A Systematic Approach for Comparing and Reusing Design Alternatives[1]

Jens Knodel, Thomas Forster, Jean-François Girard
Fraunhofer Institute for Experimental Software Engineering (IESE),
Sauerwiesen 6, D-67661 Kaiserslautern, Germany
{knodel, forster, girard}@iese.fraunhofer.de

Abstract

This work introduces an approach to mine field-tested design solutions when defining the architecture of a new product line. The design comparison approach (DCA) compares design solution alternatives implemented in existing systems and evaluates their advantages and drawbacks. This explicit comparison and analysis enables the development of high quality product line architectures by incorporating field-tested, proven concepts and strategies. The experiences gained so far in case studies support the claim that the resulting architecture profits from the application of the DCA approach.

Keywords: architecture evaluation and reconstruction, design comparison, reverse engineering, product lines.

1. Introduction

Developing software systems is a complex and difficult task. Reuse of existing solutions and artifacts is a promising solution to the challenges of today's software-developing organizations and their needs for reducing cost, effort and time-to-market, the complexity and size of the software systems, the increasing demands for high-quality software and individually customized products for each customer. This is especially true for software product lines where planned reuse is one of the main concepts.

The architectural design is the central phase for decisions that concern the achievement of requirements. The architecture of a software system is defined as the fundamental organization of a system embodied in its components, their relationships to each other and to the environment, and the principles guiding its design and evolution. Usually architects tend to reuse successful solutions based on their background and their experiences. When several software systems are available, the architecture design phase can benefit from the experience in field-tested solutions. The goal of the design comparison approach (DCA) is to explore existing alternatives, to learn about different solutions applied to similar problems, to identify advantages and drawbacks of the solutions, and to rate them with respect to their applicability in the context of the development of new systems. During a design comparison, we apply reverse engineering techniques to mine the existing systems for information needed to perform the actual comparison. DCA does not aim at comparing the complete system architectures and designs. Instead it focuses only on finding the best solutions to specific design problem present in different existing systems.

Design comparisons are initiated by requests from the architects and they produce views that highlight certain architectural aspects of the analyzed systems along with a ranking of the solutions. DCA is expert-driven, iterative and highly context-sensitive to the current design problems. The architects cooperate closely together with reverse engineers that know about the capabilities and the strengths of different reverse engineering techniques and methods.

2. The Design Comparison Approach

When the architect of a product line faces a problematic design decision for the product line architecture, a possible source to solve the problems are solutions given in existing systems. To learn about benefits and consequences of such solutions, the potential reuse candidates (either concepts or implementations) have to be compared and rated. In order to understand the solutions and their properties the product line architect demands information about them in a so-called high-level request. Responses to these requests enable learning about successful solutions and their consequences.

The DCA is initiated by such requests. Together with evaluation criteria (elicited from stakeholders) and optionally a product map these inputs are processed into a response, which summarizes the various characteristics of the solutions embodied in selected systems and rate them with respect to the given criteria. In step five of the approach the product line architect consults the rating to get the best solution. These steps exchange feedback either within or across iterations. DCA iterations produce

[1] This work is partially funded by the European Commission EUREKA 2023/ITEA-ip00009 'FAct based Maturity through Institutionalization Lessons-learned an Involved Exploitation of System-family engineering' (FAMILIES).

responses to high-level requests. A response can result in follow-up iterations or further iterations. DCA consists of the following steps

- **Request Analysis**: the breakdown of high-level request into fine-grained pieces, the elicitation of operationalized criteria and the view specification required to address a request.
- **System Selection**: the selection of the systems included in the comparison.
- **View Refinement and Reconstruction**: the refinement and the reconstruction of new, existing or missing views with the help of reverse engineering techniques.
- **Evaluation**: the assessment of the alternative solution with respect to one criterion with a selection of one of the following evaluation techniques (i.e., simulation, instrumentation, prototyping, context analysis, scenario analysis, model sensitivity analysis) and the creation of a joint abstraction by aggregating the information found in the individual systems in comparison matrices or product line views.
- **Packaging and Rating**: the overall rating of the alternative solutions, the combination of low-level results, and the recording of rationales

DCA is an iterative, expert-driven approach to learn about alternatives, solutions and strategies embodied in existing systems. DCA uses reverse engineering techniques to obtain information from existing systems. During all the steps, feedback, learning effects, and insights gained are used to improve the results of later or ongoing iterations. In short, DCA seeks at mining solutions (concepts and/or implementations) from existing system in order to support the product line architects in their migration work.

3. Case Studies

The two case studies conducted fall in the context of our effort for building the tool infrastructure from which individual tools for product line development are derived. The first case study takes place in the context of defining a graph representation supporting efficient reverse engineering techniques within our Eclipse product line infrastructure. The goal is to identify the best graph representation for our product line from multiple reverse engineering tools previously developed offering solutions: different internal representations, implemented in different languages, designed for different purposes. The second case study involves addressing a request to understand "internal model management" in Eclipse IDE plug-ins. Our IDE tools should share an infrastructure that is compatible and easy to integrate with other plug-ins not developed in-house. The goal in this case study was to reuse both, concepts and code. The hypothesis for both case studies was that the DCA approach pays off.

4. Conclusion

In this paper, we presented DCA, an approach to answer the request of a product line architect facing hard design decisions by extracting and comparing the design alternatives present in field-tested systems. In the two case studies, DCA's systematic analysis of the selected systems provided useful insights about concepts, features and patterns that helped the architect in both cases to design product line architectures, from which a number of high-quality instances are already derived.

Although applying DCA can require additional effort than designing the system without explicitly considering the design solutions of previous systems, in our case studies the architect considered that this effort was well invested. He estimated that it resulted in a better architecture that we attribute to the systematic process for analyzing mature design solutions. The architect expects fewer and less costly rework due to this more extensive analysis. We expect that this facilitates the transition of an organization towards product line engineering. DCA explicitly supports the documentation of the rationales for key design decisions.

In the case studies, we observed that creating conceptual models of distinct solutions contributed to the common understanding, shared vocabulary, and considering a broader design space. Based on this experience we are confident that the additional effort required by DCA pays off for hard design decisions. This hypothesis will be the subject of a future experiment.

Please refer for an extended version of this work to [1] where the approach and the case studies are presented in more detail.

5. References

[1] Knodel, J.; Forster, T.; Girard, J.-F.: Comparing design alternatives from field-tested systems to support product line architecture design. European Conference on Software Maintenance and Reengineering (CSMR), March 2005, Manchester, UK