ISSN 2250-3153

A Research Study on Software Quality Attributes

Dr.A.Chandrasekar [*], Mrs.SudhaRajesh [**], Mr.P.Rajesh [***]

[*] Professor, CSE Department, St.Joseph's College of Engineering
[**] Asst. Prof, CSE Dept., SRR Engineering College
[***] Asst. Prof, IT Dept, M.S.A.J College of Engineering

Abstract- Software quality is becoming an important part in software design, helping the designer to handle the complexity of large systems. While designing, the architect should analyze the system requirements before committing the resources to it. The analyzing process helps us to ensure the high quality of architecture design. For the past decade, there were many analyzing methods are used, which in turn to analyze only the views of single stakeholder. By doing so, there are many limitations that lead to critical situation in the development process. They elaborated this situation to excessive amount of time to perform the complete analysis. The scope of finding the key architectural decision is very difficult. Intend of these types of analysis gives the detailed information only after the designing phase, which makes the software unusable and not satisfied by the end-users. Generally, unusable software is useless. Customers and users won't accept un¬usable software, even if it provides the required features with the required operations.

This paper gives the survey on software quality attributes. It is also used to manage the conflicts in views by analyzing it, with finest software quality attributes such as Performance, Dependability and Safety concerns. It represents one or more structural aspects, which illustrate how the architecture addresses the concerns such as requirements, objective, intention of stakeholders for the architecture design. This paper also gives the stakeholder's views with preeminent quality attributes, which meets the non functional requirements (such as reliability, usability, maintainability and portability). Thus by having the centric-view of stakeholders with superlative software quality attributes, guarantees an optimum quality for software architecture design.

Index Terms- Quality Attributes

I. INTRODUCTION

There are many different definitions of quality. For some it is the "capability of a software product to conform to requirements." (ISO/IEC 9001) while for others it can be synonymous with "customer value" (Highsmith, 2002) or even defect level.

The first definition of quality history remembers is from Shewhart in the beginning of 20th century: There are two common aspects of quality: one of them has to do with the consideration of the quality of a thing as an objective reality independent of the existence of man. The other has to do with what we think, feel or sense as a result of the objective reality.

In the context of software engineering, software quality refers to some relations but distinct notions that exist wherever quality is defined in a business context:

Software functional quality reflects how well it complies with or conforms to a given design, based on functional requirements or specifications. That attribute can also be described as the fitness for purpose of a piece of software.

Software structural quality refers to how it meets non-functional requirements that support the delivery of the functional requirements, such as robustness or maintainability, the degree to which the software was produced correctly.

Quality Attributes

Software quality is defined as the degree to which software possesses a desired combination of attributes. [1]The quality requirements to build the software architecture have to fulfill the stakeholders. They are commonly divided in two main groups based on the quality they are requesting, i.e., development and operational qualities. A *development quality requirement* is a requirement that is of importance for the developers work, e.g., maintainability, understandability, and flexibility. *Operational quality requirements* are requirements that make the system better from the user's point of view, e.g. performance and usability. Depending on the domain and priorities of the users and developers, quality requirements can become both development and operational, such as performance in a real-time system.

A quality attribute is the property of a software system. A quality requirement is a requirement that is placed on a software system by a stakeholder; a quality attribute is what the system actually presents once it has been implemented. During the development of the architecture it is therefore important to validate that the architecture has the required quality attributes, this is usually done using one or more architecture evaluations.

Quality Attributes in Focus

The focuses are on the following quality attributes: performance, maintainability, testability, and portability.

The IEEE standard 610.12-1990 [2] defines the four quality attributes as:

Maintainability: This is defined as: "The ease with which a software system or component can be modified to correct faults, improve performance or other attributes, or adapt to a changed environment."

Maintainability is a multifaceted quality requirement. It incorporates aspects such as readability and understandability of the source code. Maintainability is also concerned with testability to some extent, as the system has to be re-validated during the maintenance.

Performance: Performance is defined as:

"The degree to which a system or component accomplishes its designated functions within given constraints, such as speed, accuracy, or memory usage."

There are many aspects of performance, e.g., latency, throughput, and capacity.

Testability: Testability is defined as: "The degree to which a system or component facilitates the establishment of test criteria and the performance of tests to determine whether those criteria have been met".

The effort needed to validate the system against the requirements. A system with high testability can be validated quickly.

Portability: Portability is defined as: "The ease with which a system or component can be transferred from one hardware or software environment to another."

The portability is not only between different hardware platforms and operating systems, but also between different virtual machines and versions of frameworks.

II. RELATED WORK

The quality attributes are very important for the software design to satisfy the users. The quality attributes are measured in different ways as follows,

A. Architecture Quality Revisited

Frank Buschmann, David Ameller, Claudia P. Ayala, Jordi Cabot, and Xavier Franch [3], a study that says "Non-Functional Requirements in Software Architecture Practice," investigates how architects deal with nonfunctional requirements (NFRs) in their daily practices. The results appear contradictory to the common belief that nonfunctional quality is fundamentally important for architecture sustainability and project success. They raised some questions as,

- *What types of NFRs are relevant to software architects?*
- *How are NFRs elicited?*
- *How are NFRs documented?*
- *How are NFRs validated?*

They also suggest that nonfunctional quality is of little relevance to users and customers but mainly a concern for architects. Nontechnical constraints appear to be driving design as prominently as quality requirements.

B. Defect Tracking Systems

Jan M.W. Kristiansen, Steria [4], introduced Defect Tracking Systems (DTS), to facilitate software quality improvement. The focus is mainly on either revising the values of existing defect classification attributes in an existing DTS or introducing new attributes. Primarily, they wanted to give project managers and developers more current, relevant, correct, and easy-to- analyze defect data for assessing software quality and finding potential SPI measures in a cost-effective way. A case study is done by collecting some data from companies to track the defects in the quality attributes. None of the companies recorded the actual effort used to fix a defect, so they perform root-cause analysis to prevent further defects, especially for those that were most costly to fix. Other problems included as incomplete data, inconsistent data, mixed data. The DTS improvement aimed to reduce the defect density and to improve defect-fixing efficiency. To achieve this goal, the DTS must provide supplementary information that the quality assurance (QA) managers could use to answer the following questions:

- *What are the main defect types?*
- *How the companies prevent defects in a project's early stages?*
- *What are the reasons for the actual defect-fixing effort?*

C. Guideline-Based Approach

Malik Hneif and Sai Peck Lee[5], their approach is to achieving Non-Functional Attributes (NFA) quality is preventive, as opposed to curative- that is, it focuses on preventing defects associated with NFAs during the software development life cycle, rather than identifying and correcting defects after testing. Practical implementation is done through an optimal set of prioritized guidelines that software engineers can identify and apply efficiently throughout system development**.** The approach has two steps as

- *Selecting Guidelines*
- *Using Guidelines to Prevent NFA Defects*

Three factors affect the guideline selection - NFA priorities, Guideline effects on NFAs, Guideline interrelationships. After selecting guidelines there should not be any overlapping or conflicts. Then there are two stages to prevent NFA defects as preparation stage and application stage.

D. Software Components Quality in Bayesian Networks

M. F. Bertoa, M. A. Moraga, M. C. Morcillo and C. Calero[6], they suggests to improve the quality of software products, which traditionally focused on improving the Internal or External Quality based on the idea that a good External Quality guarantees a good Quality in Use. To analyze the relationships between External Quality and Quality in Use with the external quality sub characteristics *Bayesian Networks* is used to model these relationships and provide a method to define them in a measurable way.

E. Risk-based requirements model

Martin S. Feather, Steven L. Cornford, and Kenneth A. Hicks, James D. Kiper, Tim Menzies[7], they proposed the Defect Detection and Prevention(DDP) model to make the early decision of requirements in the software development phases. They populate this model with three concepts as *Requirements*: What are the functional and nonfunctional requirements of the project, system, or technology? How might they factor into its development? *Risks:* What might delay attaining these requirements? *Mitigations:* What to reduce risks?

F. Issue-Oriented Approach

Norman F. Schneidewind [8], suggested this approach is to measure the software quality in two ways that address nine issues in software companies. The first approach derives knowledge requirements from a set of issues identified during two standards efforts—IEEE Std. 1061-1998 for a Software Quality Metrics Methodology and the American National Standard

Recommended Practice for Software Reliability (ANSI/AIAA R-013-1992). The second approach ties these knowledge requirements to phases in the software development life cycle. Together, these approaches define a body of knowledge that shows software engineers why (issue-oriented) and when (phase-oriented) to measure quality. By answering these issues the software engineers perform the function in life-cycle quality management plan. *The issues are goals, Cost and risk, context, Operational profile, model, data requirements, Measurement types and granularity, Product and process test and evaluation, Product and process quality prediction.* It also accounts for time, with measurements obtained during the early part of the life cycle being generally less quantitative than those obtained later. Both the product and process evolve over the lifecycle phases, so the objects measured during test and operation are not the same objects measured during requirements analysis. Not only are the objects different but requirements and design approaches can change many times during the life cycle.

G. Software Quality Measurement

Ho-Won Jung and Seung-Gweon Kim, Chang-Shin Chung [9], a survey is made on software quality measurement and to address the issues of software product quality, which is defined by *the Joint Technical Committee 1 of the International Organization for Standardization and International Electro technical Commission published a set of software product quality standards known as ISO/IEC 9126*. These standards specify software product quality's characteristics and sub characteristics and their metrics. However, some in the software engineering community have expressed concerns
about a lack of evidence to support such standards. User satisfaction is often considered a critical outcome of quality management, and studies show it as having a positive impact on organizational cost, profit, and sales growth. The defined Characteristics are Functionality, Reliability, Usability, Efficiency, Maintainability, and Portability.

Table I. Scope and Limitations of Existing Methods

Method Name	Scope	Limitations/Future Enhancements
Architecture Quality Revisited [3]	the study suggests that nonfunctional quality is of little relevance for users and customers, and is instead primarily a concern for Software architects.	The practitioners consider non -functional qualities as an afterthought, rather than as a prime driver of architecture design. Development teams underestimate the contribution of nonfunctional qualities to a system's success
Defect Tracking Systems(DTS) [4]	The improved DTS provided valuable inform –ation to initia -lize and justify software process improvements and software quality assessment.	In future it continuous the work to collect more cost and benefit data of these DTS improvements to get a comprehensive understanding of their Return On Investment (ROI)
Guideline-Based Approach [5]	an approach for improving NFA quality by identifying guide -lines to help software engineers better meet non functional requ-irements during system design, implementation, and deployment.	However, some NFAs might require a specific quality level. Quantification techniques could enable achievement of a targeted NFA quality level though not necessarily the highest level.
Software Components Quality in Bayesian Networks [6]	The aim is to avoid un -necessary costs or irrelevant characteristics for the end users who un -necessarily raise the cost and effort of product development.	Sometimes there will be confusion in choosing sub characteristics of quality attribute, that lead to more cost, which dissatisfy the end users.
Risk-based requirements model [7]	The method's name reflects its purpose: to help developers cost-effectively select assurance activities and thereby both prevent the introduction of hardware defects and detect and correct existing ones.	As a future research, they planned to continue in studying the requirements needs of a wide variety of technologies as software, hardware, and combinations of the two.
Issue-Oriented Approach [8]	The approach derives knowledge requirements from a set of issues identified and ties these knowledge requirements to phases in the software development life cycle. Together, it define a body of knowledge to software engineers why and when to measure quality.	They suggested in giving the requirements a high priority in the core body of knowledge for software engineering, adding it to the requirements for certification and licensing. So doing would help advance quality measurement from a craft to a profession
Software Quality Measurement [9]	A survey is made to empirically investigate whether the ISO/IEC 9126 categorization is correct and reliable	The survey data should be augmented with more comprehensive measures of product quality in future studies. Replications of study

	in evaluating user satisfaction with the judgment of a packaged software product's quality.	using other statistical analytic methods such as confirmatory factor analysis are also necessary to substantiate or clarify the present results.

III. PROBLEM STATEMENT

From the above study of various people's views, it is clearly shown that quality attributes are very important for the software development phase. The main purpose of the quality is to satisfy the users and it is precious in all products. But in most of the time the software architects are not serious with this quality attributes. So here the problem is to

1. *Identify the preeminent quality attribute to evaluate efficient software architecture which also to meet the non-functional requirements*
2. *Identify the measuring tools to estimate the non functional quality attributes.*

Proposed Solution

According to the problem statement, a view "Quality Centric Architectural Views (QCAV) " is proposed in which there is a need to identify potential issues in an architecture, its feasibility and to evaluate its ability to meet its quality requirements and to generate the centric view for designing the architecture with the improved quality attributes to satisfy the stakeholders.

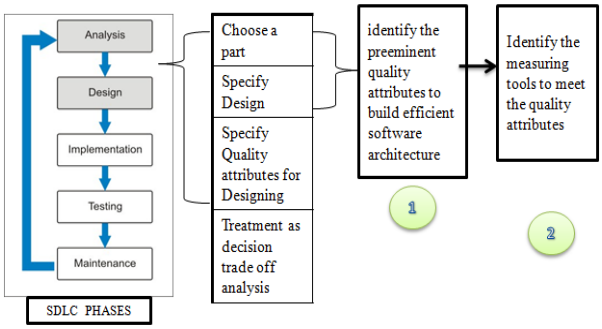

Figure1.Proposed Model

IV. CONCLUSION

Quality is the main focus of any software engineering project, because it is transparent when presented, but easily recognized in its absence [10]. Software quality is the degree to which software possesses a desired combination of attributes. A quality attributes is a property of a work product by which its quality be judged by stakeholders [10]. Without measuring, we cannot be sure of the level of quality in software architecture design. So a model is proposed to identify preeminent quality attribute and to identify the measuring tools to estimate the non functional quality attribute.

REFERENCES

[1] L. S. Maurya et al," Comparison of Software Architecture Evaluation Methods for Software Quality Attributes",Journal of Global Research in Computer Science, 1 (4), November 2010.

[2] IEEE std 610.12-1990 (n.d.). , "IEEE Standard Glossary of Software Engineering Terminology", 1990. Retrieved January 19, 2006. Web site: http://ieeexplore.ieee.org/.

[3] Frank Buschmann, David Ameller, Claudia P. Ayala, Jordi Cabot, and Xavier Franch," Architecture Quality Revisited", IEEE Software | published by the IEEE computer socie ty, 2012.

[4] Jan M.W. Kristiansen, Steria," Enhancing Defect Tracking Systems to Facilitate Software Quality Improvement", IEEE Software www.computer.org/ software, 2012.

[5] Malik Hneif and Sai Peck Lee, University of Malaya," Using Guidelines to Improve Quality in Software Nonfunctional Attributes", IEEE Software | Published by The IEEE Computer Society,2011.

[6] M. F. Bertoa, M. A. Moraga, M. C. Morcillo and C. Calero," An Analysis of the Software Components Quality in Use using Bayesian Networks", IEEE Latin America Transactions, vol. 8, no. 2, April 2010.

[7] Martin S. Feather, Steven L. Cornford, and Kenneth A. Hicks, James D. Kiper, Tim Menzies, "A Broad, Quantitative Model for Making Early Requirements Decisions", IEEE software, March/April 2008.

[8] Norman F. Schneidewind, Naval Postgraduate School," Body of Knowledge for Software Quality Measurement", IEEE research feature, 2002.

[9] Ho-Won Jung and Seung-Gweon Kim, Korea University, Chang-Shin Chung, "Telecommunications Technology Association, Measuring Software Product Quality: A Survey of ISO/IEC 9126"," IEEE software – IEEE computer society " , 2004.

[10] Leire Etxeberria MU, "Method for analysis-aided design decision making and quality attribute prediction", Embedded System for energy efficient building, 2010.

[11] Juha Savolainen, "Conflict – Centric Software Architectural Views: Exposing Trade - Offs in Quality Requirements", IEEE 2010.

[12] Alexander Egyed , "Automatically Detecting and Tracking Inconsistencies in software Design models", IEEE 2010.

[13] Frank Buschmann, "Unusable Software is Useless, Part1", IEEE 2011.

[14] Frank Buschmann,"Unusable Software is Useless, Part2", IEEE 2011.

AUTHORS

First Author – Dr.A.Chandrasekar, Professor, CSE Department, St.Joseph's College of Engineering, OMR, Chennai, India., drchandrucse@gmail.com

Second Author – Mrs.SudhaRajesh, Research Scholar, Sathyabama University, Assistant Professor, CSE Dept, SRR Engineering College OMR, Chennai, India, sudharajesh2005@gmail.com.

Third Author – Mr.P.Rajesh , Assistant Professor, IT Dept., M.S.A.J College of Engineering, OMR, Chennai, India., rajeshpadmanaban08@gmail.com.