The handle http://hdl.handle.net/1887/30105 holds various files of this Leiden University dissertation.

Author: Etemadi Idgahi (Etemaadi), Ramin
Title: Quality-driven multi-objective optimization of software architecture design: method, tool, and application
Issue Date: 2014-12-11
In this chapter the motivations, the objectives and the contributions of this study are discussed.

This chapter is structured as follows. Section 1.1 gives a brief background on software architecture optimization and describes the contributions of this dissertation. Section 1.2 introduces the research objectives of this study and the research questions which we are going to address through this dissertation. After that, Section 1.3 discusses the outline of the dissertation for the rest of chapters.

1.1 Problem Statement and Contribution

Software architecting is a people-intensive, non-trivial and demanding task for software engineers to perform. At the same time, its architecting is a fundamental activity of software development because it involves several questions such as balancing the dependencies among components, maximization modularity, and fulfilling of quality requirements.

The architecture is a key enabler for software systems. Besides being crucial for user functionality, the software architecture has deep impact on non-functional properties such as performance, safety, energy consumption and cost. Moreover, software architecture addresses fundamental design choices that are often difficult or very expensive to change in later stages of development. Hence, methods and techniques are needed

\footnote{It should be noted that we have approached the problem of architecture design coming from the area of software architectures. However, for most practical cases (including all case studies in this dissertation) an architecture comprises a hardware architecture and a software architecture. A more precise terminology would be to talk about ‘system architecture’ (as is common in system engineering).

In this dissertation, the term ‘software architecture’ generally denotes the combination of hardware and software architecture.}
Introduction

for designing good software architectures which meet various quality constraints in the early phases of development. The complexity of systems has increased sharply because of customers’ demand for more user functions. Fierce competition makes short time-to-market important. To construct a system that optimizes all its requirements simultaneously is difficult, if not impossible. Therefore, architects are faced with a complex optimization problem. During optimization, a large design space needs to be searched in an efficient manner.

In the recent years, researchers have proposed automated approaches for supporting architects in creating architectural designs: (1) Rule-based approaches and (2) Meta-heuristic-based approaches [MKBR10]. Rule-based approaches try to detect weak points (e.g. bottlenecks) in the architectural model based on predefined rules and apply predefined solutions (tactics, patterns) to alleviate these weak points. Meta-heuristic approaches [BR03] frame the challenge of designing architectures as an optimization problem and iteratively try to improve a candidate solution with regard to a given measure of quality. In this way these algorithms explore very large design spaces to find optimal architectural solutions. The component-based paradigm makes it possible to easily and automatically create variation of architectural designs. Hence, the component-based paradigm is key to meta-heuristic optimization approaches for architecture design.

Evolutionary Algorithms (EA), as a recent meta-heuristic approach, are a common optimization technique for solving software architectural problems. However, EA for generating new solutions uses generic search operators, such as Crossover or Mutate, which are blind to the problem and do not take into account the domain knowledge. To overcome this issue, domain-specific search operators have been proposed by the researchers. The downside of using domain-specific search operators, is that the algorithm might find local optimal solutions. In addition, each domain-specific operator is usually useful only for one specific objective, and this is a threat to the optimality of results in multi-objective problems.

The Software Product Line (SPL) approach focuses on the development of specific products within a well-defined domain by leveraging their commonalities and managing their variabilities in a systematic way in order to obtain large-scale reuse [vdLSR07]. The SPL development paradigm is an approach for developing variant-rich software systems that has been widely adopted in recent years. A SPL is defined as a set of software systems with common managed features [CN01]. Software product line engineering aims to develop these systems by taking advantage of the massive reuse of core assets in order to improve time to market and product quality. In the SPL context, every product on the product lines has its own characteristics, and therefore it has different quality attributes. However, the architectural design is a critical factor SPLs because, ideally, the architecture forms a common basis on top of which different products can be build. The architecture has deep impact on non-functional properties
such as performance, safety, reliability, security, energy consumption and cost.

In the context of software product lines, architects need to design an architecture that can work with different components that offer the same functionality, but differ in their behaviour, performance and other quality characteristics, and hence lead to different overall quality of the system.

In this dissertation, an automated approach for software architecture design is proposed that supports analysis and optimization of multiple quality attributes both for single products as well as for product lines.

The main contributions of this dissertation are:

- the demonstration of a meta-heuristic optimization approach for automated software architecture design in a real industrial system. More specifically, it reports the results of applying our architecture optimization framework to an automotive sub-system that was conducted based on a large-scale real world industrial case study. The framework supports multiple quality attributes including response time, processor utilization, bus utilization, safety and cost.

- the introduction of two novel degrees of freedom for the optimization of software architectures. It presents the usefulness of the topology degree of freedom as well as replication-degree of freedom. It demonstrates how the number of processing nodes and their interconnecting network can be codified to fit into a genetic algorithm genotype and thus be subject to automated synthesis. Our studies show that these extra degrees of freedom lead to better overall software architecture optimization. Moreover, it analyses the effectiveness of these two new degrees of freedom by running a very computationally-intensive experiment against our industrial case study. The results of this case study show us additional evidence for the usefulness of these two novel degrees of freedom.

- a comparison between various combinations of EA search operators (both domain-specific and generic) for multi-objective optimization of software architecture. The domain-specific operators we study are motivated by software architectural anti-patterns. However, each heuristic-based search operator improves only one quality attribute of the solution, which is challenging for multi-objective problems. To address this issue, we develop strategies for mixing generic and domain-specific search operators in evolutionary algorithms that speed up the finding of good solutions.

- the introduction of a new search-based approach for generating a set of optimal software architectural solutions for use in software product lines. This approach extends our architecture optimization framework in the direction of features (which can be considered to exist in the requirement-domain). In our new approach, we add feature models as input to the framework and take into
account the relationship between the software components in the architecture and features in the feature model. Our proposed optimizer addresses this by first searching for architectures that are optimal for individual product. After that, it analyses the commonality of the found optimal solutions and proposes a set of solutions which are suitable for the range of products defined by various feature combinations.

1.2 Research Objectives

We formalize the objectives of the current work via the following research questions:

- **RQ1**
  Can meta-heuristic optimization improve the process of designing efficient architectures for a set of given quality attributes in an industrial domain?

- **RQ2**
  Can enlargement of the optimization search space help the meta-heuristic approach to find better architectural solutions?

- **RQ3**
  In which ways can meta-heuristic optimization be improved in order to make the process of reaching optimal architectural solutions faster?

- **RQ4**
  In what aspects can search-based approaches improve the process of designing a software architecture for a family of products in a software product line?

1.3 Dissertation Outline

The rest of this dissertation is structured in these chapters: Chapter 2 introduces and defines common terminologies, Chapter 3 discusses related work, Chapter 4 introduces our proposed AQOSA framework, Chapter 5 presents three case studies, Chapter 6 introduces two novel degrees of freedom, Chapter 7 proposes problem-specific search operators, Chapter 8 introduces a new search-based approach for integration of feature models and architecture optimization together, Chapter 9 presents a parallel implementation of our framework, Chapter 10 concludes our findings. Brief summaries of these chapters are given next:
Chapter 2 defines the foundations on which this dissertation is built and introduces the used terminology. This chapter discusses (i) optimization and more specifically multi-objective optimization problems, (ii) software architecture, software component and component-based software architecture, (iii) quality of software architectures, and (iv) software product lines.

Chapter 3 discusses the state of the art of tools, methods, and approaches related to our proposed framework. The related work in this chapter is categorized into four categories: (i) the related optimization tools which tackle the software architecture optimization problem, (ii) industrial case studies which use an optimization approach in embedded systems, (iii) related heuristic-based approaches for software architecture optimization, (iv) search-based approaches and techniques in the domain of software product lines.

Chapter 4 introduces a new meta-heuristic optimization framework for automated software architecture design. The framework has been developed as a new implementation from scratch and is named AQOSA for Automated Quality-driven Optimization of Software Architectures. This chapter discusses the design of this framework.

Chapter 5 presents three software architecture design problems to which we apply our AQOSA framework. These cases show the effectiveness and usefulness of an automated quality-driven approach for the problem of software architecture design. These case studies will be the basis for the experiments in the following chapters, as well. To the best of our knowledge this is the largest case study on architecture optimization of real industrial embedded software system.

Chapter 6 introduces two novel degrees of freedom for the optimization of software architectures. We know that meta-heuristic approaches, such as genetic algorithms (GA), use degrees of freedom to automatically generate new alternative solutions. The two degrees of freedom are: (1) the topology of hardware platform, and (2) replication of software components. They can improve the results of the optimization algorithms by enlarging the design space. This chapter analyses the effectiveness of these two new degrees of freedom by running a very computationally-intensive experiment against an industrial case study.

Chapter 7 proposes an approach to make the optimization process faster by employing problem-specific search operators. The chapter discusses the use of architectural patterns and anti-patterns as heuristic-based search operators to reach the optimal solution faster. It also introduces different ways of combining the aforementioned search operators. In this chapter, an experiment was set up to compare various combinations
of heuristic-based search operators for an embedded system architecture problem with multiple objectives based on a performance measurement called Averaged Hausdorff distance.

**Chapter 8** proposes a new search-based approach for generating an optimal software architectural solution that supports a family of products that exist in the context of Software Product Line (SPL). This novel search-based method produces a set of solutions which are suitable for a range of products defined by various feature combinations. In this SPL-aware method, AQOSA will also consider feature models as input to the framework and take into account the relationship between the software components in the architecture and features in the feature model. Hence, the approach applies optimization techniques to each product of the SPL. After that, it analyses the commonality of the optimal solutions and proposes a set of solutions which are suitable for the entire range of products defined by various feature combinations.

**Chapter 9** deals with making the optimization process faster by parallelising the execution. We know that meta-heuristic approaches in multi-objective problems especially for high dimensions mostly take a very long time to be executed. One of the best solutions to speed up this process is by parallelising execution of evolutionary algorithm on multiple computing nodes. This chapter presents the results of parallel execution of evolutionary algorithm for multi-objective optimization of software architecture. This chapter studies two ways for parallelising the execution of evolutionary algorithm.

**Chapter 10** closes with a conclusion and suggestions for future work.

**Appendix** Last, but not least, in Appendix A a sample of AQOSA IR is presented. It shows the source of the AQOSA IR model of one of our case studies. It is encoded in the Eclipse EMF format.