visited 2004-03-23.

[3] R. C. Holt, A. Winter, and J. Wu. Towards a Common Query Language for Reverse Engineering. Fachberichte Informatik 8–2002, Universität Koblenz-Landau, Universität Koblenz-Landau, Institut für Informatik, Rheinau 1, D-56075 Koblenz, 2002.

[4] J. H. Jahnke, W. Schäfer, J. P. Wadsack, and A. Zündorf. Supporting iterations in exploratory database reengineering processes. *Sci. Comput. Program.*, 45(2-3):99–136, 2002.

[5] A. Marburger and D. Herzberg. E-CARES Research Project: Extraction of State Machines from PLEX Code. In *Proc. 4th Workshop Software Reengineering*, pages 21–23. Fachberichte Informatik Universität Koblenz: Koblenz, Germany, 2002.

[6] A. Marburger and B. Westfechtel. Tools for Understanding the Behavior of Telecommunication Systems. In *Proc. 25th Intl. Conference on Software Engineering (ICSE 2003)*, pages 430–441, Portland, Oregon, USA, May 2003. IEEE Computer Society: Los Alamitos CA, USA.

[7] A. Schürr. Programmed graph replacement systems. In G. Rozenberg, editor, *Handbook of Graph Grammars and Computing by Graph Transformation. Vol. I: Foundations*, chapter 7, pages 479–546. World Scientific, 1997.

[8] A. Winter, B. Kullbach, and V. Riediger. An overview of the GXL graph exchange language. In S. Diehl, editor, *Software Visualization: International Seminar*, number 2269 in Lecture Notes in Computer Science, pages 324–. Springer, 2002.