REPORT

Sabaragamuwa University of Sri Lanka

Faculty of Computing

Department of Software Engineering

SE5103 – Product Assurance

Name : **Nilakshi Samarasekara**

Reg. No : **20APSE4877**

Academic Year : **2020/2021**

Degree Program : **Software Engineering**

Date : **11.08.2024**

Contents

1. Introduction 2

2. McCall's Quality Model 2

2.1. History and Development 2

2.2 Key Components and Characteristics 2

2.3 Findings from Reviewed Articles 4

2.4 Applications and Relevance 5

3. Boehm's Quality Model 5

3.1 History and Development 5

3.2 Key Components and Characteristics 6

3.3 Findings from Reviewed Articles 7

3.4 Applications and Relevance 8

4. Comparison and Contrast 8

4.1 Components and Approaches 8

4.2 Strengths and Weaknesses 9

5. Conclusion 10

5.1 Summary 10

5.2 Importance of Understanding These Models for Software Quality Assurance 10

6. References 11

Literature Review on McCall's and Boehm's Models

1. Introduction

Software quality is integral to the development and success of any software product, as it ensures the software meets user expectations, adheres to required standards, and performs reliably across various environments. As software systems have become more complex, the need for systematic approaches to evaluating and improving software quality has grown. Over the years, several models have been introduced to address this need, offering frameworks to understand, evaluate, and enhance various aspects of software quality.

Among these models, McCall's Quality Model and Boehm's Quality Model stand out as two of the most influential frameworks in the field. Both models were developed in the late 1970s, a period marked by rapid growth in the software industry, when ensuring software reliability, maintainability, and overall quality became critical concerns.
McCall's Quality Model was one of the first attempts to formalize software quality evaluation. It sought to bridge the gap between users and developers by identifying key quality factors that reflect both user needs and developer priorities.
Boehm's Quality Model introduced a more hierarchical approach to software quality, considering the software's utility as the primary measure of quality and breaking it down into multiple characteristics.

2. McCall's Quality Model

2.1. History and Development

McCall's Quality Model was developed by Jim McCall in 1977, during a time when software systems were becoming increasingly complex, and the need for reliable and maintainable software was more pressing than ever. The U.S. Air Force, facing challenges in software reliability and maintainability, commissioned the development of this model to establish clear guidelines for evaluating software quality. McCall's model was groundbreaking in its attempt to formalize the concept of software quality, focusing on bridging the communication gap between users and developers.

The model's development was driven by the recognition that software quality is multi-dimensional and cannot be assessed by a single metric. Instead, it needed a structured approach that considers various quality attributes that are relevant to both users and developers. By translating user needs into technical requirements that developers could understand and implement, McCall's model laid the foundation for a more systematic approach to software quality.

2.2 Key Components and Characteristics

McCall's Quality Model is organized around three primary aspects of software quality, each of which is further divided into specific quality factors:

1. Product Operation:
    - Correctness: Measures the software's ability to perform its intended functions accurately.
    - Reliability: Assesses the software's consistency in maintaining performance under varying conditions over time.
    - Efficiency: Evaluates the software's ability to achieve its objectives using minimal resources.
    - Integrity: Concerns the software's ability to protect against unauthorized access and data loss.
    - Usability: Assesses how easy it is for users to learn and operate the software.

2. Product Revision:
    - Maintainability: Reflects how easily the software can be modified or corrected after delivery.
    - Flexibility: Measures the software's ability to accommodate changes in requirements or environments.
    - Testability: Assesses how easily the software can be tested to ensure it functions correctly.

3. Product Transition:
    - Portability: Evaluates the ease with which the software can be transferred to different hardware or software environments.
    - Reusability: Measures the software's ability to be used in different applications or environments.
    - Interoperability: Assesses the software's ability to work with other systems or components.

These aspects encompass 11 quality factors, each representing a specific dimension of software quality. Additionally, the model includes 23 quality criteria that offer more detailed metrics for evaluating these factors. The hierarchical structure of McCall's model allows for a comprehensive assessment of software quality, taking into account both the user's perspective and the developer's technical requirements.

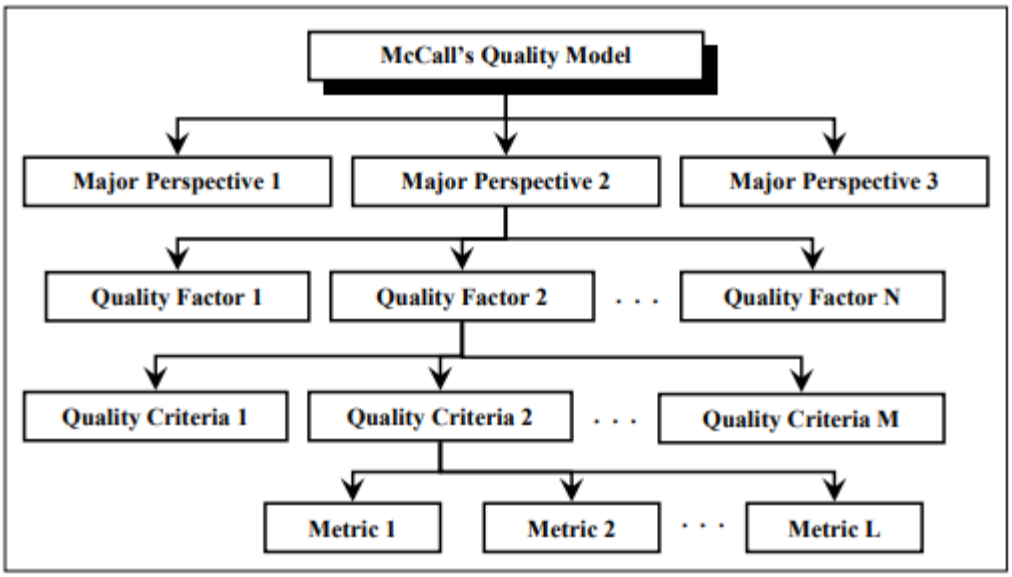

Figure 01-The structure of McCall's quality model

The following diagram represents the structure of **McCall's Quality Model**. It shows the hierarchical organization of the model, starting from high-level characteristics down to specific metrics.

2.3 Findings from Reviewed Articles

Various studies and articles have analyzed McCall's Quality Model, highlighting both its strengths and limitations.

Strengths:

- The model provides a structured framework for evaluating multiple dimensions of software quality, making it applicable to a wide range of software systems.
- It is particularly effective in ensuring that software meets user expectations by translating these into technical requirements.
- The hierarchical structure allows for both high-level and detailed evaluations, enabling a more nuanced understanding of software quality.

Limitations:

- Some critiques point out that McCall's model may be too rigid, particularly in the context of modern, agile development practices that require more flexibility.
- The model's focus on traditional software systems may limit its applicability to newer types of software, such as cloud-based applications or mobile apps.
- The emphasis on certain quality factors, like maintainability and testability, may not fully capture the complexities of contemporary software systems.

2.4 Applications and Relevance

Despite being developed over four decades ago, McCall's Quality Model remains relevant in today's software engineering landscape. Its structured approach to software quality is particularly useful in environments where clear communication between users and developers is essential.

- Application in Modern Software Development:
    - McCall's model is often used in the early stages of software development to define quality requirements and guide the design process. By identifying potential quality issues early on, it reduces the risk of costly revisions later in the development cycle.
    - The model's focus on usability, reliability, and efficiency aligns well with the user-centric nature of today's software market, where these factors significantly impact a product's success.
    - In regulated industries, such as aerospace, defense, and healthcare, where software quality is critical, McCall's model continues to be a valuable tool for ensuring compliance with stringent quality standards.

- Relevance in Agile and DevOps:
    - Although McCall's model was developed in a traditional waterfall context, its principles can be adapted to agile and DevOps methodologies. For example, its emphasis on maintainability and testability aligns well with continuous integration and continuous delivery practices in DevOps.
    - The model's focus on product transition aspects, such as portability and interoperability, is increasingly important in today's diverse and interconnected software ecosystems.

In conclusion, McCall's Quality Model has had a lasting impact on the field of software engineering. Its structured approach to evaluating software quality remains relevant, providing a solid foundation for both traditional and modern software development practices.

3. Boehm's Quality Model

3.1 History and Development

Boehm's Quality Model was introduced by Barry Boehm in 1978, a year after McCall's model. Boehm's model responded to the need for a more comprehensive approach to software quality that could address both the product itself and the processes used to develop it. The model introduced the concept of "general utility," which considers the software's overall usefulness to its intended users as the primary measure of quality.

Boehm's model was developed with a focus on providing a detailed framework for understanding and evaluating software quality across multiple dimensions. It was designed to be flexible and adaptable, allowing for a more granular analysis of quality at various levels of abstraction.

3.2 Key Components and Characteristics

Boehm's Quality Model employs a hierarchical structure that breaks down software quality into three levels of characteristics:

1. High-Level Characteristics:
   - General Utility: Measures the overall usefulness of the software to its intended users.
   - As-Is Utility: Assesses the software's ability to meet current requirements and provide value to users.
   - Maintainability: Reflects the ease with which the software can be modified and updated to meet changing requirements.

2. Intermediate-Level Characteristics:
   - Portability: Evaluates the software's ability to operate in different environments.
   - Reliability: Assesses the software's ability to maintain performance under varying conditions.
   - Efficiency: Measures the software's resource utilization.
   - Human Engineering: Concerns the user interface and the ease with which users can interact with the software.

3. Lowest-Level Characteristics:
- Testability: Reflects how easily the software can be tested.
- Understandability: Measures how easy it is for users and developers to comprehend the software's structure and behavior.
- Modifiability: Assesses the software's ability to accommodate changes.

Each level of the hierarchy is linked to specific metrics that provide measurable criteria for evaluating software quality. This structure allows for a detailed examination of both the product and the processes involved in its creation.

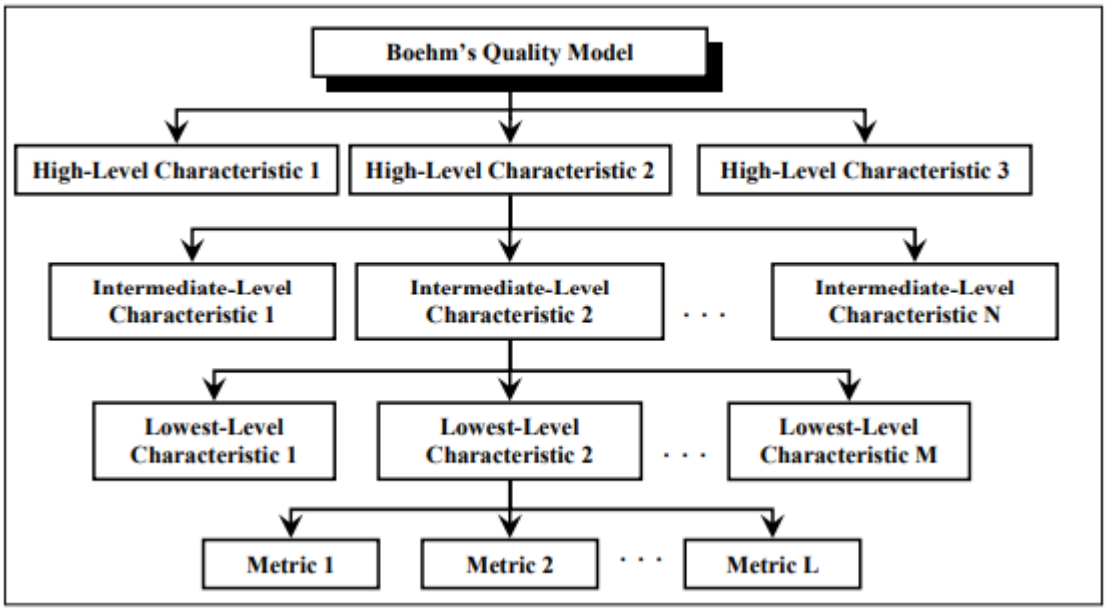

Figure 02-The structure of Boehm's quality model

The diagram below illustrates **Boehm's Quality Model**. It shows the hierarchical structure from high-level characteristics to specific metrics used to evaluate the software.

3.3 Findings from Reviewed Articles

Boehm's Quality Model is widely recognized for its comprehensive and flexible nature. Key findings from studies and articles include:

- Strengths:
  - The model provides thorough coverage of software quality, encompassing a wide range of attributes from high-level utility to specific engineering considerations.
  - Its hierarchical structure allows for adaptability, making it suitable for different types of projects and enabling a more nuanced understanding of quality.
  - By considering both product and process quality, the model offers a more holistic view of software quality.

- Weaknesses:
  - The model's complexity can make it challenging to implement, particularly in smaller projects or teams with limited resources.

- Tailoring the model to specific projects may require significant time and effort, which can be impractical in fast-paced or resource-constrained environments.

3.4 Applications and Relevance

Boehm's Quality Model is particularly well-suited for complex, large-scale projects where a detailed understanding of software quality is necessary. Its flexibility allows it to be adapted to different types of software systems and development processes.

- Application in Large-Scale Projects:
  - Boehm's model is often used in projects where the software's utility is critical, such as enterprise systems, government applications, and mission-critical software.
  - Its comprehensive coverage of quality attributes makes it valuable for projects with diverse requirements, where multiple aspects of quality must be considered simultaneously.

- Relevance in Modern Development Practices:
  - In agile environments, Boehm's model can be used to establish quality goals and guide iterative development processes, ensuring that quality is built into the software from the outset.
  - The model's focus on human engineering and usability is particularly relevant in today's user-centric software market, where the user experience is a key determinant of success.

In conclusion, Boehm's Quality Model provides a robust and adaptable framework for evaluating software quality. Its comprehensive approach makes it particularly valuable for complex, large-scale projects, while its flexibility allows it to be applied in various contexts, including modern development methodologies.

4. Comparison and Contrast

4.1 Components and Approaches

McCall's Quality Model

- Components

  McCall's model is structured around three major perspectives: Product Operation, Product Revision, and Product Transition. These perspectives are further divided into quality factors, which are then broken down into quality criteria and measurable metrics.
- Approach

The model focuses primarily on the operational aspects of software, emphasizing factors like reliability, maintainability, and usability. It is designed to be straightforward, with a clear emphasis on the end-product quality.

- Applications

McCall's model is often applied in environments where product quality is the primary concern, and it is particularly useful in assessing and improving specific quality attributes in a linear, well-defined manner.

Boehm's Quality Model

- Components

  Boehm's model employs a hierarchical structure, consisting of high-level characteristics, intermediate-level characteristics, and lowest-level characteristics. These characteristics are linked to specific metrics that measure the quality of software at different levels of abstraction.

- Approach

  Boehm's model takes a more comprehensive and flexible approach by considering a wider range of software attributes, including utility, maintainability, and human engineering. It allows for a detailed examination of both the product and the processes involved in its creation.

- Applications

The model is often applied in complex, large-scale projects where a detailed understanding of various quality dimensions is necessary. Its hierarchical nature makes it adaptable to different types of software systems..

4.2 Strengths and Weaknesses

Quality Model	Strengths	Weaknesses
McCall's Quality Model	Simple and clear structure.	Limited focus on the development process.
	Strong focus on product quality	May be too rigid for modern agile practices.
	Widely applicable across different software systems.	Less comprehensive coverage of quality attributes.

Boehm's Quality Model	Comprehensive coverage of quality attributes.	Complexity can be a barrier to implementation, especially in smaller or resource-constrained projects.
	Flexibility and adaptability to different project needs.	The model's focus on specific attributes may not fully address all non-functional aspects of software quality.
	Considers both product and process quality.	

5. Conclusion

5.1 Summary

McCall's and Boehm's Quality Models have made significant contributions to the field of software engineering, providing structured frameworks for understanding and improving software quality. While McCall's model offers a clear and user-oriented approach to product quality, Boehm's model provides a more comprehensive and flexible framework that considers both product and process quality.

Both models remain relevant in modern software development, with their principles being adapted to suit contemporary practices like Agile and DevOps. By understanding and applying these models, software professionals can make informed decisions that enhance software quality, leading to more successful and reliable software products.

In a rapidly evolving software landscape, the enduring relevance of McCall's and Boehm's Quality Models underscores the importance of systematic approaches to software quality. As new technologies and methodologies emerge, these models provide a solid foundation for ongoing quality improvement, helping to ensure that software continues to meet the needs and expectations of users in an increasingly complex and interconnected world

5.2 Importance of Understanding These Models for Software Quality Assurance

Understanding McCall's and Boehm's quality models is the very root of software quality assurance. These models open up at least an argument on dimensions of software quality, thereby letting a developer or a quality assurance team take a structured approach toward

evaluation and betterment. These models would let organizations have a framework for systematic evaluations of software quality and point out areas that most need enhancement, along with strategies to overcome deficiencies.

These models can be integrated into software development practices to increase software performance, user satisfaction, and maintainability or adaptability. Besides, by having an idea about the strengths and limitations of each model, teams could adapt their quality assurance process to project needs and context to meet the quality objectives efficiently and effectively. At the very least, these models are what a software engineer and quality assurance professional can arm themselves with to give high-quality software products that would be evolving according to the changing standards in this industry and user expectations.

6. References

[1] Rahmawati, F., Musyafa, M. A., Fauzi, M. D., & Mulyanto, A. (2016). Quality testing of order management information system based on mccall's quality factors. *IJID (International Journal on Informatics for Development)*, *5*(2), 12-20.

[2] Yolandari, S., & Rahmawati, S. (2024). Evaluation of the Quality of Additional Employee Income using the McCall Method. *Journal of Computer Scine and Information Technology*, 1-6.

[3] Rawashdeh, A., & Matalkah, B. (2006). A new software quality model for evaluating COTS components. *Journal of computer science*, *2*(4), 373-381.

[4] Jamwal, D. (2010). Analysis of software quality models for organizations. *International Journal of Latest Trends in Computing*, *1*(2), 19-23.

[5] Ortega, M., Pérez, M., & Rojas, T. (2003). Construction of a systemic quality model for evaluating a software product. *Software Quality Journal*, *11*, 219-242.

[6] Al-Qutaish, R. E. (2010). Quality models in software engineering literature: an analytical and comparative study. *Journal of American Science*, *6*(3), 166-175.

[7] Romera, A. J., McCall, D. G., Lee, J. M., & Agnusdei, M. G. (2009). Improving the McCall herbage growth model. *New Zealand Journal of Agricultural Research*, *52*(4), 477-494.

[8] Ing, M. A. C. M., & Georgiadou, E. (2003). Software quality model requirements for software quality engineering. In *14th international conference on the software quality requirement*.

[9] Fitzpatrick, R. (1996). Software quality: definitions and strategic issues.

[10] Roihan, A., & Rafika, A. S. (2019, July). Statistical Analysis in System Evaluation of ilearning Media With Mccall's Quality Model. In *Journal of Physics: Conference Series* (Vol. 1179, No. 1, p. 012018). IOP Publishing.