OPTIMIZING THE SELECTION OF ARCHITECTURE FOR COMPONENT-BASED SYSTEM

ADIL ALI ABDELAZIZ SAED

A thesis submitted in fulfilment of the
requirement for the award of the degree of
Doctor of Philosophy (Computer Science)

Faculty of Computing
Universiti Technologi Malaysia

MAY 2013

To my beloved mother and soul of my father

ACKNOWLEDGEMENT

In the Name of Allah, Most Gracious, Most Merciful

All praise and thanks are due to Allah, and peace and blessings be upon his messenger, Mohammed (peace be upon him).

I would like to express my heartfelt gratitude to my supervisor Assoc. Professor Dr. Wan Mohd Nasir Wan Kadir who has guided me and without whose guidance and advice this study would not have been possible. He has been incredibly wise, helpful, understanding, and generous throughout the process. He has truly been a mentor and I owe him my deepest thanks. I also wish to thank my co-supervisor, Assoc. Professor Dr.Siti Zaiton Bt. Mohd Hashim for her help and encouragement throughout this study. The International Doctoral Fellowship has played an important role in this study, accordingly, many thanks from the depths of my heart to UTM for granting me the IDF for four semesters.

I have made many friends during my time in UTM and I thank them for their support and encouragement. A great deal of useful information pertaining to the work was found via the World-Wide Web; I thank those who made their materials available by means of this medium and those who kindly answered back to my roll-calls of help sent over the World-Wide Web, among them staff of the postgraduate office and the faculty for the use of facilities and Lab. I also would like to thank the software engineering group members for their co-operation. Finally, I would like to thank my wife and kids Ahd, Amro, Aubada, and Ola for their sacrifices, patience, encouragement, support and understanding.

ABSTRACT

Redundant components are commonly used for solving Redundancy Allocation Problems (RAP) and improving the reliability of complex systems. However, the use of such a strategy to minimize development costs while maintaining high quality attributes for building software architecture is a research challenge. The selection for an optimal architecture to meet this challenge is an inherently complex task due to the high volume of possible architectural candidates and the fundamental conflict between quality attributes. Current software evaluation methods focus on predicting the quality attributes and selecting Commercial-Off-the Shelf (COTS) components for COTS-Based applications rather than utilizing additional architectural evaluation methods that could increase the opportunity for obtaining a cost-effective solution for RAP. In this thesis, an architecture-based approach called Cost-Discount and Build-or-Buy for RAP (CD/BoB-RAP) is introduced to support the decision making for selecting the architecture with optimal components and level of redundancy that satisfies the technical and financial preferences. This approach consists of an optimization model that includes two architectural evaluation methods (CD-RAP and BoB-RAP) and applies three variants of Particle Swarm Optimization (PSO) algorithms. Statistical results showed a 74% reduction on the development cost using CD-RAP on an embedded system case study. Moreover, the application of a maximum possible improvement on the algorithms showed that Penalty Guided PSO (PG-PSO) had enhanced the quality of obtained solutions by 70% to 84% in comparison to other algorithms. The results of the CD-RAP and BoB-RAP were superior when compared to the results obtained from similar approaches. The overall results of this research have proven the potential benefits of the CD/BoB-RAP approach for software architecture evaluation, particularly, in selecting software architecture for minimizing the development cost maintaining a highly reliable system.

ABSTRAK

Komponen lewah sering digunakan untuk menyelesaikan Masalah Peruntukan Lewahan (MPL) dan memperbaiki keutuhan sistem yang kompleks. Walau bagaimanapun penggunaan strategi ini untuk meminimumkan kos pembangunan dan mengekalkan ciri-ciri kualiti yang tinggi dalam pembangunan seni bina perisian masih menjadi cabaran kepada penyelidikan. Pemilihan seni bina yang optimum adalah satu tugas yang sukar oleh sebab jumlah calon seni bina munasabah yang tinggi dan konflik antara sifat-sifat kualiti. Kaedah penilaian perisian pada masa ini memberikan tumpuan kepada peramalan sifat-sifat kualiti dan pemilihan komponen Tersedia Komersial (TK) bagi aplikasi berasaskan TK berbanding dengan penggunaan kaedah penilaian seni bina tambahan yang boleh meningkatkan peluang untuk mendapatkan penyelesaian dengan kos efektif bagi MPL. Dalam kajian ini pendekatan berasaskan seni bina yang dikenali dengan Diskaun-Kos (DK) dan Beli-atau-Bina (BaB) untuk MPL diperkenalkan untuk menyokong pengambilan keputusan semasa memilih seni bina yang mempunyai bilangan komponen dan tahap lewahan optimum yang memenuhi kehendak teknikal dan kewangan. Pendekatan ini terdiri daripada satu model pengoptimuman yang mempunyai dua kaedah penilaian (DK-MPL dan BaB-MPL) dan menggunakan tiga varian algoritma Pengoptimuman Kerumunan Zarah (PKZ). Keputusan statistik menunjukkan pengurangan sebanyak 74% kos pembangunan menggunakan DK-MPL dalam kajian kes sistem terbenam. Sementara itu penggunaan peningkatan maksimum yang mungkin algoritma-algoritma menunjukkan bahawa Pengoptimuman Kerumunan Zarah Berpandukan Denda (PKZ-BD) telah meningkatkan kualiti penyelesaian yang diperoleh daripada 70% kepada 84% berbanding dengan algoritma lain. Keputusan bagi DK-MPL dan BaB-MPL adalah lebih unggul berbanding dengan keputusan yang diperoleh dengan pendekatan lain yang sama. Hasil keseluruhan kajian ini membuktikan potensi kelebihan pendekatan DK/BaB-MPL bagi penilaian seni bina perisian, terutamanya dalam pemilihan seni bina perisian untuk meminimumkan kos pembangunan yang dapat mengekalkan keutuhan sistem yang tinggi.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS/ SYMBOLS

ABAS	-	Attribute-Based Architectural Styles
ABS	-	Anti-Lock Brake System
ACO	-	Colony Algorithm
ADD	-	Attribute-Driven Design
ATAM	-	Architecture tradeoff analysis method
BCA	-	Bee colony algorithm
BoB-RAP	-	“Build-or-Buy” strategy to solve RAP problem
CBAM	-	Cost Benefit Analysis Model
CBAM	-	Cost Benefit Analysis Model
CBS	-	Component-Based Software
CBSD	-	Component-Based Software Development
CD-RAP	-	cost-discount model to solve RAP problem
COTS	-	Commercial-Off-the Shelf
CTMC	-	Continuous Time Markov Chain
DE	-	Differential Evaluation
DTMC	-	Discrete Time Markov Chain
EC	-	Evolutionary Computation
ECU	-	Electronic Control Unit
ES	-	Embedded System
GF	-	General Features
IAS	-	Immune Artificial System
IP	-	integer programming
IS	-	Information System
LQN	-	Linear Queuing model
MINLP	-	Mixed Integer Nonlinear Programming

MIP	-	mixed integer problem
MOO	-	Multi-objective Optimization
PG-PSO	-	Multi-Objective Penalty guided algorithm based on Particle Swarm Optimization
MO-WS_PSO	-	Multi-Objective Weighted-Sum based on Particle Swarm Optimization
MPI	-	Maximum Possible Improvement
NLIP	-	Non-linear integer programming
OF	-	Optimization Features
POC	-	Proof of Concept
RAP	-	Redundancy Allocation Problem
RIS	-	Information Retrieval System
RRAP	-	Reliability Redundancy Allocation Problem
RUP	-	Rational Unified Process
SA	-	System Architecture
SBSE	-	Search-Based Software Engineering
SMP	-	Semi-Markov Process
SOO	-	Single Objective Optimization
SOO-PSO	-	Single Objective Optimization based on Particle Swarm Optimization
SPN	-	Stochastic Petri Net
SI	-	Swarm Intelligent
TS	-	Tabu Search
VNS	-	Variable Neighborhood Search

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

This chapter introduces the research work reported in this thesis. Several important topics relevant to the research work are presented. First, backgrounds of the study followed by the background of the problem are demonstrated. Next, the problem statement, research objectives, and scope of study respectively are described. Finally, the significance of the research is explained.

1.1 Background of Study

Component-Based Software Development (CBSD) paradigm is one of crucial efforts made to improve the quality of software system. CBSD helps organizations to simplify the development of large and complicated systems. Moreover, CBSD helps to lower the development cost, delivers a shorter time to market the product, and improves the quality of the system [1]. From CBSD point of view, applications are accumulation of deployed software parts. These parts, which are known as components, can be used and reused to develop uncounted numbers of applications. Software applications are built by fitting more or less standardized software

components into a single software application [2]. This means the components are arranged together as black-box objects.

Although CBSD promises a faster time-to-market and increased productivity [3], many risks such as failure to satisfy quality attributes have occurred if the composition is not managed properly. The use of good-components to develop a system does not guarantee a system with satisfaction quality attributes. Indeed, bad quality components will not produce a high-quality product, and even good components can damage a good product if the composition is not managed properly. It is believed that the failure to satisfy quality attributes means, a financial loss, increased expenses of hardware, higher cost of software development, and loss of relationships with consumers. In the real world, such as the industrial automation domain, this probability is unacceptable. Hence, additional measures, time, efforts, and costs are required to minimize the risks.

Indeed, whenever quality issues are addressed at implementation or integration time, correction of problems impacts on cost, schedule, and quality of the software. For example, Hoch *et al.* [4] reported that, large Japanese car manufacturer had to recall 160,000 vehicles due to software failure. Furthermore, the observation reported in the same reference showed that about 25 percent of software problems are related to software architecture, which can be detected at an earlier stage of development life cycle. In fact the decisions made during architecture design have significant implications for economic and quality goals. Examples of architecture-level decisions include the selection of software and hardware components, their replication, the mapping of software components to available hardware nodes, and the overall system topology.

Architectural decisions have a great impact on the consequent quality of software systems. As a result, it is important to evaluate how software architecture meets its quality demands. Though much focus has been placed on modeling and describing the software architecture as a design artifact, automation of architecture

selection has not gained enough study. The selection of the architecture candidate for the next step of a software product is an inherently complex task due to the high volume of possible solutions and to the integral conflict between quality attributes. Additionally, real software projects suffer ever more from limited budgets, and the decisions taken by software developers are heavily affected by cost issues. Therefore, selecting an appropriate optimization technique to solve the problem is essential to nominate an optimal design.

The earliest level of software architecture approaches are used to define the top hierarchical or modular components of the system and assess if they are sufficient to represent the system [5]. The approaches draw out and analyze the quality attributes. For example, an architectural approach is aimed at meeting performance and reliability goals. Through its processes, the architectural approaches identify the architectural risks, sensitivity points, and trade-off points. Actually, obtaining a short list of appropriate architectural options from thousands of possible solutions to do the trade-off is the most difficult task in architectural design approaches.

On the other hand, the main goal of optimization is to find the best possible solutions which could satisfy all objectives and software constraints. In general, optimization techniques can be classified into Single Objective Optimization (SOO) and Multi-Objective Optimization (MOO) [6, 7]. The main goal of SOO is to obtain the best solution relating to the minimum or maximum value of a single objective function (also known as cost function) and which joins all different objectives into a single one. This type of optimization is useful as a tool, which provides decision makers with insights into the nature of the problem. It is usually executed several times rather than a single run in order to produce a set of Pareto solutions [8]. This method has been implemented successfully using different optimization algorithms in [9-11]. The penalty strategy is used to eliminate the inequalities in constraints and formulate new objective operators, which can guarantee feasibility within a reasonable execution time.

This research is motivated by the author own experience as a software engineer in Civil Aviation Authority (CAA)-Sudan. The author had been appointed as a committee member to recommend solutions that improve and integrate all software systems of the CAA. CAA has more than 27 departments vary in their responsibilities but they are interrelating to each other. The possible ways to build and integrate all systems include outsourcing, upgrading old systems, and in-house system development. The task was complicated since there were too many possible alternatives that had to be evaluated based on different criteria. An ad-hoc method had been prepared to perform the task. Finally, the decision was made, the report recommended few alternatives. Unfortunately, the financial and technical preferences for the suggested solutions could not be satisfied together, for example some of suggested solutions were beyond the budged. Therefore, none of suggested solutions was applied by that time.

It is believed that such complicated problems should be automated and optimized to evaluate all possible combinations in order to select an appropriate set of solutions that satisfies the requirements within budget constraints. Inspired by this problem and due to rising complexity challenging quality requirements and demands to reduce the development cost, this study proposes an architectural evaluation approach and optimization technique to consider the evaluation of all possible solutions. The proposed approach aims to optimize the selection of architecture design to solve Reliability Allocation Problem (RAP) in order to satisfy the reliability and cost for developing component-based System Development (CBSD) at early stage of development.

1.2 Reliability Redundancy Allocation Problem (RAP)

Software Architecture (SA) is important in order to evaluate the quality attributes of applications. Reliability is one of the key quality features in a

Component-Based System (CBS) [12]. Many software systems are distributed across a network, comprehensively providing diverse kinds of services for their consumers. These systems must be extremely reliable and provide services when required. Reliability must be engineered into software from the beginning of its development. The software architecture design phase is the first stage of software development in which it is feasible to evaluate how well the quality requirements are being met. Many architecture-based approaches have been proposed to handle reliability from the early stage of design [13-16].

Three strategies have been proposed in literature to improve reliability of complex systems, namely: selecting components with higher reliability; by using additional components in subsystems; or by combining the two strategies. The first strategy often does not reach a satisfactory improvement even if most current reliable components are used to build the system. The second strategy is based on optimizing redundancy levels in subsystems. Redundancy means appending extra computational or component nodes (so-called redundancy allocation) [10]. However, redundancy may lead to additional life-cycle costs, energy, and weight of the system; although the reliability is improved. This problem is commonly known as “Redundancy Allocation Problem (RAP)” [17]. The third strategy is feasible and often used to provide optimal solutions. This strategy is based on selecting optimal components and appropriate redundancy level for each subsystem to maximize system reliability [14, 18]. Optimizing reliability and redundancy simultaneously is known as Reliability Redundancy Allocation Problem (R-RAP) [14]. Early evaluation method at architectural level is essential to solve such problem.

Redundancy allocation strategy is widely used as a design tactic to improve reliability of system at the architecture level [19-21]. Such approaches are often used in industry, where systems are developed using variant (yet functionally similar) components in parallel. For instance, airplanes have primary electronic gyroscope and secondary mechanical gyroscope working in parallel. Anti-Lock Brake System (ABS) in automotive systems domain also uses redundant components; most new automobiles have a redundant (spare) tires with different size and weight

characteristics. Besides, retrieval information systems require higher availability and good performance for its services, which, in turn, necessitates the use of redundant components. These configurations should be better handled in the early stage of development.

Analytical approaches, which take into account all architectural alternatives, can support and speed up the whole stakeholder decision making process. The first optimization techniques introduced in this area were published in [22, 23]. The both methods were built based on the knapsack model, a classical model for cost management used in integer-linear/non linear programming models. In [22] a variant of the knapsack model was introduced to select software architecture that obtained value with minimum cost, while Jung [23] has proposed a model for reliability maximization under budget constraints. More applications are found in [9, 19, 24, 25]. All of the presented approaches are general methods and they mainly focus on maintaining the reliability of components so as to increase the reliability of the whole system.

Practically, the common way to solve RAP in industry is based on ad-hoc or expert opinions, especially in the automotive domain. The major focus of recent researches in the area of software evaluation is the development of approaches based on heuristic/metaheuristic algorithms for solving RAP [26, 27]. For mixed-integer reliability problems, the number of redundant components and the corresponding component reliabilities are to be decided upon simultaneously so as to minimize the system cost whereas the reliability would remain within the accepted levels. For example, components used in the automotive domain are expensive and one of the reasons for the high cost of electronics is the use of large numbers of Electric Control Unit (ECU) [28]. Meedeniya *et al.* [29] have proposed a method by which to evaluate architecture of an embedded system considering redundant components within its subsystems. However, the approach did not provide a solution to reduce the number of components or the development cost.

The consideration of probable discounts to optimize the architecture could minimize the development cost. Cost-discount methods using parameters of quantity discount policies have been introduced in [30]. Few studies [9, 31] have proposed the application of similar methods in the architecture optimization domain to benefit from the probable discount offered by vendors in components' prices.

Another possible way to reduce the development cost is by using mixed components to compose the system, wherein architect has the capability to use different types of components from different sources to compose the system. For example, an approach based on "Build-or-Buy" strategy has been introduced in [32, 33]. This method allows the use of in-house developed and outsourced components to build the system. Since this method is developed utilizing exact algorithms using linear programming method, it has inherited limitations of linear programming on representation of complex problems such as consideration of redundancy components.

A considerable number of attempts to efficiently and effectively solve RAP problems have been found in literature. Although these assets are valuable to date, it appears that there is still room for improvement. Therefore, this thesis proposes an approach which aims to reduce the development cost while keeping reliability within an accepted level. The approach deals with an analytical model of software system, in which applications are constructed from a glued set of components with well-defined behaviors and interfaces. This model of software development is becoming more powerful, and gaining growing confidence of large organizations regarding out–sourcing and in-house development of software components, service–based approaches, and the construction of architectures into which trusted and semi-trusted components are assembled [27].

The approach proposed in this thesis is an architecture-based approach that utilizes optimization techniques to support the selection of optimal design at the architectural level based on a cost-discount model and a build-or-buy strategy. Three

Particle Swarm Optimization (PSO) algorithms varying in their dimensionality and objective functions are proposed to be applied in this study. These are, namely: Single objective optimization (SOO-PSO); Weighted-Sum optimization method (MO-WS_PSO); and Penalty Guided algorithm (PG-PSO) methods.

The results reported in the thesis were drawn from the application of the approach to case studies from two different domains representing both the Information System (IS) domain and the Embedded System (ES) domain. Additionally, a numeric example is utilized for verification and to perform sensitivity analysis for parameters of the “Build-or-Buy” strategy.

1.3 Problem Statement

SA, CBSD, and optimization techniques are separate but related topics in software engineering research and practice [34]. In general, SA defines system components, their co-operation and the basic structure and design. It is concerned with the high-level organization and structure of systems in general. CBSD focuses on the realization of systems through integration of pre-existing components, while optimization techniques are used to search solution space for the best, or near best, solution. However, several challenges must be handled in order to select an architectural design that satisfies financial and technical preferences. These challenges can be classified into three groups. These are, namely: challenges related to SA, challenges relevant to the optimization process, and general challenges related to the components and their interactions.

Architecture challenges include the challenge of adapting a single model for multiple attributes. This issue has motivated researchers for various reasons: First, most cases of real-world systems require the analysis of more than one quality

attribute; second, it is useful to build the prediction model on one single model rather than wasting effort in identifying several models; the built quality model must consider the dependability relating to the remaining quality attributes when defining a metric for a specific quality attribute. For example, timing behavior on a software model might be required to predict reliability of the system [15, 35]. This is essential to select an appropriate model, which can be adapted and applied on software application in order to successfully evaluate the "goodness" of the architectural candidates.

Another architecture challenge is the selection of an appropriate strategy to perform the evolution. Two different techniques have been introduced in [36] to control quality aspects. The first one is based on embedding the quality element into the method; alternatively, another technique relies on extracting the method from the quality features. Consequently, those quality attributes could be modularized regardless of what combination of quality attributes would be used. Rational Unified Process (RUP) [37] is an example for the first technique, while reasoning framework [38] is an example of the second one. Non-experts can use the reasoning framework; hence, it includes the mechanisms needed to use sound analytic theories to analyze the behavior of a system with respect to some quality attributes.

Optimization challenges should be approached to enable architects to explore design space and in order to find out an optimal design based on an evaluation strategy. In fact, the number of different designs for a complicated system can be very large indeed if not infinite. Even with detailed design, there are usually vast numbers of possibilities, far too many to be considered and evaluated individually. On the other hand, for most software designs, the optimization of one quality attribute will result in a deterioration of other quality attributes. Therefore, an efficient design decision strategy is required and it can have dramatic effects upon the cost and quality of the system [3]. Such techniques should have the capability to: analyze the impact of an individual component in the composite model; evaluate its effect on overall behavior of the software system based on the required quality

criteria and, finally, restructure and process the model over and over to try all possible solutions until the requested improvement is achieved.

However, since employing any of the traditional analytical optimization approaches might not be practicable in most complex cases, metaheuristics can be an alternative solution as it is a stochastic-based search technique with a solid base in artificial life, social psychology, as well as in engineering and computer science. Additional examples of the important issues related to the optimization challenges are; the transformation of architectural problems to optimization problems and the selection of a programming method (Linear/ Non-linear and integer/mixed-integer programming methods). Moreover, optimizing software architecture that has conflicting objectives is the main challenge in any optimization process, thus the careful selection of optimization type is essential to obtain solutions to suit the problem.

General challenges as identified in this study are those challenges related to the overall capabilities of the approach in context of components' properties and their interactions, namely; hybrid components, multi-instances, redundancy, cost discounts, and additional attributes or constraints. Optimizing the software architecture using only available and costly COTS components is risky; there is a need to find out ways that can decrease development cost. One way to decrease the development cost is to benefit from the discount that is probably offered by vendors [16]. Another alternative is to produce in-house components and apply a "Build-or-Buy" strategy [32, 33] to select the appropriate component for each subsystem. This will impact on the cost, quality and the time to deliver the system. Other reasons to produce in-house components are, namely: the component might not be as readily available as a COTS component and the cost to search the component might be higher than the price itself [39].

In fact, the involvement of cost discount and "Build-or-Buy" strategies in an approach to optimize the architecture requires additional configuration to the

architecture of the system. Since several new variables and parameters should be included, the great challenge is the formulation and transformation of new parameters that are relevant to changing an architectural problem into an optimization problem.

The current high interest in SA is mainly motivated by the possibility of managing complex software components. The following research question is related to SA and CBSE from one side and decision-making using optimization algorithms from the other side. Mainly, this problem, like most practical optimization problems, requires the coincident optimization of more than one objective function. Similar to the traditional optimization problem that deals with the challenge to simultaneously minimize risks and maximize benefits, facilitating the trade-off between the quality attributes is essential to obtain a design that satisfies the required attributes. Therefore, the main research question is: “How to develop an architecture-based approach for optimizing the selection of architecture for component-based system in order to support the development of reliable and cost-effective applications?”

The sub-questions of the main question are as follows:

(i) What are the state-of-the-art software architecture evaluation approaches in supporting the development of reliable and cost-effective applications?

(ii) What is an effective evaluation approach to evaluate the reliability and cost-effectiveness for architecture candidates of a component-based system at architecture level?

(iii) How can effective evaluation strategies be developed to optimize the software architecture for selecting reliable and cost-effective architecture?

(iv) How can an optimization algorithm be capable of searching design space for optimal architecture based on the developed evaluation strategies?

(v) How can we evaluate the applicability and effectiveness of the proposed approach?

1.4 Research Objectives

The objectives to answer the stated research questions and to achieve the above goal are described below:

(i) To identify the problems in the state-of-the-art software architecture evaluation approaches in supporting the optimization of software architecture for developing reliable and cost-effective application.

(ii) To develop an effective architecture-based approach that integrates necessary aspects and strategies to optimize software architecture of component-based system for developing reliable and cost-effective application.

(iii) To benchmark the proposed approach and evaluate its applicability in developing a reliable and cost-effective application using the selected case studies.

1.5 Scope of the Study

The scope of this study encompasses five facets, i.e. quality attributes that have been handled, optimization types performed, the architectural style used to represent system architecture, the designs and tools, and case study implementations and their assumptions.

Quality attributes: The main software features handled by this study are software reliability and cost. Reliability is defined as a probability of "failure-free" software operation for a specified period of time in a specified environment [40]. The common methods used to achieve reliable software systems are, namely; fault prevention, fault tolerance, fault removal and fault/failure forecasting. The latter is the main focus of this study. Some techniques used in fault/failure forecasting are, namely; developing models, collecting data, calibrating models and reliability

prediction. The main technique used in this research is a reliability prediction model. The cost is important in this study since it aims to produce cost-effective application. Therefore, cost models to estimate the development costs are built based on the cost of their components. In addition, response time, as one of the performance measures, has been modeled based on interactions between components to investigate probable trade-offs between reliability, response time and cost.

Dimensions of optimization: The study focuses on optimizing the architecture problem using Single Objective Optimization algorithms (SOO) and Multi-Objective Optimization algorithms (MOO). Multiple functions are aggregated into a scalar function using the weighted sum method, and a penalty guided algorithm to reform the MOO problem. All algorithms in this study were developed based on the general algorithm of the Particle Swarm Optimization algorithm (PSO). The effectiveness of the proposed algorithms was examined in terms of the quality of solution using statistical tests and Maximum Possible Improvement (MPI) method.

Architecture representation: Software architecture of the sequential applications can be modeled by: Discrete Time Markov Chain (DTMC), Continuous Time Markov Chain (CTMC), Stochastic Petri Net (SPNs) and Semi-Markov Process (SMP). The Serial Parallel system is ideal modeling for solving an RAP problem. DTMC is commonly used to represent the serial-parallel system. DTMC and SMP are used to represent the architecture of systems in this study.

The implementation and the tailored tool: The implementation of the approach on the case studies and customization of parameters were developed and performed using a tailored tool. The tool was designed using Borland Delphi 5 for the purpose of experiments in order to repeat the executions while varying the parameters.

Case studies and their assumptions: The numerical case study is a simple one for the purpose of examining the sensitivity of parameters relevant to the “Build-or-Buy”

strategy. The second case study, Anti-Lock Brake System (ABS), from embedded system domain (ES), is used to validate the applicability of the approach for the ES domain, as well as to evaluate the impact of using the cost-discount model to the quality of the obtained solutions. The third case study is Web-Based Data Retrieval System (WBDRS) from Information system domain, which is used to demonstrate the applicability of the approach in IS domain. This case study is also used to compare the proposed approach with similar approaches from literature.

The first case study is composed of a simple structure with numeric data representing the parameters of software architecture. The simple structure is used to simplify the analysis and to show reactions to changes in the parameters.

For the second case study, ABS, some of the parameters are associated with each component (such as the cost of components), while other parameters need to be estimated. Estimated time per visit, redundancy overheads, execution initiation probability, and transfer probability are either estimated based on profiles of expert knowledge or based on different operating profiles of results using sensitivity analysis. The annotated parameters such as failure rates have been estimated using a model based approach and sensitivity analysis, as applied in a previous work by Meedeniya *et al.* [41]. The rest of the parameters can be calculated using the estimated and given parameters such as Sojourn Time per Visit and expected number of visits.

The WBDRS case study has been partly based on the monitoring of an existing data retrieval system at University of L'Aquila, Italy. The data has been taken from software artifacts of the same system. Cortellessa *et al.* [32] have applied an approach similar to the proposed one and using this case study. In fact, due to the incomplete documentation, an exploration technique has been applied by the same authors to provide convinced values. For example, to identify the number of invocations, the researchers have analyzed partial scenarios and compared the result by monitoring the average number of interactions. For the purpose of comparing the

proposed approach with [32], a similar case study and the same data have been utilized.

The proposed approach applied to optimize architecture of the selected case study in order to evaluate its applicability and effectiveness. Quantitative evaluation methods have been used for evaluation. Independent t-Test is used to evaluate the efficiency of CD_RAP. Additionally, a model of comparison is used to qualitatively evaluate and indicate the effectiveness of the approach in compared to well known practice in literature. Moreover, Maximum Possible Improvement (MPI) method and statistical tests have been used to compare between the proposed algorithms in order to select best optimization dimensionality for the proposed approach. Statistical test is used to statistically measure the quality of obtained solutions from based on the selected algorithm and to ensure the robustness of the results.

1.6 Significance of the Study

Selecting the appropriate set of components and connectors to make the system meeting functional and accommodate non-functional requirements remains a hard task to be accomplished. In practice, most selection decisions are subject to current joint ventures, commercial benefits, and successful vendor marketing. Moreover, the competitiveness of business depends usually on very strict development schedules. Traditional selection of components is time-consuming since considerable time is needed to investigate and study the available components [39]. As stated before, real software projects suffer ever more from limited budgets, and the decisions taken by software developers are heavily affected by cost issues. Therefore, selecting an appropriate optimization technique to solve the problem is essential to nominate optimal design.

There are many studies in literature associated with the evaluation of software architecture. Architecture Tradeoff Analysis Method (ATAM) [42], which is a common method in the software evaluation domain, provides a quantitative framework by which to reason about software trade-offs at architecture level. Further, the Architecture Tradeoff Analysis Method / Attribute-Based Architectural Styles (ATAM/ABAS) [43] method and Cost Benefit Analysis Model (CBAM) [44] have been proposed with ATAM to provide quantitative and qualitative reasons about quality attributes. An approach guided by a quality attribute known as Attribute-Driven Design (ADD) method [45] has been proposed to produce systems with high quality. It is a systematic step-by-step method for designing the software architecture of a software-intensive system. At each stage in the development, these approaches use scenarios, tactics and architectural patterns to assess the satisfaction of a set of quality attributes. However, the perspective of this study differs from ATAMs and ADD. Although ADD is guided by quality attributes, there is no solution for cases with no prescribed scenario or pattern. Additionally, ATAMs and ADD are manual methods and based on experience, while this study is an automated method based on optimization techniques to select an optimal design.

Many real-world decision-making problems use optimization techniques to attain their goals [46]. These include, for example, minimizing time to deliver product, maximizing reliability, minimizing deviations from desired levels, minimizing costs etc. Although real-world software systems have increased in size and complexity, the cost of application failures grows and hence business performance increasingly deteriorates. Components used in the automotive domain are expensive and one of the reasons for the high cost of electronics is the large number of ECU used [28]. Thus, optimization of architecture design has become crucial.

However, few efforts have been directed towards optimization of software based on reliability-cost trade-offs to produce cost-effective applications. This releases the need to expand prescribed relationships between architectural design decisions and quality attributes under cost constraints, which in turn could conduct a

realistic evaluation and support automated architecture design [47]. In addition, earlier fault corrections and precise predictions that needed to the consumers from the system could be delivered. Consequently, there is a need for an architecture-based approach that enables the selection of architectural design. The employment of an optimization technique into such approaches will aid in simplifying evaluation of software architecture, save software development cost and efforts, and it will play an essential role in producing cost-effective applications.

1.7 Thesis Outlines

This thesis encompasses some aspects relevant to software architecture and optimization techniques to support software architecture evaluation for solving RAP problems. The Proposed approach in this thesis provides two different ways to evaluate the software architecture based on one optimization model. The approach is noted as (CD/BoB-RAP), which stands for Cost-Discount /Build-or-buy for RAP, (CD/BoB-RAP). The thesis consists of nine major chapters, including the introductory chapter which commences the report. The remainder of the thesis is composed of eight chapters and an appendix followed by published papers related to the topic.

Chapter 2 discusses the literature review of software architecture evaluation and optimization particularly to support the software evaluation for the selection of optimal architectural design. It opens with explanations of the basic concepts of software architecture, component based development, and related problems such as span of design space and Reliability Redundancy Allocation Problem. This is followed by discussion on the main approaches for the evaluation of architecture, which are grouped into three main categories, i.e. anti-pattern, rule-based, and search-based approaches.

Chapter 3 aims to evaluate the current search-based approach for software architecture optimization. It illustrates the proposed taxonomy for the software architecture optimization to evaluate the previous studies and to put this study in context. The main features of the taxonomy are based on features of component interactions, software architecture and the optimization process. The chapter reports on comparative evaluation for a number of optimization-based approaches to architectural design selection. The evaluation conducted was based on an optimization feature extracted from the taxonomy. In addition, some general features proposed in this study have been used for the evaluations. The critical discussion and summary of the evaluation of the comparison focus on the capabilities of the proposed approach to effectively support the selection of optimal architectural design. The evaluation results of general features are crucial to identify gaps in the current works, while the results of evaluations represent the corner stone by which to build a solution that fills the identified gaps.

Chapter 4 presents the research methodology established to handle this work. The chapter includes research design, operational framework, and overview concerning verification and evaluation of the proposed approach. The research design is visualized as a flowchart to illustrate the plan and sequence steps to conduct the research. However, the operational framework, which is built based on the research questions and research objectives, describes the action plan by which to perform the study. The chapter also describes the verification and evaluation methods and the case studies used to evaluate the proposed approach.

Chapter 5 describes the CD/BoB-RAP approach, optimization algorithm, as well as the varying optimization strategies used for the approach. First, the optimization model and its main elements are outlined. The chapter then highlights the architecture representation and architectural evaluation methods. These include “Cost-Discount” and “Build-or-Buy” and the optimization process, as well as the proposed algorithms for the approach to enhance the capabilities in selecting a reliable and cost effective design.

Chapter 6 demonstrates the evaluation of the applicability of CD/BoB-RAP to optimize the software architecture. The CD/BoB-RAP has been applied on Anti-lock Brake System (ABS), a case study from the ES domain. The case study is described and utilized for the evaluation of the approach in order to develop a cost-effective application based on the Cost-Discount (CD) evaluation method. The chapter reports on the comparison between the quality attributes of architecture obtained when the approach is executed based on a simple optimization model and performed based on the CD/BoB-RAP. The results are analyzed and the impact of CD/BoB-RAP approach on the quality of the obtained solutions is discussed.

Chapter 7 reports the results and discussions of the application of CD/BoB-RAP on a numerical case study to evaluate the sensitivity of parameters of Build-or-Buy strategy to changes. The applicability of the CD/BoB-RAP then builds a cost effective application from mixed components based on Build-or-Buy strategy. This has been evaluated by applying the approach on a Web-based Data Retrieval System (WBDRS) used on a retrieval system case study to evaluated its applicability in the IS domain. The results of the evaluation have been shown and discussed.

Chapter 8 demonstrates evaluation of the algorithm and the selection of optimization dimensions for the CD/BoB-RAP. In addition, the chapter presents the process and discussions on comparing the CD/BoB-RAP to similar approaches in order to show the effectiveness of the proposed approach.

Finally, Chapter 9 presents the thesis summary. In addition, it outlines the achievements, contributions, future works, and related publications. The chapter ends with the conclusion followed by reference to future works.

REFERENCES

1. Fukuzawa, K. and M. Saeki. *Evaluating software architectures by coloured petri nets*. in *Proceedings of the 14th international conference on Software engineering and knowledge engineering*. 2002: ACM.
2. Brun, Y., *Smart redundancy for distributed computation.* IEEE, 2011.
3. Voas, J., *COTS software: the economical choice?* Software, IEEE, 1998. **15**(2): p. 16- 19
4. Hoch, D., Huhn,W., Naher, V., and A. Zielke *The race to master automotive embedded systems development.* 2006, McKinsey Company, Automotive and assembly sector business technology office: Germany.
5. Sharafi, S.M., G.A. Ghazvini, and S. Emadi. *An analytical model for performance evaluation of software architectural styles*. in *Software Technology and Engineering (ICSTE), 2010 2nd International Conference on*. 2010.
6. Banks, A., J. Vincent, and C. Anyakoha, *A review of particle swarm optimization. Part II: hybridisation, combinatorial, multicriteria and constrained optimization, and indicative applications.* Natural Computing, 2008. **7**(1): p. 109-124.
7. Sülflow, A., N. Drechsler, and R. Drechsler. *Robust multi-objective optimization in high dimensional spaces*. 2007: Springer.
8. Cortellessa, V., F. Marinelli, and P. Potena, *Automated selection of software components based on cost/reliability tradeoff.* Software Architecture, 2006: p. 66-81.
9. Hsieh, T.-J. and W.-C. Yeh, *Penalty guided bees search for redundancy allocation problems with a mix of components in series-parallel systems.* Computers & Operations Research, 2012. **39**(11): p. 2688-2704.

10. Chen and TaCheng, *Penalty Guided PSO for Reliability Design Problems, PRICAI 2006: Trends in Artificial Intelligence*, Q. Yang and G. Webb, Editors. 2006, Springer Heidelberg,Berlin. p. 777-786.
11. Joines, J.A. and C.R. Houck. *On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with GA's*. 1994.
12. Pelliccione, P, *An architectural approach to the correct and automatic assembly of evolving component-based systems*. Journal of Systems and Software, 2008. **81**(12): p. 2237-2251.
13. Goševa-Popstojanova, K. and K.S. Trivedi, *Architecture-based approach to reliability assessment of software systems.* Performance Evaluation, 2001. **45**(2-3): p. 179-204.
14. Hikita, M., Nakagawa, Y., Harihisa, H., *Reliability optimization of systems by a surrogateconstraints algorithm.* IEEE Transactions on Reliability, 1992. **R-41**(3): p. 473-480.
15. Ralf Reussner, Judith Stafford , and C.A. Szyperski, *Architecting Systems with Trustworthy Components* 04511 Abstracts Collection -- Architecting Systems with Trustworthy Components, ed. D.S.P.R.a.J.S.a.C.A. Szyperski. 2006, Schloss Dagstuhl, Germany: Internationales Begegnungs- und Forschungszentrum f{\"u}r Informatik (IBFI).
16. Lee, M.J.R.H.L., *Improving Profitability with Quantity Discounts under Fixed Demand.* IIE Transactions 1985. **17**(4): p. 388-395
17. Zia, L. and D.W. Coit, *Redundancy Allocation for Series-Parallel Systems Using a Column Generation Approach.* Reliability, IEEE Transactions on, 2010. **59**(4): p. 706-717.
18. Kuo, W. and V.R. Prasad, *An annotated overview of system-reliability optimization.* Reliability, IEEE Transactions on, 2000. **49**(2): p. 176-187.
19. Ouzineb, M., M. Nourelfath, and M. Gendreau, *Tabu search for the redundancy allocation problem of homogenous series–parallel multi-state systems.* Reliability Engineering & System Safety, 2008. **93**(8): p. 1257-1272.
20. Chen, T., You,And PengSheng, *Immune algorithms-based approach for redundant reliability problems with multiple component choices.* Computers in Industry, 2005. **56**(2): p. 195-205.

21. Leandro dos Santos, C., *An efficient particle swarm approach for mixed-integer programming in reliability–redundancy optimization applications.* Reliability Engineering & System Safety, 2009. **94**(4): p. 830-837.
22. Jung, H.W., C.S. Chung, and K.O. Lee. *Selecting optimal COTS products considering cost and failure rate*. 1999: Citeseer.
23. Jung, H.-W. and B. Choi, *Optimization models for quality and cost of modular software systems.* European Journal of Operational Research, 1999. **112**(3): p. 613-619.
24. Yung-Chain Liang, Min-Hua Lo, and Yi-Ching Chen, *Variable Neighbourhood Search For Redundancy Allocation Problems.* IMA Journal of Management Mathematics, 2007. **18** p. 135−155.
25. Liang, Yun-Chia Chen, and Yi-Ching., *Redundancy allocation of series-parallel systems using a variable neighborhood search algorithm.* Reliability Engineering & System Safety, 2007. **92**(3): p. 323-331.
26. Daniel Dominguez Gouv, D., Muniz., Gilson Pinto, Alberto Avritzer, Rosa Maria Meri Le, Edmundo de Souza e Silva, Morganna Carmem Diniz, Luca Berardinelli, Julius C.B. Leite, Daniel Mosse, Yuanfang Cai, Mike Dalton, Lucia Kapova, And Anne Koziolek,. *Experience building non-functional requirement models of a complex industrial architecture*, in *Proceeding of the second joint WOSP/SIPEW international conference on Performance engineering*. 2011, ACM: Karlsruhe, Germany. p. 43-54.
27. Baker, P., Harman, M., Steinhofel, K., And Skaliotis, A. *Search Based Approaches to Component Selection and Prioritization for the Next Release Problem*. in *Software Maintenance, 2006. ICSM '06. 22nd IEEE International Conference on*. 2006.
28. Patil, S. and L. Kapaleshwari, *Embedded Software - Issues and Challenges.* 2009.
29. Indika Meedeniya, Aldeida Aleti, and B. Zimmerova, *Redundancy Allocation in Automotive Systems using Multi-objective Optimisation* in *Symposium on Automotive/Avionics Systems Engineering SAASE*. 2009.
30. Guidong, L. *Coordinating Three-Level Supply Chain with Quantity Discount Policies*. in *Management and Service Science (MASS), 2010 International Conference on*. 2010.

31. Ebrahimipour, V. and M. Sheikhalishahi. *Application of multi-objective particle swarm optimization to solve a fuzzy multi-objective reliability redundancy allocation problem*. in *Systems Conference (SysCon), 2011 IEEE International*. 2011.
32. Cortellessa, V., F. Marinelli, and P. Potena, *An optimization framework for "build-or-buy" decisions in software architecture.* Comput. Oper. Res., 2008. **35**(10): p. 3090-3106.
33. Cortellessa V, Marinelli F, and P. P, *Automated selection of software components based on cost/reliability tradeoff.* Software Architecture, 2006: p. 66-81.
34. Pande, J., *On Some Critical Issues in Component Selection in Component based Software Development.* International Journal of Computer Applications, 2012. **46**(4): p. 44-50.
35. Hasselbring, W. and R. Reussner, *Toward trustworthy software systems.* Computer, 2006. **39**(4): p. 91-92.
36. Etessami, K., *The ComFoRT Reasoning Framework*, in *Computer Aided Verification*. 2005, Springer Berlin / Heidelberg. p. 164-169.
37. Krutchen, P., *The Rational Unified Process: An Introduction*. 2003, 3rd ed.Addison-Wesley: Boston.
38. Bass, L., Ivers, J.,Klein, M.H., and Merson, P.F., *Reasoning frameworks*. 2005, Software Engineering Institute Carnegie Mellon University.
39. Alves, C. and A. Finkelstein. *Challenges in COTS decision-making: a goal-driven requirements engineering perspective*. 2002: ACM.
40. Lyu, M.R., *Software Reliability Engineering: A Roadmap*, in *2007 Future of Software Engineering*. 2007, IEEE Computer Society. p. 153-170.
41. Indika Meedeniya, B.B., Aldeida Aleti, and Lars Grunske, *Reliability-driven deployment optimization for embedded systems*. Journal of Systems and Software, 2011. **84**(5): p. 835-846.
42. Goševa-Popstojanova, K. and K.S. Trivedi, *Architecture-based approach to reliability assessment of software systems.* Performance Evaluation, 2001. **45**(2): p. 179-204.
43. Goševa-Popstojanova, K. and S. Kamavaram. *Uncertainty Analysis of Software Reliability Based on Method of Moments*. 2002.

44. Helander, M.E., M. Zhao, and N. Ohlsson, *Planning models for software reliability and cost.* Software Engineering, IEEE Transactions on, 1998. **24**(6): p. 420-434.

45. Bass, L., M. Klein, and F. Bachmann, *Quality attribute design primitives and the attribute driven design method.* Software Product-Family Engineering, 2002: p. 323-328.

46. Sahoo, N.C., S. Ganguly, and D. Das, *Multi-objective planning of electrical distribution systems incorporating sectionalizing switches and tie-lines using particle swarm optimization.* Swarm and Evolutionary Computation, 2012. **3**(0): p. 15-32.

47. Shaw, M. and P. Clements, *The golden age of software architecture.* Software, IEEE, 2006. **23**(2): p. 31-39.

48. Szyperski, C., D. Gruntz, and S. Murer, *Component software: beyond object-oriented programming*. 2002: Addison-Wesley Professional.

49. Ritzsche, M.a.J.J., *Putting performance engineering into model-driven engineering: Model-driven performance engineering.* Nashville, TN, United states, Springer Verlag., 2008.

50. Zeng, L., Benatallah, B.,Ngu, A.H.H.,Dumas, M.,Kalagnanam, J., and Chang, H., *QoS-aware middleware for web services composition.* Software Engineering, IEEE Transactions on, 2004. **30**(5): p. 311-327.

51. Becker, S., H. Koziolek., *The Palladio component model for model-driven performance prediction.* Journal of Systems and Software, 2009. **82(1): 3-22.localization in model transformation, Sofia, Bulgaria, INSTICC Press.**

52. Diaz-Pace, A., Kim, H.,Bass, L.,Bianco, P., and Bachmann, F., *Integrating quality-attribute reasoning frameworks in the ArchE design assistant.* Quality of Software Architectures. Models and Architectures, 2008: p. 171-188.

53. Goseva-Popstojanova, K., A.P. Mathur, and K.S. Trivedi. *Comparison of architecture-based software reliability models*. in *Software Reliability Engineering, 2001. ISSRE 2001. Proceedings. 12th International Symposium on*. 2001.

54. Pressman, R.S., *Software Engineering - A Practitioner's Approach* 6th ed. 2004: McGraw Hill

55. Clements, P., *Documenting software architectures: views and beyond*. 2003: IEEE.

56. Dueñas, J., W. de Oliveira, and J. de la Puente, *A software architecture evaluation model.* Development and Evolution of Software Architectures for Product Families, 1998: p. 148-157.
57. Kruchten, P., H. Obbink, and J. Stafford, *The past, present, and future for software architecture.* Software, IEEE, 2006. **23**(2): p. 22-30.
58. Avizienis, A., *Basic concepts and taxonomy of dependable and secure computing.* Dependable and Secure Computing, IEEE Transactions on, 2004. **1**(1): p. 11-33.
59. Reussner, R.H., H.W. Schmidt, and I.H. Poernomo, *Reliability prediction for component-based software architectures.* Journal of Systems and Software, 2003. **66**(3): p. 241-252.
60. Avizienis, A., *Fundamental concepts of dependability.* TECHNICAL REPORT SERIES-UNIVERSITY OF NEWCASTLE UPON TYNE COMPUTING SCIENCE, 2001.
61. Mari, M. and N. Eila. *The impact of maintainability on component-based software systems*. 2003: IEEE.
62. Immonen, A. and E. Niemelä, *Survey of reliability and availability prediction methods from the viewpoint of software architecture.* Software and Systems Modeling, 2008. **7**(1): p. 49-65.
63. Martens, A. and H. Koziolek, *Automatic, Model-Based Software Performance Improvement for Component-based Software Designs.* Electronic Notes in Theoretical Computer Science, 2009. **253**(1): p. 77-93.
64. A .A. Abdelaziz, W MWN Kadir, Citizaiton Mohmd Hashim, *Metaheuristic Search Approach Based on In-house/Out-sourced Strategy to Solve Redundancy Allocation Problem in Component-Based Software Systems* International journal of software engineering and its applications, 2012. **Vol.6**(No.3, 2012).
65. Yeh, W.-C. and T.-J. Hsieh, *Solving reliability redundancy allocation problems using an artificial bee colony algorithm.* Comput. Oper. Res., 2011. **38**(11): p. 1465-1473.
66. Yeh, W.-C., *A two-stage discrete particle swarm optimization for the problem of multiple multi-level redundancy allocation in series systems.* Expert Systems with Applications, 2009. **36**(5): p. 9192-9200.

67. Sabane, A. *Improving system testability and testing with microarchitectures*. 2010: IEEE.

68. Cortellessa, V. and L. Frittella, *A framework for automated generation of architectural feedback from software performance analysis*, in *Proceedings of the 4th European performance engineering conference on Formal methods and stochastic models for performance evaluation*. 2007, Springer-Verlag: Berlin, Germany. p. 171-185.

69. Rodrigues, G., D. Rosenblum, and S. Uchitel. *Using scenarios to predict the reliability of concurrent component-based software systems*. 2005. Edinburgh, United kingdom: Springer Verlag.

70. Xu, J., *Rule-based automatic software performance diagnosis and improvement*, in *Proceedings of the 7th international workshop on Software and performance*. 2008, ACM: Princeton, NJ, USA. p. 1-12.

71. Harman, M. and B.F. Jones, *Search-based software engineering.* Information and Software Technology, 2001. **43**(14): p. 833-839.

72. Aguilar-Ruiz, J.S.,, *An evolutionary approach to estimating software development projects.* Information and Software Technology, 2001. **43**(14): p. 875-882.

73. Antoniol, G., M. Di Penta, and M. Harman. *Search-based techniques applied to optimization of project planning for a massive maintenance project*. 2005: IEEE.

74. Briand, L.C., J. Feng, and Y. Labiche. *Using genetic algorithms and coupling measures to devise optimal integration test orders*. 2002: ACM.

75. Guo, Q., *Computing unique input/output sequences using genetic algorithms.* Formal Approaches to Software Testing, 2004: p. 1098-1100.

76. McMinn, P, *The species per path approach to SearchBased test data generation*. 2006: ACM.

77. Bouktif, S,.*A novel approach to optimize clone refactoring activity*. 2006: ACM.

78. Fatiregun, D., M. Harman, and R.M. Hierons. *Search-based amorphous slicing*. 2005: IEEE.

79. G. Canfora, M.D.P., R. Esposito, and M. L. Villani. , *An approach for qoS-aware service composition based on genetic algorithms.* In H.-G. Beyer and

U.-M. O'Reilly, editors, Proc.of Genetic and Evolutionary Computation Conference 2005, pages 1069–1075. ACM, , 2005.

80. Cohen, M., S.B. Kooi, and W. Srisa-an. *Clustering the heap in multi-threaded applications for improved garbage collection*. 2006: ACM.
81. Bouktif, S., H. Sahraoui, and G. Antoniol, *Simulated annealing for improving software quality prediction*, in *Proceedings of the 8th annual conference on Genetic and evolutionary computation*. 2006, ACM: Seattle, Washington, USA. p. 1893-1900.
82. Khoshgoftaar, T.M., Y. Liu, and N. Seliya, *A multiobjective module-order model for software quality enhancement.* Evolutionary Computation, IEEE Transactions on, 2004. **8**(6): p. 593-608.
83. Praditwong, K., M. Harman, and Y. Xin, *Software Module Clustering as a Multi-Objective Search Problem.* Software Engineering, IEEE Transactions on, 2011. **37**(2): p. 264-282.
84. Shannon, C.E., *A mathematical theory of communication.* ACM SIGMOBILE Mobile Computing and Communications Review, 2001. **5**(1): p. 3-55.
85. CanforaHarman, G. and M. Di Penta. *New frontiers of reverse engineering*. 2007: IEEE Computer Society.
86. Cooper, K.D., P.J. Schielke, and D. Subramanian. *Optimizing for reduced code space using genetic algorithms*. 1999: ACM.
87. Ruan, N. and X. Sun, *An exact algorithm for cost minimization in series reliability systems with multiple component choices.* Applied Mathematics and Computation, 2006. **181**(1): p. 732-741.
88. David, Alice, and Ieee, *Reliability Optimization of Series-Parallel Systems Using a Genetic Algorithm.* IEEE Transactions on Reliability, 1996. **45**: p. 254--260.
89. Kulturel-Konak, S., A.E. Smith, and D.W. Coit, *Efficiently solving the redundancy allocation problem using tabu search.* IIE transactions, 2003. **35**(6): p. 515-526.
90. website. *Swarm and Evolutionary Computation*. 2011 [cited 2011 13-11-2011]; Available from: http://web.mysites.ntu.edu.sg/epnsugan/PublicSite/Shared%20Documents/Swarm%20and%20Evolutionary%20Computation_2.pdf.

91. Meedeniya, I. *OptimizationSurvey*. 2011 6 DECEMBER 2009, AT 17:34. [cited 2011 20-11-2011]; Available from: https://sdqweb.ipd.kit.edu/wiki/OptimizationSurvey.

92. Martens Anne, D.A., Heiko Koziolek, Raffaela Mirandola, and Ralf Reussner, , *A Hybrid Approach for Multi-attribute QoS Optimisation in Component Based Software Systems*, in *Research into Practice – Reality and Gaps*, G. Heineman, J. Kofron, and F. Plasil, Editors. 2010, Springer Berlin / Heidelberg. p. 84-101.

93. Noorshams, Q., A. Martens, and R. Reussner. *Using quality of service bounds for effective multi-objective software architecture optimization*. 2011. Oslo, Norway: Association for Computing Machinery, 2011.

94. Martens, A., F. Brosch, and R. Reussner, *Optimising multiple quality criteria of service-oriented software architectures*, in *Proceedings of the 1st international workshop on Quality of service-oriented software systems*. 2009, ACM: Amsterdam, The Netherlands. p. 25-32.

95. Ashrafi, N. and O. Berman, *Optimization models for selection of programs, considering cost and reliability.* Reliability, IEEE Transactions on, 1992. **41**(2): p. 281-287.

96. Taboada, H.A., J.F. Espiritu, and D.W. Coit, *MOMS-GA: A Multi-Objective Multi-State Genetic Algorithm for System Reliability Optimization Design Problems.* Reliability, IEEE Transactions on, 2008. **57**(1): p. 182-191.

97. Wadekar, S.A. and S.S. Gokhale, *Exploring Cost and Reliability Tradeoffs in Architectural Alternatives Using a Genetic Algorithm*, in *IEEE Computer Society, Proceedings of the 10th International Symposium on Software Reliability Engineering*. 1999.

98. Li, J.Z., *Fast scalable optimization to configure service systems having cost and quality of service constraints*. 2009: ACM.

99. Zheng, T. and M. Woodside, *Heuristic optimization of scheduling and allocation for distributed systems with soft deadlines.* Computer Performance Evaluation. Modelling Techniques and Tools, 2003: p. 169-181.

100. El-Sayed, H., D. Cameron, and M. Woodside. *Automation support for software performance engineering*. 2001: ACM.

101. Zeng, L., *QoS-aware middleware for web services composition.* Software Engineering, IEEE Transactions on, 2004. **30**(5): p. 311-327.

102. Cortellessa, Marinelli, and Potena, *Automated selection of software components based on cost/reliability tradeoff*. Software Architecture, 2006: p. 66-81.

103. El-Sayed, H., D. Cameron, and M. Woodside. *Automation support for software performance engineering*. in *ACM SIGMETRICS Performance Evaluation Review*. 2001: ACM.

104. Lisnianski, A., Levitin, G.,Ben-Haim, H., and Elmakis, D., *Power system structure optimization subject to reliability constraints.* Electric Power Systems Research, 1996. **39**(2): p. 145-152.

105. Debardelaben, J.A. and A.J. Gadient, *Incorporating Cost Modeling in Embedded-System Design.* IEEE DESIGN & TEST OF COMPUTERS, 1997.

106. Barry, W.B., Chris, A.,A. Winsor Brown,Sunita Chulani, Bradford K. Clark, Ellis Horowitz, Ray Madachy,Donald J. Reifer, and Bert Steece *Software Cost Estimation with COCOMO II*. 2009: SPrentice Hall Press Upper Saddle River, NJ, USA ©2009.

107. Helander, Zhao, and Ohlsson, *Planning models for software reliability and cost.* Software Engineering, IEEE Transactions on, 1998. **24**(6): p. 420-434.

108. WILLIAMS, D.R.P., *Study of the Warranty Cost Model for Software Reliability with an Imperfect Debugging Phenomenon.* Turk J Elec Engin, VOL.15, NO.3 , http://journals.tubitak.gov.tr/elektrik/issues/elk-07-15-3/elk-15-3-5-0604-5.pdf 2007.

109. Gokhale, S.S. *Cost constrained reliability maximization of software systems*. in *Reliability and Maintainability, 2004 Annual Symposium - RAMS*. 2004.

110. Aneja, Y.P., R. Chandrasekaran, and K.P.K. Nair, *Minimal-cost system reliability with discrete-choice sets for components.* Reliability, IEEE Transactions on, 2004. **53**(1): p. 71-76.

111. Gokhale, S.S., Wong, W.E., Trivedi, K.S., and Horgan, JR. *An analytical approach to architecture-based software reliability prediction*. 1998: IEEE.

112. Ramirez-Marquez, J.E. and D.W. Coit, *A heuristic for solving the redundancy allocation problem for multi-state series-parallel systems.* Reliability Engineering & System Safety, 2004. **83**(3): p. 341-349.

113. Saed, A.A.A., W.M.W N., *An automated support for evaluating alternative design decisions.* Journal of Theoretical and Applied Information Technology, 2012. **36**(2): p. 234-246.

114. Saed, A.A.A. and W.M.N.W. Kadir. *Applying particle swarm optimization to software performance prediction an introduction to the approach*. 2011.
115. Fukuzawa, K. and M. Saeki. *Evaluating software architectures by coloured petri nets*. 2002: ACM.
116. Sharafi, M., F. Shams Aliee, and A. Movaghar. *A review on specifying software architectures using extended automata-based models*. 2007: Springer.
117. Littlewood, B., *Software reliability model for modular program structure.* IEEE Transactions on Reliability, 1979. **5**(10): p. 241-246.
118. Asad, C.A., M.I. Ullah, and M.J.U. Rehman. *An approach for software reliability model selection*. 2004: IEEE.
119. Xie, M. and C. Wohlin. *An additive reliability model for the analysis of modular software failure data*. in *Software Reliability Engineering, 1995. Proceedings., Sixth International Symposium on*. 1995.
120. Zeshan, F. and R. Mohamad. *Software architecture reliability prediction models: An overview*. 2011: IEEE.
121. Trcka, N., *Integrated model-driven design-space exploration for embedded systems*. 2011: IEEE.
122. Liang, Y.C. and M.H. Lo, *Multi-objective redundancy allocation optimization using a variable neighborhood search algorithm.* Journal of Heuristics, 2010. **16**(3): p. 511-535.
123. Haverkort, B.R. and A.M.H. Meeuwissen, *Sensitivity uncertainty analysis of Markov-reward models.* IEEE Transactions on Reliability, 1995. **44**(1): p. 147-154.
124. Trivedi, K.S., *Probability and statistics with reliability, queuing, and computer science applications, 2nd Edition*. 2001: John Wiley.
125. Bass, L., *Reasoning frameworks.* 2005.
126. Diaz-Pace, A., *Integrating quality-attribute reasoning frameworks in the ArchE design assistant.* Quality of Software Architectures. Models and Architectures, 2008: p. 171-188.
127. Gokhale, S.S. and K.S. Trivedi, *Analytical models for architecture-based software reliability prediction: A unification framework.* IEEE Transactions on Reliability, 2006. **55**(4): p. 578-590.

128. Gokhale, S. and K.S. Trivedi. *Structure–based software reliability prediction*. in *In Proc.of Fifth Intl. Conference on Advanced Comput-ing (ADCOMP 97)*. 1997. Chennai,India.

129. Cheung, R.C., *A User-Oriented Software Reliability Model.* Software Engineering, IEEE Transactions on, 1980. **SE-6**(2): p. 118-125.

130. Laprie, J.-C., *DEPENDABILITY EVALUATION OF SOFTWARE SYSTEMS IN OPERATION.* IEEE Transactions on Software Engineering, 1984. **SE-10**(6): p. 701-714.

131. Kubat, P., *Assessing reliability of modular software.* Operations Research Letters, 1989. **8**(1): p. 35-41.

132. Coit, D.W. and A.E. Smith, *Redundancy allocation to maximize a lower percentile of the system time-to-failure distribution.* Reliability, IEEE Transactions on, 1998. **47**(1): p. 79-87.

133. Coit, D.W. and A.E. Smith, *Solving the redundancy allocation problem using a combined neural network/genetic algorithm approach.* Computers & Operations Research, 1996. **23**(6): p. 515-526.

134. David W. Coit and A.E. Smith, *Genetic algorithm to maximize a lower-bound for system time-to-failure with uncertain component Weibull parameters.* Computers and Industrial Engineering, 2002. **41**(4).

135. Coit, D.W. and A.E. Smith, *Solving the redundancy allocation problem using a combined neural network/genetic algorithm approach.* Comput. Oper. Res., 1996. **23**(6): p. 515-526.

136. Tian, Z., G. Levitin, and M.J. Zuo, *A joint reliability–redundancy optimization approach for multi-state series–parallel systems.* Reliability Engineering & System Safety, 2009. **94**(10): p. 1568-1576.

137. Nakagawa, Y. and S. Miyazaki, *Surrogate Constraints Algorithm for Reliability Optimization Problems with Two Constraints.* Reliability, IEEE Transactions on, 1981. **R-30**(2): p. 175-180.

138. Cao, L., J. Cao, and M. Li, *Genetic algorithm utilized in cost-reduction driven web service selection.* Computational Intelligence and Security, 2005: p. 679-686.

139. Bhunia, A., L. Sahoo, and D. Roy, *Reliability stochastic optimization for a series system with interval component reliability via genetic algorithm.* Applied Mathematics and Computation, 2010. **216**(3): p. 929-939.

140. SAED, A.A.A., W. KADIR, and A. YOUSIF. *A Prediction Approach to Support Alternative Design Decision for Component-Based System Development*. in *the 11th WSEAS International Conference on Software Engineering, Parallel and Distributed Systems (SEPADS'12)*. 2012.

141. Taboada, H.A., *Practical solutions for multi-objective optimization: An application to system reliability design problems.* Reliability Engineering & System Safety, 2007. **92**(3): p. 314-322.

142. Breedam, A.V., *Comparing descent heuristics and metaheuristics for the vehicle routing problem.* Computers & Operations Research, 2001. **28**(4): p. 289-315.

143. Luenberger, D.G. and Y. Ye, *Linear and nonlinear programming*. Vol. 116. 2008: Springer Verlag.

144. Bussieck, M.R. and A. Pruessner, *Mixed-integer nonlinear programming.* SIAG/OPT Newsletter: Views & News, 2003. **14**(1): p. 19-22.

145. Postolache, R. *Linear and Nonlinear Optimization Programming*. 2007 [cited 2012 15-02-2012]; Available from: http://www.capital.edu/uploadedFiles/Capital/Academics/Schools_and_Departments/Natural_Sciences,_Nursing_and_Health/Computational_Studies/Educational_Materials/Finance_and_Economics/linear30085.pdf.

146. Holland, J.H., *Adaptation in natural and artificial systems*. 1975: University of Michigan press.

147. Adil yousif, a.h.a., sulaiman mohd nor, adil ali abdelaziz *Scheduling jobs on grid computing using firefly algorithm.* Journal of Theoretical and Applied Information Technology , 14570 -JATIT, p 155 - 164 Vol 33. No. 2 2011.

148. Eberhart and S. Yuhui. *Particle swarm optimization: developments, applications and resources*. in *Evolutionary Computation, 2001. Proceedings of the 2001 Congress on*. 2001.

149. Eberhart, R.C. and Y. Shi. *Comparing inertia weights and constriction factors in particle swarm optimization.* in *Proceedings of the Congress on Evolutionary Computation*. 2000.

150. Shi, Y. and R.C. Eberhart. *Parameter selection in particle swarm optimization*. in *Proceedings of Evolutionary Programming VII (EP98).* 1998.

151. Shi, Y. and R. Eberhart. *A modified particle swarm optimizer*. in *Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on*. 1998.

152. Eberhart, R. and Y. Shi, *Comparison between genetic algorithms and particle swarm optimization Evolutionary Programming VII*, V. Porto, Editors. 1998, Springer Berlin / Heidelberg. p. 611-616.

153. Kennedy, J. and R.C. Eberhart. *A discrete binary version of the particle swarm algorithm*. in *Systems, Man, and Cybernetics, IEEE International Conference on Computational Cybernetics and Simulation.* 1997.

154. Kennedy, J. and R. Eberhart. *Particle swarm optimization*. in *Proceedings, IEEE International Conference on Neural Networks*. 1995.

155. Liang, J.J., Qin, A. K.,Suganthan, P. N., and Baskar, S., *Comprehensive learning particle swarm optimizer for global optimization of multimodal functions.* IEEE Transactions on Evolutionary Computation, 2006. **10**(3): p. 281-295.

156. Qasem, S.N. and S.M. Shamsuddin, *Radial basis function network based on time variant multi-objective particle swarm optimization for medical diseases diagnosis.* Applied Soft Computing, 2011. **11**(1): p. 1427-1438.

157. ZHOU, C., *Particle Swarm Optimization (PSO) Algorithm [J]*. Application Research of Computers, 2003. **12**: p. 7-11.

158. Mostaghim, S. and J. Teich. *Strategies for finding good local guides in multi-objective particle swarm optimization (MOPSO).* in *Swarm Intelligence Symposium, 2003. SIS '03. Proceedings of the 2003 IEEE*. 2003.

159. Coello Coello, C.A. and M.S. Lechuga. *MOPSO: a proposal for multiple objective particle swarm optimization.* in *Evolutionary Computation, 2002. CEC '02. Proceedings of the 2002 Congress on*. 2002.

160. Carlisle, A. and G. Dozier, *An Off-The-Shelf PSO.* Proceedings of the Particle Swarm Optimization Workshop, 2001: p. 1–6.

161. Kotinis, M., *A particle swarm optimizer for constrained multi-objective engineering design problems.* Engineering Optimization, 2010. **42**(10): p. 907-926.

162. delValle, V., Mohagheghi,Hernandez , and Harley, *Particle Swarm Optimization: Basic Concepts, Variants and Applications in Power Systems.* Evolutionary Computation, IEEE Transactions on, 2008. **12**(2): p. 171-195.

163. Xiaohui, H. *PSO Tutorial*. 2006 [cited 2012 12-6-2012]; Available from: http://www.swarmintelligence.org/tutorials.php.

164. Zhou, L. and J. Zheng, *A New Immune Clone Algorithm to solve the constrained optimization problems.* WSEAS Transactions on Computers, 2011. **10**(4): p. 105-114.

165. Harman, M., *The Current State and Future of Search Based Software Engineering*, in *2007 Future of Software Engineering*. 2007, IEEE Computer Society. p. 342-357.

166. Miettinen, K., *Nonlinear multiobjective optimization*. Vol. 12. 1999: Springer.

167. Ishibuchi, H., T. Doi, and Y. Nojima, *Incorporation of scalarizing fitness functions into evolutionary multiobjective optimization algorithms.* Parallel Problem Solving from Nature-PPSN IX, 2006: p. 493-502.

168. Talbi, E.-G., *METAHEURISTICS FROM DESIGN TO IMPLEMENTATION*. 2009, Hoboken, New Jersey: John Wiley & Sons, Inc., .

169. Karl O. Jones, G., and goire Boizant, *Comparison of Firefly algorithm optimisation, particle swarm optimisation and differential evolution*, in *Proceedings of the 12th International Conference on Computer Systems and Technologies*. 2011, ACM: Vienna, Austria. p. 191-197.

170. Saed, A.A.A., W.M.N.W. Kadir, and S.Z.M. Hashim, *Metaheuristic Search Approach Based on In-house/Out-sourced Strategy to Solve Redundancy Allocation Problem in Component-Based Software Systems.* International journal of software engineering and its applications, 2012. **6**(3).

171. van den Bergh, F., *An Analysis of Particle Swarm Optimizers.*, in *Faculty of Natural and Agricultural Science*. 2001, University of Pretoria.

172. Zelkowitz, M.V. and D.R. Wallace, *Experimental models for validating technology*. Computer, 1998. **31**(5): p. 23-31.

173. Lázaro, M. and E. Marcos. *Research in software engineering: Paradigms and methods*. 2005: Citeseer.

174. J. M. Van Den Akker, S.B., G. Diepen, and J. Versendaal. *Determination of the Next Release of a Software Product: an Approach using Integer Linear*

Programming. in *Proceeding of the 11th International Workshop on Requirements Engineering: Foundation for Software Quality REFSQ'05*. 2005.

175. Avizienis, A., Laprie, J.C.,Randell, B., and Landwehr, C., *Basic concepts and taxonomy of dependable and secure computing.* Dependable and Secure Computing, IEEE Transactions on, 2004. **1**(1): p. 11-33.
176. McIlroy, M.D., Buxton, JM,Naur, P., and Randell, B. *Mass-produced software components*. in *Proceedings of the 1st International Conference on Software Engineering, Garmisch Pattenkirchen, Germany*. 1968: sn.
177. Hissam, kml., *Enabling predictable assembly*. Journal of Systems and Software, 2003. **65**(3): p. 185-198.
178. Avizienis, A., J.C. Laprie, and B. Randell, *Fundamental concepts of dependability.* Technical Report Series-University Of Newcastle Upon Tyne Computing Science, 2001.
179. Derrac, J., García, S.,Molina, D., and Herrera, F., *A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms.* Swarm and Evolutionary Computation, 2011. **1**(1): p. 3-18.
180. Witkowski, T., A. Antczak, and P. Antczak, *Comparison of Optimality and Robustness between SA, TS and GRASP Metaheuristics in FJSP Problem.* Advanced Intelligent Computing Theories and Applications, 2010: p. 319-328.
181. Burke, E., E.K. Burke, and G. Kendall, *Search methodologies: introductory tutorials in optimization and decision support techniques*. 2005: Springer Verlag.
182. Smith, C.U. and L.G. Williams. *Building responsive and scalable web applications*. 2000: Computer Measurement Group; 1997.
183. Shatz, S.M., J.P. Wang, and M. Goto, *Task allocation for maximizing reliability of distributed computer systems.* Computers, IEEE Transactions on, 1992. **41**(9): p. 1156-1168.
184. Meedeniya, B., Aleti, and Grunske, *Architecture-Driven Reliability and Energy Optimization for Complex Embedded Systems Research into Practice – Reality and Gaps*, G. Heineman, J. Kofron, and F. Plasil, Editors. 2010, Springer Berlin / Heidelberg. p. 52-67.

185. Associations, T., *Technician Guidelines for Antilock Braking Systems*, V. The Maintenance Council American Trucking Associations 2200 Mill Road Alexandria, Editor. 1998, U.S. Department of Transportation Federal Highway Administration.
186. Vibhu Saujanya Sharmaa and Kishor STrivedib, *Quantifying softwareperformance, reliability and security: An architecture-based approach.* Journal of Systems and Software (2007), 2006. **80**(4): p. 493-509.
187. Grunske, L., *Towards an integration of standard component-based safety evaluation techniques with saveCCM*. 2006.
188. kerholm , Johan Fredriksson, and K. Sandstr, *Quality Attribute Support in a Component Technology for Vehicular Software.* Fourth Conference on Software Engineering Research and Practice in Sweden, 2004.
189. Li, J.Z., Chinneck, J., Woodside, M., and Litoiu, M. *Fast scalable optimization to configure service systems having cost and quality of service constraints*. in *Proceedings of the 6th international conference on Autonomic computing*. 2009: ACM.
190. Brun, Y., *Smart redundancy for distributed computation*. 2011: IEEE.
191. Sarmenta, L.F.G., *Sabotage-tolerance mechanisms for volunteer computing systems.* Future Generation Computer Systems, 2002. **18**(4): p. 561-572.
192. Sun-Devil-Auto. *ABS Brakes*. 2012 [cited 2012 15-12-2012]; Available from: http://www.sundevilauto.com/auto-diagrams/abs-brakes.