Analysis of Software Quality Models

Mayank Dineshkumar Patel
mdpatel@clemson.edu
Clemson University
Clemson, SC, USA

Anant Sharma
anants@clemson.edu
Clemson University
Clemson, SC, USA

Yaswanth Rreddy Poreddy
yporedd@clemson.edu
Clemson University
Clemson, SC, USA

Sai Vinay Nandigam
snadig@clemson.edu
Clemson University
Clemson, SC, USA

1 INTRODUCTION

Quality is defined as a degree of superiority or inferiority of something compared to other things of the same type. In every aspect of human life, software products are being used today. For many companies evaluating and measuring the quality of a software product is a challenging task. However, software quality is a very subjective topic, with many elements influencing it both before and after creation, such as production costs, maintenance costs, etc. Was it successful in resolving the difficulties it was supposed to? Is it feasible in the current market? Is there any better software that can do the job better than this piece of software? Standalone software is not always a good thing; sometimes, companies want software that can help them with multiple issues, so we cannot simply judge whether a piece of software is good or just a poor product. The quality of the product determines a software product's success or failure. For a software product to sustain itself in today's world, quality is a key factor [22]. In order to determine this, we must first understand a few concepts. Each software has two types of requirements functional and non-functional. There are two categories of needs in software: functional and non-functional. The basic system behavior is defined by functional requirements. Calculations, data input, and business processes are frequently included in functional requirements, which define if/then actions. Non-functional requirements outline how the system should operate, rather than what it must or must not perform. They are, in essence, the system's generic properties or quality attributes. Basically, They are the general properties or quality attributes of the system. Apart from these requirements, software must be compliant with the following criteria:

(1) The final product or the service specifications should conform to the original specifications [20].
(2) The final product should satisfy all the implicit and explicit user needs [20].
(3) The final product should run smoothly without flaw [20].

Quality is the degree to which the software meets these requirements. However, there can be several requirements that need to be fulfilled, some more important than others. Due to this, we have software quality models. During software engineering development, a collection of Software Quality Models was introduced [9].

In most places, a quality model is defined as a set of characteristics like document clarity, design traceability, integrity, program reliability, test integrity, and relationships, which provide the foundation for identifying quality requirements and calculating quality. These models play a critical role in the quality assurance process of a software development life cycle and have been evolving for more than 40 years [19]. So, a quality model is used to define the quality attributes and then build and measure the resulting product's quality. Typically, every ten years, New significant Software quality model appears in the real world.

In 1976-77 US Air force Electronic System Division (ESD), the Rome Air Development Centre (RADC), and General Electric [13] sought to increase the quality of software and make it measurable [2]. Hence McCall's model was developed. This model used 55 software characteristics used to determine the quality of the software and called it factors. Later, these factors were consolidated into 11 factors in order to make the implementation simple. However, these 11 factors can be classified into three main high-level factors: the ability to undergo changes, adaptability to the new environment, product operation characteristics. As part of the ongoing development process, many new factors were added to the original model, and later on, another model was introduced in 1978, known as the Boehm model, which was defined using a different set of quality factors. McCall's model differs from Boehm's in that it focuses on the exact measurement of high-level attributes. Boehm's quality model, on the other hand, is based on a broader set of attributes [5]. There were many other software quality models introduced depending on software requirements. SPARDAT is another commercial quality model developed for the banking environment, and it had three significant factors: applicability, maintainability, and adaptability.

Now, several quality models have been developed in order to satisfy distinct requirements. They were developed using well-known models such as McCall, Boehm, FURPS, Dromey, and ISO, these are also known as the basic quality models. The new quality models developed using these models can be classified as tailored quality models. The development of tailored software quality models was achieved by conducting a comparison between select well-known models focusing on the desired characteristic. Explicit attention to characteristics of software quality can lead to significant savings in software life-cycle costs [4]. The new model would be customized based on the requirement of the intended scope.

In this paper, we intend to compare the basic software quality models. We will determine the characteristics and quality factors associated with each model by doing a literature review of similar works. Then we will determine which quality model is best for which condition using a survey questionnaire.

2 PROBLEM

One of the best ways to develop new quality models is to compare well-known existing models and selectively pick the desired characteristics. This is the reason that the comparison of various software quality models is necessary. In the present, many existing research has compared the various software quality models, but many have produced conflicting results.

The McCall, Boehm, and ISO models have been compared by three separate researchers *Hamada et al.* [11], *Rawashdeh et al.* [1], and *Haig uang et al.* [7]. All three researches focus on the same five quality factors of the quality models Integrity, Efficiency, Reusability, Changeability, and Testability. Hamada et al. has concluded that integrity quality factor is included under all the three quality models McCall, Boehm, and ISO, Rawashdeh et al. concludes that integrity is included in McCall and ISO models but not in Boehm model, and Haig uang et al. has concluded that integrity is included in McCall and Boehm model but may or may not be included in ISO model. Hamada et al. and Rawashdeh et al. also showed that the efficiency quality factor is included in all three quality models. But, Haig Huang et al. show that it is included in the Boehm and ISO models but not the McCall models. This is the kind of conflicting results that are found in many different comparisons.

This inconsistency in the quality factors of the software quality models results in an erroneous development of the required model such that the desired quality factor could either be completely absent or not be present to the required degree. Hence, to accurately analyze and compare the different quality models, we refer to previous studies done on the quality model and the studies done to compare those models.

3 BACKGROUND

In this section, a quick introduction of words that are used in software quality models is discussed. A summary of each famous software quality model is introduced. Definitions of characteristics of quality models are presented.

Software quality: The working capacity of a software product under certain circumstances [15]. It often decomposes into different quality characteristics.

Quality model: A specific collection of characteristics and correlation among them, implemented in a particular order, helps assess software quality [15].

Quality Characteristics: These are the attributes that aid in assessing the quality of software products [15].

McCall's Quality Model:

The first famous software quality assurance model was McCall's Quality model in 1977. It was founded in the US military and is also known as the General Electric model [2]. This model was developed based on software developers' and users' priorities for enhancing the quality of software products. The characteristics that McCall's model used for the operations of software are efficiency, integrity, reliability, correctness, and usability. For testing and maintenance of the software: - Maintainability, flexibility, and testability are some of the characteristics proposed by McCall's Model. Finally, Re-usability, portability, and interoperability are the features proposed by McCall's model for adapting to different environments [2]. This model is mainly for the system development process and the developers. However, the model did not examine the functionality of the software products directly[12].

Boehm's Quality Model:

After a year of the release of McCall's model, a second new model named Boehm's model was introduced in 1978 [21]. The main difference between these two quality models is: Boehm's quality model also implemented the same pre-defined attributes as McCall's model but included attributes of hardware performance. The three main characteristics that stand out in Boehm's quality model are: - Maintainability, As is Utility, and Portability. It uses a top-down approach for assessing the quality of software [12]. Boehm's model focuses on the cost-effectiveness of software maintenance and satisfying user needs. According to Boehm, bad software projects are caused by a lack of user interaction, insufficient specifications, and executive backing [10]. However, this model holds only a structure without any proposal concerning measuring the quality attributes [2].

FURPS Quality Model:

In the year 1987, Robert Grady and Hewlett Packard proposed the FURPS model. FURPS stands for Functionality, Usability, Reliability, Performance, and Supportability [6]. This model categorized its characteristics into two requirements: one is a functional requirement, and the other is a non-functional requirement [2]. Where usability, reliability, performance, and supportability come under non-functional requirements [2]. Later FURPS+ has introduced, which is a modified version of FURPS, which is among one of the most being used software models in the industry. However, attributes such as maintainability and portability are neglected to take into the software.

Dromey's Quality Model:

In 1995 Dromey introduced a quality model called Dromey's quality model. According to this model, each product's quality assessment is unique, necessitating a flexible approach for a quality model. so a flexible approach is required for a quality model. Dromey proposed three models: requirements, design, and implementation. In addition, each model is dependent on the final products at each phase of the development process [16]. The model comprehends the link between quality characteristics and sub-characteristics. Sub-characteristics can be referred to as sub-attributes, and characteristics can be referred to as attributes [17]. Dromey's model also aims to improve knowledge of the relationship between these two characteristics. Dromey says no quality process can exist without a product quality model [16]. Efficiency, reliability, maintainability, portability, usability, and functionality are among the quality attributes present in this approach. The fundamental flaw with this paradigm is that it fails to assess software quality [12].

ISO-9126 Quality Model:

In the period where there were many available quality models, confusion happened in selecting quality models, and a new standard model was required [2]. Then ISO started its work in the year 1978. In the development process, ISO began the ISO-9126 quality model in 1985 [2]. The ISO 9126 was introduced in 1991, proposed a layered model and includes quality characteristics, sub-quality characteristics, and metrics. It defines three types of quality classes: internal quality, external quality, and quality in use. The low-level characteristics are sub-divided from the high-level characteristics [12]. The following table (See Table-1) shows the decomposed sub-characteristics from high-level characteristics:

Product Quality Model	
Characteristics	**Sub-characteristics**
Functionality	• Accuracy • Security • Suitability • Interoperability
Usability	• Attractiveness • Operability • Understandability • Learnability
Reliability	• Recoverability • Fault Tolerance • Maturity
Portability	• Adaptability • Coexistence • Installability • Replaceability
Maintainability	• Stability • Testability • Analyzability • Changeability
Efficiency	• Resource Utilization • Time Behaviour

Table 1: ***Quality Model's Characteristics and Sub-Characteristics.***

ISO/IEC 25010 Quality Model:

Software Quality Model ISO 25010 is the newest of all Software Quality models published in 2011 and is considered a cornerstone of a product quality evaluation system.It takes into account the most important modern software irregularities when evaluating quality [5]. This model is referred to as the top-level international standard. Software quality is divided into six characteristics by the ISO/IEC 9126 software quality model, further subdivided into sub-characteristics (**See Figure 1**). They did, however, add two new components to ISO 25010, namely Security and Compatibility. The distinction between ISO 9126 and ISO 25010 is how they recognize and define the non-functional features of software quality criteria.

Evaluating Software Quality, in general, is hard, as each metric has their own interpretation, range of values, scale or some other measurement methods [23]. And it's not easy to combine all the Software Quality metrics and creating a single quality score as it limits the evaluation of specific quality aspects. As we already know that Software Quality can be decomposed into different characteristics and sub-characteristics(hierarchically) and these can be used to assess external and internal metrics of a software. Ulan et al. [23] have taken an approach that uses distributions of all the quality metrics(ISO 25010 model), and they used Visualization of these distributions which is later used by users to examine and compare the quality metrics of software systems and their artifacts, as well as spot trends, correlations, and anomalies, using visualizations of these distributions. Furthermore, because our visualization technique allows rich interactivity for visual queries to the quality models' multivariate data, it is feasible to find common traits and defects. Their results gives us strong inference that the characteristics of ISO 25010 model should be considered as the fundamental model for the future models.

3.1 Evaluation

The evaluation of quality models can be done in different ways. Some of them are by Mapping and comparing the quality attributes of quality models [11] [15] [19], internet-based surveys [14], interviews [3], etc.

For example, Systematic Mapping [15] identifies, structures, and classifies the software quality models based on four research questions. Because each model has a limited number of traits and metrics, comparing them did not require a sophisticated method. In paper [11], The technique employed is a basic parsing and elimination procedure to decide which traits and metrics are common and which are not.

3.2 Cost of Quality

The significant growth of the software industry in recent years has brought attention to long-standing issues with software development, such as uncontrollable costs, missed deadlines, and unpredictably high quality. Software companies must provide high-quality solutions on schedule and on a budget to be competitive. On the other hand, software managers may skip quality improvement techniques like design reviews and code inspections to get their products to market faster, believing that these processes merely add time to the development cycle. Indeed, in the software development sector, the economics of enhancing quality is not well known. Even outside of the software development context, there is considerable ambiguity concerning the business value of quality. It is sometimes assumed that high levels of quality are uneconomical and that quality must be sacrificed to attain other goals, such as shorter development cycles. A study on the Software Engineering Institute's Capability Maturity Model (CMMTM) adoption, for example, quotes a software manager as saying, "I'd rather have it wrong than late. We can always fix it later" [18]. Consumer satisfaction and the cost of development are two essential factors of final product quality. Finding and fixing faults and inadequacies as near to the source as feasible, or better yet, preventing them from occurring in the first place, is the key strategy for lowering the cost of software quality.

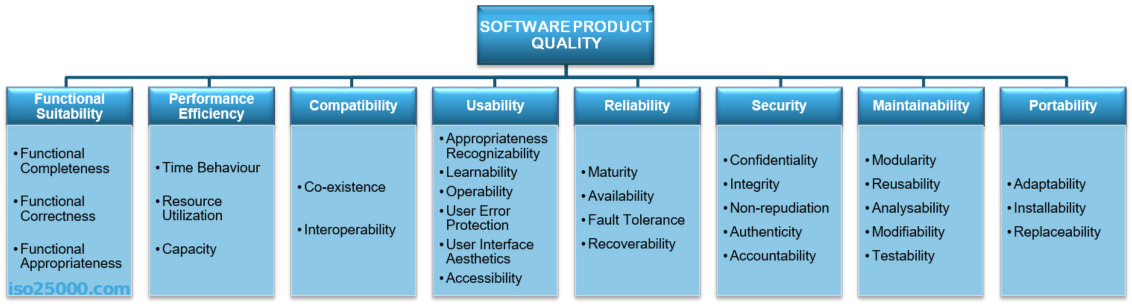

Figure 1: ***Software Product Quality***

3.3 Previous Studies:

Several studies have been conducted to compare various well-known quality models. Each has used different approaches and produced a different set of results.

Hemayat et al [20] have done a comprehensive analysis of many software quality models, and in this paper, they have chosen 19 different quality models. They have conducted a literature survey to study and classify those models into different categories. They have produced tables showing different quality factors associated with each model and several corollaries related to quality models.

AL-Badareen et al. [2] have taken a mathematical approach to compare the quality models. Their main intention was to formalize an approach towards quality models comparison, which they have demonstrated in their paper.

Galli et al. [8] work identifies entire software product quality models that have been published since 2000, as well as attempting to quantify the importance of each model by introducing indicators relevant to the scientific and industrial communities. There are 23 software programs that have been identified in terms of publication intensity, publication range, and average quality score, product quality model classes differ significantly. The Google Relative Search Index 12-month average and the relevancy score the outcomes provide a framework for deciding on the best option. The findings provide a basis for deciding which software product quality model to utilize and an extension if recently discovered quality features need to be linked to a broader context.

4 METHODOLOGY

4.1 Objective

When analyzing the software quality models to find the best software quality model, we decided to do a literature review and survey. Our primary focus was on the literature review. We used a survey study to support our reasoning on the literature review.

This study used a literature review to support the survey results to decide the current state of software quality models and their characteristics in use. The goal of the literature review and survey is to examine software quality models and their attributes. The literature research clarifies the application of a specific quality model as well as particular characteristics such as when and where it should be deployed.

4.2 Research Questions

To analyze the software quality models, we focused on some significant areas of study in quality models. These areas will help analyze the best software quality model.
Here is the list of the study concerns:

- Which quality model should be adopted in particular situations?
- Which Quality Characteristics are Important Regardless of Quality Models?
- What are the issues that quality models face in today's world?
- What is the solution to such problems?

4.3 Methodology Design

This study is split up into two sections:
Part-1: Literature Review

In part one, we looked at a lot of publications about software quality models. In order to discover solutions to the questions above, we searched the IEEE (Institute of Electrical and Electronics Engineers) Explore, ACM Digital Library, Google Scholar, and Research Gate websites for articles relating to software quality models.

Part- 2: Survey

In the second part, we conducted a questionnaire-based survey. The questions in the questionnaire were based on our study. The population of this study consists of working professionals employed at software development organizations and students who know Software Quality Models. In this survey, participants were asked to answer the questions in the form of multiple-choice and comments. The respondents' experience is also a critical factor in ensuring that the respondents can answer the questionnaire meaningfully.

4.4 Data Collection

Data collection is a process that helps figure out and analyze better software quality models. In this study, one of the methods is a survey where we ask questions to an individual who has any knowledge about software quality models or who has real-life experience in using software quality models.

Collecting data from a user can be biased, and not every user has the required knowledge to take part in this Analysis of Software Quality Model research study. To analyze the best software quality model, we need the data from the user who knows and understands the use of software quality models or has some knowledge. One of the questions assisted us in determining which Software Quality Models are employed in the actual world. We gave the user six quality models as options to pick: Mccall's Quality Model, Boehm's Quality Model, FURPS Quality Model, Dromey's Quality Model, ISO 9126 Quality Model, and ISO/IEC 25010 Quality Model. These are the famous and primarily usable software quality models in the present day. Our study used these data to find which software quality model was the most used and led us to the reason behind why they used the most.

Users' opinions are not always accurate, but the majority always wins regarding opinion-based questions. For example, What Software Quality Models do you apply to ensure that your products are better? This was a fundamental opinion question to figure out which of the Software Quality Model is best for the product to be better. We used this data collection to determine which is the best Software quality Model in the user's opinion. Regardless of whether Software Quality Models are used for a sole product or multiple products, adaptation is an essential effort.

4.5 Methods

In the literature review, the quality models are evaluated in terms of their features, highlighting the benefits and drawbacks of each model.

To complete the survey, one must answer both personal and general questions. In this research, the respondents' responses were examined, and helpful information was presented. The survey took about 15-20 minutes to complete on average.

4.6 Threats to Validity

Regulations are expected in the field of research. We looked at many research papers and prepared a questionnaire to supplement the literature review to collect reliable answers. The survey included both seasoned professionals and students with a background in computer science. Because we were conducting a survey, there was no way for participants to get to know each other. We also made certain that the responses remained confidential. Because the participants may have been concerned about the outcomes, they may have attempted to manage them. The research was done in a broad sense, with people from various countries and companies participating.

5 RESULTS

5.1 Literature review

When we do a literature review of papers related to study in Software Quality Models, each paper has different conclusions but similar information of Software Quality Models. All the quality factors that are considered in a quality model can be distributed into two categories, static elements, which measures a software's conformity to the original requirements, ease of use, basically whatever is expected of software from a user's perspective, and the dynamic factors which measure the software's maintainability, portability, performance under stress, etc. which are a software's requirements from the future perspective or the non-specified requirements. We studied and reviewed many papers on Software Quality Models. From those papers, we draw the following analysis conclusion:

- **McCall Model:** Out of the 31 quality factors, the McCall model consists of 11. The main idea behind this model is about the relationship between the external quality factors of the product and the internal quality factors. This model was developed to prioritize the factors that make the owner's job easier, like maintainability. Hence, this model focuses more on the dynamic aspects of the software rather than the static factors. This means that the software with better dynamic factors will be ranked better according to this quality model. This model does give enough consideration to the static aspects and more so that user satisfaction is not even considered an important factor according to this model, nor is the end product's conformity to the original specifications. Hence, this model does not provide a very accurate representation of the quality of the software.
  The factors it consists of are Performance efficiency, Usability, Reliability, Maintainability, portability, Testability, reusability, correctness, integrity, interoperability, and flexibility.
- **Boehm Model:** This is another widely used model, much like the McCall model, as both follow the same hierarchal structure. This model also gives more weightage to the dynamic factors than the static ones. The only static factor it requires is the overall completeness of the end product, which is very basic. Hence, even though the quality is a relative concept, this model cannot provide a very accurate quality comparison.
  Focuses on maintainability rather than software functionality and compliance with specifications, it only considers seven quality factors: Performance efficiency, reliability, portability, testability, understandability, Human engineering, and Modifiability. It does not consider customer satisfaction.
- **FURPS Model:** FURPS model is an acronym for Functionality, Usability, Reliability, Performance, and Supportability. This model tried to cover both the static and dynamic factors but is more concerned with the static requirements of the software.
  FURPS model supports eight quality factors: Functionality, performance efficiency, Usability, reliability, supportability, business implementation, interface, and Physicists.
- **Dromey model:** This model is an extension of the above ISO9126 model. This model considers both the dynamic and static factors. Still, it also considers the coding standards, source code quality, and development life cycle efficiency to determine the quality of the software.
  The quality factors that it considers are functionality, reliability, usability, efficiency, maintainability, portability, and reusability.
- **ISO9126 model:** This model pays attention to maintainability and conformance to requirements, considering the static aspects of the software. It also pays attention to factors such as compliance and usability, which indirectly measure user's satisfaction, so it pays attention to the dynamic factors as well.

However, although this model considers both factors, it does so barely. Also, it lacks code and lifecycle-specific quality factors. Hence, although this model provides a high-level quality measure, it lacks an in-depth analysis.
The quality factors it considers are functionality, reliability, usability, efficiency, maintainability, and portability.

- **ISO25010 model:** This model covers both dynamic and static aspects of quality. It also considers customer satisfaction and overall development cost as different quality factors, and it also finds the software security and performance of the software as a quality factor. Overall, this model is quite thorough while determining the quality of the software.

- The most important contribution of a McCall quality model for software quality is establishing the association between software quality characteristics. However, this model neglected functionality, a crucial component of software quality. What a software product can accomplish for a user is functionality. A feature like functionality is important to fulfill the needs of the user.

6 DISCUSSION

6.1 Quantitative Analysis

The responses from the survey are described in this section. They are grouped in the same way as the questionnaire is. Each topic is initially explained in broad terms, followed by descriptive statistics. This involves the significance of various quality models and the application of quality models to particular objectives. Furthermore, prospective areas for improvement are explored.

To answer Research Question 2, which Quality Characteristics are Important Regardless of Quality Models? In a multiple-choice style, the participants were given a list of attributes from which to choose. Functionality is at the top of the list, followed by reliability, performance efficiency, and usability, as seen in **figure 2**.

When it came to quality models, McCall's and Boehm's quality models overlooked functionality. Quality models such as Furps, Dromey, ISO-9126, and ISO-2510, on the other hand, emphasized functionality. Each quality model emphasized in this study contained the quality characteristic reliability as a significant element. In addition, 19 of the 28 respondents considered reliability an essential factor in ensuring software quality.

Participants were asked to pick one or more quality models from the alternatives they are familiar with or have worked in the past. The question was, "Please choose software quality models that you are familiar with or have worked with." And the options were McCall's quality model, Boehm's quality model, Furps quality model, Dormey's quality model, ISO-9126 quality model, ISO-25010 quality model, and None of the Above. Those whose quality model was not listed in the options were asked to respond to the following question.

From the survey results, the ISO-9126 quality model was picked by most of the participants, and also that Dormey's quality model is no longer in use.

Participants were asked, "What variables do you think influences software quality the most?" to answer research questions 3&4. The majority of the participants agreed that the education, experience, and training of development staff are essential elements that influence software quality. The proper use of software quality models may not occur if there is no suitable education and training. As a result, it could be a problem for software quality models. The aspects of the development process were the second most popular choice. If a software quality model lacks certain vital qualities, it may pose a quality concern for the software product. The below is the **figure3** where you can see the pie chart of the various responses from participants.

The questionnaire also had a question, "which quality model do you apply to ensure that your products are better?" The ISO-9126 and ISO-25010 quality models were the most common responses to this question. The respondents may have responded to this question based on their job experience or knowledge of software quality models. ISO-9126 and ISO-20510 have practically all of the essential characteristics, according to the study.

Most of the participants (89.3%) agrees with the following statement- " Regardless of whether quality models are used for a single product or a set of products, adaptation is an essential effort." From these results, we may infer that regardless of whether quality models are used for a single product or a set of products, adaptation is an important activity when using quality models. It might also be a sign that current standards are insufficient for practical needs. The **Figure4** shows the results of responses from the participants.

"How important are the following quality attributes for your products?" was another question about quality attributes. To rate the relevance of significant quality attributes, respondents were given options such as Extremely important, Important, Neutral, Somewhat Important, and Not important. For functioning and reliability features, the majority of participants rated them as extremely significant. The majority of participants rated performance efficiency as an essential attribute. Except for the first two quality models, McCall's and Boehm's, every quality model has a functionality attribute, indicating that functionality is an important character. When it came to the reliability character, every well-known quality model found it helpful. A few questions were asked to determine which of the given characteristics were most important for evaluating and establishing the quality of a Software. Every time Functionality comes top, followed by Reliability and Usability.

In reference to the above question, one more following question was asked- "If you were in charge of establishing a quality improvement plan, which three characteristics would you prioritize?" and as expected, most of the responses were redundant, and Functionality came top followed by Reliability and Usability. And every time, the least number of votes were given to Portability, Testability, and Reusability, which are also significant for a viable software. The below **figure 5** shows the responses of the participants.

Significance of Portability: Even with small software teams, environments increase quickly: developers need to run the application locally, quality engineers on a test environment, and operators on staging and production settings. The cost of the developing, testing, and production lifecycle is proportional to the software portability. As new versions progress through the lifecycle, software that requires a lot of environment-related configuration and tweaking will cost time and effort. For everyone involved in migrating new versions of software between environments, portability saves time and mental effort.

Characteristics	McCall	Boehms	FURPS	Dromey	ISO-9126	ISO-25010
Accuracy					×	×
Adaptability			×			×
Analyzability					×	×
Attractiveness					×	×
Changeability					×	×
Correctness	×					×
Efficiency	×	×		×	×	×
Flexibility	×					
Functionality			×	×	×	×
Human Engineering		×				
Installability					×	×
Integrity	×					×
Interoperability	×					×
Maintainability	×			×	×	×
Maturity					×	×
Modifiability						×
Operability					×	×
Performance			×		×	×
Portability	×	×		×	×	×
Reliability	×	×	×	×	×	×
Resource Utilization					×	×
Reusability	×			×		×
Stability					×	×
Suitability					×	×
Supportability			×		×	×
Testability	×	×			×	×
Transferability						×
Understandability		×			×	×
Usability	×		×	×	×	×

Table 2: Comparison of Quality Models by Characteristics

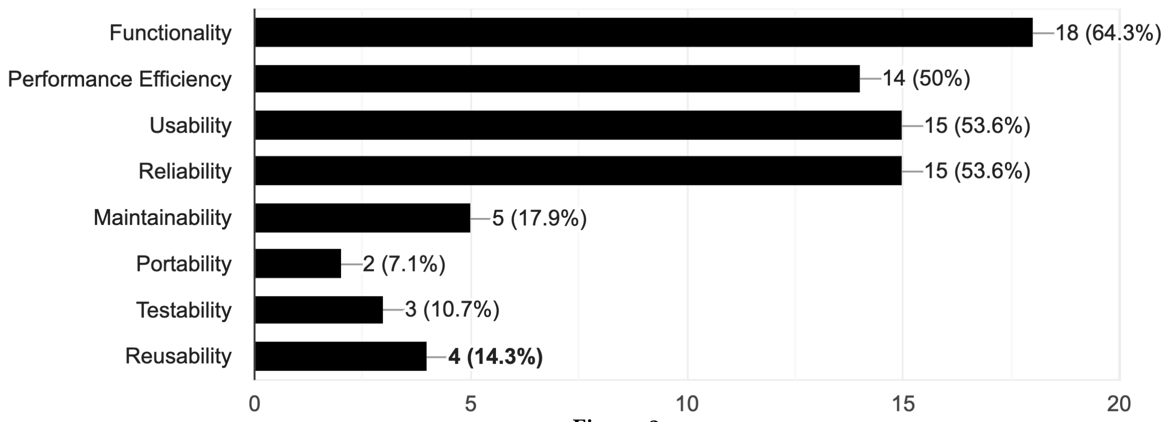

Figure 2

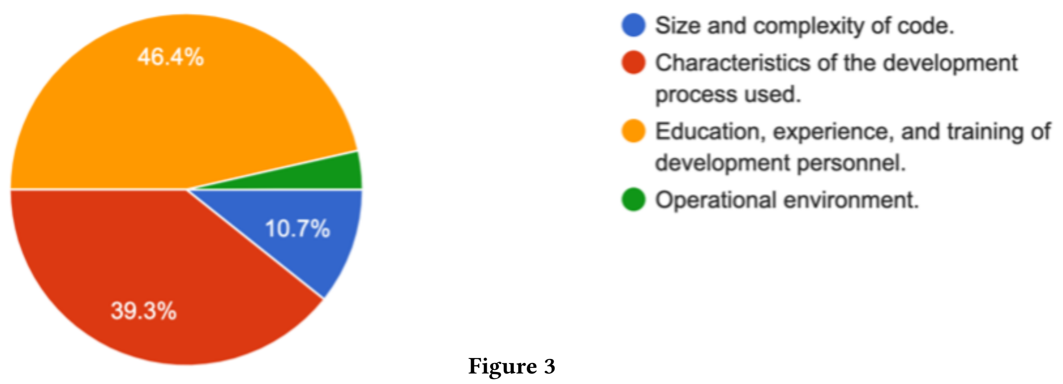

Figure 3

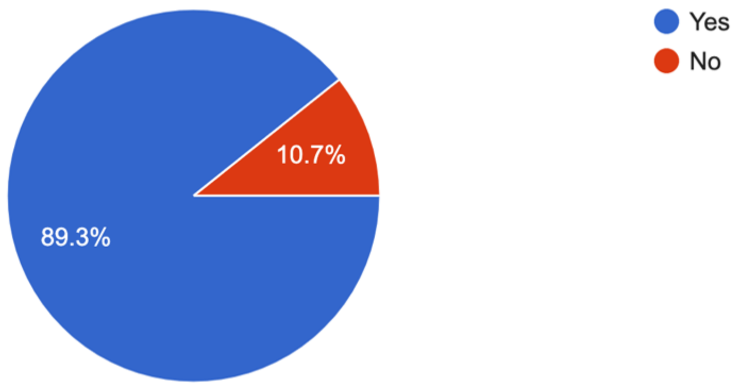

Figure 4

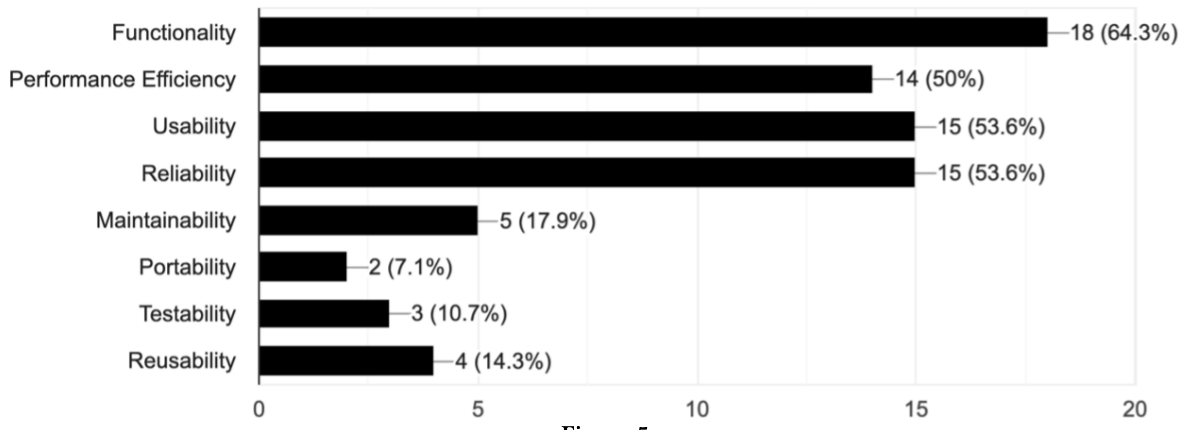

Figure 5

Is testability not important? Because the application is frequently created for the user rather than the QA team, testability is often disregarded in favor of other objectives such as usability and functionality throughout the design and development phases. Creating highly tested software, on the other hand, will ultimately help the user. Deliverability is influenced by testability. When it's easier for testers to find issues, it's debugged faster, and the software reaches the user faster and with fewer bugs. Furthermore, testability will benefit product and development teams. Those teams will gain from faster feedback as a result of increased testability, allowing for more frequent fixes and modifications.

Reusability: The idea of reusability is commonly utilized to save software development costs, effort, and time. Reusability also improves the software products' maintainability, portability, and reliability, one of the most sought quality factors. The reusable software components can be tested in multiple different systems.

To determine the significance of functional and reliable characteristics. A question was posed in the survey: "Most of the Software Quality Models have functionality and reliability factors. Do you agree that these two factors are important? Why?" On the statement, the respondent was requested to add a comment. "I do believe these two factors are important. Functionality is arguably the most important. Software needs to work as it is intended to be useful. Reliability is also advantageous to clients, as the software needs to be available and work appropriately often. However, reliability may be slightly less important with online features, as many accept downtime and scheduled maintenance. We can deduct from this comment that the functionality characteristic is more essential than reliability in determining software quality. Another interesting comment made on the statement was, "I believe Functionality solely defines the use case of product and reliability helps build the trust of product." From this comment, we can understand that both are equally important.

A study on the Software Engineering Institute's Capability Maturity Model (CMMTM) adoption, for example, quotes a software manager as saying, "I'd rather have it wrong than late. We can always fix it later". Do you agree or disagree with the above statement? Why/why not? The survey asked this question to know the general opinion on the quote. Some of the comments commented by the participants were:

- "It really depends. To me, quality is one of the biggest factors for a product. There is a difference between buggy code that can be patched and broken code that isn't usable. One is more acceptable than the other. I do understand that when contracts and marketing budgets are involved, sometimes you have to submit what you have. In general, a broken product shouldn't be given to the customer."
- "It depends on the situation. If a customer is happy with a first iteration being more of a concept than an actual reliable solution, than getting it to them in a timely manner is more important than having all functionality working exceptionally well."
- "It may be naive, but I believe writing bad code or using bad practices with the intent of fixing it later just invites problems. These issues may never be fixed and having to weed through spaghetti code to add a new feature is never fun. I admit, sometimes you have to get the product deployed, but I don't think this practice should be encouraged."
- "I would disagree with this. I believe that having a wrong output is worst as it will take more time in correction. A little late increment deliverable is better as it puts you in the right track and does not confuse the end user."
- "No the quality vs speed discussion should be a feature of the product specifications. Sometimes speed is more important than perfection, but sometimes perfection is a functional requirement. This isn't typically a decision made by the swe, but by the organization requesting the product."

Some of the participants agreed with the quote, but most of them disagreed with it. We can infer that we cannot always fix the problem later. Also, it may not be possible to fix the problem earlier. It depends on the software development stage and the requirements that the software product and the customer needs.

7 CONCLUSION

This paper conducted a survey and examined research papers relating to software quality models. The literature review and the survey results made us identify the quality models and characteristics in practice. This study explored the significance of quality models such as ISO-9126 and ISO-25010. This paper explained each quality model and the characteristics that each quality model possesses. The importance of characteristics over quality models was also addressed and the essential attributes required to ensure the quality of a software product. This study helps researchers and working professionals to understand the importance of software quality models and their characteristics.

8 FUTURE WORK

In the future, besides the quality models and their characteristics, it would be helpful to find out the other feasible techniques for ensuring the quality of a software product.

REFERENCES

[1] Rawashdeh Adnan and Matalkah Bassem. 2006. A New Software Quality Model for Evaluating COTS Components. *Journal of Computer Science* 2 (04 2006). https://doi.org/10.3844/jcssp.2006.373.381

[2] Anas Bassam AL-Badareen, Mohd Hasan Selamat, Marzanah A. Jabar, Jamilah Din, and Sherzod Turaev. 2011. Software Quality Models: A Comparative Study. ReaearchGate. https://www.researchgate.net/publication/283103329_Evolution_of_software_quality_models_Green_and_reliability_issues

[3] Claudia Ayala, Øyvind Hauge, Reidar Conradi, Xavier Franch, and Jingyue Li. 2011. Selection of third party software in Off-The-Shelf-based software development—An interview study with industrial practitioners. *Journal of Systems and Software* 84, 4 (2011), 620–637. https://doi.org/10.1016/j.jss.2010.10.019 The Ninth International Conference on Quality Software.

[4] B. W. Boehm, J. R. Brown, and M. Lipow. 1976. Quantitative Evaluation of Software Quality. In *Proceedings of the 2nd International Conference on Software Engineering* (San Francisco, California, USA) *(ICSE '76)*. IEEE Computer Society Press, Washington, DC, USA, 592–605. chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/viewer.html?pdfurl=http%3A%2F%2Fwww.ptidej.net%2Fseminars%2F2013%2F131014%2520-%2520Yann-Gael%2520Gueheneuc%2520-%2520Quality%2C%2520Patterns%2C%2520and%2520Multi-language%2520Systems%2FPaper%2520-%2520Boehm%2520et%2520al.pdf&clen=1436198

[5] danf7861. [n. d.]. *Boehm's Software Quality Model*. https://www.geeksforgeeks.org/boehms-software-quality-model/

[6] Tharashasank Davuluru, Jayapal Medida, and V.S.K Reddy. 2014. A study of software quality models. In *2014 International Conference on Advances in Engineering Technology Research (ICAETR - 2014)*. 1–8. https://doi.org/10.1109/ICAETR.2014.7012958

[7] Haiguang Fang. 2008. Modeling and Analysis for Educational Software Quality Hierarchy Triangle, Vol. 31. 14 – 18. https://doi.org/10.1109/ICWL.2008.19

[8] T. Galli, F. Chiclana, and F Siewe. 2020. Software Product Quality Models, Developments, Trends, and Evaluation. (2020), 154. https://doi.org/10.1007/s42979-020-00140-z

[9] Oleksandr Gordieiev, Vyacheslav Kharchenko, and Mario Fusani. 2015. Evoluation of software quality models: Green and reliability issues. ReaearchGate. https://www.researchgate.net/publication/283103329_Evolution_of_software_quality_models_Green_and_reliability_issues

[10] Narasimhaiah Gorla and Shang-Che Lin. 2010. Determinants of software quality: A survey of information systems project managers. *Information and Software Technology* 52, 6 (2010), 602–610. https://doi.org/10.1016/j.infsof.2009.11.012

[11] Ahmed Hamada, Mohamed Moustafa, and Hussein Shaheen. 2008. Software Quality model Analysis Program. *2008 International Conference on Computer Engineering and Systems, ICCES 2008*, 296–300. https://doi.org/10.1109/ICCES.2008.4773015

[12] Klaus Lochmann, Jasmin Ramadani, and Stefan Wagner. 2013. Are Comprehensive Quality Models Necessary for Evaluating Software Quality?. In *Proceedings of the 9th International Conference on Predictive Models in Software Engineering* (Baltimore, Maryland, USA) *(PROMISE '13)*. Association for Computing Machinery, New York, NY, USA, Article 3, 9 pages. https://doi.org/10.1145/2499393.2499404

[13] Jim. A. McCall, Paul. K. Richard, and Gene. F. Walters. 1977. *Factors in Software Quality: Concept and Definitions of Software Quality*. Defence Technical Information Center.

[14] Vlastimir Nikolić, Jelena Kaljevic, Srđan Jović, Dalibor Petković, Miloš Milovančević, Ljubomir Dimitrov, and Pancho Dachkinov. 2018. Survey of quality models of e-learning systems. *Physica A: Statistical Mechanics and its Applications* 511 (2018), 324–330. https://doi.org/10.1016/j.physa.2018.07.058

[15] Padmalata Nistala, Kesav Vithal Nori, and Raghu Reddy. 2019. Software Quality Models: A Systematic Mapping Study. In *2019 IEEE/ACM International Conference on Software and System Processes (ICSSP)*. 125–134. https://doi.org/10.1109/ICSSP.2019.00025

[16] M. Ortega, M. Pérez, and T Rojas. 2003. Construction of a Systemic Quality Model for Evaluating a Software Product. In *Software Quality Journal 11*. SpringerLink, 219-242 pages. https://doi.org/10.1023/A:1025166710988

[17] José P. Miguel, David Mauricio, and Glen Rodríguez. 2014. A Review of Software Quality Models for the Evaluation of Software Products. *International Journal of Software Engineering & Applications* 5, 6 (Nov 2014), 31–53. https://doi.org/10.5121/ijsea.2014.5603

[18] Mark Paulk, Bill Curtis, Mary Chrissis, and Charles Weber. 1993. *Capability Maturity Model for Software, Version 1.1.*

[19] Muhammad Qasim Riaz1 and Zeeshan Asif. 2017. Review and Comparison of Different Software Quality Models. (2017), 15. https://www.ukessays.com/essays/computer-science/review-comparison-software-quality-6861.php

[20] M. Sadeghzadeh Hemayati and H. Rashidi. 2017. Software Quality Models: A Comprehensive Review and Analysis. *Journal of Electrical and Computer Engineering Innovations (JECEI)* 6, 1 (2017), 59–76. https://doi.org/10.22061/jecei.2019.1076

[21] Aman Sharma, Arvind Kalia, and Hardeep Singh. 2012. An Analysis of Optimum Software Quality Factors. *IOSR Journal of Engineering* 02 (04 2012). https://doi.org/10.9790/3021-0204663669

[22] J. Tian. 2004. Quality-evaluation models and measurements. *IEEE Software* 21, 3 (2004), 84–91. https://doi.org/10.1109/MS.2004.1293078

[23] Maria Ulan, Sebastian Hönel, Rafael M. Martins, Morgan Ericsson, Welf Löwe, Anna Wingkvist, and Andreas Kerren. 2018. Quality Models Inside Out: Interactive Visualization of Software Metrics by Means of Joint Probabilities. In *2018 IEEE Working Conference on Software Visualization (VISSOFT)*. 65–75. https://doi.org/10.1109/VISSOFT.2018.00015