# Quality-driven Multi-objective Optimization of Software Architecture Design: Method, Tool, and Application

Etemadi Idgahi, R.

**Author**: Etemadi Idgahi (Etemaadi), Ramin
**Title**: Quality-driven multi-objective optimization of software architecture design : method, tool, and application
**Issue Date**: 2014-12-11

# Chapter 3

# State of the Art

In this chapter, the state of the art of tools, methods, and approaches are discussed which are related to our proposed framework and its related approaches in the following chapters. Besides the following related works from other research groups, the work presented in this dissertation is built on the previous works [BCdW06] and [LCL10].

In this chapter, related work is categorized into four categories which are structured as follows. Firstly, Section 3.1 details the related optimization tools which tackle the software architecture optimization problem. It should be noted that because each tool supports different objectives it is not practically feasible for us to directly compare the achieved results of our proposed framework with others. Then, in Section 3.2 some industrial case studies are listed which use an optimization approach in embedded systems. Although they use different optimization approaches at different levels, it is useful to compare our proposed approach with them. Section 3.3 discusses over related heuristic-based approaches for software architecture optimization. After that, a few approaches that use evolutionary algorithms to tackle software design problem are listed in Section 3.4. And lastly, Section 3.5 discusses different search-based approaches and techniques in the domain of software product lines.

## 3.1 Related Optimization Tools

### 3.1.1 PerOpteryx

Martens et al. [MKBR10] introduced an approach which can automatically improve software architectures based on trade-off analysis of performance, reliability, and cost by using a multi-objective genetic algorithm. The tool is a part of the Palladio project and is dedicated to the Palladio Component Model (PCM) [BKR09] for modelling and quality prediction. More technically, Layered Queuing Networks (LQN) methods are used for performance property prediction and Markov models are adopted for analysis of reliability. The Palladio component model (PCM) specifies component-based software architectures in a parametric way. Its simulation tool is capable of making performance predictions. Hence, this approach is suitable for systems which are modelled by PCM and focused on performance quality attributes. Instead, our approach is independent from any specific component model or modelling language.

### 3.1.2 ArcheOpteryx

ArcheOpteryx [ABGM09] [MBAG10] is a generic framework which optimizes architecture models by means of evolutionary algorithms. It works on the basis of AADL [FGH06] (Architecture Analysis & Design Language) specifications. Two quality criteria (data transmission reliability and communication overhead) are defined and the evaluation of architectures is based on analytical methods, rather than simulation. It supports only one degree of freedom for exploration, which is allocation (or deployment) of software components to hardware components. ArcheOpteryx specializes in probabilistic analysis of quality attributes and software architecture under uncertainty. Therefore, their tool and their analysis techniques are very suitable for situations with uncertainty. Their approach has different focus than ours, since they focus on probabilistic analysis and uncertainty conditions.

### 3.1.3 Multicube Explorer

Multicube Explorer [PSZ05] [YCAM+11] proposes a design space exploration framework for supporting platform-based design. It allows optimization of parametrized system architectures. It presents a generic and structured framework for run-time resource management of embedded multi-core platforms. Based on the results of design-time multi-objective exploration, it defines a methodology to optimize the run-time allocation and scheduling of different application tasks. The design exploration flow results in a Pareto-optimal set of design alternatives in terms of power and performance trade-offs. The set of operating points can then be used at run-time to decide how the system resources should be distributed over different tasks running on the multiprocessor system on chip. So, it focuses on the link between design-time design space exploration and run-time configuration. However, our approach is not targeted for run-time optimization of configurations. The degrees of freedom for Multicube's optimization process mostly can be characterized as related to the configuration parameters (e.g. scheduling).

### 3.1.4 Artemis

Pimentel presented an overview of the Artemis workbench in [Pim08]. This workbench provides high-level modelling and simulation methods and tools for efficient performance evaluation and exploration of heterogeneous embedded multimedia systems. It consists of a set of methods and tools conceived and integrated in a framework to allow designers to model applications and System-on-Chip-based (multi-processor) architectures at a high level of abstraction, to map the former onto the latter, and to estimate performance through co-simulation of application and architecture models. It supports only one degree of freedom for exploration, which is allocation of software components.

### 3.1.5 KlaperSuite

The KlaperSuite [CFD+11] is a family of tools that offers designers a means to analyse performance and reliability at the architecture or design stage. The approach uses model transformations to automatically generate analysis models (queuing networks) out of an annotated UML design model. The approach focuses on validation of non-functional properties during early stages of design and does not offer optimization.

### 3.1.6 MOSES

MOSES [CPSV08] is a methodology proposed by Cortellessa et.al. which aims at the automated generation of feedback targeted at performance. It aims at systematically evaluating performance prediction results using step-wise refinement. It is built on top of UML-RT (UML for Real-Time) and presents a new implementation based on the UML2 meta-model. A unique feature of this approach is that it tries to diagnose the cause of performance problems in the system model level.

### 3.1.7 Other Related DSE Approaches

Hegedüs et al. [HHRV11] defined Design Space Exploration (DSE) as a process to analyse several "functionally equivalent" implementation alternatives, which meets all design constraints in order to identify the most suitable design choice (solution) based on quality metrics such as cost or dependability. They stated that in model-driven engineering, DSE is applied to find instance models that are (i) reachable from an initial model by a sequence of transformation rules and (ii) satisy a set of structural and numerical constraints. However, our approach using Genetic Algorithm (GA) approach does not start from any specific initial state. The initial population in GA can be spread out in the design space by means of randomized generation. Another difference is that, those approaches use a sequence of transformations to evolve the solution. But in GA, it is possible that a revolutionary solution is proposed as a result of some random mutation(s) in the genotype.

Räihä et al. in [RKM11] introduced a multi-objective genetic synthesis approach for software architecture. However, they cover two different quality attributes of software architectures; modifiability and efficiency. Therefore, although they use GA like our approach, we target different quality attributes.

## 3.2 Related Industrial Case Studies

### 3.2.1 Reliability Optimization for Embedded Systems

Glaß et al. [GLHT10] presented an automatic reliability-aware system-level design methodology to tolerate hardware defects. Real-life case studies from the automotive domain have been used to validate the effectiveness of the proposed techniques. For the reliability analysis and optimization at design time, a real-life specification for an adaptive light control has been explored. For the performance of the proposed online algorithm, a specification that combines six automotive applications including an adaptive cruise control and brake-by-wire system has been used. Similar to our case study, it studies real-life cases from the automotive domain. However, their optimization process tries to optimize two objectives: reliability and cost. Instead, our case study explores five objectives. Moreover, their approach does not use genetic algorithms.

### 3.2.2 Symbolic Multi-objective Design Space Exploration

Lukasiewycz et al. [LGHT08] formalized the problem of design space exploration as multi-objective $0-1$ integer linear programming problem. They proposed a heuristic approach based on Pseudo Boolean (PB) solvers and a complete multi-objective PB solver based on a backtracking algorithm that incorporates the non-dominance relation from multi-objective optimization, called Heuristic multi-objective PB Solver (HPB). To emphasize the efficiency of the proposed algorithms, they applied the algorithms to a real adaptive light control from the automotive area. So, similar to the previous work, it comes from the automotive domain. However, their focus is on the proposed HPB approach and comparison with other algorithms, instead of the industrial case study. In contrast, our case study does not compare design space exploration algorithms.

### 3.2.3 Multiprocessor Systems-on-Chip Synthesis

In [CFL+10] Ceriani et al. compared three evolutionary approaches for real-time, embedded, heterogeneous, multiprocessor systems (Multiprocessor Systems-on-Chip or

MP-SoCs). They considered three approaches: a multi-objective genetic algorithm, multi-objective Simulated Annealing, and multi-objective Tabu Search. In an experimental evaluation, they applied their method to a realistic JPEG compressor on the Xilinx FPGA toolchain. Again, similar to the previous work, although they worked on a real-life case study, their focus is on comparing three exploration algorithms.

### 3.2.4 Exploring Embedded System Architectures

Pimentel et al. [PEP06] presented the Sesame framework, which provides high-level modelling and simulation methods and tools for system-level performance evaluation and exploration of heterogeneous embedded systems. They demonstrated the framework by using a case study which traverses Sesame's exploration trajectory for a Motion-JPEG encoder application. They focused on multiple abstraction levels in their method. So, their framework allows architectural exploration at different levels of abstraction. A Motion-JPEG (M-JPEG) encoder application has been used as industrial case study to illustrate Sesame's modelling methodology and trajectory. Similar to our approach, they used a genetic algorithm for design space exploration. However, they focus on performance simulation. Instead, our approach aims to address other quality attributes in addition to performance as well.

## 3.3 Related Heuristic-based Approaches

### 3.3.1 Architectural Tactics

Koziolek et al. [KKR11] introduced a hybrid approach that incorporates architectural performance "tactics" into an evolutionary optimization process. They defined architectural tactics to improve two quality attributes: Performance and Cost. They implemented those tactics as part of the evolutionary optimization process. They showed that by using tactics, optimization algorithms can achieve better solutions. However, their experiment was conducted in an information systems context and considered 2-objective optimization.

Martens at el. in [MAK+10] proposed another way of hybridization for optimization of software architectures. They propose two-step optimization: first analytic optimization and then a subsequent evolutionary optimization based on the Pareto candidates from the first step. However, this way is totally different from our approach. Because in our approach, we use the domain knowledge as the search operators and therefore, we integrated that knowledge as part of the evolutionary optimization process.

### 3.3.2 Anti-patterns in Palladio

Turbiani et al. [TK11] discussed the advantages of using software performance anti-patterns in an iterative manner. They introduced a couple of performance anti-patterns and defined an automatic approach to detect and solve the bottlenecks in software architectures. They demonstrated that, by applying this technique iteratively, the system performance can be improved significantly. However, they did not discuss quality attributes other than performance. They also did not integrate their approach within an evolutionary optimization process. Thus, the downside of their approach is that without having generic degrees of freedom and involving randomness in the optimization iterations, the optimality of the results is highly dependent on the initial architecture.

## 3.4 Software Design using Evolutionary Algorithms

### 3.4.1 Evolutionary Search in Object-Oriented Class Design

Simons et al. [SPG10] reported the findings of two experimental early life cycle design episodes wherein a human designer and software agents interact and collaborate to jointly steer an evolutionary, multi-objective search toward useful and interesting class designs. Their interactive search-based design approach takes place during upstream design before the design is realized in source code. It involves both a representation of the design problem and a representation of the design solution which are inherently traceable; and the representation of the solution space is a UML class model. Their approach changes the software design through an evolutionary process. Objective fitness functions available to the designer relate to the structural integrity of the solution class designs and include cohesion of classes, coupling between classes, and number of classes in a design. In their approach, they focus on interaction between human designer and software agents during evolutionary process which is different from our goals.

### 3.4.2 Class Responsibility with Genetic Algorithms

Bowman et al. in [BBL10] provided a decision-making approach to re-assign methods and attributes to classes in a class diagram. Their solution is based on a multi-objective genetic algorithm and uses class coupling and cohesion measurement for defining fitness functions. Their approach takes as input a class diagram to be optimized and suggests possible improvements to it. Their solution to the problem of class responsibility assignment in the context of object-oriented analysis and domain class models, suggests that the multi-objective genetic algorithm can help correct suboptimal class responsibility assignment decisions and perform far better than simpler alternative heuristics such as hill climbing and a single objective GA. However, our approach does not aim to optimize at the class diagram level. So, we have different focus from this study.

## 3.5 Related Software Product Lines (SPL)

### 3.5.1 Variability-driven Quality Evaluation in SPL

Etxeberria et al. in [EM08] proposed a novel method for evaluation of a product line. It is clear that the quality evaluation in software product lines is much more complicated than in single-systems because different products within the same product line can require different quality levels and the product line can have variabilities in its design that affect the quality. However, we know that the evaluation of all the products of a product family is very expensive. They presented a method for facilitating cost-effective quality evaluation of a product line taking into consideration variability on quality attributes.

They mentioned that the assessment of all the instances of the product line may not be worthwhile due to the high cost. However, it is possible to shorten product architecture evaluations because *"The product architecture evaluation is a variation of the product-line architecture evaluation as the product architecture is a variation of the product-line architecture and the extent to which product evaluation is a separate, dedicated evaluation depends on the extent to which the product architecture differs in quality-attribute-affecting ways from the product line architecture"* . Therefore, their approach is to create a generic evaluation model with variability that helps to evaluate the product family, thus reducing efforts, as several products can be evaluated together with that model.

### 3.5.2 Search-based Design of SPL

Colanzi in [Col12] proposed an approach for search-based design of the software product line architectures (PLA). She introduced a multi-objective optimization approach to the PLA design to ease the SPL development and to automate the PLA design. So, this approach focuses on search-based design of product lines; however in our approach, we assume that a product line is already designed and fixed. Then, we search to find optimal architectural solutions for the products of that fixed product line. Hence, our approach is not aimed at optimizing the features in a product line.

In another extended work, Colanzi et al. in [CV12] discussed the lessons they have leaned from applying search-based optimization to software product line architectures. They concluded that the results point out it is necessary to use SPL-specific measures and evolutionary operators more sensitive to the SPL context.

### 3.5.3 SPL Configuration

Sayyad et al. in [SIMA13] proposed an evolutionary algorithm, based on the Indicator-Based Evolutionary Algorithm (IBEA), for the problem of software product lines configuration. Specially, they targeted for large scale product lines. They mentioned that software product lines are hard to configure and techniques which work for medium sized product lines fail for much larger product lines such as the Linux kernel with 6000+ features. They found 30 sound solutions for this very large product line; however, we assume any arbitrary sound configuration as an input to our tool and our approach is not aimed at searching for sound configurations in a product line. In short, they showed an approach to automate the configuration of the very large variability models available from LVAT (Linux Variability Analysis Tools) feature model repository.

Moreover, Sayyad et al. in [SMA13] extended the aforementioned work to bring the user preferences in to the perspective. In this work, they presented a high-dimensional multi-objective search approach, which puts each user preference in focus without aggregation, and then incorporates the user preferences in the Pareto dominance criteria, using the Indicator-Based Evolutionary Algorithm (IBEA). They used 5 optimization objectives, namely: to maximize logical (syntactic) correctness, maximize richness of feature offering, minimize cost, maximize code reuse and minimize known defects. As can be seen, they are completely different from our approach objectives.

### 3.5.4 Software Evolution in SPL

Abrahão et al. in [AGHIR12] presented a framework to support the development and evolution of high-quality software product lines. The framework is based on several interrelated models or system views (eg, functionality, variability, quality) and a production plan defined by model transformations that generate a software system that meets both functional and quality requirements. The variability management involves the manipulation of features, represented as cardinality-based feature models, and the support of such variability in the so-called product line core assets. Specifications of the system variability, functionality, and quality can be dealt with models that are independent from each other but where their inter-consistency is assured by means of the relationships defined in this multi-model. The SPL production plan is parametrized by means of the multi-model which specifies the corresponding model transformations that are needed to obtain a specific product with the desired features, functions, and quality. Therefore, they focus on software product lines evolution by using a multi-model approach, which is completely different from our targets.

### 3.5.5 Quality Engineering for SPL

Kolb et al. in [KM06] discussed the quality engineering for software product lines. They provided an understanding on how to perform quality engineering and to systematically assure the quality of product lines and reusable artefacts. They discussed about how quality engineering for software product lines and generic software components needs to be different from traditional software systems and how quality engineering can be performed in the context of product lines. In addition to that, they provided a discussion of the difficulties and challenges of quality engineering for product lines and investigates the implications of product lines and reusable components on quality engineering. To summarize, their goal was to provide an understanding of existing constructive and analytical quality engineering techniques and methods and how these techniques and methods can be applied in a software product line context. The methods and techniques addressed include systematic architecture design and evaluation, architecture conformance checking, inspections, testing, metrics and measurement evaluation, and risk management. For each technique or method, it is discussed what the problems and challenges in the context of software product lines are, which benefits and limitations the technique or method has, and how it has to be extended to be applicable for assuring the quality of software product lines and reusable artefacts. Their approach can be considered as a the translation of classical manual software quality assurance techniques to the domain of software product lines.