Automated Design of Software Architectures for Embedded Systems using Evolutionary Multiobjective Optimization

R. Li ∗ R. Etemaadi ∗ M.T.M. Emmerich ∗ M.R.V. Chaudron ∗

∗ Leiden Institute of Advanced Computer Science (LIACS), Leiden University
Postbus 9512, 2300RA, Leiden, The Netherlands
{ruili, etemaadi, emmerich, chaudron}@liacs.nl

ABSTRACT
The design of software architecture for embedded system is one of the big challenges in the research field of modern software engineering. It requires software architects to address a large number of non-functional requirements that can be used to quantify the operation of system. Furthermore, these quality attributes often conflict with each other, for instance, improving system performance often needs more powerful hardware, which could increase the production cost and power consumption in the meantime. In most cases, software designers try to find a set of good architectures by hand. However because of large and combinatorial design space, this process is very time-consuming and error-prone. As a consequence, architects could easily end up with some suboptimal designs. In this paper, we introduce our AQOSA (Automated Quality-driven Optimization of Software Architecture) toolkit which can improve these aforementioned non-functional properties in an automated manner. More precisely, beginning with some initial architectures, AQOSA toolkit can use its optimizer to not only produce several alternatives, but also apply trade-off analysis to these newly created architectures according to multiple attributes of interests.

1. INTRODUCTION
Modern embedded systems are large and complicated and therefore difficult to develop and maintain. For example, real-time systems, which nowadays are intensively applied to application domains such as automobile and multimedia, are often built to guarantee the safety and robustness requirements. To meet these requirements makes the design of real-time systems very challenging.

Under such circumstances, software architecture which is an important field of study in software engineering receives more and more attentions in the last few years. More technically speaking, software architectures describe various aspects of the system, mostly their deployment, behavioral, and structural features. With them, designers have the opportunity to analyze the quality properties of software at a high level and thus can make optimal architectural decisions to satisfy the quality attributes at the very early architectural stage of the project.

In many cases, quality properties conflict with each other, that is, improving one quality property can have a negative impact on others, and thus to construct a system that satisfies all its requirements could be difficult. One possible solution is to use optimization techniques to generate several feasible architectures according to initial models and then select optimal solutions from all alternatives through the trade-off analysis with respect to all quality requirements.

In current practice, this process is normally performed manually to the system design. The drawback of this is that it can be time-consuming and error-prone work, especially for large and complex architectures. For complex applications, having some of this work automated could be a considerable cost saver. To this end we propose our AQOSA toolkit which was developed to automatically improve the non-functional properties of an architectural design and thus enable architects to focus on the higher-level design decisions.

The paper is organized as follows. Section 2 summaries some existing methods which are different from ours. Section 3 explains our proposed AQOSA toolkit, especially the execution procedure, in detail. The case study as well as some experimental results is presented in Section 4. Finally, conclusions and future works are given in Section 5.

2. RELATED WORK
As we emphasized at the very beginning of this paper, it is almost impossible for software architects to manually find optimal architecture designs from not only large but also discontinuous design search space. Researchers have proposed several approaches, especially some metaheuristic-based methods which can automate this process. For instance, Martens et al. [1] introduced approach which could automatically improve software architectures modelled with the Palladio Component Model based on trade-off analysis of performance, reliability, and cost.

ArcheOpterix [2] is another generic framework which optimize architecture models with evolutionary algorithms. It supports only one degree of freedom for exploration, that is allocation of software components. Besides, two quality criteria (data transmission reliability and communication overhead) are defined and the evaluation is based on formal mathematical analysis. Similar to Marten’s approach, ArchiOpterix suffers from the limitation on search freedom and has chance to be trapped by some suboptimal solutions.

To alleviate this issue, our proposed AQOSA toolkit, which deploys both advanced model technology and evolutionary multi-objective optimization algorithms with specially designed genetic encoding scheme, allows not only more quality attributes but also more complex degrees of freedom like exploration of architecture topology.

ALIO/EURO-1
3. AQOSA TOOLKIT

The detailed working process of AQOSA toolkit is illustrated in Figure 1. As shown, the automated optimization process starts with some initial software architectures, which could be designed by domain experts by using some advanced model design tools. Next, these architectures are evaluated and corresponding multiple quality criteria of interests are obtained. More specific, processor utilization, data flow latency, and cost metrics are addressed in this study. At the current stage, the simulation-based approach [1] is used for AQOSA evaluator. Note that the precision of evaluation is highly dependent on the modeling details and the features supported by simulator.

As mentioned earlier, some conflicting quality attributes, such as utilization and cost, are often involved in performance analysis. Thus the domination principle could be adopted by evolutionary optimizer for doing trade-off analysis on quality attributes which are extracted through an extractor based on our performance metrics. Some good architectures are then selected from current available solutions. Furthermore, the evolutionary optimizer could automatically produce new candidate architectures by using reproduction operators like “crossover” and “mutation”.

![Figure 1: The detailed working scheme of AQOSA (Automated Quality-Driven Optimization of Software Architecture) toolkit.](Image)

Next, we will explain some key components and related techniques in detail.

3.1. Modeling and Evaluation Engine

For software architecture modeling, as a natural extension of previous work [3] AQOSA integrates ROBOCOP [4] (Robust Open Component Based Software Architecture for Configurable Devices Project) modeling language. Furthermore, AQOSA also supports AADL [5] (Architecture Analysis & Design Language) which is now widely recognized industrial standard in modeling embedded and real-time architectures. The architect can easily design the initial architecture in OSATE (Open Source AADL Tool Environment) and then import it into AQOSA framework. To use ADeS [6] as the core part of our AQOSA simulation engine, we made some modifications of ADeS in scheduling and added new features for data flow latencies evaluating. More specifically speaking, our evaluation engine first loads an AADL model and creates necessary objects for simulation. After that, it generates system events based on the behaviour annex of the model and follow the events through the model connections till end of flows. For complex and concurrent events, the scheduling module decides which process can take the processor.

At present, we implement three quality properties: processor utilization, data flow latency and architecture cost. By design, AQOSA toolkit can be easily extended to support other quantitative quality criteria of software architectures by introduce new evaluation plug-ins, i.e. for communication lines loads evaluation, we just needed to add a new listener which implements the measurement of the bus load to our simulation engine. Another advantage of AQOSA is that it provides some very flexible API for the interaction between evaluator and various optimization frameworks such as Opt4J and JMetal [7].

3.2. Evolutionary Optimizer

3.2.1. Evolutionary multiobjective optimization

Evolutionary multiobjective optimization (EMO) [7] derives from single objective evolutionary optimization (EO) algorithms and is recognized as a fast growing fields of research. It is relatively simple to implement and wide-spread applicable. In this work, two representative multiobjective optimization algorithms (NSGAII [8] and SPEA2 [9]) from literatures are chosen and applied to one architecture design task for the car radio navigation (CRN) system.

3.2.2. Search problem formulation

From EMO algorithm perspective, architecture design problem can be generalized as following optimization task (see Equation 3.2.2):

$$\text{min } f_m(x), \quad m = 1, 2, \ldots, M$$

$$\text{s.t. } g_j(x) \geq 0 \quad j = 1, 2, \ldots, N$$

(1)

Here, $x$ is a solution and can be of any domain, e.g., real or binary. In the given context, $x$ could be a valid architecture from embedded system design domain. For each solution $x$, there exists $m = 3$ objectives, i.e. $f_1 :$ Processor utilization, $f_2 :$ Cost, and $f_3 :$ Data flow latency. $g_j(x)$ represents a number of constraints which any feasible solution must satisfy. The aim is not only to provide one optimal solution but rather to provide a broad variety of non-dominated solutions representing trade-offs in the three objectives.

3.2.3. Generic degree of freedom to exploration

With specially designed genotype representation, the following degrees of freedom to exploration are implemented: (1) System hardware topology (hypergraph), i.e. processor/bus can be added or removed from the system, (2) Allocation of service instances, (3) Replacement between different hardwares, i.e. one component can be replaced by its counterparts from available hardware repository. Figure 2 shows three system topologies which are supported and valid for car radio navigation (CRN) architecture design (i.e. case study in Section 3).
4. CASE STUDY AND EXPERIMENTAL RESULTS


To validate our proposed AQOSA toolkit, we applied it to one benchmark application - the car radio navigation (CRN) system [10]. The CRN system is constructed according to the component-based paradigm. An overview of the software architecture is depicted in Figure 3.

As can be seen, the CRN system contains three major functional blocks:

- The Man-Machine Interface (MMI), that takes care of all interactions with the end-user, such as handling key inputs and graphical display output.

- The Navigation functionality (NAV) is responsible for destination entry, route planning and turn-by-turn route guidance giving the driver visual advices. The navigation functionality relies on the availability of map database and positioning information.

- The Radio functionality (RAD) is responsible for tuner and volume control as well as handling of traffic message channel information services.

The major challenge is to determine a set of optimal architectures with respect to quality attributes such as processor utilization, data flow latency, and cost. Technically speaking, we investigate how to distribute these aforementioned functionalities over the available resources (processor node in Figure 2) to meet some global requirements. Vector representation in Figure 4 illustrates how the genotype is used to describe possible architecture topologies (Figure 2) as well as mapping of services.

4.2. Experimental Setup and Results

The experimental setup is as follows: two standard evolutionary multiobjective optimization algorithms from Opt4J, Non-dominated Sorting Genetic Algorithm (NSGA-II) and Strength Pareto Evolutionary Approach 2 (SPEA2), will be used. Furthermore, the following parameter settings are adopted: initial population size: 50, parent population size: 25, number of offspring: 25, archive size: 100, number of generation: 50, crossover rate is set to 0.95, constant mutation probability is 0.01. For each algorithm we run AQOSA 20 runs (≈ 10 hours). The resulting archive of optimal solutions can be visualized in the 3-D Pareto front with respect to processor utilization, cost, and data flow latency in Figure 5.

An interesting finding is that resulting pareto front consists of three segmentation (with clearly gap in between). This could be the result of discontinuities in the search space caused by structural transitions. By identifying and mapping each individual from archive back to corresponding design architecture, solutions from same segmentation share the same architectural topology (i.e. Figure 2). This discovery is consistent with our understanding of CRN system, for instance, solutions with topology 3 (solutions with blue color) normally have lower processor utilization and higher cost for the hardware. On the contrary, solutions with topology 1 (red color) have higher processor utilization and lower cost.

The 2-D plot of two quality attributes is presented in Figure 6. In this way, the software architect can make trade-off decision much easier. For instance, the left plot shows the processor utilization over the cost per candidate architecture while the right one

3All three algorithms which we studied show the same behaviour.
indicates the data flow latency over cost. There is no obvious conflict between processor utilization and data flow latency and the corresponding plot is excluded here. Further more, both the attainment surface of one typical run of SPEA2 and the box-plots of the hypervolume indicator \cite{11} for ref. point \((1,1,1)^T\) of archive population for NSGA-II, SPEA2, and random search over 20 runs are presented in Figure \ref{fig:7}.

![Figure 7: The dominated Hypervolume approximation of one typical run of SPEA2 and the box-plots of the hypervolume indicator for NSGA-II, SPEA2, and random search on CRN design problem over 15 runs.](image)

From figure \ref{fig:7} (left), it gets clear that final solutions from archive are mutually non-dominated with respect to three quality attributes investigated. Another observation is that NSGA-II and SPEA2 show the comparable performance with each other (student’s t-test with 1% confidence level), and the results are very similar. Random search, by contrast, shows worst performance.

5. CONCLUSIONS AND OUTLOOK

We presented so-called AQOSA (Automated Quality-driven Optimization of Software Architecture) toolkit. It not only can help software architects to reduce the workload for modeling and evaluating real-world problems, but also can automatically improve quality attributes by using evolutionary multiobjective optimizers. We applied AQOSA on the car radio navigation (CRN) system. It not only can help software architects to reduce the workload for modeling and evaluating real-world problems, but also can automatically improve quality attributes by using evolutionary multiobjective optimizers.

6. ACKNOWLEDGEMENTS

This work has been supported by the Dutch national project OMECA (Optimization of Modular Embedded Computer-vision Architectures) and European project SCALOPES (an ARTEMIS project on SCalable LOw Power Embedded platformS).

7. REFERENCES


