CHAPTER 1

INTRODUCTION

The long-term success of a software system is ultimately governed by its ability to fulfill requirements on quality attributes [1]-[4]. Quality attributes include three key aspects; operational aspects inclusive of performance and reliability, structural aspects inclusive of ease of maintenance and flexibility, development aspects inclusive of cost and time. Over the past three decades, research has been going on to predict the quality of a software system from a high level design description. In 1968, Dijkstra introduced the notion of layered structure in operating systems in which related programs were grouped into separate layers, communicating with groups of programs in adjacent layers [5]. In 1972, Parnas [6] proposed the idea of modularization and information hiding as a means of high level system decomposition for improving the flexibility and understandability of a software system. In 1974, Stevens et al. [7] introduced the notions of module coupling and cohesion to evaluate alternatives for program decomposition. Currently, software architecture is emerging as an appropriate design document for maintaining quality of software systems.

Kazman and Clements attempts to calculate the values for quality attributes of the software architectures [8][9], and this is notoriously difficult. Other evaluation methods focus on providing qualitative descriptions of the benefits and liabilities of single software architecture like ATAM [10], ADD [11], Global Analysis and QASAR [12]. But these methods lack the ability to quantitatively compare different software architectures with each other using a set of quality attributes. It is important to ensure that the best possible architecture is selected by comparing the set of quality attributes.

The purpose of the research work is to present an improved method for quantitative evaluation, comparison, analysis and selection among different software architecture candidates based on an arbitrary set of quality attributes. In this chapter, the research problem is briefly introduced by recounting concepts of software architecture and their evaluation. The motivation, objectives and the problem statement of the
research work are also formulated and explained. The contribution based on this research and the organization of the chapters is presented in a nutshell.

1.1 Software Architecture

Software architecture design provides a high level abstraction of a system. The popular definition is [1]:

*The software architecture of a program or computing system is the structure or structures of the system, which comprise of software elements, the externally visible properties of those elements, and the relationships among them.*

It lays the structural foundation of a system. It allows designers to visualize that a design is viable and that it would satisfy key requirements. When performing architectural design, architects need to consider both the functional and non-functional requirements of a system. Other aspects such as project schedule, budget, information technology strategies and design trade-off are some of the considerations in architectural design. Such diversity of considerations makes it challenging to balance the conflicting interests in an architecture design. The question to address is how architects should organize and trade-off among a wide range of considerations to provide a quality design. Moreover, the quality of the architecture design depends on the architect’s experience, knowledge and decision making abilities [13]. Very few methodologies prescribe on how architecture design decisions could be made and verified, and as such decision making is largely dependent on software architects. Good decisions result in high quality system design and bad decisions result in poor quality system design. Therefore, design decisions are an important aspect in architecture design.

1.2 Architectural Knowledge

A recent development in software architecture research is the notion of Architectural Knowledge (AK). There has been an increasing awareness that not only the architectural design is important to capture, but also the knowledge pertaining to it. Often, this so-called architectural knowledge is defined as the set of design decisions [14][15], including the rationale for these decisions [16], together with the resulting
architectural design [17]. Architectural knowledge is vital for the architecting process, as it improves the quality of this process and of the architecture itself [18]. Establishing ways to manage and organize such architectural knowledge is a key challenge [19]. Software architecting is a highly knowledge-intensive process. Due to the increase in size and complexity of software systems, architecting means collaborating with different stakeholders [20]. Hence, it is often the case that there is no single all-knowing architect; instead the architect’s role is fulfilled by more than one person [21]. In order to make well-founded decisions, all involved stakeholders need to obtain relevant architectural knowledge at the right place, at the right time. Consequently, sharing such architectural knowledge is crucial, especially for reusing best practices, attaining a more transparent decision making process, providing traceability among design artifacts, recalling past design decisions and their rationale. The absence of a disciplined approach for managing architecture knowledge has many consequences such as [22]:

- Evolution of a software system becomes complex and cumbersome, resulting in violations of the fundamental design decisions.
- Inability to identify design errors.
- Inadequate clarification of arguments and less information sharing about the design and process.

1.3 Software Architecture Evaluation

Software architecture evaluation is a technique which determines the quality properties of software architectures, or software architectural styles, or software design patterns by analyzing them. This evaluation verifies architectural decisions against problem statements and requirement specifications [23]. It determines the degree to which software architecture or an architectural style satisfies its quality requirements. Software qualities are not completely independent and may interact with each other positively or negatively. e.g., modifiability has negative impact on performance and positive aspect on reliability and security. Architecture evaluation helps to identify and analyze these trade-offs among quality attributes.
The goal of *Software Architecture Analysis Method (SAAM)* [24] is to verify basic architectural assumptions and principles against the documents describing the desired properties of an application. SAAM was developed for modifiability but is being used for various quality attributes. Business drivers, software architecture description and quality requirements are the main inputs to this method. The outputs include quality sensitive scenarios, mappings between those scenarios and software architecture components and effort associated with each changing scenario.

*The Architecture Tradeoff Analysis Method (ATAM)* [10] helps the stakeholders to understand the consequences of architectural decisions with respect to the system’s quality attribute requirements and business goals. ATAM reveals how well the architecture satisfies particular quality goals. It also provides insight into how these quality goals interact with each other – how they trade-off against each other.

*Cost Benefit Analysis Method (CBAM)* [25] is an architecture-centric method for analyzing the benefits, costs and schedule implications of architectural decisions. It also assesses the uncertain surrounding, the cost and benefits, based on which a decision can be made on architectural design.

Software architecture evaluation is a more flexible way to uncover problems in software architecture before the software architecture has been committed to code. There are some challenges in architecture evaluation includes:

- Lack of common understanding of the high level design. Software architects often do not document design rationale. When they do, they may not follow a systematic way of expressing the rationale [26]. There is no standard notion to describe software architectures. Recent research suggests that design decisions and the rationale should be recorded along with the architectural description.

- The purpose of the architecture evaluation of a software system is to analyze the architecture to identify potential risks and to verify that the quality requirements have been addressed in the design. However, it is too difficult to analyze an architecture based on these abstract qualities which are too vague and provide very little procedural support for evaluating an architecture [27].
1.4 Motivation

Through research study, it was found that software architects were constantly challenged to address the conflicting but the essential requirements of various stakeholders. Moreover, the architects need a knowledge source to experienced based design decisions to assist in making quality decisions or discovering the rationale for the past decisions. Appropriate knowledge management support is needed for improving the quality of software architecture process, by providing the knowledge about quality attributes in a particular domain. It has been mentioned that one of the main challenges in evaluating software architecture is a precise definition of the quality attributes that are used in architecture evaluation and architectural knowledge about the decisions taken for evaluation. A framework is needed to empower a software architecture evaluation method called the Architecture tradeoff Analysis Method thereby effectively reusing the software architecture knowledge.

Given a set of architectures for a software system, the most suitable software architecture providing maximum satisfaction to the stakeholders is to be identified. Choosing the architectural design that guarantees the satisfaction of the quality attributes is a multi criteria decision making problem. As existing software architecture evaluation methods fail to quantitatively analyze, a quantitative evaluation method is needed to evaluate the candidate architectures over a set of quality attributes that can assess the systems’ ability to deliver its intended functionality.

The integrated ATAM/CBAM [28] combination evaluates the software architecture by considering the business goals and requirement. Here the CBAM evaluation will be subsequently performed after the ATAM valuation. The point of maximum quality attainment for the minimum amount of investment is exactly the point of interest to the software manager. The optimization framework is needed to consider both the cost and quality implication of different design options. It is observed that reasonably high quality architectures can be produced by genetic synthesis. To find the solutions, an automated approach is needed to search the design space for good optimal solutions. Metaheuristic search techniques improve software architectures in finding new alternatives with respect to quality factors and cost. Starting with a given initial
architectural model, new architectures are automatically generated and evaluated for the quality criteria. This technique needs less manual effort to find good design alternatives and eases the work of the software architect in choosing the high quality design.

1.5 Objectives of the Research work

Effective analysis and evaluation of software architecture is needed to extract the incomplete specifications from the stakeholder’s viewpoint. It is obvious that it is actually the documentation of software architecture which transforms the evaluation process to an effective one. Architectural decisions take different quality attributes into account and are the key factors in exposing risks and sensitivity points of the architecture while the evaluation is in progress. The architectural styles play an important role during evaluation to extract the possible combination of design decisions from earlier experience. It is important to have well formed and expressive infrastructure to document the current architectural knowledge to reuse the past evaluation experience in a structured way.

The objectives of the research are:

• To devise an Architectural Knowledge framework with the intent of accessing Software Architecture Knowledge thereby improving the architecture evaluation.

• To devise a Quantitative evaluation method for analyzing the quality of the decisions using the retrieved Architectural Knowledge.

• To enhance the above evaluation method, Multi-objective optimization approach is involved in selecting the most optimal design with maximum quality within the given budget constraints.

1.6 Problem statement

The limitations existing in the above architectural knowledge based evaluation techniques have paved way to explore a quantitative and automated evaluation to select an optimal design. The research problem is defined as follows:
• Given a prioritization of quality requirements, the most suitable software architecture has to be suggested following a Quantitative approach based on architectural knowledge.

• Given the stakeholder budget, the optimal architectural design has to be identified so that it satisfies the quality requirements for the given budget.

1.7 Summary of the Research Contributions

Based on the research objectives the following evaluation techniques have been developed in this research work.

1.7.1 Ontology Based Architectural Knowledge Framework (OBAKF)

Existing architectural knowledge management tools have been analyzed to identify their limitations. To overcome the limitations and challenges in the existing methods, the software architectural knowledge needs to be managed for improving the architectural capabilities. Architectural Development using Architectural Knowledge has been developed to support a framework for capturing and using managed architectural knowledge to improve the architecture design process. The developed framework proposes an ontology to conceptualize the issues related to the architecture evaluation process and guides the architect in selecting the most appropriate architecture, where the interactions among quality attributes, tactics should be analyzed for the quality driven architectural design process. The stakeholders need to acquire the knowledge that resides in the software audit documents. After applying the developed ontology representation framework for a typical case study, it was found that the framework offers the following advantages:

• Architectural Knowledge can be accessed faster with the above framework as it reduces the design time.
• Access to the rationale improves the quality of the decisions.
• Effective sharing and reusing the architectural knowledge is facilitated.
• The software architecting process can be improved by providing competing design alternatives.
1.7.2 Quantitative Quality Driven Software Architecture Evaluation using FAHP

The quality attributes of large software systems are determined by the system’s software architecture. Architecture Tradeoff Analysis Method (ATAM) is a scenario based software architecture evaluation method extended from SAAM. ATAM is a method for evaluating architecture-level designs that considers multiple quality attributes such as modifiability, performance, reliability and security. The evaluation of architectural designs is a multiple-criteria decision making (MCDM) problem. Analytic Hierarchy Process (AHP) has been successfully used in the areas of software engineering. However, the experiences and judgments of humans are normally linguistically represented and it is not quantitatively digital (well defined condition). The human assessment on qualitative attributes is always subjective and thus imprecise. Hence, conventional AHP seems inadequate to capture decision makers' requirements explicitly [29]. As an alternative, the Fuzzy Analytic Hierarchy Process (FAHP) [30] approach is chosen to evaluate the architectural design for the achievement of qualities. The usage of the proposed quantitative quality driven software architecture evaluation using FAHP has been exemplified through the suitable case studies.

1.7.3 Software Architecture Selection Framework based on Genetic Algorithm

Choosing a portfolio of architectural designs that meets an organization's objectives without exceeding available capital investments is recognized as a critical issue for which the decision maker needs to take several aspects into consideration. As most of these aspects may be conflicting, this problem can be considered as a multi-objective one [31]. The multi-objective optimization algorithm with Genetic Algorithm (GA) allows us to identify a set of optimal solutions providing the decision maker with the complete spectrum of optimal solutions with respect to the various targets. Instead of evaluating software architecture in the formal manner, genetic algorithm optimizes the architecture with respect to certain quality requirements in the context of a particular system. GAs can be applied to the problem of the architectural design, interpreting the possible solutions which can be modeled as the chromosomes and features as genes. The evolution happens through generations with the idea of producing new offspring which combines the characteristics of the two individual solutions. Software architecture design
with the GA approach ensures that the solutions are quality consistent. The ability to handle the large search space and automating the identification of good architecture is handled. This will reduce the production costs and improve the customer satisfaction. The gain obtained using the new improved crossover encourages to use this crossover. Experimental results indicate that fitness function scores are proportional to the relevance of the obtained architectural solutions compared to the expected ones.

1.8 Organization of Chapters in the Thesis

This thesis comprises of six chapters.

Chapter 1 introduces the research problems and presents the general overview along with the challenges of the problem domain. The motivation and the objectives of the thesis are also formulated and explained in this chapter.

Chapter 2 deals with the existing evaluation methods and the architectural knowledge management tools. It explains the fundamental concepts of evaluation, architectural knowledge and the metaheuristic algorithm. Further, this chapter discusses the strengths and weaknesses of the existing methods.

Chapter 3 focuses on the development of the architectural knowledge development framework to overcome the limitations and challenges of existing architectural knowledge management tools. The usage of the framework is exemplified through a suitable case study. The benefits and limitations of the framework are discussed in detail.

Chapter 4 deals with the analysis of a quantitative quality driven software architecture evaluation using fuzzy analytical hierarchical process. The usage of the model is exemplified through a suitable case study. Validation of the framework is also discussed.

Chapter 5 deals with the development of the selection framework using genetic algorithm for software architectures. The usage of the framework is exemplified through a suitable case study.

Chapter 6 concludes the thesis by highlighting the findings that facilitated to accomplish the objectives of the research work. The limitations of the research work have been stated and an insight into the possible future research directions is also given.