

**METRIC-ORIENTED QUALITY MODEL  
FOR  
ARCHITECTURE TRADEOFF ANALYSIS  
METHOD**

T. 4997  
=



**Asif Javed  
Abdul Hafeez**

Supervised By

**Prof. Dr. Khalid Rashid**



**Department of Computer Science  
Faculty of Applied Sciences  
International Islamic University Islamabad  
2006**

*In the Name of ALLAH*

**The Most Merciful**

**The Most Beneficent**

**INTERNATIONAL ISLAMIC UNIVERSITY ISLAMABAD  
DEPARTMENT OF COMPUTER SCIENCE**

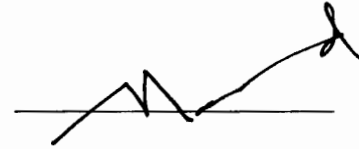
Dated: \_\_\_\_\_

**FINAL APPROVAL**

It is certified that we have read the research thesis submitted by Mr. Asif Javed, registration no. 56-FAS/MSSE/F04 and Mr. Abdul Hafeez, registration no. 65-FAS/MSSE/F04 and it is our judgment that this thesis is of sufficient standard to warrant its acceptance by International Islamic University Islamabad for the Master of Science (MS) in Software Engineering.

**External Examiner**

**Dr. Naveed Ikram**  
Assistant Professor,  
Muhammad Ali Jinnah University,  
Jinnah Avenue Blue Area,  
Islamabad.




**Internal Examiner**

**Prof. Dr. Sikandar Hayat Khiyal,**  
Head,  
Department of Computer Science  
International Islamic University,  
Islamabad-Pakistan



**Supervisor**

**Prof. Dr. Khalid Rashid**  
Dean,  
Faculty of Applied Sciences,  
International Islamic University,  
Islamabad-Pakistan



*A*

*Dissertation*

*Submitted as Partial Fulfillment*

*of Requirements for the Award of The Degree of*


**MS in Software Engineering**

---

*To*  
*The Holiest man ever born,*  
*Prophet Muhammad (Peace Be Upon Him)*  
*&*  
*Our Dear Parents & Family*  
*Who are an embodiment of diligence and honesty,*  
*Without their prayers and support*  
*This dream could have never come true*

## DECLARATION

We hereby declare that this research neither as a whole nor as a part thereof has been copied out from any source. It is further declared that we have conducted this research and proposed an improved modification to software architecture evaluation method, case study and accompanied thesis on the basis of our personal efforts, under the sincere guidance of our supervisor. If any artifact of this research is proved to be copied out from any source or found to be reproduction of someone else, we shall stand by the consequences.



*Asif Javed*  
56-FAS/MSSE/F04



*Abdul Hafeez*  
65-FAS/MSSE/F04

## ACKNOWLEDGEMENTS

Countless gratitude to the Almighty Allah, who is Omnipotent and He, who blessed us with the ability to read and write. He blessed us with a chance and choice, health and courage to achieve this goal.

We would like to pay special regards to our parents. Without their support and prayers we could have never completed this research work. We also want to thank Dr. Hafiz Farooq Ahmad who blessed us with the guidelines for proving our research idea. We owe a debt of gratitude to Prof. Dr. Len Bass, professor at Software Engineering Institute, Carnegie Mellon University USA, for encouraging us in our research work and providing the research material which helped us to prove our work. Let us not forget the prayers of our brothers, sisters and beloved friends. They all always helped us in our down times and boosted our morals. We would like to pay them the very special thanks for their best wishes, encouragement and support in not only this research but throughout our lives without that we would have not been able to achieve anything worthwhile.

We would like to pay special thanks to our supervisor Prof. Dr. Khalid Rahsid, Dean, Faculty of Applied Sciences for his timely guidance, encouragement and personal interest to maintain the quality of work apart from his duties throughout the research. Thanks to Dr. Sikandar Hayat Khiyal, Head, Department of Computer Science and Dr. Sy. Afaq Hussain, Head, Department of Telecommunication & Computer Engineering, Mr. Muhammad Amir Aman and Mr. Muhammad Sulayman, Lecturers of Software Engineering, International Islamic University Islamabad without their proper guidance we could have never achieved this milestone.

We would also like to mention the support of International Islamic University Islamabad who generously provided us R&D Funds for the international conference held at Berlin, Germany. We also owe a great deal to all the respected colleagues who extended their help whatsoever we needed.

*Asif Javed*  
*Abdul Hafeez*

## **PROJECT IN BRIEF**

**Project Title:**

Metric-Oriented Quality Model for Architecture Tradeoff Analysis Method

**Objectives:**

- An Improved modification to software architecture evaluation method.
- Application of the proposed model on a case study.
- To discover and explore the benefits with implication of this proposed idea.

**Undertaken By:**

Asif Javed  
56-FAS/MSSE/F04

Abdul Hafeez  
65-FAS/MSSE/F04

**Supervised By:**

Prof. Dr. Khalid Rashid  
Dean,  
Faculty of Applied Sciences,  
International Islamic University Islamabad

**Started On:**

September 2005

**Completed On:**

August 2006

**Research Area:**

Software Engineering, Software Architecture,  
Software Architecture Evaluation, Software Quality and  
Software Metrics



## **ABSTRACT**

It has been realized by the software engineering communities that software architecture serves as an important artifact in producing quality software products. Architecture Tradeoff Analysis Method (ATAM) proposed by Software Engineering Institute (SEI) is considered to be a well-known methodology for evaluating software architectures. Current research observes a lack of proper guidelines and a well-defined approach in generation of quality attributes utility tree. This lack of standard has limited the usefulness of this method. We proposed a quality model based on ISO 9126-1 quality model. We support architecture evaluation using Goal Question Metric (GQM) approach to answer the questions raised by the stakeholders to fulfill business drivers in the light of ISO 9126-1 quality model. We also applied the method to a Case study Battlefield Control System (BCS) and performed the architecture evaluation by presenting results to overcome the existing problem.

# TABLE OF CONTENTS

Ch. No.	Contents	Page No.
1	Introduction.....	2
1.1	Background.....	3
1.1.1	System Software Architecture .....	3
1.1.2	Software Architecture Evaluation.....	4
1.2	Software Architecture as a Research Area .....	6
1.3	Research Agenda.....	6
1.3.1	Objectives.....	7
1.3.2	Approach.....	7
1.4	Validation of the Research.....	7
1.4.1	Battlefield Control System (BCS) – Case Study.....	8
1.5	Overview of Thesis.....	8
1.6	Literature Survey.....	9
1.6.1	Software Quality .....	9
1.6.2	Software Quality Models.....	11
1.6.2.1	McCall’s Model .....	11
1.6.2.2	Boehm’s Model.....	12
1.6.2.3	ISO/IEC 9126 .....	13
1.6.3	Architecture Tradeoff Analysis Method (ATAM).....	14
1.6.4	Goal Question Metric Approach.....	20
2	Problem Definition.....	23
2.1	Hypothesis.....	23
2.2	Problem Scope .....	24
2.3	Problem Statement .....	25
2.4	Significance of the Research .....	26
3	Proposed Solution.....	28
3.1	Conceptual Model with respect to ATAM .....	31
3.2	Conceptual Model with respect to GQM approach .....	33
4	Application of MoQaMo to a Case Study.....	37
4.1	Driving Architectural Requirements.....	37
4.2	High-Level Architectural Views .....	38
4.3	Architectural Evaluation .....	40
4.4	Comparison and Results .....	41

5 Conclusion.....	48
5.1 The Benefits.....	48
References & Bibliography.....	51
Appendix A- Research Paper	

## **Chapter 1**

# **INTRODUCTION**

## 1 Introduction

Architecture usually focuses on a set of business and technical decisions. There are many influences at work in its design, and the realization of these influences is subject to change according to the environment in which the architecture is required to perform. An architect designing a system for which the low budget is allocated and short deadline is specified will opt to take one set of decisions; the same architect, designing a similar system in which the deadlines can be easily satisfied, will make different choices. Even with the same requirements, hardware, support software, and human resources available, an architect designing a system today is likely to design a different system that might have been designed five years ago.

The purpose of architecture evaluation of software systems is to analyze the architecture in order to identify potential risks and verify that quality requirements have been addressed in the design. Architecture Tradeoff Analysis Method (ATAM) focuses on understanding the consequences of architecture decisions with respect to quality attribute requirements of the system [1]. Quality is one of the major issues [2]. This has been the oldest practice in the software industry to predict the quality of a software product from higher-level design [3]. Currently software architecture is considered to deal with software quality. It has been realized by the software engineering community that software architecture serves as an important artifact in producing quality software products [4]. The Architecture Tradeoff Analysis Method (ATAM) defines the attribute utility tree to provide a top down mechanism for directly and efficiently interpreting the business drivers of the system into concrete quality attribute scenarios [1]. Non-functional requirements, like performance, maintainability and reliability are represented as architectural drivers. The utility tree is formed from the combination of these architectural drivers. These quality requirements are refined into relevant attributes in order to get a prioritized list of scenarios that serves as a plan for the remaining architecture evaluation process.

ATAM uses one level of quality characteristics. However there is not any specific guideline to define the utility tree. Expression of quality view and the reason for one level of refinement is ambiguous [5]. Attributes defined in the utility tree are measured in terms of stimuli, parameters and responses. After having a utility tree, we see that there is

a lack of coordination among quality characteristics, their refined attributes and resulting scenarios which also attempt to specify measures to attributes.

The purpose of this research is to propose Metric-Oriented Quality Model (MoQaMo) based on the ISO 9126-1 [6] framework. MoQaMo utilizes the Goal Question Metric (GQM) approach to support software architecture evaluation by answering the quantifiable questions raised by the stakeholders to fulfill business drivers (goals) over the ISO 9126 framework. The work attempts to replace the utility tree used in Architecture Tradeoff Analysis Method (ATAM) with the MoQaMo model. MoQaMo guides the architecture evaluation process by clearly emphasizing the major quality characteristics. This helps to identify quantifiable questions and a relevant set of metrics derived in order to fulfill the goals.

Architectural tactics and patterns in particular explain the known properties to the systems in which they are used. Hence, design choices that is to say, architecture are analyzable. Given architecture, we can deduce things about the system, even if it has not been built yet.

## **1.1 Background**

It is better to have a right perception about the meaning of software architecture before going into the details of architectural evaluation. Different people have taken different meanings from software architecture. We will mention a few definitions of software architecture prevailing in the software engineering community. There has been a wide spread growth in the industry in the methods for software architecture evaluation. The major purpose of any architectural evaluation methodology is to assess the candidate architecture whether it fits according to the business and technical requirements of the organization or not?

### **1.1.1 System Software Architecture**

We see various schools of thought regarding the concepts of system software architecture. Software Engineering Institute (SEI), Carnegie Mellon University has played a tremendous role in the evolution of the field of software engineering. The researchers have given lot of attention to the research work on software architecture at the

Software Engineering Institute (SEI), Carnegie Mellon University. SEI has been considered as a valuable source of knowledge in software engineering field.

We mention the definition for software architecture proposed by the software engineering Institute. The Software Engineering Institute (SEI) defines what software architecture is. This definition comprises of what constitutes software architecture.

*"The software architecture of a program or computing system is the structure or structures of the system, which comprises software elements, the externally visible properties of those elements, and the relationships among them. [7]"*

Software architecture forms the backbone for building successful software-intensive systems [8]. Architecture largely permits or precludes a system's quality attributes such as performance or reliability. Architecture represents a capitalized investment, an abstract reusable model that can be transferred from one system to the next. Architecture represents a common vehicle for communication among system's stakeholders, and is the arena in which conflicting goals and requirements are mediated. The right architecture is the linchpin for software project success. The wrong one is a recipe for disaster.

### **1.1.2 Software Architecture Evaluation**

An organization should analyze software or system architecture, because it is a cost-effective way of mitigating the substantial risks associated with this highly important artifact [9]. Architectures are the blueprints for a system, and the carriers of the system's quality attributes.

It is always cost-effective to evaluate software quality as early as possible in the life cycle. If problems are found early, they are easier to correct a change to a requirement specification, or design is all that is necessary. Software quality cannot be appended late in a project, but must be inherent from the beginning, built in by design. It is in the project's best interest for prospective candidate designs to be evaluated (and rejected, if necessary) during the design phase.

However, architecture evaluation can be carried out at many points during a system's life cycle. If the architecture is still embryonic, we can evaluate those decisions that have already been made or are being considered. We can choose among architectural alternatives. If the architecture is finished, or nearly so, we can validate it before the

project commits to lengthy and expensive development. It also makes sense to evaluate the architecture of a legacy system that is undergoing modification, porting, integration with other systems, or other significant upgrades. Finally, architecture evaluation makes an excellent discovery vehicle: Development projects often need to understand how an inherited system meets (or whether it meets) its quality attributes requirements.

Quality in architecture is reported by the organizations that practice architecture evaluation as a standard part of their software development life cycle. As development organizations learn to anticipate the questions that will be asked, the issues that will be raised, and the documentation that will be required for evaluations, they naturally preposition themselves to maximize their performance on the evaluation. Architecture evaluations result in better architectures not only after the fact but before the fact as well. Over time, an organization develops a culture that promotes good architectural design.

Most complex software systems are required to be modifiable and must exhibit high performance. They may also need to be secure, interoperable, portable, and reliable. For any particular system, what precisely do these quality attributes - modifiability, security, performance, reliability - mean? Can a system be analyzed to determine these desired qualities? How soon can such an analysis occur? How do we know if software architecture for a system is suitable without having to build the system first?

Experience has shown that the quality attributes of large software systems live principally in the system's software architecture. In such systems the achievement of qualities attributes depends more on the overall software architecture than on code-level practices such as language choice, detailed design, algorithms, data structures, testing, and so forth. It is therefore a critical risk mitigation measure to try to determine, before a system is built, whether it will satisfy its desired qualities.

The Software Engineering Institute (SEI) has developed several methods for analyzing system and software architecture Active Reviews for Intermediate Designs (ARID), the Architecture Tradeoff Analysis Method (ATAM), and the Cost-Benefit Analysis Method (CBAM). These techniques can be used in combination to obtain early and continuous benefits. ARID can be used to evaluate early designs or portions of designs for their viability in satisfying stakeholder concerns. Once the software architecture is more fully developed, the ATAM can be used to reveal how well the architecture satisfies particular



quality attribute requirements and the risks, sensitivities, and tradeoffs involved in satisfying those requirements. The CBAM guides system engineers and other stakeholders to determine the economic tradeoffs associated with the architectural decisions that result in the system's qualities.

## 1.2 Software Architecture as a Research Area

We observe an enormous amount of research work in the field of software architecture during the last decade. This area has emerged as the principled study of the overall structure of software systems, especially the relations among subsystems and components. From its roots in qualitative descriptions of useful system organizations, software architecture has matured to encompass broad explorations of notations, tools, and analysis techniques. Whereas initially the research area interpreted software practice, it now offers concrete guidance for complex software design and development.

We can understand the evolution and prospects of software architecture research by examining the research paradigms used to establish its results. These are, for the most part, the paradigms of software engineering. We advance our fundamental understanding by posing research questions of several kinds and applying appropriate research techniques, which differ from one type of problem to another, yield correspondingly different kinds of results, and require different methods of validation. Unfortunately, these paradigms are not recognized explicitly and are often not carried out correctly; indeed not all are consistently accepted as valid. This retrospective on a decade-plus of software architecture research examines the maturation of the software architecture research area by tracing the types of research questions and techniques used at various stages. We will see how early qualitative results set the stage for later precision, formality, and automation and how results build up over time. This generates advice to the field and projections about future impact. [10]

## 1.3 Research Agenda

A lot of research work exists in the area of software architecture evaluation and a large number of contributions from Software Engineering Institute (SEI) at Carnegie Mellon University, USA in this area. SEI should be called as an eminent leader in this research domain. Many software architecture analysis methods have been developed by SEI that has opened up many interesting areas for research. Turning the pages of recent SEI

reports and Software Architecture conferences reveals a wealth of research directed towards architecture analysis. Architectural Tradeoff Analysis Method (ATAM) is one of the methodologies proposed by SEI. Our work focuses on this particular method and offers an improved modification by providing a quality model based on a well defined standard. We propose a quality model for this architecture evaluation method in this thesis.

### **1.3.1 Objectives**

We focus on the following objectives:

- An improved modification to Architecture Tradeoff Analysis Method (ATAM)
- Guidelines for generating utility tree
- Comparison between classic ATAM and the proposed MoQaMo for ATAM.

### **1.3.2 Approach**

The selection of appropriate research methodology is an important decision in a research project. Software engineering as a field offers many competing research paradigms and methodologies. The methodology employed in this thesis is to develop a new method for analysis or evaluation research paradigm. Software engineering research answers questions about methods of development or analysis, about details of designing or evaluating a particular instance, about generalizations over whole classes of systems or techniques, or about exploratory issues concerning existence or feasibility. Methods of analysis or evaluation help software engineers and architects to answer and to set their priorities and decisions in selecting a particular product or service.

## **1.4 Validation of the Research**

To prove MoQaMo model, we selected case study BattleField Control System (BCS). This case study was applied by Software Engineering Institute (SEI) to a method for software architecture evaluation; Architecture Tradeoff Analysis Method [ATAM]. We find this case study as an authentic source of information and fits very much in the validation of our research work. The approach to apply case study was similar to that of SEI, Carnegie Mellon University.

### **1.4.1 Case Study: BattleField Control System (BCS)**

This system was proposed for army battalions to control the movement, strategy, and operations of troops in real time in the battlefield. We divided case study into two phases. In phase 1, we presented the ATAM basic steps along with customer business case. The business case reflected information about the mission and its requirements. The requirements state that the system supports a commander who commands a set of soldiers and their equipment, including different kinds of weapons and sensors. The system need to interface with numerous other systems. Phase 2 consisted of executing architecture evaluation process by following MoQaMo model (replacing the utility tree in traditional ATAM), collecting architectural approach information, exploring stakeholders' questions for every scenarios and then satisfying these questions by subjective and objective metrics.

## 1.5 Overview of Thesis

This section provides an overview of this research thesis as follows:

Chapter 2 mainly describes the Problem Definition. Research Questions are identified which makes the focus of the research clearer to find answers.

Chapter 3 discusses the newly devised improved modification to Architecture Tradeoff Analysis Method (ATAM). An example is also described so as to clearly elaborate the proposed idea.

Chapter 4 describes the application of MoQaMo model to a case study, Battlefield Control System (BCS). A set of risks, tradeoff points and sensitivity points are identified after the evaluation of software architecture for BCS.

Chapter 5 concludes the thesis by summarizing the main findings and benefits of this model.

## 1.6 Literature Survey

This section discusses literature survey i.e. Software Quality, Software quality models, Architecture Tradeoff Analysis Method (ATAM) and Goal Question Metric approach.

### 1.6.1 Software Quality

Everyone agrees that software quality is the most important element in software development because high quality could reduce the cost of maintenance, test and software reusing. Quality has different meanings for customers, users, management, marketing, developers, testers, quality engineers, maintainers, and support personnel. Many institutes and organizations have their own definitions of quality and their own quality characteristics. The software industry is growing up daily and “it is rather surprising that more serious and definitive work has not been done to date in the area of evaluating software quality” [11]. Moreover, Kitchenham (1989) notes that “quality is hard to define, impossible to measure, easy to recognize” [12,13]. Also, Gilles states that quality is “transparent when presented, but easily recognized in its absence” [14,15]. Furthermore, Kan (2000) explains that “Quality is not a single idea, but rather a multidimensional concept. The dimensions of quality include the entity of interest, the viewpoint on that entity, and quality attributes of that entity” [16]. Some organizations try to develop standard definitions for quality. Some Definitions of international and standard organizations are [17]:

- **ISO 9126:** “Software quality characteristic is a set of attributes of a software product by which its quality is described and evaluated”.
- **German Industry Standard DIN 55350 Part 11:** “Quality comprises all characteristics and significant features of a product or an activity which relate to the satisfying of given requirements”.
- **ANSI Standard (ANSI/ASQC A3/1978):** “Quality is the totality of features and characteristics of a product or a service that bears on its ability to satisfy the given needs”.

**ISO/IEC 9126-1** defines a quality model as a framework which explains the relationship between different approaches to quality" [18]. A quality model decomposes in hierarchical elements. An approach to quality is to decompose quality in Factors, Sub-

factors, and criteria. Evaluation of a program begins with measuring each quality criteria with numerical value from metrics. Then, each quality sub-factor is assessed using their criteria. Finally, numerical values are assigned to quality characteristics from their quality sub-factors. Figure 1.1 presents a meta-model of the relationships among quality model elements.

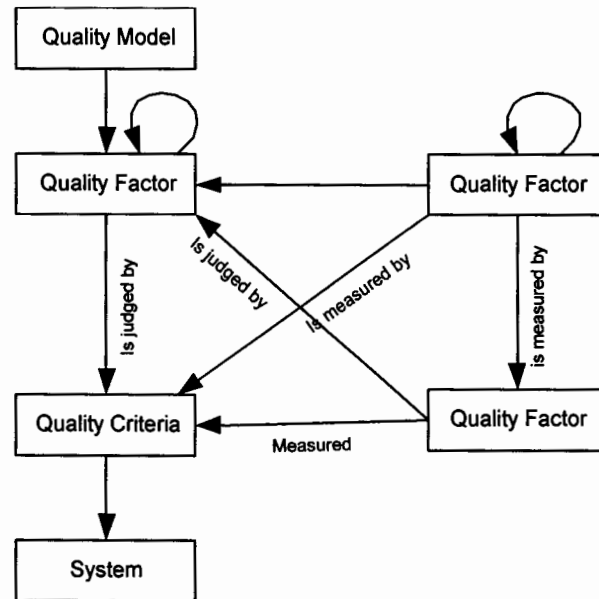


Figure 1.1: Relationship among Quality model Components

## 1.6.2 Software Quality Models

Several quality models have been defined by different people and organizations. In the following, we summarize briefly some of the most standard and well known quality models.

### 1.6.2.1 McCall's Model (1977)

McCall's model (See Figure 1.2) for software quality combines eleven criteria around product operations, product revisions, and product transitions. The main idea behind McCall's model is to assess the relationships among external quality factors and product quality criteria. McCall's Model is used in the United States for very large projects in the military, space, and public domain. It was developed in 1976-7 by the US Air- force Electronic System Decision (ESD), the Rome Air Development Center (RADC), and General Electric (GE), with the aim of improving the quality of software products [17].

One of the major contributions of the McCall model is the relationship created between quality characteristics and metrics, although there has been criticism that not all metrics are objective. One aspect not considered directly by this model was the functionality of the software product [19].

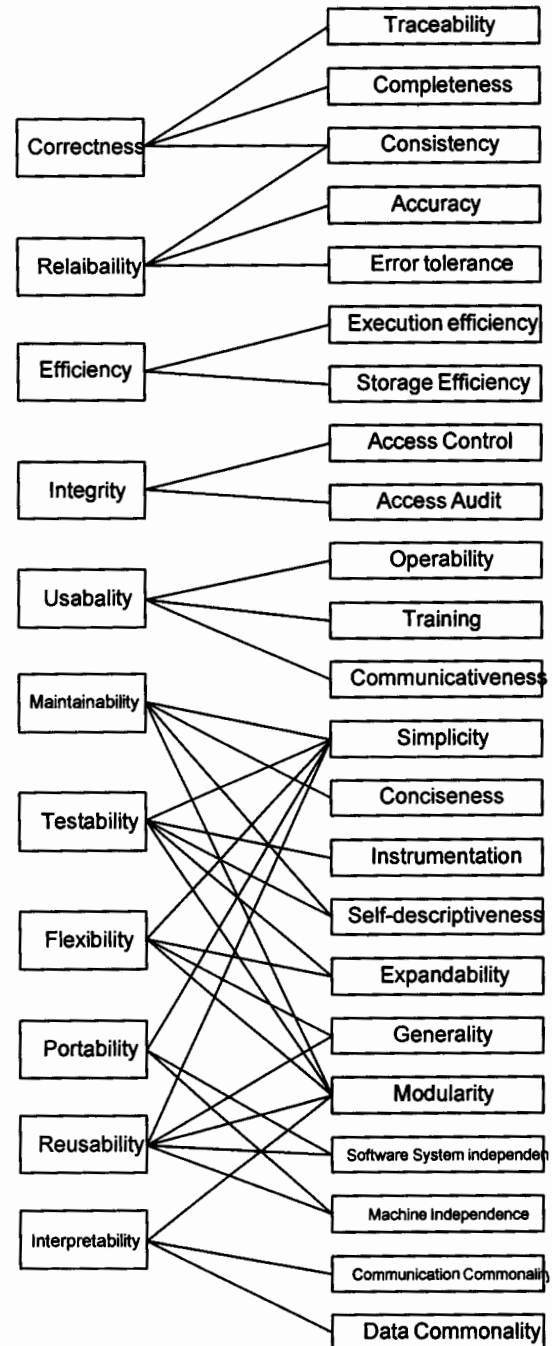


Figure 1.2: McCall's Quality Model

The layers of quality model in McCall are defined as [20]:

- Factors.
- Criteria.
- Metrics.

### 1.6.2.2 Boehm's Model (1978)

Boehm added some characteristics to McCall's model with emphasis on the maintainability of software product. Also, this model includes considerations involved in the evaluation of a software product with respect to the utility of the program, The Boehm model is similar to the McCall model in that it represents a hierarchical structure of characteristics, each of which contributes to total quality. Boehm's notion includes user's needs, as McCall's does; however, it also adds the hardware yield characteristics not encountered in the McCall model" [19].

However, Boehm's model (See Figure 1.3) contains only a diagram without any suggestion about measuring quality characteristics. The layers of quality model in Boehm are defined as [20]:

- High-level characteristics.
- Primitive characteristics.

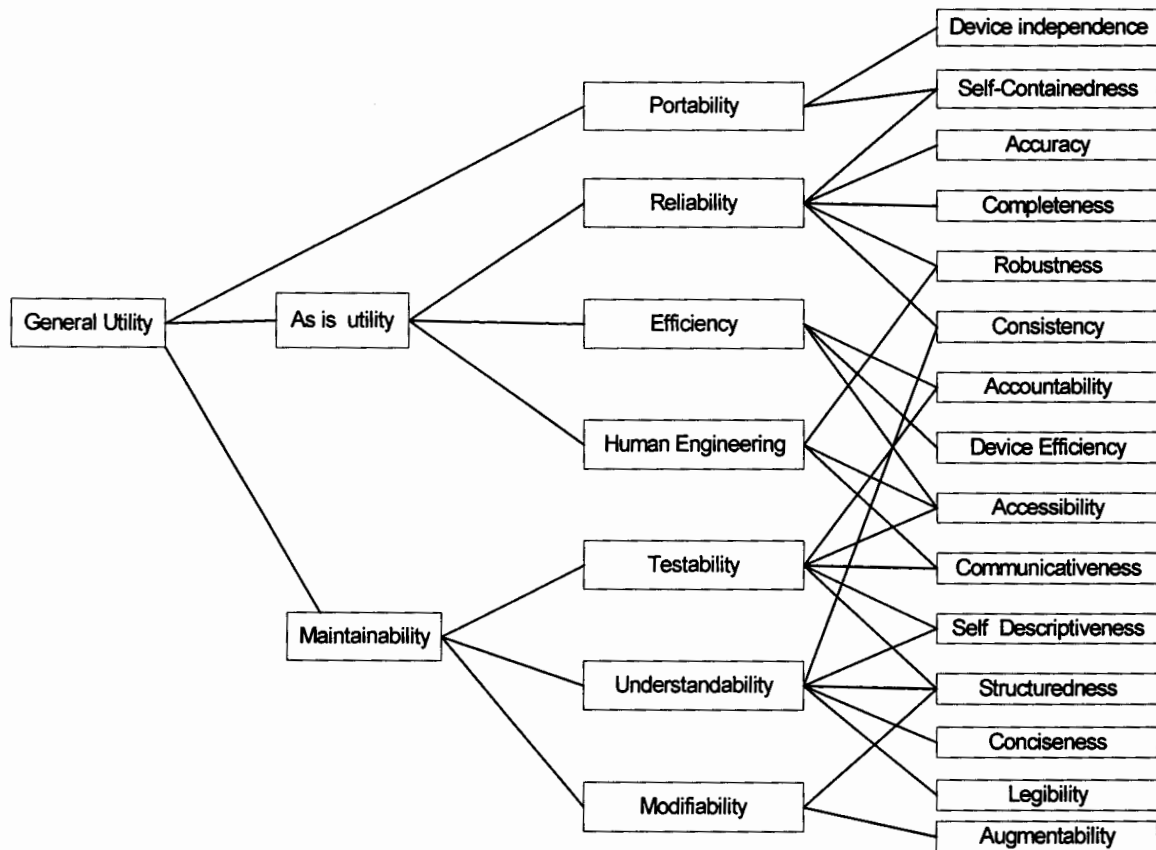


Figure 1.3: Boehm's Quality Model

### 1.6.2.3 ISO/IEC 9126 (1991)

With the need for the software industry to standardize the evaluation of software products using quality models, the ISO (International Organization for Standardization) proposed a standard which specifies six areas of importance for software evaluation and, for each area, specifications that attempt to make the six areas measurable.

One of the advantages of the ISO 9126 model (See Figure 1.4) is that it identifies the internal characteristics and external quality characteristics of a software product. However, at the same time it has the disadvantage of not showing very clearly how these aspects can be measured [19].

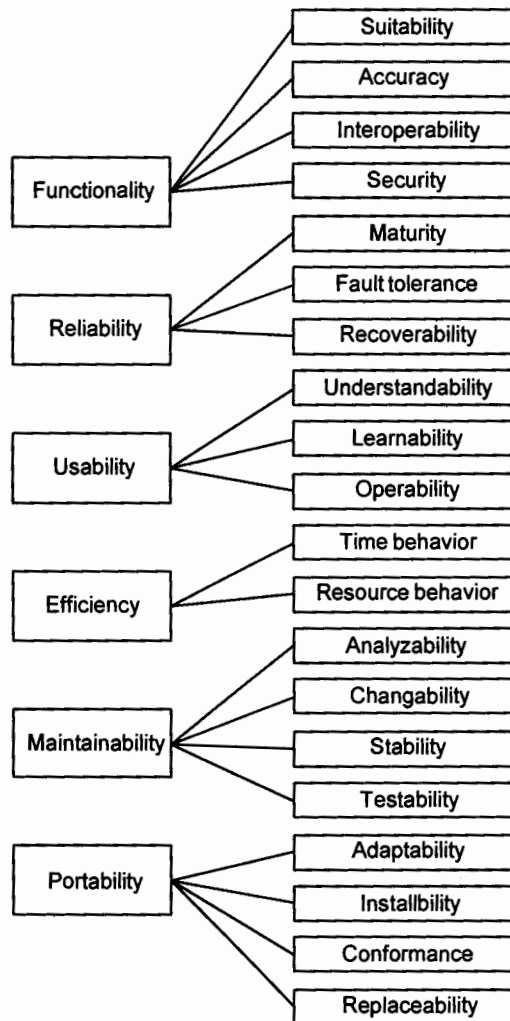


Figure 1.4: ISO/IEC 9126 Quality Model



### 1.6.3 Architecture Tradeoff Analysis Method (ATAM)

The ATAM is an analysis method organized around the idea that architectural styles are the main determiners of architectural quality attributes. The method focuses on the identification of business goals which lead to quality attribute goals. Based upon the quality attribute goals, we use the ATAM to analyze how architectural styles aid in the achievement of these goals.

#### **Purpose**

The purpose of the ATAM is to assess the consequences of architectural decision alternatives in light of quality attributes. The method ensures that the right questions are asked early to discover

- risks: alternatives that might create future problems in some quality attribute
- sensitivity points: alternatives for which a slight change makes a significant difference in a quality attribute
- tradeoffs: decisions affecting more than one quality attribute

The ATAM is intended to analyze architecture with respect to its quality attributes, not its functional correctness. It involves a wide group of stakeholders (including managers, developers, maintainers, testers, end users, and customers) in an effort to surface the relevant stakeholders' quality goals for the system. It is a method for mitigating architecture risks, a means of detecting areas of potential risks within the architecture of a complex software intensive system, and not a precise mathematical analysis. As such, the ATAM can be done early in the software development life cycle, and it can be done inexpensively and quickly.

It does not need to produce detailed analysis of any measurable quality attribute of a system (such as latency or mean time to failure) to be successful, but it instead identifies trends where some architectural parameter is correlated with a measurable quality attribute of interest.

## Steps

The ATAM process consists of nine steps, as is briefly presented. Sometimes there must be dynamic modifications to the order of steps to accommodate the availability of personnel or architectural information. Although the steps are numbered, suggesting linearity, this is not a strict waterfall process. There will be times when an analyst will return briefly to an earlier step, or will jump forward to a later step, or will iterate among steps, as the need dictates. The importance of the steps is to clearly delineate the activities involved in ATAM along with the outputs of these activities. Figure 1.5 shows steps of ATAM in four phases.

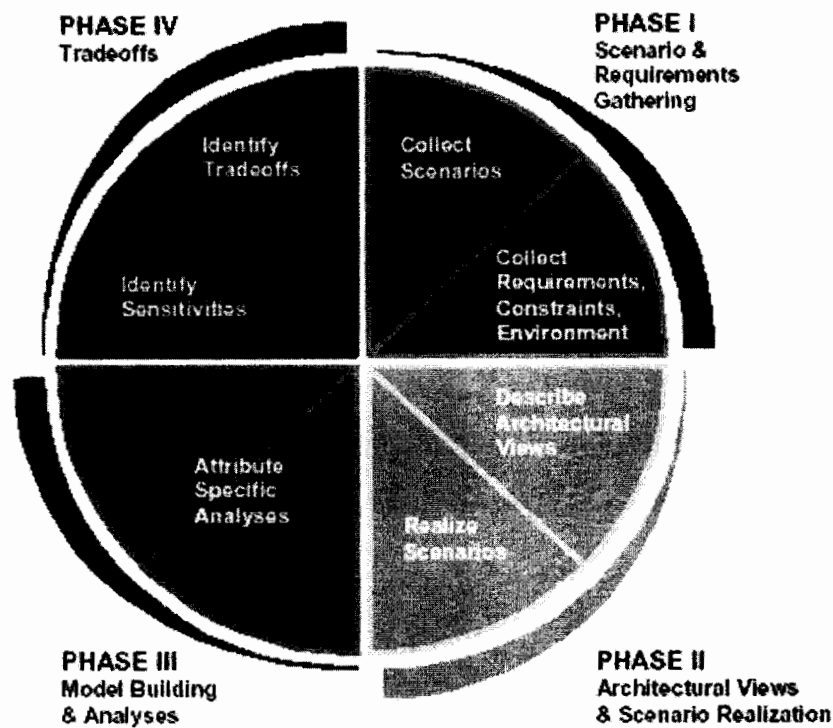


Figure 1.5: Steps of the Architecture Tradeoff Analysis Method

### Step 1 - Present the ATAM

In this step the evaluation team lead presents the ATAM to the assembled stakeholders to explain the process that everyone will be following, allows time to answer questions, and sets the context and expectations for the remainder of the activities. It is important that everyone knows what information will be collected, how it will be scrutinized, and to whom it will be reported. In particular, the presentation will describe

- ATAM steps in brief
- Techniques that will be used for elicitation and analysis: utility tree generation, architectural approach-based elicitation/analysis, and scenario brainstorming/mapping.
- Outputs from the evaluation: the scenarios elicited and prioritized, the questions used to understand/evaluate the architecture, a “utility tree”, describing and prioritizing the driving architectural requirements, set of identified architectural approaches and styles, set of risks and non-risks discovered, set of sensitivity points and tradeoffs discovered.

### **Step 2 - Present Business Drivers**

The system to be evaluated needs to be understood by all participants in the evaluation. In this step the project manager presents a system overview from a business perspective. The system itself must be presented, initially at a high level of abstraction, typically describing its

- most important functional requirements
- technical, managerial, economic, or political constraints
- business goals and context
- major stakeholders
- architectural drivers (major quality attribute goals that shape the architecture)

### **Step 3 - Present Architecture**

The architecture will be presented by the lead architect (or architecture team) at an appropriate level of detail. What is an appropriate level? This depends on several factors: how much information has been decided upon and documented? How much time is available? How much risk the system faces? This is an important step, as the amount of architectural information available and documented will directly affect the analysis that is possible and the quality of this analysis. Frequently the evaluation team will need to specify additional architectural information that is required to be collected and documented before a more substantial analysis is possible.

In this presentation the architecture should cover:

- Technical constraints such as an OS, hardware, or middleware prescribed for use
- Other systems with which the system must interact
- Architectural approaches used to meet quality attribute requirements.
- At this time the evaluation team begins its initial probing of architectural approaches.

#### **Step 4 - Identify Architectural Approaches**

The ATAM focuses on analyzing architecture by understanding its architectural approaches. In this step they are identified by the architect, and captured by the analysis team, but are not analyzed. We concentrate on identifying architectural approaches and architectural styles because these represent the architecture's means of addressing the highest priority quality attributes; that is, the means of ensuring that the critical requirements are met in a predictable. These architectural approaches define the important structures of the system and describe the ways in which the system can grow, respond to changes, withstand attacks, integrate with other systems, and so forth.

#### **Step 5 - Generate Quality Attribute Utility Tree**

In this step the evaluation team works with the architecture team, manager, and customer representatives to identify, prioritize, and refine the system's most important quality attribute goals, this is a crucial step in that it guides the remainder of the analysis. Analysis, even at the level of software architecture, is not inherently bound in scope. We need a means of focusing the attention of all the stakeholders on the aspects of the architecture that are most critical to the system's success. We do this by building a utility tree. The output of the utility tree generation process is a prioritization of specific quality attribute requirements, realized as scenarios. This prioritized list provides a guide for the remainder of the ATAM. It tells the ATAM team where to spend its limited time, and in particular where to probe for architectural approaches and their consequent risks, sensitivity points, and tradeoffs. Additionally, the utility tree serves to concretize the quality attribute requirements, forcing the evaluation team and the customer to define their "Utility" requirements precisely.

**Step 6 - Analyze Architectural Approaches**

Once the scope of the evaluation has been set by the utility tree elicitation, the evaluation team can then probe for the architectural approaches that realize the important quality attributes. This is done with an eye to documenting these architectural decisions and identifying their risks, sensitivity points, and tradeoffs.

**Step 7 - Brainstorm and Prioritize Scenarios**

Scenarios are the motor that drives the testing phase of the ATAM. Generating a set of scenarios has proven to be a great facilitator of discussion and brainstorming, when greater numbers of stakeholders are gathered to participate in the ATAM

**Step 8 - Analyze Architectural Approaches**

After the scenarios have been collected and so analyzed, the architect then begins the process of mapping the highest ranked scenarios onto whatever architectural descriptions have been presented. Ideally this activity will be dominated by the architect's mapping of scenarios onto previously discussed architectural approaches. In fact the whole point of the hiatus between the two phases is to ensure that this is the case. If this is not the case then either the architect has no approach- or style-based (and hence no architecture-wide) solution for the stimulus that the scenario represents, or the approach exists but was not revealed by any activities up until this point.

**Step 9 - Present Results**

Finally, the collected information from the ATAM needs to be summarized and presented back to the stakeholders. This presentation typically takes the form of a verbal report accompanied by slides but might, in addition, be accompanied by a more complete written report delivered subsequent to the ATAM. In this presentation we recapitulate the steps of the ATAM and all the information collected in the steps of the method including: the business context, driving requirements, constraints, and the architecture. Most important, however, is the set of ATAM outputs:

- the architectural approaches/styles documented
- the set of scenarios and their prioritization

- the set of attribute-based questions
- the utility tree
- the risks discovered
- the non-risks documented
- the sensitivity points and tradeoff points found

### **ATAM Strengths and Weaknesses**

According to Mugurel et al [21], the strengths of the ATAM are:

- Stakeholders understand more clearly the architecture.
- Improved software architecture documentation. In some cases the architecture documentation must be recreated.
- Enhanced communication among the stakeholders.

The remarks of ATAM are:

- Requires detailed technical knowledge
- Doesn't look easy to me.

The problem with scenario based approach is that it will have scenarios haven't considered. What do we do about these? Will this in fact create risk by itself? Consider a software team done a good software architecture evaluation process, and it is perfect. Two months down the track, a scenario became an important one that hadn't been considered before, what happens next? Same old story, re-do the architecture again, maybe. It can be argued that evaluation process is to minimize the risk, event though the risk is always there. However, the point is that the fundamental problem of scenario based evaluation method is that it will have scenario haven't considered.

ATAM seems shifting the focus of analysis towards estimating risks and uncertainty associated with the system's requirement, architectural decisions and strategies a great step forward.

The ATAM is general, so that it can apply to all application system. At the same time, because it is general, we found that event it is not too difficult to apply, but some information might be missing in some problem domain.

### 1.6.4 The Goal Question Metric Approach

The Goal Question Metric (GQM) approach is based upon the assumption that for an organization to measure in a purposeful way it must first specify the goals for itself and its projects, then it must trace those goals to the data that are intended to define those goals operationally, and finally provide a framework for interpreting the data with respect to the stated goals. Thus it is important to make clear, at least in general terms, what informational needs the organization has, so that these needs for information can be quantified whenever possible, and the quantified information can be analyzed whether or not the goals are achieved. The Conceptual Model for GQM approach is presented in Figure 1.6

The approach was originally defined for evaluating defects for a set of projects in the NASA Goddard Space Flight Center environment. The application involved a set of case study experiments [22] and was expanded to include various types of experimental approaches [23]. Although the approach was originally used to define and evaluate goals for a particular project in a particular environment, its use has been expanded to a larger context. It is used as the goal setting step in an evolutionary quality improvement paradigm tailored for a software development organization, the Quality Improvement Paradigm, within an organizational framework, the Experience Factory, dedicated to build software competencies and supplying them to projects. The result of the application of the Goal Question Metric approach application is the specification of a measurement system targeting a particular set of issues and a set of rules for the interpretation of the measurement data. The resulting measurement model has three levels:

**1. Conceptual level (GOAL):** A goal is defined for an object, for a variety of reasons, with respect to various models of quality, from various points of view, relative to a particular environment. Objects of measurement are

- Products: Artifacts, deliverables and documents that are produced during the system life cycle; e.g., specifications, designs, programs, test suites.
  - Processes: Software related activities normally associated with time; e.g., specifying, designing, testing, interviewing.
  - Resources: Items used by processes in order to produce their outputs; e.g., personnel, hardware, software, office space.
-

**2. Operational level (QUESTION):** A set of questions is used to characterize the way the assessment/achievement of a specific goal is going to be performed based on some characterizing model. Questions try to characterize the object of measurement (product, process, resource) with respect to a selected quality issue and to determine its quality from the selected viewpoint.

**3. Quantitative level (METRIC):** A set of data is associated with every question in order to answer it in a quantitative way. The data can be

**Objective:** If they depend only on the object that is being measured and not on the viewpoint from which they are taken; e.g., number of versions of a document, staff hours spent on a task, size of a program

**Subjective:** If they depend on both the object that is being measured and the viewpoint from which they are taken; e.g., readability of a text, level of user satisfaction.

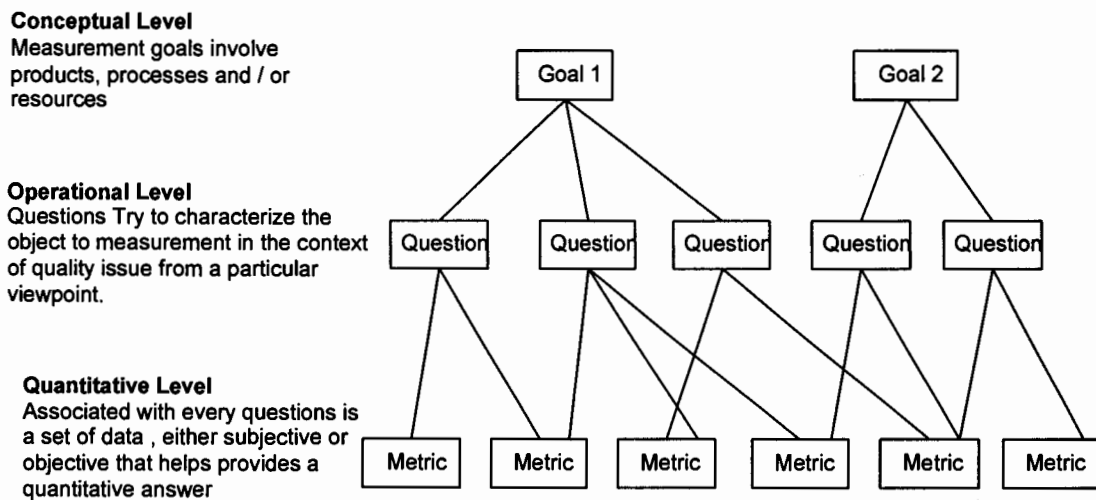


Figure 1.6: The GQM Paradigm



## **Chapter 2**

# **PROBLEM DEFINITION**

## 2 Problem Definition

The study addresses an important problem identified in the Architecture Tradeoff Analysis Method (ATAM). It attempts to specify the problem Statement, specific object and approached used for solving the problem. Architecture Tradeoff Analysis Method is defined in order to assess the effect of architecture decisions in the light of Quality attribute requirements. ATAM does not give any path to approach a well-defined structure of quality attributes. We shall try to answer the questions given below:

- Defining a quality attribute utility tree proved to be difficult, time consuming and tedious task.
- Lack of clear guidelines of how to build up such a tree hindered and slowed down the discussion.

### 2.1 Hypothesis

The primary measure of success of a software system is the degree to which it meets the purpose for which it was intended. It is said that Software Architecture is a key business asset for an organization and architecture is key practice for that organization. This is because architectures are complex and involved many design tradeoffs (ATAM). There are a number of inherent difficulties in this process. Architectural analysis involves the participation of many stakeholders that may include developers, customers, users, architects and project Managers. The output of utility tree generation process is a long list of scenarios that must realize the quality attributes. It has been clearly stated in research area that output from utility tree serves as a foundation for the remaining part of Architecture Tradeoff Analysis Method. Therefore, lack of guidelines for generating a utility tree could result in the ambiguous and uncertain result. It is important to have a very sound framework (a long list of scenarios along with response measures) to arrive at right architectural decisions in the light of quality attributes. The quality of software architecture may be more realistic if the architecture analysis is supported with a well-defined software measurement approach.

Software metrics may help to draw the real picture of software architecture by defining functional and non-functional requirements in terms of scenarios. High involvement of stakeholders in an architectural analysis, which is explicitly based on, standardized procedures and approaches may definitely results in getting the right set of architectural decisions.

## 2.2 Problem Scope

Architecture Tradeoff Analysis Method (ATAM) is considered to be a well-known architectural evaluation approach. When evaluating an architecture using ATAM, The goal is to understand the consequences decisions with respect to the quality attributes requirements of the systems. The ATAM is intended for analysis of architecture with respect to its quality attributes.

Generation of quality attribute utility Tree is an important step in ATAM as it helps to guide the architecture evaluation process. Utility tree provides a top-down mechanism for directly and efficiently translating the business drivers of a system into concrete quality attribute scenarios. We identified a problem in the utility tree structure. In classic ATAM, the utility tree has been defined upon the concept of quality attribute characterization. With quality attribute characterization, every quality attribute is realized in terms of external stimuli, architectural decision and response measure. Lack of proper guidelines resulted in identifying the right set of external stimuli, architectural decisions and response measures. Architectural evaluation team experienced difficulty in deriving scenarios which is the ultimate out of the utility tree. If quality attribute characterization is the principle logic to establish the framework for utility tree generation, then we propose a quality model. The quality model provides us with the set of guidelines for the achievement of detailed and unambiguous scenarios. Every scenario is supported with its response measure. These measurements help to realize quality attributes.

In ATAM, the evaluation approach is supported by the scenarios based analysis. Each quality characteristics is refined into its corresponding sub- characteristics and then a scenario gives the definition and measure for those specific quality sub-characteristics. This measurement approach focuses to measure specific quality characteristics from one

possible perspective. We also observed that metric assignment to the quality characteristics is not based on a well define approach.

### 2.3 Problem Statement

“ATAM uses one level of quality characteristics. However there is not any specific guideline to define the utility tree. An expression of quality view and the reason for one level of refinement is ambiguous. Attributes defined in the utility tree are measured in terms of stimuli, parameters and responses. After having a utility tree, we see that there is a lack of coordination among quality characteristics; their refined attributes and resulting scenarios which also attempt to specify measures to attributes. According to AVG case study, it is found difficult to come up with a quality attribute utility tree. Furthermore it preparation is time consuming and tedious. Without clear guidelines, there occur obstacles in building up such a tree. A quality model is proposed that takes the support of software metrics to clearly identify the architectural decisions and their consequences in the light of quality attribute requirements.”

This problem statement provides a base line for that research. The proposed quality model will provide a comprehensive solution to remedy of the mentioned problem.

We used Goal Question Metric (GQM) approach to specify the measures for quality attributes. The measures can be subjective as well as objective. We do not get the measures from ISO quality model because in that case we will skip the exploration of scenarios. We prefer to use GQM approach as it would give us the measures that could serve as the response measures for the new explored scenarios. This way, we are attempting to strengthen the quality attribute characterization process which serves as the basic concept of utility tree creation. Metrics help to arrive at the right architectural decisions because metrics are derived from the questions raised by the stakeholders. GQM approach requires the involvement of stakeholders in architecture evaluation process. Thus the model sticks to the basic philosophy of ATAM and helps to carry out architecture analysis while keeping participant’s interest in focus.

## **2.4 Significance of the Research**

This research attempts to offer an improvement to an existing architecture evaluation model ATAM. This model would make it more realistic to arrive at right architectural decisions. This model is based on a well-defined software quality standard ISO 9126. Therefore it paves the way to carryout architecture analysis that assists in getting a quality software product. The standard model encourages software professionals to follow a standard quality model, which is clearly defined and elaborated. Architecture evaluation carried out by one team would be fully understandable to another team if both teams follow MoQaMo model. In research area, we do not see any such model that offers an improved modification to a famous model ATAM.

## **Chapter 3**

# **PROPOSED SOLUTION**

### 3 Proposed Solution

Proposed Metric-Oriented Quality Model (MoQaMo) is based on the ISO 9126-1 standard. The model is designed with a view to support evaluation of architecture using ATAM. The model guides in the process of architecture evaluation. We do not attempt to replace the architecture tradeoff analysis method (ATAM) but we proposed a modification to this method. For any evaluation process, it is necessary to have well-defined evaluation criteria. Evaluation criteria helps to identify what characteristics of the target (architecture to be evaluated) are of interest for evaluation purposes [24]. One of the steps in ATAM is the generation of the quality attribute utility tree. This utility tree serves as evaluation criteria for discovering architectural risks associated with architectural approaches. The utility tree in ATAM is elicited by focusing on the business drivers. Business drivers represent the goals that have to be achieved. In practice, we see that major quality characteristics like performance, maintainability, portability, etc. are captured as business drivers and put in the utility tree to define evaluation criteria. We also observe that these major characteristics are refined into sub-characteristics without any guiding principle. The definition of quality attributes is only supported through scenarios only, which are gathered from stakeholders. The quality attributes get different definitions each time they are applied within ATAM. Due to varying interpretations of these quality attribute requirements, the usefulness of ATAM becomes limited within the software industry. It is seen that the same quality attribute names vary from evaluation to evaluation. One organization's "maintainability" is another organization's "changeability". Reliability and availability are often interchanged [7]. It proved to be difficult, time-consuming and disappointing to come up with a utility tree. A lack of clear and concrete guidelines hindered the generation of a quality attribute utility tree [25]. As a result, communication among stakeholders, which is a crucial activity of ATAM, was badly affected.

MoQaMo Model is proposed with a view to bring an improved modification in ATAM. We attempt to replace the utility tree by the proposed model. MoQaMo is based on the ISO 9126 standard. This model incorporates all the major six characteristics and their respective sub-characteristics. These quality characteristics define the scope of evaluation criteria,

which support the architecture evaluation team in an organized and systematic manner. The quality attributes are taken from the ISO 9126-1 quality model because these describe the high-level characteristics in a more relevant way and guide to achieve the quality of a software product within a well-defined scope. High-level quality characteristics can be represented as architectural drivers. High-level quality characteristics are refined into their corresponding sub-characteristics. Sub-characteristics for each major quality characteristic explain its meaning and purpose. Measures are derived for these sub-characteristics. It is clear that the purpose of ATAM is to evaluate the consequences of architectural decisions in the light of architectural drivers. We consider these architectural drivers as architectural goals. If the quality factor which is serving as architectural or business driver is not found in the ISO quality model, then we recommend utilizing Goal Question Metric approach. GQM methodology treats the quality factor as a goal and helps us to produce relevant scenarios along with the response measures. We prefer to use the measures for these quality characteristics obtained from the GQM method rather than ISO quality model. ISO quality model gives us the opportunity to derive measures for internal quality characteristics. But doing so would avoid us in getting scenarios which is very important output of utility tree process. In both the cases whether the required quality factor is found in ISO 9126 model or not, we prefer GQM method in getting to the scenarios.

7A-4997 We support our proposition by giving a few justifications for employing the ISO quality model and GQM method:

## **Justifications**

### **ISO 9126 Quality Model:**

- 1- ISO 9126 is an international standard for the evaluation of software.
- 2- ISO 9126 Part one, referred to as ISO 9126-1 is an extension of previous work done by McCall (1977), Boehm (1978), FURPS and others in defining a set of software quality characteristics.



- 3- The fundamental objective of this standard is to address some of the well known human biases that can adversely affect the delivery and perception of a software development project.
- 4- In ATAM, we evaluate software architecture on the basis of quality characteristics. ISO model serves to be the best tool because it gives a complete quality model.
- 5- ISO 9126 model is product-oriented, focusing the product external quality characteristics that must be accomplished when the product is in operation. However, the internal characteristics, which influence the external ones are taken into account. These internal characteristics arise during the development process and can be used to evaluate the architecture, which is a sub-product of the development process.

### **Goal Question Metric Approach**

- 1- Goal Question Metric approach is used for the transformation of quality factors into the scenarios.
- 2- With GQM, users are not restricted to the predefined meaning of quality characteristics specified in ISO 9126 quality model. Rather, they can generate scenarios from GQM approach.
- 3- Stakeholders can derive meaning to the quality attributes according to their requirements and context.
- 4- If there does not exist the quality attribute in the pool of quality attributes provided in ISO 9126 quality model, then the stakeholders can take advantage of the GQM approach. Stakeholders may pick quality attributes from ISO quality model according to the business driver and apply the GQM approach to describe it in detail.

To equip our model with the ability to define appropriate metrics, the Goal Question Metric (GQM) approach [26] has been applied. This approach is based on the assumption that organization must specify the goals for itself and its projects, then it must trace those goals

operationally and finally provide a framework for interpreting the data with respect to the stated goals.

We identify business goals that need to be achieved in the desired software product. These goals usually refer to functional and non-functional requirements of software. The goals are defined under the light of ISO quality model. That is, high level quality characteristics help us to define and document the goals in a clear context. Once we have a set of well-defined goals for software. We then, start exploration of scenarios. Stakeholders of the software system are prompted to raise questions in order to achieve the goals. Questions raised are answerable and metrics are derived to answer questions. The answers take the form of subject and objective metrics. The more the relevant questions are raised, the more the chance are to achieve goals. Scenarios are elicited to explore the system. These scenarios help to identify those questions which pave the way for achievement of goals. The goals are defined by focusing on high-level quality characteristics and their refined sub-characteristics. Thus quality of the product is achieved by identifying goals on the basis of software quality dimensions. Metrics are devised to answer these questions. Thus metrics are assigned indirectly to internal quality attributes. Questions are always raised by tracing sub-characteristics. MoQaMo model develops close coordination among high level quality characteristics, sub-characteristics, questions, scenarios and metrics. The sub-characteristics serve as the goals if the quality factors exist in ISO 9126 quality model. If they do not exist then high level characteristics will be become the goals and would results in refined sub-characteristics as well as scenarios. This coordination highly supports the architecture evaluation process. MoQaMo model involves the participation of stakeholders in this process and provides the opportunity to take correct architectural decisions while keeping focus on stakeholders' interest towards the system.

### **3.1 Conceptual Model with respect to ATAM**

From the conceptual model as shown in Fig. 3.1 we see that business drivers are treated as goals. The architecture evaluation process is exercised to attain these goals. The high-level quality attributes follow the structure provided by the ISO 9126 standard.

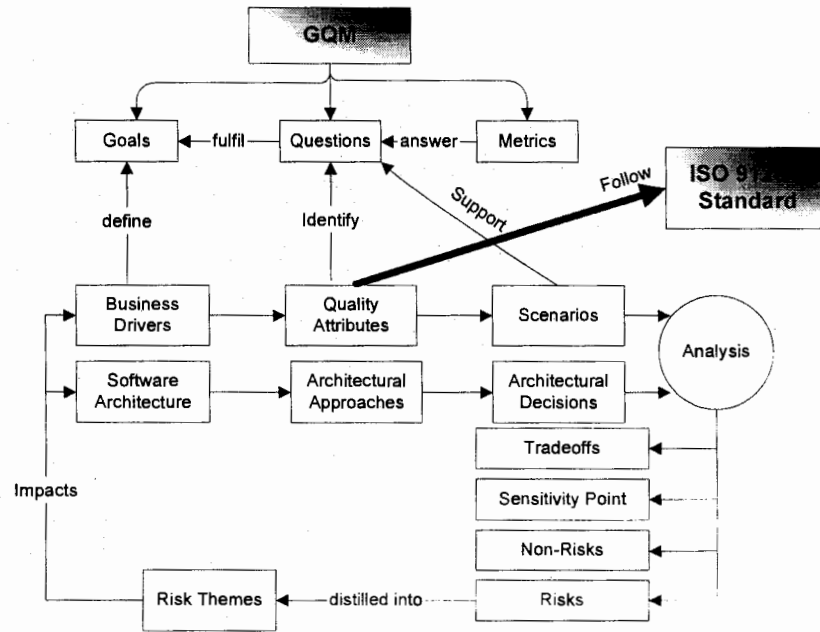


Figure 3.1: MoQaMo Conceptual Model with respect to ATAM

The well-defined structure of the ISO 9126-1 quality model motivates to use quality characteristics in the MoQaMo model. These quality characteristics in ISO 9126 are structured with a view to ensure quality of software products. The sub-characteristics (internal quality attributes) help to identify quantifiable questions, which are asked in order to fulfill goals (high-level quality characteristics). The application of the MoQaMo Model appreciates the involvement of stakeholders in the architecture evaluation process. Questions are raised by the stakeholders in order to observe whether architectural approaches address quality attributes in question or not. In ATAM, we see that quality attributes are defined and covered up by the elicitation of scenarios, that is, scenarios define the context of quality attributes according to the evaluating system. These scenarios pave the way for quantifiable questions to be asked according to relevant sub-characteristics. Once all the relevant questions are defined for a particular goal, metrics have to be specified in order to answer the quantifiable questions. Each question can be answered with a single or multiple metrics. After having a collection of metrics, we can start analyzing the architecture with the candidate architectural approaches and can assess the consequences of architectural decisions. This particular suite of metrics helps to take justifiable architectural decisions. The analysis which is a core process in ATAM gets improved due to high

involvement of stakeholders, a well-defined evaluation criteria and availability of appropriate set of metrics. The same metrics can be used in order to answer different questions for the same goal.

### 3.2 Conceptual Model with respect to GQM approach

We also propose another conceptual model [Fig. 3.2] for MoQaMo with respect to the GQM approach that shows its different components and the interaction among them. Quality characteristics are refined into sub-characteristics. The set of Sub-characteristics provide the field where relevant scenarios can be defined. Scenarios and sub-characteristics help to define scope of questions which are to be raised by the system's stakeholders. Stakeholders play an important role in the execution of ATAM. Stakeholders use to ask questions about system working from the dimensions they expect the system to work. The end-user of software system is always interested to have a software which is easy to learn and easy to use. So he would be raising questions by taking "usability" quality characteristic into consideration. The questions are always asked with a view to achieve a business driver and architecture driver (goal of the system). Stakeholders are expected to answer question by providing answers in the form of subjective and objective metrics. If system's architecture is designed in such a manner where it fulfills business goals using MoQaMo model, then we claim to say that stakeholders would be satisfied with system's operation.

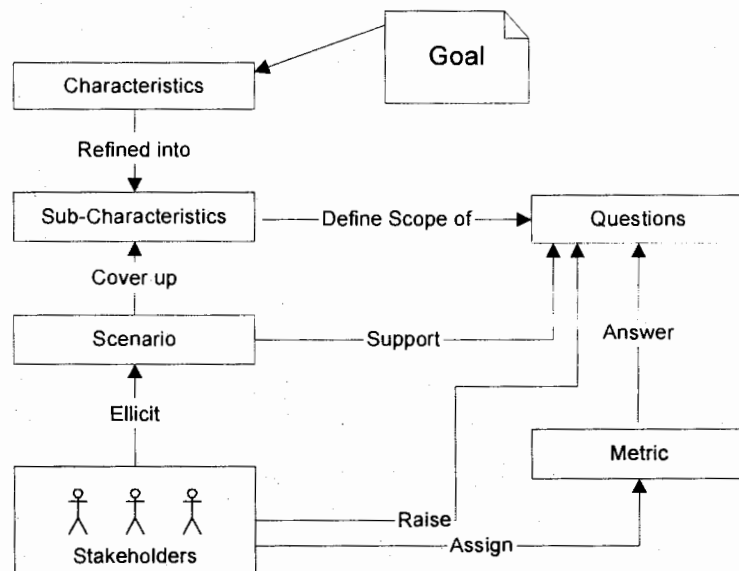


Figure 3.2: MoQaMo Conceptual Model with respect to GQM approach

The process of setting goals is critical to the successful application of the MoQaMo model. A goal should be expressible with four dimensions; (i) Issue, (ii) Objects, (iii) Purpose, (iv) Viewpoints. The Measurement Goal Template (MGT) is helpful in the operationalization of quality goals. It also addresses the characteristics that are to be included in an evaluation. MGT [Table 3.1] makes the MoQaMo model more flexible by adjusting it according to the context of a particular software project [27].

No	Dimensions	Definition
1	Issue (Quality Focus)	Which characteristics of the object are taken into consideration?
2	Object	What is the artifact to be evaluated?
3	Purpose	Why is the object analyzed?
4	Viewpoint	Who will evaluate?

Table 3.1: Measurement Goal Template (MGT)

In order to provide an example of the application of the MoQaMo model, let's assume we want to improve the performance of the altitude monitoring device of an airplane. The resulting goal will specify the purpose (improve), an object (altitude monitoring device), the viewpoint (aeronautical engineer) and a quality issue (efficiency). The MoQaMo Model is presented in Table 3.2.

Characteristic/Goal	Improve the efficiency of altitude monitoring device from an aeronautical engineer's viewpoint.	
Sub-Characteristics	Time Behavior	
<b>Time Behavior</b>		
S-1	<i>The device should return information within specified time.</i>	
Q-1	How much time does the device take to send information to the screen?	
M-1	0.25 ms	
Q-2	What is the response time of the device in calculating the altitude?	
M-2	0.5 ms	
Q-3	What is the frequency of information production?	
M-3a	20 times / minutes	
M-3b	Once in every 3 sec.	

Table 3.2: The MoQaMo Model for altitude monitoring device

First we identify the business driver and that is to increase the efficiency of altitude monitoring device in an airplane. This business driver is translated into a quality characteristic of the system. Here we have an "Efficiency" quality characteristic with its refined sub-characteristic "time behavior" and "resource behavior". We only take "time behavior" sub-characteristic only and ask stakeholders of the system to raise relevant questions. The questions are raised under the scope of a particular scenario. The questions are always asked in the context of a well defined scenario. Stakeholders are encouraged to raise only those questions which can help in achieving a goal. Quality sub-characteristics (in our case time behavior) restrict the definition of scenarios. Scenarios are selected, defined, and analyzed under the scope of internal quality attributes.

## **Chapter 4**

# **APPLICATION OF MOQAMO TO A CASE STUDY**

## 4 Application of MoQaMo Model to a Case Study

We have selected the case study for which the Software Engineering Institute (SEI) has carried out architecture evaluation using ATAM. The details of the case study can be found at the SEI's technical report [1]. We have applied the MoQaMo model to a Battlefield Control System (BCS). This system is designed with an aim to support army battalions in controlling the movement, strategy and operation of troops in real-time Battlefield. We focus on two major quality requirements (Efficiency and Reliability) in the system. We carry out evaluation by selecting different architectural approaches. Questions with metrics as their answers were identified from the stakeholders' perspective.

### 4.1 Driving Architectural Requirements

A commander commands a set of soldiers and their equipments, including different kinds of weapons and sensors. External systems need to be interfaced with the BCS system in order to capture its commands and intelligence information. The commander communicates with all the soldiers on a real-time basis. The system is considered to be working if there is a working commander along with a number of soldiers. The failure of the system depends upon the life of the commander. The commander fights the battle according to mission plan and utilizes its available resources during the battle. A radio modem with 9600-baud speed supports the communication between the commander and the soldiers. There is a need for extreme level of robustness and a number of performance goals should be considered. The system is also subject to frequent modifications. The evaluation of this system requires careful consideration of different architectural approaches, quality requirements of the system, stakeholders' expectations in the form of documented questions and scenarios.

### 4.2 High-Level Architectural Views

We give a few high-level architectural views of the BCS to flesh out our understanding about its working structure. We are not going to present detailed architectural documentation of the system. From hardware view shown in Figure 4.1, we observe that the commander is central to the system. In fact, the commander node acts as a server and fulfils the clients (soldiers) requests. Inter-node communication between the clients and the



server is only through encrypted messages sent via a radio modem. The soldier with shaded box represents a backup soldier. A Backup soldier takes the responsibility of the commander in case of death of commander. Thus a backup soldier needs to be updated frequently with all the information that commander possess during the battle, so that a backup soldier should be smart enough to replace the position of commander.

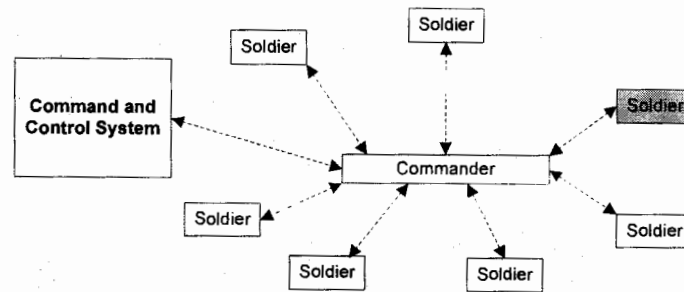


Figure 4.1: System view of BCS

The module decomposition view shown in Figure 4.2 identifies major components of the system. This view may include many more modules but for better understanding, we have mentioned only a few. The module decomposition view consists of Decision Support System (DSS), Communication Manager (Comm. Mgr) and Battle Controller (B.Controller).

**Decision Support System:** This part of the system facilitates the provision of intelligent information to other components like Battle Controller (B.C) and Communication Manager (Com.Mgr). The commander uses intelligent information during the battle. For example, the commander should know which weapon to use at which time and which soldier need to be moved forward and which soldiers need to be kept in defense. Decision support system has a knowledge base which contains all the mission plans, weapon information, soldiers' profiles and other relevant system information.

**Communication Manager:** This is responsible for streamlining and floating all information among different parts of the system. Radio modem is the device used to transfer information within soldiers and commander. Com.Mgr encrypts, decrypts, moves, formats and distributes information within the system. The battle requires a quick transfer of

messages (data with different size and structure) among software system. Thus to achieve efficiency quality perspective of the system, it is necessary to understand the complete working structure of communication manager.

**Battle Controller:** Battle Controller (B.Controller) helps to provide coordination among all the components of the system. The commander needs to move soldiers within battlefield according to mission plan and battle strategy. B.Controller also facilitates in managing the switching position of backup soldiers to commander position.

Figure 4.3 shows a module layered view. In this view, the BCS application layer is using the services of its lower layers. The BCS Application Programming Interface (API) layer is developed over .Net framework and provides a programmable interface to the layer above it.

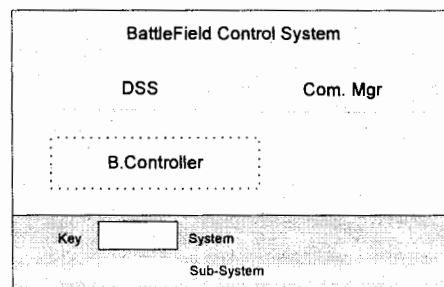


Figure 4.2: Module decomposition view

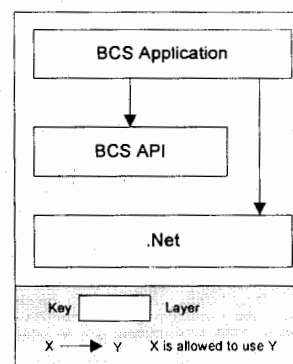


Figure 4.3: Module Layered View

Architectural approaches are considered with respect to modifiability, availability, and performance. To achieve availability, a backup commander approach was described. Availability and performance of the system are found to be a high priority of the stakeholders. The MoQaMo model attempts to organize the architectural evaluation process by eliciting information for desired quality characteristics through high involvement of stakeholders. This organized and efficient structure helps an evaluation team to realize those architectural approaches and architectural decisions that cater for the desired quality characteristics (quality goals). The MoQaMo model serves as a guideline and well-defined structure to evaluate software architectures using ATAM.

### 4.3 Architectural Evaluation

From the case study, we observed that the stakeholders were highly interested in system availability, modifiability and performance. In order to achieve availability, an architectural approach was needed to be employed which could sustain the existence of the commander in the battlefield. We will only consider availability and performance of the system in evaluating architecture. We observe that system availability is primarily affected by the failure rate of the commander, the repair rate of the commander (the time required for the backup to become commander) and the repair rate of the backup (the time required for the soldier to become a backup). The shaded soldier node indicates the backup. According to the existing architecture, availability of the system is assured by provoking the backup soldier node to mirror the commander's state through acknowledged communication (state messages) with commander. Upon failure of the commander, the backup takes over as the new commander.

To achieve high availability (a high level of readiness), an alternative architecture proposed that the multiple soldier nodes could be put to monitor the commander-to-backup communication. The backup soldiers could be acknowledged backups (requesting resends of missed packets) or could be passive backups (silent receiver packets) or a mixture of these concepts. In the case where packets are not acknowledged, the state of the backup database would increasingly drift from that of the commander. If one of these backups is called upon to become the commander, it would need to engage in some negotiation (with the external systems and/or the other Soldier nodes) to complete its database.

It is clear to say that the availability of the system increases as the number of backups is increased, because the system can survive multiple failures of individual nodes without failing its mission.

## 4.4 Comparison and Results

### Classic ATAM:

In classic ATAM, information on the architectural approaches with respect to modifiability, availability and performance scenarios was elicited. The system was loosely organized around the notion of clients and servers. This dictated both the hardware architecture and the process architecture and affected the system's performance characteristics. In addition to this style

- For availability, a backup commander approach was described
- For modifiability, standard subsystem organizational patterns were described
- For performance, and an independent communicating components approach was described.

The stakeholders in this ATAM were most interested in modifiability and performance. Upon probing, however, they admitted that availability was also of great importance to them. Based upon stated concerns and elicitation, a utility tree was created. As part of elicitation process they ensured that each of the scenarios in the utility tree had a specific stimulus and response associated with it.

Evaluation team requested specific additional architectural information to address the gaps in the documentation produced by the contractor. These requests were in the form of questions such as:

What is the structure of the message-handling software (i.e., how is the functionality is broken down in terms of modules, functions, APIs, layers, etc.)?

What facilities exist in the software architecture (if any) for self-testing and monitoring of software components?

- What facilities exist in the software architecture (if any) for redundancy, liveness monitoring, failover, and how data consistency is maintained (so that one component can take over from another and be sure that it is in a consistent state with the failed component)?
- What is the process and/or task view of the system, including mapping of these processes/tasks to hardware and the communication mechanisms between them?
- What functional dependencies exist among the software components (often called a "uses" view)?

- What data is kept in the database (which was mentioned by one of your stakeholders), how big is it, how much does it change, and who reads/writes it?
- What is the anticipated frequency and volume of data being transmitted among the system components?

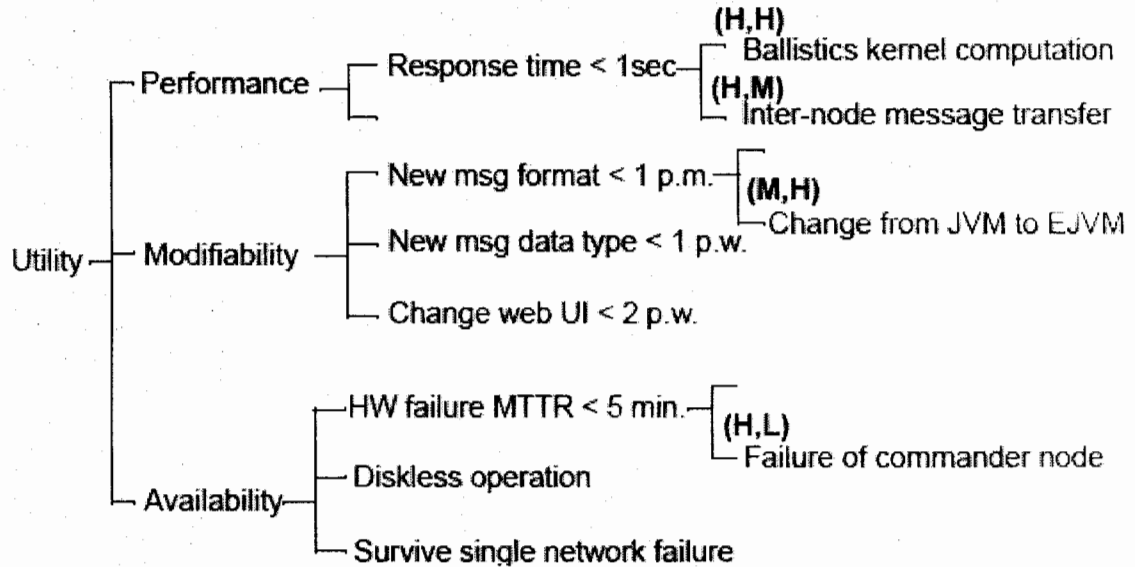


Figure 4.4 ATAM Utility tree

The utility tree shown in Figure 4.4 above contains utility as the root node. This is an expression of the overall “goodness” of the system. Typically the quality attributes of performance, modifiability, security, and availability are the high-level nodes immediately under utility, although different stakeholder groups may add their own quality attributes or may use different names for the same ideas (for example, some stakeholders prefer to speak of maintainability). Under each of these quality factors are specific sub-factors. For example, performance is broken down into “data latency” and “transaction throughput”. This is a step toward refining the attribute goals to be concrete enough for prioritization. Notice how these sub-factors are related to the attribute characterizations. Latency and throughput are two of the types of response measures noted in the attribute characterization. Data latency is then further refined into “Minimize storage latency on customer database” and “Deliver video in real-time.” Throughput might be refined into “Maximize average throughput to the authentication server.”

Further work on these scenarios would, in fact, make these architectural response goals even more specific, e.g., “Storage latency on customer database no more than 10 ms. on average.”

The output of utility tree generation provides a prioritized list of scenarios that serves as a plan for the remainder of the ATAM. It tells the ATAM team where to spend its (relatively limited) time, and in particular where to probe for architectural approaches and risks. The utility tree guides the evaluators to look at the architectural approaches involved with satisfying the high priority scenarios at the leaves of the utility tree. Additionally, the utility tree serves to concretize the quality attribute requirements, forcing the evaluation team and the customer to define their “XYZ-ility” requirements very precisely.

### **Our Proposition**

On the basis of the stated requirements, we attempt to generate utility tree by following a set of guidelines. We employ ISO 9126 quality model and GQM approach in order to achieve quality attribute characterization.

### **Guidelines:**

- 1- Search for the quality attribute in the ISO 9126 framework.
- 2- Pick up the sub-characteristics for that quality attribute.
- 3- Declare each sub-characteristic as a goal according to the GQM approach.
- 4- Raise a question against each sub-characteristic such that its answer can realize the achievement of that goal.
- 5- Provide answer to the question in the form of well-defined measures. Measures should serve as response measures in the characterization of sub-characteristics.
- 6- In case, quality attribute, that needs to be characterized, is not found in ISO 9126 framework then declare quality factor as a goal and raise questions in order to realize the goal.
- 7- Organize the quality factor with its elicited information in the form of a hierarchy. The leaf node should be scenario.

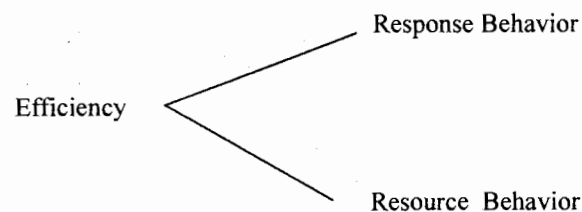
Using the BCS case study, we observe that Performance and Availability take high importance in the architecture evaluation. So we follow the proposed guidelines in order to generate ATAM utility tree.

### **Performance Quality Attribute**

In order to characterize Performance Quality attribute, we refer to ISO 9126 quality model. There we find Efficiency quality attribute with its refined sub-characteristics i.e. Response Behavior and Resource Behavior.

We declare response behavior as a goal and we will raise the questions against this quality attribute. Derived metrics would justify the answers to the raised questions. From the sub-characteristic in the form of goal, its relevant questions and metrics would help us to get the set of well defined scenarios.

#### From ISO 9126 Quality Model



Response Behavior represents the architectural stimulus. We need to accommodate the change in architecture according to this stimulus. So achievement of sub-characteristic response behavior motivates us to declare it as a goal.

**Goal:** Improve the Response behavior of the system from evaluation team's viewpoint.

**Question:** What is the communication speed between the commander and the soldiers?

**Answer:** 9600 baud (9600 bits/sec) or 9.6 kbits / sec

**Question:** How much time is required to download mission plans?

**Answer:**  $280 \text{ kbits} / 9.6 \text{ kbits/sec} = 29.17 \text{ sec}$

**Question:** How much time does it take to make updates to environmental database?

**Answer:**  $280 \text{ kbits} / 9.6 \text{ kbits/sec} = 29.17 \text{ sec}$

**Question:** How much time is required to acquire issued orders (for 24 soldiers)?

**Answer:**  $24 \text{ soldiers} * (18 \text{ kbits} / 9.6 \text{ kbits/sec}) = 45 \text{ sec}$

**Question:** What is the time needed to acquire information about the inventories (for 24 soldiers)?

**Answer:**  $24 \text{ soldiers} * (42 \text{ kbits} / 9.6 \text{ kbits/sec}) = 105.0 \text{ sec}$

**Question:** How much time is required for a soldier to become a backup (in case of 24 soldiers)?

**Answer:** It takes 216.05 sec for a soldier to become a backup

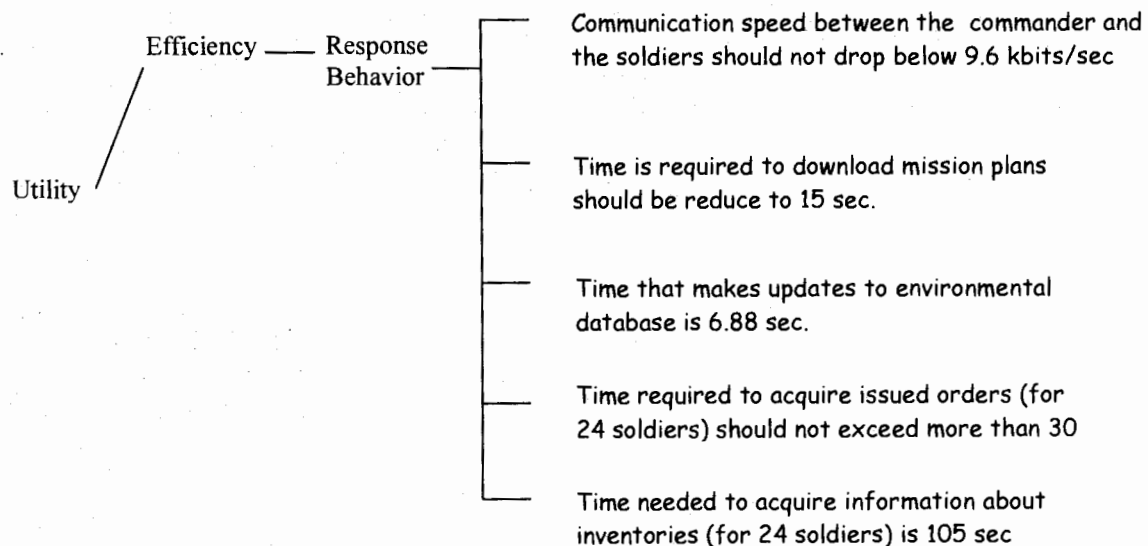


From the above discussion about efficiency quality factor's sub-characteristic Response Behavior by applying the GQM approach, we derive scenarios that would serve as the leaf nodes of the utility tree.

### Scenarios

- 1- Communication speed between the commander and the soldiers should not drop below 9.6 kbits/sec.
- 2- Time is required to download mission plans should be reduce to 15 sec.
- 3- Time that makes updates to environmental database is 6.88 sec.
- 4- Time required to acquire issued orders (for 24 soldiers) should not exceed more than 30 sec.
- 5- Time needed to acquire information about the inventories (for 24 soldiers) is 105.0 sec.
- 6- Time required for a soldier to become a backup (in case of 24 soldiers) should be minimized to 150 sec.

### Utility tree generation:



## **Chapter 5**

# **CONCLUSION**

## 5 Conclusion

Adapting the architecture in earlier steps of development is recommended but its evaluation is more important. The MoQaMo model supports the quantifiable evaluation of software architecture by giving measures to the refined quality attributes. The Goal Question Metric approach assists in defining the subjective as well as objective metrics with a high involvement of stakeholders of the system. The model does not restrict itself only to measurement specification but is flexible enough to be applied to any attributed-based architecture analysis method. The proposed model focuses on the analysis phase of the ATAM and guides all the remaining evaluation process. Another important aspect that will be explored in the near future is the definition of metrics for architectural approaches. This would make it possible to map appropriate architectural approaches to meet quality attributes in an architecture evaluation process.

The architectures of substantial software-intensive systems are large and complex. They involve many stakeholders, each of whom has their own agenda. These architectures are frequently incompletely thought out or only partially documented. A method for architecture evaluation has to consider and overcome all of these daunting challenges. MoQaMo model is formed by following a set of guidelines to generate utility tree. Well-defined utility tree lays out the foundation for evaluating architecture.

### 5.1 Benefits

We present the following benefits that were realized after the application of MoQaMo model. These are enlisted below:

- Architecture Evaluation Method (ATAM) will be based on a Quality model.
- A clear exploration of user's requirements in different contexts helps to approach right software architecture.
- Architecture Evaluation process following MoQaMo model is easily understood, manipulated and applied in different teams as well as different organizations.
- The model allows producing objective and subjective metrics.
- A huge number of scenarios can be generated for each and every quality attribute.
- On the basis of well-defined utility tree, we can approach to a detailed level of quality attribute characterization of quality characteristics on Architecture

Tradeoff Analysis Method (ATAM) following MoQaMo model gives the right set of architectural decisions and hence resulting in quality product.

- Quality of a software product can be assured at a very early stage of software development life cycle.
- New requirements are identified in the form of scenarios.
- Important quality requirements are identified and analyzed.
- The method provided the stakeholders with a chance to give a critical look at the system. It validated some architectural decisions and raised questions about others.

## **REFERENCES & BIBLIOGRAPHY**

## References &amp; Bibliography

- [1] Klein, M., Clements, P. and Kazman, R., "**ATAM: Method for Architecture Evaluation**", August 2000, TECHNICAL REPORT, CMU/SEI-2000-TR-004, ESC-TR-2000-004
- [2] K. Khosravi, Y. Gu'eh'eneuc: **On Issues with Software Quality Models**. 19th European Conference on Object-Oriented Programming SECC. 2005
- [3] J. Bosch: **Design and Use of Software Architecture**. Harlow. 2000
- [4] L. Dobrica, E. Niemela: **A Survey on Software Architecture Analysis Methods**. *IEEE Transactions On Software Engineering*. 2002
- [5] F. Losavio, L. Chirinos, N. Lévy, A. Ramdane-Cherif: **Quality Characteristics for Software Architecture**. *Journal Of Object Technology*. 2003
- [6] ISO/IEC FCD 9126-1.2: **Information Technology – Software Product Quality**. Part 1: Quality Model, draft. 1998
- [7] Bass, Len; Clements, Paul; & Kazman, Rick. **Software Architecture in Practice, Second Edition**, Boston, MA: Addison-Wesley, 2003
- [8] <http://www.sei.cmu.edu/architecture/>
- [9] [http://www.sei.cmu.edu/architecture/ata\\_eval.html](http://www.sei.cmu.edu/architecture/ata_eval.html)
- [10] Mary Shaw, **The coming-of-age of software architecture research**, International Conference on Software Engineering, Proceedings of the 23rd International Conference on Software Engineering, Toronto, Ontario, Canada , Page: 656, Year of Publication: 2001, ISBN ~ ISSN:0270-5257 , 0-7695-1050-7
- [11] James M. Bieman and Byung-Kyoo Kang. **Measuring design-level cohesion**. Number 2, pages 111-124, Feb 1998.
- [12] Barbara Kitchenham and Shari Lawrence Peeger. **Software quality: The elusive target**. *IEEE Software*, pages 12-21, 1996.
- [13] Jan Tretmans and Peter Achten. **Quality of information systems**, 2003.
- [14] Alan Gillies. **Software Quality: Theory and Management**. International Thomson Publishing, 1992
- [15] W. J. Salamon and D. R. Wallace. **Quality characteristics and metrics for reusable software** (preliminary report). Technical report, National Institute of Standards and Technology, may 1994.
- [16] Stephan H. Kan. **Metrics and Models in Software Quality Engineering**. Addison-Wesley publishing Company, 2000.
- [17] Donald Firesmith. **A hard look at quality management software. OPEN Process Framework (OPF)**, April 2004.
- [18] ISO. **Iso/iec 14598-1. International Standard, Information technology software product evaluation(2nd)**, 1999
- [19] Sassan Pejhan, Alexandros Eleftheriadis, and Dimitris Anastassiou. **Distributed multicast address management in the global internet**. *IEEE Journal of Selected Areas in Communications*, 13(8):1445-1456, 1995

- [20] Lionel C. Briand, John W. Daly, and Jürgen K. Wüst. A unified **framework for coupling measurement in object-oriented systems**. IEEE Transactions on Software Engineering, 25(1):91-121, January/February 1999
- [21] Mugurel T. Ionita, Dieter K. Hammer, Henk Obbink, "**Scenario-Based Software Architecture Evaluation Methods: An Overview**", <http://www.win.tue.nl/oas/architecting/aimes/papers/Scenario-Based%20SWA%20Evaluation%20Methods.pdf>
- [22] R. Basili, D. M. Weiss, "**A Methodology for Collecting Valid Software Engineering Data**," IEEE Transactions on Software Engineering, vol. SE-10,
- [23] V.R. Basili, R.W. Selby, "**Data Collection and Analysis in Software Research and Management**," Proceedings of the American Statistical Association and Biomeasure Society, Joint Statistical Meetings, Philadelphia, PA, August 1984
- [24] L. Marta. **An Evaluation Theory Perspective of the Architecture Tradeoff Analysis Method (ATAM)**. Pittsburgh. 2000
- [25] N. Boucke, T. Holvoet, T. Lefever, R. Sempels, K. Schelfhout, D. Weyns, T. Wielemans: **AVG case study Applying the Architecture Tradeoff Analysis Method (ATAM) to an industrial multi-agent system Application. SEI Software Architecture Technology User Network (SATURN) Workshop**. 2006
- [26] V. Basili, Caldiera, Gianluigi, Rombach, H. Dieter: **The Goal Question Metric Approach** Encyclopedia of Software Engineering. 1994
- [27] A. Trendowicz, T. Punter, **Quality modeling for software Product Lines**. Darmstadt. 2003
- [28] Klein, M. & Kazman, R. **Attribute-Based Architectural Styles** (CMU/SEI-99-TR-022, ADA371802). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1999. WWW. URL: <<http://www.sei.cmu.edu/publications/documents/99.reports/99tr022/99tr022abstract.html>>
- [29] Iannino, A. "**Software Reliability Theory**." Encyclopedia of Software Engineering, New York, NY: Wiley, May 1994, 1237-1253
- [30] Kazman, R.; Abowd, G.; Bass, L.; & Webb, M. "**SAAM: A Method for Analyzing the Properties of Software Architectures**," 81-90. Proceedings of the 16th International Conference on Software Engineering. Sorrento, Italy, May 1994.

**Appendix A**

**RESEARCH PAPER**



## **Appendix A. Publication**

Our research publication “Metric-Oriented Quality Model for Architecture Tradeoff Analysis Method” has been published in the 9th International Conference on Quality Engineering in Software Technology (CONQUEST'2006), Berlin Germany, Sep 27-29, 2006. This research paper proposes a modification to Architecture Tradeoff Analysis Method (ATAM), one of the famous architecture evaluation method defined by the Software Engineering Institute (SEI), Carnegie Mellon University.

The copy of the research paper is taken from the conference proceedings and is provided in this appendix.

ASQF e.V. (ed.)  
Arbeitskreis Software-Qualität und -Fortbildung e.V.

# Software Quality in Service-Oriented Architectures

Proceedings of the CONQUEST 2006

9th International Conference on  
Quality Engineering in Software Technology –  
Berlin 2006

 dpunkt.verlag

teme

 ons

## Foreword



**Dagmar Wöhrl,  
Parliamentary State Secretary in the German Ministry of  
Economics and Technology,  
Conference Patron:**

With a 6.8 % share of the global ICT market in 2005, Germany was the world's third largest country market after the United States (28 %) and Japan (14.7 %), and by far the largest in Europe. The ICT sector has developed into our most important industrial branch, accounting for sales of 134 billion Euros in 2005 and a 6.2 % share of gross domestic product. It is growing significantly more rapidly than the economy as a whole and thus turning into a driving force for economic performance in the Federal Republic.

Employment figures in the ICT branch have also expanded – by some 4,000 jobs to a total of 750,000. Nearly every tenth job in Germany is now found in the provider and user sides of the ICT business. This increase is particularly the result of the prospering fields of IT services and software. At a good 5 %, software is now the fastest growing market segment. Figures for individuals beginning to study informatics have unfortunately been declining for some years now. All sides must therefore do more for education and training in this field to make sure that German ICT companies remain competitive.

Under the lead management of the Federal Ministry of Economics and Technology, the German government will develop a new Action Program (iD2010 – Information Society Germany 2010) by the end of summer 2006. This will help further improve framework conditions for innovation, growth, and employment,

Nationalbibliografie;  
lib.de> abrufbar.

emarks or registered  
s book in editorial fashion  
ny trade name, is intended

duced or utilized in any  
ay any information storage  
ner.

ASQF e.V.  
Wetterkreuz 19a  
D-91058 Erlangen  
Fon: +49 (0)9131-91910-0  
Fax: +49 (0)9131-91910-10  
E-Mail: info@asqf.de  
www.asqf.de

Copy-Editor: Julia Neumann, Dunedin, NZ  
Producer: Birgit Bäuerlein  
Cover Design: Helmut Kraus, www.exclam.de  
Printer: Koninklijke Wöhrmann B.V., Zutphen, Netherland

Bibliografische Information Der Deutschen Bibliothek  
Die Deutsche Bibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie;  
detaillierte bibliografische Daten sind im Internet über <http://dnb.ddb.de> abrufbar.

ISBN 3-89864-432-4

1st Edition  
Copyright © 2006 dpunkt.verlag GmbH  
Ringstraße 19 b  
69115 Heidelberg  
Germany

All product names and services identified throughout this book are trademarks or registered trademarks of their respective companies. They are used throughout this book in editorial fashion only and for the benefit of such companies. No such uses, or the use of any trade name, is intended to convey endorsement or other affiliation with the book.  
No part of the material protected by this copyright notice may be reproduced or utilized in any form, electronic or mechanical, including photocopying, recording, or by any information storage and retrieval system, without written permission from the copyright owner.

5 4 3 2 1 0

# Contents

<b>Verification of SOA Orchestration</b> <i>P. M. Heck</i>	<b>1</b>
<b>Metric-Oriented Quality Model (MoQaMo) for Architecture Tradeoff Analysis Method (ATAM)</b> <i>A. Javed · A. Hafeez · K. Rashid · H. Farooq Ahmad</i>	<b>17</b>
<b>Automated Software Testing in Service-Oriented Architecture</b> <i>T. Kovacheva</i>	<b>37</b>
<b>Towards SOA-based approaches for IT Quality Assurance</b> <i>A. Farooq · R. Braungarten · M. Kunz · A. Schmietendorf · R. R. Dumke</i>	<b>45</b>
<b>Why Software Project Management is so Challenging</b> <i>P. A. McQuaid</i>	<b>55</b>
<b>Key Success Factors for Project Management Excellence Case Study of Implementing Level 3 of the Project Management Maturity Model (PMMM)</b> <i>E. Siegeris · N. Feuring</i>	<b>63</b>
<b>Experiences with offshore projects within the software development process</b> <i>A. Schmietendorf</i>	<b>77</b>
<b>Statistical Process Control for Software Development</b> <i>T. Fehlmann</i>	<b>89</b>
<b>Testing your Regression Testing: A Case Study on Quality Control of Test Cases at UC4 Software GmbH</b> <i>B. Burger · A. Hammerschmid</i>	<b>103</b>

---

<b>Moderating for Success</b> <i>M. van der Zwan</i>	111
<b>TRex – An Open-Source Tool for Quality Assurance of TTCN-3 Test Suites</b> <i>B. Zeiss · H. Neukirchen · J. Grabowski · D. Evans · P. Baker</i>	117
<b>Requirements Analysis using Unified Modeling Language</b> <i>R. Schönwald</i>	129
<b>Model Transformers for Test Generation from System Models</b> <i>M. Busch · R. Chaparadza · Z. R. Dai · A. Hoffmann · L. Lacmene · T. Ngwangwen · G. C. Ndem · H. Ogawa · D. Serbanescu · I. Schieferdecker · J. Zander-Nowicka</i>	135
<b>Quality Assurance by XMI – Catalyst of a new Age!? Automated Model Reviews ensure Quality in the early Phase of Software Development</b> <i>M. Scholze</i>	151
<b>Functional Size and Reuse Evaluation in ERP Requirements Engineering</b> <i>M. Daneva</i>	161
<b>A Software Development Environment for Ground Segment Operation Software</b> <i>F. C. Cuadrado · E. Gómez · F. Delhaise</i>	181
<b>A Mapping between the CMMI<sup>®</sup> approach and ISO/IEC 15504 with respect to Quality and Efficiency of Requirements Engineering in the Automotive Sector</b> <i>C. Salazar Dorn</i>	193
<b>Web Service Quality Descriptions for Web Service Consumers</b> <i>J. Rückert · B. Paech</i>	203
<b>Securing Web Services</b> <i>R. Groenboom</i>	215
<b>SOA is not enough</b> <i>A. Löffler · T. Löffler</i>	229
<b>Optimizing the Contribution of Testing to Project Success</b> <i>N. Malotau</i>	243
<b>NetQGate – Tool Support for Quality Gate Processes</b> <i>T. Flohr</i>	261

# Metric-Oriented Quality Model (MoQaMo) for Architecture Tradeoff Analysis Method (ATAM)

A. Javed, A. Hafeez, K. Rashid, H. Farooq Ahmad

International Islamic University Islamabad

## Abstract

*It has been realized by the software engineering communities that software architecture serves as an important artifact in producing quality software products. Architecture Tradeoff Analysis Method (ATAM) proposed by the Software Engineering Institute (SEI) is considered to be a well-known methodology for evaluating software architectures. Current research observes a lack of proper guidelines and of a well-defined approach in the generation of a quality attributes utility tree. This lack of standards has limited the usefulness of this method. We proposed a quality model based on the ISO 9126-1 quality model. We support architecture evaluation using the Goal Question Metric (GQM) approach to answer the questions raised by the stakeholders to fulfill business drivers in the light of the ISO 9126-1 quality model. We have also applied the method to the case study Battlefield Control System (BCS) and performed the architecture evaluation by presenting results to overcome the existing problem.*

## 1 Introduction

The purpose of architecture evaluation of software systems is to analyze the architecture in order to identify potential risks and verify that quality requirements have been addressed in the design. Architecture Tradeoff Analysis Method (ATAM) focuses on understanding the consequences of architecture decisions with respect to quality attribute requirements of the system [Kaz00]. Quality is one of the major issues [Kho05]. This has been the oldest practice in the software industry to predict the quality of a software product from higher-level design [Bos00]. Currently software architecture is considered to deal with software quality. It has been realized by the software engineering community that software architecture serves as an important artifact in producing quality software products [Dob02]. The Architecture Tradeoff Analysis Method (ATAM) defines the attribute utility tree to provide a top down mechanism for directly and efficiently interpreting the business drivers of the system into concrete quality attribute scenarios [Kaz00]. Non-functional requirements, like performance,

maintainability and reliability are represented as architectural drivers. The utility tree is formed from the combination of these architectural drivers. These quality requirements are refined into relevant attributes in order to get a prioritized list of scenarios that serves as a plan for the remaining architecture evaluation process.

ATAM uses one level of quality characteristics. However there is not any specific guideline to define the utility tree. Expression of quality view and the reason for one level of refinement is ambiguous [Los03]. Attributes defined in the utility tree are measured in terms of stimuli, parameters and responses. After having a utility tree, we see that there is a lack of coordination among quality characteristics, their refined attributes and resulting scenarios which also attempt to specify measure to attributes.

The purpose of this research is to propose Metric-oriented Quality Model (MoQaMo) based on the ISO 9126-1 [ISO01][ISO98] framework. MoQaMo utilizes the Goal Question Metric (GQM) approach to support software architecture evaluation by answering the quantifiable questions raised by the stakeholders to fulfill business drivers (goals) over the ISO 9126-1 framework. Our work attempts to replace the utility tree used in Architecture Trade off Analysis Method (ATAM) with the MoQaMo model. MoQaMo guides the architecture evaluation process by clearly emphasizing the major quality characteristics. This helps to identify quantifiable questions and a relevant set of metrics derived in order to fulfill the goals.

The remaining part of the paper is as follows. Section 2 discusses different quality models and identifies their weak aspects. The ISO 9126-1 quality model is elaborated with a view as how it supports our work. Section 3 explains the proposed MoQaMo model and gives an overview of the application of the model. Section 4 describes the MoQaMo model using a case study BattleField Control System (BCS) taken from SEI's technical report [Kaz00]. Finally, conclusion and directions for future work are presented in Section 5.

## **2 Related Work**

### **2.1 Quality Models**

A quality model represents an interaction between a set of characteristics and sub-characteristics. This relationship serves as a foundation for specifying quality requirements to assess quality [Kho05]. Despite the growth in software industry, it is surprising that no serious attention is paid to the area of evaluating software quality [Kho04]. Quality is hard to define, impossible to measure and easy to recognize [Bar96]. KAN [Kan03] states "Quality is not a single idea, but rather a multidimensional concept". Furthermore the ISO 9126-1 quality model [ISO01] explains that software quality characteristics are a combination of attributes of a software product and that these attributes determine its quality. All these definitions provide different views on quality. In order to arrive at the best definition, quality





The definition of high-level quality characteristics of the ISO 9126-1 standard is given in Tab. 1.

Tab. 1: ISO 9126-1 Generic Quality Model

Characteristics	Description
Functionality	capability of the software product to provide functions which meet stated and implied needs when the software is used under specified conditions (what the software does to fulfill needs)
Reliability	capability of the software product to maintain its level of performance under the stated conditions for a stated period of time
Usability	capability of the software product to be easy to use when used under specified conditions (the effort needed for use)
Efficiency	capability of the software product to provide appropriate performance, relative to the amount of resources used, under the stated conditions
Maintainability	capability of the software product to provide appropriate performance, relative to the amount of resources used, under the stated conditions
Portability	This describes the capability of the software product to be transferred from one environment to another. The environment may include organizational, hardware or software environment.

### 2.3 Architecture Tradeoff Analysis Method (ATAM)

The purpose of the Architecture Tradeoff Analysis Method (ATAM) is to assess the consequences of architectural decisions in the light of quality attribute requirements [Kaz 00]. The method focuses on the identification of business drivers that lead to quality attribute goals. ATAM helps to analyze how architectural styles support the achievement of these quality attribute goals.

Quality attribute utility tree generation is an important step in ATAM because it guides the remainder of the analysis. However, it is observed that there is no guideline for how to arrive at the utility tree. The quality attributes defined in the utility tree do not show their meanings due to their undefined hierarchical structure. That is, it does not indicate the real meaning of a particular quality attribute in the required context. The stakeholders can take many interpretations from these quality attributes defined in the utility tree which could result in an ambiguous architecture evaluation. Furthermore, we see that the placement of quality attributes in the utility tree is not based on a well



To equip our model with the ability to define appropriate metrics, the Goal Question Metric (GQM) approach [Bas94] has been applied. This approach is based on the assumption that organization must specify the goals for itself and its projects, then it must trace those goals operationally and finally provide a framework for interpreting the data with respect to the stated goals.

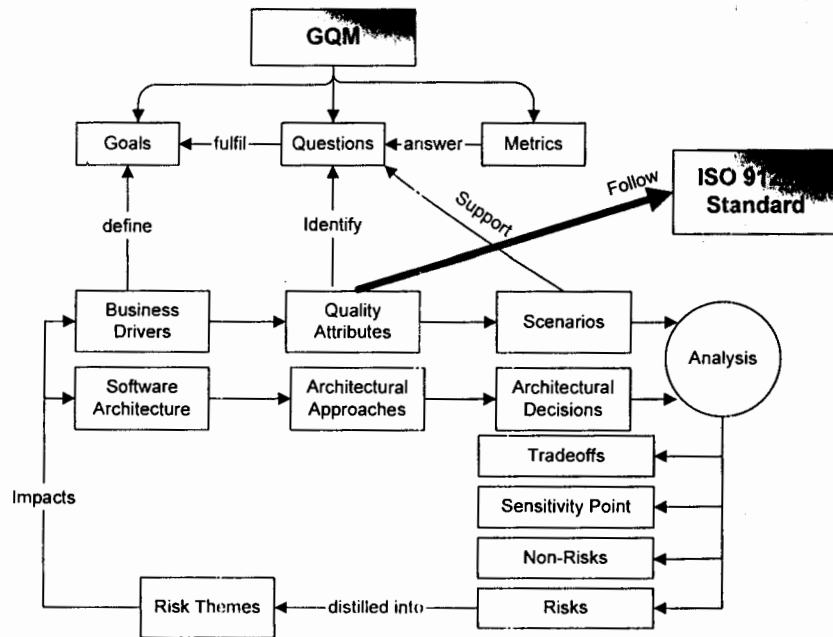


Fig. 2: MoQaMo Conceptual Model with respect to ATAM

From the conceptual model as shown in Fig. 2 we see that business drivers are treated as goals. The architecture evaluation process is exercised to attain these goals. The high-level quality attributes follow the structure provided by the ISO 9126-1 standard. The well-defined structure of the ISO 9126-1 quality model motivates to use quality characteristics in the MoQaMo model. These quality characteristics in ISO 9126-1 are structured with a view to ensure quality of software products. The sub-characteristics (attributes) help to identify quantifiable questions, which are asked in order to fulfill goals (high level quality characteristics). The application of the MoQaMo Model appreciates the involvement of stakeholders in the architecture evaluation process. Questions are raised by the stakeholders in order to observe whether architectural approaches address quality attributes in question or not. In ATAM, we see that quality attributes are defined and covered up by the elicitation of scenarios, that is, scenarios define the context of quality attributes according to the evaluating system. These scenarios pave the way for quantifiable questions to be asked according to relevant sub-characteristics. Once all the relevant questions are defined for a particular goal, metrics

rics, the Goal Question approach is based on the and its projects, then it framework for interpreting

have to be specified in order to answer the quantifiable questions. Each question can be answered with a single or multiple metrics. After having a collection of metrics, we can start analyzing the architecture with the candidate architectural approaches and can assess the consequences of architectural decisions. This particular suite of metrics helps to take justifiable architectural decisions. The analysis which is a core process in ATAM gets improved due to high involvement of stakeholders, a well-defined evaluation criteria and availability of appropriate set of metrics. The same metrics can be used in order to answer different questions for the same goal. We also give another conceptual model [Fig. 3] for MoQaMo with respect to the GQM approach that shows its different components and the interaction among them.

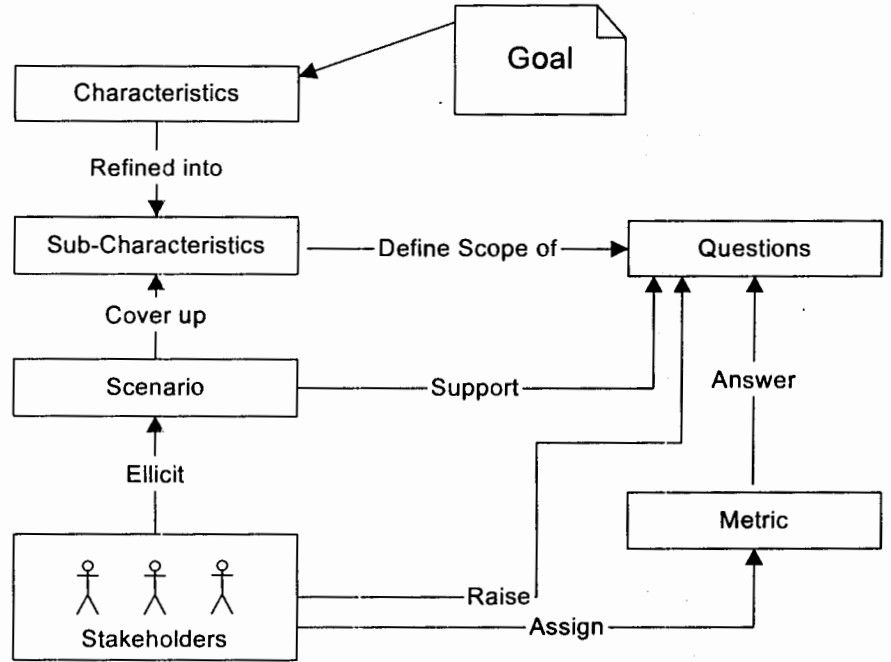
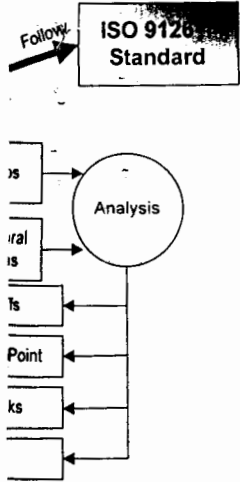


Fig. 3: MoQaMo Conceptual Model with respect to GQM approach

business drivers are treated to attain these goals. The by the ISO 9126-1 standard. del motivates to use quality characteristics in ISO 9126-1 are ucts. The sub-characteristics are asked in order to fulfill on of the MoQaMo Model itecture evaluation process. ervice whether architectural ATAM, we see that quality scenarios, that is, scenarios e evaluating system. These ed according to relevant sub- for a particular goal, metrics

The process of setting goals is critical to the successful application of the MoQaMo model. A goal should be expressible with four dimensions; (i) Issue, (ii) Objects, (iii) Purpose, (iv) Viewpoints. The Measurement Goal Template (MGT) is helpful in the operationalization of quality goals. It also addresses the characteristics that are to be included in an evaluation. MGT, shown in Tab. 2, makes the MoQaMo model more flexible by adjusting it according to the context of a particular software project [Tre03].

Tab. 2: Measurement Goal Template (MGT)

No	Dimensions	Definition
1	Issue (Quality Focus)	Which characteristics of the object are taken into consideration?
2	Object	What is the artifact to be evaluated?
3	Purpose	Why is the object analyzed?
4	Viewpoint	Who will evaluate?

In order to give an example of the application of the MoQaMo model, let's suppose we want to improve the performance of the altitude monitoring device of an airplane. The resulting goal will specify the purpose (improve), an object (altitude monitoring device), the viewpoint (aeronautical engineer) and a quality issue (efficiency). The MoQaMo Model is shown in Tab. 3.

Tab. 3: The MoQaMo Model for altitude monitoring device

Characteristic/Goal	Improve the efficiency of altitude monitoring device from an aeronautical engineer's viewpoint.	
Sub-Characteristics	Time Behavior	
<b>Time Behavior</b>		
S-1	<i>The device should return information within specified time.</i>	
Q-1	How much time does the device take to send information to the screen?	
M-1	0.25 ms	
Q-2	What is the response time of the device in calculating the altitude?	
M-2	0.5 ms	
Q-3	What is the frequency of information production?	
M-3a	20 times / minutes	
M-3b	Once in every 3 sec.	

### 4 Application of MoQaMo Model to a Case Study

We have selected the case study for which the Software Engineering Institute (SEI) has carried out architecture evaluation using ATAM. The details of the case study can be found at the SEI's technical report [Kaz00]. We have applied the MoQaMo model to a BattleField Control System (BCS). This system is designed with an aim to support army battalions in controlling the movement, strategy and operation of troops in real-time Battlefield.

#### 4.1 Driving Architectural Requirements

There is a commander who commands a set of soldiers and their equipments, including many different kinds of weapons and sensors. External systems need to be interfaced with the BCS system in order to capture its commands and intelligence information. The commander communicates with all the soldiers on a real-time basis. The system is considered to be working if there is a working commander along with a number of soldiers. The failure of the system depends upon the life of the commander. The commander fights the battle according to mission plan and utilizes its available resources during the battle. A radio modem with 9600-baud speed supports the communication between the commander and the soldiers. There is a need for extreme level of robustness and a number of performance goals should be considered. The system is also subject to frequent modifications.

#### 4.2 High-Level Architectural Views

We give a few high-level architectural views of the BCS to flesh out our understanding about its working structure. We are not going to present detailed architectural documentation of the system. From hardware view shown in Fig. 4, we observe that the commander is central to the system. In fact, the commander node acts as a server and fulfils the clients (soldiers) requests. Inter-node communication between the clients and the server is only through encrypted messages sent via a radio modem.

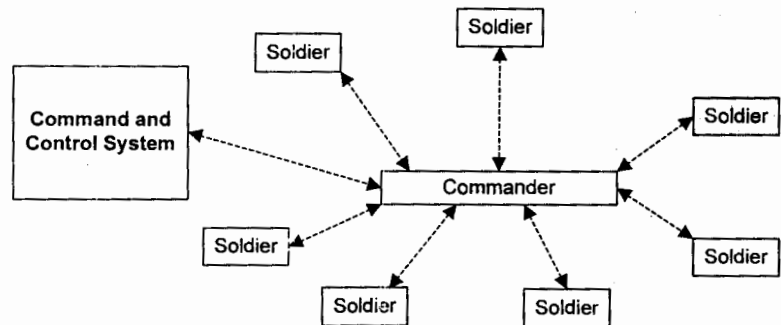


Fig. 4: System view of BCS

are taken into

model, let's suppose we  
device of an airplane. The  
ect (altitude monitoring  
issue (efficiency). The

monitoring device viewpoint.
ed time.
mation to the screen?
ting the altitude?

The module decomposition view shown in Figure 5a. identifies the major components of the system. This view may include many more modules but for better understanding, we have mentioned only a few. The module decomposition view consists of Decision Support System (DSS), Communication Manager (Comm. Mgr) and Battle Controller (B.Controller). Figure 5b shows a module layered view. In this view, the BCS application layer is using the services of its lower layers. The BCS Application Programming Interface (API) layer is developed over .Net framework and provides a programmable interface to the layer above it.

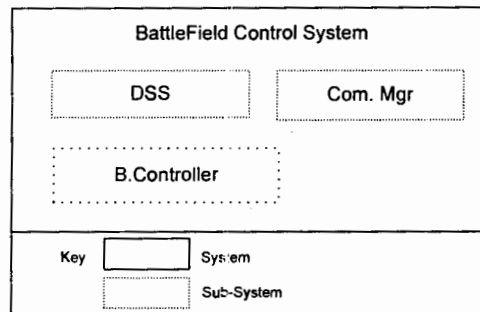


Fig. 5a: Module decomposition view

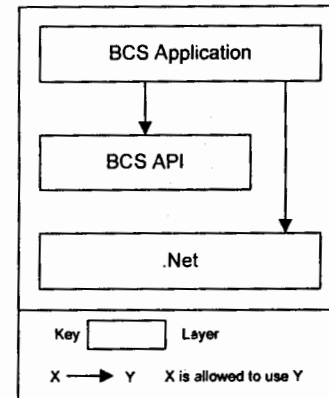


Fig. 5b: Module layered view

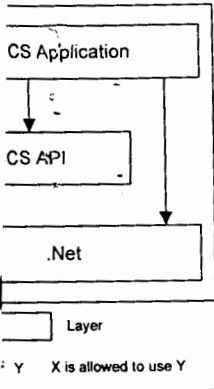
Architectural approaches are considered with respect to modifiability, availability, and performance. To achieve availability, a backup commander approach was described. Availability and performance of the system are found to be a high priority of the stakeholders. The MoQaMo model attempts to organize the architectural evaluation process by eliciting information for desired quality characteristics through high involvement of stakeholders. This organized and efficient structure helps an evaluation team to realize those architectural approaches and architectural decisions that cater for the desired quality characteristics (quality goals). The MoQaMo model serves as a guideline and well-defined structure to evaluate software architectures using ATAM.

### 4.3 Architectural Evaluation

From the case study, we observed that the stakeholders were highly interested in system availability, modifiability and performance. In order to achieve availability, an architectural approach was needed to be employed which could sustain the existence of the commander in the battlefield. We will only consider availability and performance of the system in evaluating architecture. We observe that system availability is primarily affected by the failure rate of the commander, the repair rate of the commander (the time required for the backup to become commander) and the repair rate of the backup



the major components for better understanding, consists of Decision and Battle Controller. In this view, the BCS Application network and provides a



Module layered view

reliability, availability, and approach was described. As a high priority of the architectural evaluation characteristics- through high architecture helps an evaluation of architectural decisions that cater for the MoQaMo model serves as a tool for architectures using ATAM.

highly interested in system to achieve availability, an architecture should sustain the existence of system availability and performance of system availability is primarily dependent on the repair rate of the commander (the repair rate of the backup

(the time required for the soldier to become a backup). The shaded soldier node indicates the backup. According to existing architecture, availability of the system is assured by provoking the backup soldier node to mirror the commander's state through acknowledged communication (state messages) with commander. Upon failure of the commander, the backup takes over as the new commander.

To achieve high availability (a high level of readiness), an alternative architecture proposed that the multiple soldier nodes could be put to monitor the commander-to-backup communication. The backup soldiers could be acknowledged backups (requesting resends of missed packets) or could be passive backups (silent receiver packets) or a mixture of these concepts. In the case where packets are not acknowledged, the state of the backup database would increasingly drift from that of the commander. If one of these backups is called upon to become the commander, it would need to engage in some negotiation (with the external systems and/or the other Soldier nodes) to complete its database.

It is clear to say that the availability of the system increases as the number of backups is increased, because the system can survive multiple failures of individual nodes without failing its mission.

The MoQaMo model for reliability, shown in Tab. 4, documents the reliability quality attribute in addition to scenarios and relevant questions that aim to achieve this goal. The symbols used are S, Q, and M that represent Scenarios, Questions and Metrics respectively. We know that reliability concerns to the successful working of the system, therefore availability is used as a substitute to the recoverability sub-characteristic while documenting the MoQaMo model.

Tab. 4: MoQaMo Model for Reliability Quality Characteristic (Goal)

Characteristic/Goal	Increase the reliability of the system from an evaluation team's viewpoint.		
Sub-Characteristics	Recoverability/Availability	Maturity	Fault-Tolerance
<b>Availability</b>			
S-1	<b>Failure of the commander node. (H,H)</b>		
Q-1	What is the failure rate of the commander?		
M-1	Unknown		
Q-2	What is the repair rate of the commander?		
M-2	10 sec		
Q-3	What is the repair rate of the backup?		
M-3a	5 min		
M-3b	300 sec		

Characteristic/Goal	Increase the reliability of the system from an evaluation team's viewpoint.		
Sub-Characteristics	Recoverability/Availability	Maturity	Fault-Tolerance
<b>Availability</b>			
S-1	<i>Failure of the commander node.(H,H)</i>		
Q-4	What is the number of backup soldiers in current battlefield?		
M-4	24		
Q-5	What is the number of acknowledging backups and passive backups?		
M-5	15 acknowledging backups and 9 passive backups		
Q-6	How is the failure of the commander detected?		
M-6	The backup soldier has employed "Pull Heart Beat" mechanism to see the health status of the commander.		
Q-7	Which availability tactic is used to achieve availability?		
M-7	"Fail Over Cluster Pattern"		
S-2	<i>Backup Soldiers needs to be synchronized with the commander.(H,H)</i>		
Q-8	Which approach is used for state synchronization between commander and soldiers?		
M-8	"Hot Standby" technique is used: the internal state of the commander is immediately copied to backup soldier.		
Q-9	Do the commander and the soldiers use shared storage device to maintain their states?		
M-9	No storage area network exists.		

At the same time, performance of the system was also a considerable architectural driver. The communication between the commander and the soldier was made via a radio modem with 9600-baud speed. Due to this constraint, the performance model was focused on capturing those architectural decisions that affected message sizes and distributions. To turn a soldier node into a backup, the backup acquires information about all missions, updates to the environmental database, issued orders, current soldier locations and status, and detailed inventories from the soldiers. The MoQaMo model for efficiency shown in Tab. 5 represents the efficiency quality attribute with a supporting set of scenarios and questions.

With a 6.8 % third largest and by far the most important in a 6.2 % and rapidly than economic per Employ jobs to a total provider and of the prospective now the fast study information must therefore German ICT, Under the technology, the Information S further improv



Tab. 5: MoQaMo Model for Efficiency Quality Characteristic (Goal)

Characteristic/Goal	Improve the efficiency of the system from evaluation team's viewpoint.	
Sub-Characteristics	Time Behavior	Resource Behavior
<b>Time Behavior</b>		
S-3	<i>Turning a Soldier node into a backup (H,L)</i>	
Q-10	What is the communication speed between the commander and the soldiers?	
M-10	9600 baud (9600 bits/sec) or 9.6 kbits / sec	
Q-11	What is the size of mission plans to be downloaded?	
M-11	280 kbits	
Q-12	How much time is required to download mission plans?	
M-12	280 kbits / 9.6 kbits/sec = 29.17 sec	
Q-13	What is the size of updates to environmental database to be made by the backup soldier?	
M-13	66 kbits	
Q-14	How much time does it take to make updates to environmental database?	
M-14	66 kbits / 9.6 kbits/sec = 6.88 sec	
Q-15	How much time is required to acquire issued orders (for 24 soldiers)?	
M-15	24 soldiers * (18 kbits / 9.6 kbits/sec) = 45 sec	
Q-16	How much time does it take to get the location and status of soldiers (for 24 soldiers)?	
M-16	24 soldiers * (12 kbits / 9.6 kbits/sec) = 30 sec	
Q-17	What is the time needed to acquire information about the inventories (for 24 soldiers)?	
M-17	24 soldiers * (42 kbits / 9.6 kbits/sec) = 105.0 sec	
Q-18	How much time is required for a soldier to become a backup (in case of 24 soldiers)?	
M-18	It takes 216.05 sec for a soldier to become a backup.	

evaluation  
 ult-Tolerance  
 backups?  
 nism to see the  
 der. (H,H)  
 commander and  
 mmander is  
 vice to maintain

erable architectural  
 tier was made via a  
 ormance model was  
 message sizes and  
 equires information  
 orders, current soldier  
 MoQaMo model for  
 te with a supporting

Characteristic/Goal	Improve the efficiency of the system from evaluation team's viewpoint.	
Sub-Characteristics	Time Behavior	Resource Behavior
<b>Time Behavior</b>		
S-4	<i>The System is required to run with 35 soldiers.(L,H)</i>	
Q-19	How much time is required to acquire issued orders (for 35 soldiers)?	
M-19	35 soldiers * (18 kbits / 9.6 kbits/sec) = 65.63 sec	
Q-20	How much time does it take to get the location and status of soldiers (for 35 soldiers)?	
M-20	35 soldiers * (12 kbits / 9.6 kbits/sec) = 43.75 sec	
Q-21	What is the time needed to acquire information about the inventories (for 35 soldiers)?	
M-21	35 soldiers * (42 kbits / 9.6 kbits / sec) = 153.12 sec	
Q-22	How much time is required for a soldier to become a backup (in case of 35 soldiers)?	
M-22	It takes 298.55 sec for a soldier to become a backup.	
S-5	<i>Acknowledging and passive backups need periodic updates from the commander to give a high state of readiness.</i>	
Q-23	Can acknowledging and passive backups give a high state of readiness?	
M-23	Yes, the acknowledging backups are more ready to assume the responsibilities of the commander much more quickly. Passive backups need to negotiate with other nodes for missed information.	
Q-24	From Scenario S-6, what is the average message size in every (10 minutes/ per minute / per second)?	
M-24a	59,800 kbits every 10 minutes	
M-24b	99.67 bits / sec (1% of the system's overall communication bandwidth)	

Once the scope of the evaluation has been set by the MoQaMo model elicitation process, we attempt to probe for the architectural approaches that realize the important quality attributes. Architectural decisions are documented and their relevant risks, sensitivity points, and tradeoffs are identified. We associate the highest priority quality attribute requirements (as identified in the MoQaMo model) with the architectural approaches to realize them. As shown in Tab. 6, we capture an architectural approach for availability quality attribute and present architectural decisions as well as some reasoning.

ASQF e.V.  
Wetterkreuz 19a  
D-91058 Erlangen  
Fon: +49 (0)9131-  
Fax: +49 (0)9131-  
E-Mail: info@asqf.  
www.asqf.de

Copy-Editor: Julia I  
Producer: Birgit Bä  
Cover Design: Helr  
Printer: Koninklijke

Bibliografische Info  
Die Deutsche Bibli  
detaillierte bibliogr.

ISBN 3-89864-432-4

1st Edition  
Copyright © 2006 dt  
Ringstraße 19 b  
69115 Heidelberg  
Germany

All product names ar  
trademarks of their re  
only and for the bene  
to convey endorseme  
No part of the materi  
form, electronic or me  
and retrieval system, v

5 4 3 2 1 0

in evaluation
source Behavior
us of soldiers
ne inventories
ckup
tes from the
ate of readiness?
sume the Passive backups on.
in every
tion bandwidth)

Tab. 6: Architectural Approach description for Availability

<p><b>High-level quality characteristic (goal):</b> reliability  <b>Quality attribute:</b> recoverability/availability  <b>Scenario S1</b> (failure of the commander node)                  Stimulus: death of commander                  Downtime: 10 sec (M2)                  Number of acknowledging backups = 15 (M-5)                  Number of passive backups = 9 (M-5)                  Repair rate of the backup = 300 sec (M-3b)</p>			
Architectural decisions	Risks	Sensitivity points	Tradeoff points
Failover cluster (M-7)		S1	
Pull Heart Beat (M-6)		S2	T1
Ping/echo			T2
Hot standby (M-8)		S4	
Transaction log	R1		
Backup network channel (M-10)	R2		
Acknowledging backups	R3	S6	
Passive backups		S7	
<p><b>Reasoning:</b></p> <ul style="list-style-type: none"> <li>- Failover cluster pattern is aimed to achieve the availability. This involves ... (See S1).</li> <li>- The Pull Heart Beat mechanism requires the backup soldier ... (See S2, T1).</li> <li>- According to ping/echo approach, the backup soldiers send a ping to the commander at specific intervals of time and the ... (See T2).</li> <li>- The shared communication channel between commander and soldiers is a major risk ... (See R2).</li> <li>- Hot Standby technique gives the availability ... (See S4).</li> <li>- The transaction logs mechanism, which requires the commander ... (See R1).</li> <li>- A high number of acknowledging backups can negatively affect ... (See R3).</li> <li>- Acknowledging and passive backups assure availability ... (See S6, S7).</li> </ul>			

AMO model elicitation at realize the important d their relevant risks, highest priority quality with the architectural architectural approach sions as well as some

By having an appropriate set of metrics we can arrive at the right set of tradeoff points, sensitivity points and risks. In Tab 6; S, R and T are mentioned as pointers to sensitivity points, risks, tradeoff points respectively. The risks, sensitivity points and tradeoff points are enlisted in Tab. 8.

We also provide reasoning for efficiency quality characteristic. Tab. 7 presents architectural decisions that were proposed in an architectural evaluation process.

Tab. 7: Architectural Approach description for Efficiency

<p><b>High-level quality characteristic (goal):</b> efficiency  <b>Quality attribute:</b> time behavior  <b>Scenario S1</b> (turning a Soldier node into a backup)  Stimulus: Death of commander  Communication speed = 9600 baud (M-10)  Mission plan size = 280 kbits (M-11)  Time for soldier to become backup (in case of 24 soldier) = 216.05 sec (M-18)  Time for soldier to become backup (in case of 35 soldier) = 298.55 sec (M-22)  Time required to download mission plan (with Q15) = 29.17 sec (M-12)  Time required to download mission plan (for 560 kbits mission size) = 58.34 sec</p>			
Architectural decisions	Risks	Sensitivity points	Tradeoff points
Hot standby (M-8)		S8	
Transaction log	R4		
Ping/echo			T2
Communication channel			T4
Acknowledging backups (M-5)		S9	
Passive backups (M-5)		S10	
<p><b>Reasoning:</b></p> <ul style="list-style-type: none"> <li>- Hot standby technique involves a high communication ... (See S8).</li> <li>- The transaction log mechanism though would tend to improve efficiency by reducing message transmission ... (See R4).</li> <li>- Ping/echo approach does not involve high communication overhead ... (See T2).</li> <li>- Communication load was affected by various information exchanges ... (See T4).</li> <li>- A high number of acknowledging and passive backups could result in extreme performance degradation and may result in a system breakdown ... (See S9, S10).</li> </ul>			

Under the German ICT must therefore study in form now the fast of the prospective provider and jobs to a total. Employment economic per rapidly than and a 6.2% important in and by far the third largest. With a 6.8%



tradeoff points, sensitivity points and tradeoff points.

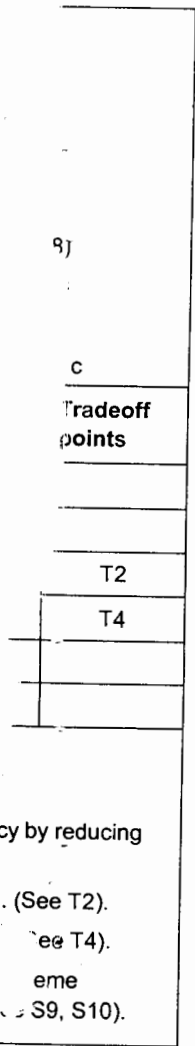
Fig. 7 presents the process.

In Tab. 8, all the architectural decisions are specified with their relevant candidate risks. A clear picture of all the relevant items including metrics, risks, and scenarios offer a better reasoning opportunity about architectural approaches. The risks, sensitivity points and tradeoff points are specified in Tab. 8.

The MoQaMo model does not eliminate the importance of scenario prioritization and brainstorming process. Instead, it supports the involvement of the stakeholders by inclining them to raise a set of approach-specific and quality-attribute-specific questions. These questions catalyze deeper analysis of architecture with respect to the desired quality characteristics. The prioritized scenarios can then be put in a MoQaMo model for exploration of architectural approaches.

Tab. 8: List of risks, sensitivity and tradeoff points

R1	The transaction log mechanism, which requires the commander to update its status in its own database, is a risk if commander fails, as its status would also be lost.
R2	The shared communication channel between commander and soldiers is a major risk in case of communication channel breakdown.
R3	A high number of acknowledging backups can negatively affect availability due to high resource demands (high communication bandwidth could result in system hang status)
R4	The transaction log mechanism though would tend to improve efficiency by reducing message transmission within nodes but it is a risk if commander fails, as its status would also be lost.
S1	Failover cluster pattern is aimed to achieve the availability. This involves switching of a backup soldier to the commander.
S2	Pull HeartBeat mechanism requires the backup soldiers to monitor the status of the commander periodically.
S4	Hot standby technique gives the availability of commander's state in real-time manner.
S6	Acknowledging and passive backups assure availability as these support multiple backups (switchover backups) to show a high state of readiness.
S7	Passive backups provide availability to the system by not capturing high communication bandwidth.
S8	Hot standby technique involves a high communication overhead because commander's status is copied to backup soldiers in real-time.
S9	A high number of acknowledging and passive backups could result in extreme performance degradation and may result in system breakdown.
S10	Passive backups are not very resource demanding as compared to acknowledging backups.



ASQF e.V.  
 Wetterkreuz 19a  
 D-91058 Erlange  
 Fon: +49 (0)9131  
 Fax: +49 (0)9131-  
 E-Mail: info@asqf  
 www.asqf.de

Copy-Editor: Julia  
 Producer: Birgit B  
 Cover Design: He  
 Printer: Koninklijj

Bibliografische In  
 Die Deutsche Bibl  
 detaillierte bibliog

ISBN 3-89864-432

1st Edition  
 Copyright © 2006  
 Ringstraße 19 b  
 69115 Heidelberg  
 Germany

All product names  
 trademarks of thei  
 only and for the bi  
 to convey endorse  
 No part of the mat  
 form, electronic or  
 and retrieval syste

5 4 3 2 1 0

T1	The Pull Heart Beat mechanism is a resource demanding approach.
T2	According to the ping/echo approach, the backup soldiers send a ping to the commander at specific intervals of time and the commander responds with an echo to confirm it exists. This approach does not consume bandwidth cost.
T4	Communication load was affected by various information exchange requirements and availability

## Conclusion

The MoQaMo model supports the quantifiable evaluation of software architecture by giving measures to the refined quality attributes. The Goal Question Metric approach assists in defining the subjective as well as objective metrics with a high involvement of stakeholders of the system. The model does not restrict itself only to measurement specification but is flexible enough to be applied to any attributed-based architecture analysis method. The proposed model focuses on the analysis phase of the ATAM and guides all the remaining evaluation process.

Another important aspect that will be explored in the near future is the definition of metrics for architectural approaches. This would make it possible to map appropriate architectural approaches to meet quality attributes in an architecture evaluation process.

## References

- [Bar96] K. Barbara, P. Shari Lawrence: Software quality: The elusive target. IEEE Software. 1996.
- [Bas03] L. Bass, P. Clements, R. Kazman, Software Architecture in Practice. Addison Wesley. 2003
- [Bas94] V.. Basili, Caldiera, Gianluigi, Rombach, H. Dieter: The Goal Question Metric Approach. Encyclopedia of Software Engineering. 1994.
- [Boe78] B. Boehm, J.. Brown, H. Kaspar, H. Lipow, M.. McLeod, M. Merritt: Characteristics of Software Quality, North Holland. 1978
- [Bos00] J. Bosch: Design and Use of Software Architecture. Harlow. 2000
- [Bou06] N. Boucke, T. Holvoet, T. Lefever, R. Sempels, K. Schelfhout, D. Weyns, T. Wielemans: AVG case study Applying the Architecture Tradeoff Analysis Method (ATAM) to an industrial multi-agent system Application. SEI Software Architecture Technology User Network (SATURN) Workshop. 2006
- [Dob02] L. Dobrica, E. Niemela: A Survey on Software Architecture Analysis Methods. IEEE Transactions On Software Engineering. 2002
- [ISO01] ISO/IEC 9126 Software engineering – Product quality Part1: Quality model, International Organization for Standardization. 2001.
- [ISO98] ISO/IEC FCD 9126-1.2: Information Technology – Software Product Quality. Part 1: Quality Model, draft. 1998
- [Kaz00] R. Kazman, M. Klein, P. Clements: ATAM: Method for Architecture Evaluation. Pittsburgh. 2000
- [Kan03] H. KAN Stephen: Metrics and Models in Software Quality Engineering. Pearson Education Singapore. 2003
- [Kho05] K. Khosravi, Y. Gu'eh'eneuc: On Issues with Software Quality Models. 19th European Conference on Object-Oriented Programming SECC. 2005



g to the ts with width

[Kho04] K. Khosravi, Y. Guéhéneuc: A Quality Model for Design Patterns. University of Montreal. 2004  
 [Los03] F. Losavio, L. Chirinos, N. Lévy, A. Ramdane-Cherif: Quality Characteristics for Software Architecture. Journal Of Object Technology. 2003  
 [Mar02] O. Maryoly, A. Maria, R. Teresita: A systemic quality model for evaluating software products. Laboratorio de Investigacin en Sistemas de Informacin. 2002  
 [Mar00] L. Marta. An Evaluation Theory Perspective of the Architecture Tradeoff Analysis Method (ATAM). Pittsburgh. 2000  
 [McC77] J. McCall., P. Richards, G. Walters: Factors in Software Quality. Nat'l Tech. Information Service. 1977  
 [Tre03] A. Trendowicz , T. Punter, Quality modeling for software Product Lines. Darmstadt. 2003

architecture by  
 Metric approach  
 lvement of  
 to measurement  
 architecture  
 ATAM and  
 definition of  
 appropriate  
 valuation process.

ware. 1996.  
 Wesley. 2003

software

AVG case  
 agent system  
 Workshop. 2006  
 ds. IEEE Transactions

International

Quality Model,

burgh. 2000  
 tion

Conference

