Quality-driven Multi-objective Optimization of Software Architecture Design: Method, Tool, and Application
Etemadi Idgahi, R.

Citation

Version: Corrected Publisher’s Version
License: Licence agreement concerning inclusion of doctoral thesis in the Institutional Repository of the University of Leiden
Downloaded from: https://hdl.handle.net/1887/30105

Note: To cite this publication please use the final published version (if applicable).
The handle http://hdl.handle.net/1887/30105 holds various files of this Leiden University dissertation.

Author: Etemadi Idgahi (Etmaadi), Ramin  
Title: Quality-driven multi-objective optimization of software architecture design: method, tool, and application  
Issue Date: 2014-12-11
In this work we have presented a meta-heuristic optimization approach for automated software architecture design. We have studied 4 research questions in this dissertation. This chapter summaries the findings of this study.

This chapter is structured as follows. Section 10.1 presents our findings with regards to each research question of the dissertation. Section 10.2 proposes some directions for future work in this research area.

10.1 Summary of Findings

In this dissertation we described a meta-heuristic optimization approach for automated software architecture design and its associated tooling that we developed. This approach offers a new tool to aid architects in finding good designs in complex design situations with potentially conflicting multiple quality requirements. Furthermore, the tool reduces the development time and improves the quality of the architecture design. The AQOSA framework supports multiple quality attributes for optimization including response time, processor utilization, bus utilization, safety, and cost. As such it supports more quality properties than any other automated software architecture design tool.

Inspired by the model-driven approach, the framework uses a single integrated representation of the software architecture (AQOSA IR) which is the basis for performing multiple quality analysis based. Moreover, this framework is designed so that it can be easily extended with additional quality attributes.

The approach has been applied on industrial case studies with the aim of finding better solutions than the current realization while fulfilling the same requirements and constraints.
The use of the AQOSA framework improves over the state of the art because:

- AQOSA is modelling language independent. It can interoperate with various architectural modelling languages, in particular UML, AADL. Because AQOSA is based on the component-based paradigm but makes only few assumptions on the component model used.

- AQOSA supports multiple degrees of freedom for automatically generating alternative architectures. In particular, we developed an approach to support variation of the hardware topology as a new degree of freedom.

- AQOSA optimizes multiple quality attributes at once. To the best of our knowledge AQOSA is the first approach which supports evaluation and optimization of five quality attributes simultaneously.

In the following sections, the conclusions collected throughout this dissertation are summarized and used to address the research questions central to this dissertation (Section 1.2).

10.1.1 Research Question 1

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

The case studies in this dissertation demonstrated the usefulness of the AQOSA software architecture optimization framework. These case studies range from business information systems to embedded systems in the automotive industry.

The main case in the dissertation reported on a real-world large-scale industrial case study applying a meta-heuristic optimization approach for automated software architecture design which supports multiple quality attributes. The case showed in an industrial context how meta-heuristic optimization approaches can improve software architecture design with respect to multiple quality attributes, and could suggest a wide range of optimized architectural solutions. Comparing the solutions proposed by AQOSA to the existing realization showed that AQOSA is able to synthesize solutions that are efficient in all quality attributes while fulfilling given constraints. Also, in contrast to human architects who tend to propose solutions based on previous architectures, AQOSA proposes revolutionary solutions. This revolutionary suggestions may open new possibilities to architects.

Although the case study shows the saving of manual effort which is beneficial for time-to-market and development cost, it also shows that the proposed architecture solutions needs to be assessed by human architects. Hence, this dissertation demonstrated how an architecture design framework like AQOSA complements the domain knowledge and experience of the architect, rather than replaces the architect.
The case studies showed that evolutionary multi-objective optimization (EMO) can improve the process of designing efficient architectures for a set of given quality attributes in the following aspects:

- EMO can efficient search through large solution spaces,
- EMO can consider multiple system quality attributes in design and analysis,
- EMO can propose revolutionary solutions.

10.1.2 Research Question 2

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

To address the enlargement of the search space, in this dissertation two novel degrees of freedom were introduced:

1. a new method for varying the topology of software architectures,
2. allowing architecture to replicate software component instances.

Introducing more degrees of freedom essentially enlarges the solution space, and thus the search space. If this enlargement is 'in the right direction', then it allows evolutionary algorithms to find better solutions. We could show by running a very computationally-intensive experiment on an industrial case study, that optimization using this approach indeed finds better software architectures. These experiments bring empirical evidences that show that better solutions can be found by using these new degrees of freedom.

10.1.3 Research Question 3

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

To answer this research question, we consider two different points of view:

Firstly, in this dissertation: (i) the usefulness of problem-specific operators in the software architecture domain was discussed, and (ii) a comparison between various approaches for combining heuristic-based search operators was performed. To do so, knowledge of architecture anti-patterns was implemented by means of problem-specific search operators within an evolutionary algorithm. The results of the case study experiment showed that search operators for improving one objective can be used in multi-objective optimization context. The results indicated that proper combination strategies for heuristic-based search operators can lead optimization algorithms to
optimal solutions faster. However, in order to prevent getting trapped in suboptimal solutions, randomness should always be included in the optimization, especially in the offspring mating process.

Secondly, the dissertation presented the results of two different strategies for parallel execution of our evolutionary optimization approach. The achieved results showed that parallel execution of evolutionary algorithm for software architecture optimization can improve execution time significantly with acceptable efficiency in multi-objective optimization context. The results show that for cases in which the evaluation calculation takes significantly more time compared to the selection calculation (of new candidate solutions), the efficiency of parallelization is considerable. When performed a comparison between an actor-based approach and a MapReduce approach to parallelizing the AQOSA computations. In our case study, the actor-based approach shows the better speedup of these two. This is probably due to the synchronization policy of the MapReduce paradigm that does not match well with the concurrent evaluation tasks that vary a lot in computation time.

10.1.4 Research Question 4

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?

For this research question, this dissertation proposes a novel search-based method for finding optimal software architectural solutions which are applicable for a range of products in a product line. To achieve this, we extended the AQOSA framework to support multiple products at the same time. Moreover, we introduced a notion of “distance” between solutions that is an indicator for the number of changes that need to be made to create one architecture out of another. We demonstrated the application of our proposed approach on an exploratory case study based on an existing sub-system from the automotive industry. The case study showed that our method identifies similar optimal solutions that are applicable to the range of products in the software product line.

To summarize, we showed that our search-based approach can improve the process of designing a set of software architectures for a range of products in a software product line in the following aspects:

- modelling the relationship between feature model and component model,
- evolving architectural solutions for the range of products in the SPL at the same time,
- find similar optimal architecture solutions that are applicable to the range of products in the SPL.
10.2 Future Work

One interesting direction for the future work is interactive search-based approaches. By the means of interactive search, the architect would be able to guide optimization process interactively to a specific solution space area. In this way, search is steered jointly by architect preferences and software optimization algorithm.

As future work regarding the heuristic-based search operators, it is interesting to study situations with an unbalanced number of operators which each change different objectives. For example, 3 operators in favour of one objective and 2 operators in favour of some conflicting objectives. Also, another topic for future work can be studying effects of adding weights to heuristic-based search operators on the results of the optimization process.

For future work in the area of software architecture optimization for multiple products, we suggest the following two directions: Firstly, this approach can be improved in the optimization process by using a co-evolving Pareto fronts technique [KCV02]. Hence, the algorithm can switch some of the archive solutions from one population to the another one, every few generations. By doing that, it might be the case that the optimization process gives us better results in terms of commonality with other Pareto fronts. Secondly, it would be interesting to investigate the results of employing a search-based approach for exploring feature combinations. We know large feature models lead up to million of possibilities for feature combinations. For those situations, going one by one through all of the configurations is not a feasible approach and a search-based approach is needed for that exploration as well. In other words, it would be a complex two-level optimization problem: both at the feature-level and at the architecture-level.

Regarding the future work in the parallel execution, it is interesting to extend the parallelization to include the selection step of the evolutionary algorithm as well. In this way, in addition to the evaluation process, also the selection algorithm could execute in parallel which helps efficiency of parallelization even more.