

Unified framework of attributes in software quality models

Sherif M. Tawfik and Nadine M. El-Mekky

Information and Documentation Center Arab Academy for Science,
Technology and Maritime Transport, Miami, Alexandria, Egypt
e-mail: Sheriftawfik@eng.aast.edu
e-mail: Nadine @eng.aast.edu

Software quality models are frequently used in large projects. They give guidance in what requirements to collect, architectural qualities to consider, and what to test. In the literature, several quality models have been defined. Some of them are focusing on model-specific quality attributes, while others vary in scope, level of details. Some use elaborate frameworks taking many perspectives into account; other use traditional frameworks based on a hierarchical breakdown of quality. The paper's purpose is to compile quality attributes recognized by researchers in framework quality into a common and established framework. A clear picture of quality attributes is presented in the form of a software manual. Such a picture would be useful for researchers as it defines the relationships between the attributes and identifies some properties for each of them. In addition, one of the advantages of the framework is that it can be easily updated by any new attributes discovered by researchers.

تستخدم نماذج جودة البرمجيات تكرارا في المشاريع الكبيرة، حيث انها تعطى دلائل حقيقية لمتطلبات النظام المطلوب معها وكذلك يجب أن يراعى أسلوب الجودة ومعرفة ماذا يراد أن يختبر. نماذج عديدة لجودة البرمجيات اقترحت في كثير من مواضيع الأبحاث السابقة، بعض منها يركز على صفات الجودة للنموذج، بينما يتتبع الآخرون في رؤية مستوى التفاصيل في بعض المناطق واستخدامها في نطاق الجودة (فبعضهم يستخدم هيكلا منفصلا يأخذ في الاعتبار أكثر من وجهة نظر والبعض الآخر يأخذ الهيكل التقليدي القائم على الهيكل الهرمي التسلسلي للجودة). الهدف من هذا البحث هو جمع العديد من العناصر التي سبق استخدامها في تعاريف نماذج الجودة المختلفة ووضعها في تصنيف مشترك وانشاء هيكل موحد لهم. وبهذا يمكن تقديم صورة واضحة لعناصر الجودة واستخدام هذا الهيكل كدليل لدراسة علاقة العناصر بعضها ببعض وتأثير هذه العناصر على البرمجيات.

Keywords: Quality model, Quality attributes

1. Introduction

Quality is a functional and artistic measurement used to specify user satisfaction with a product, or how well the product performs compared to similar products. A model is an abstract form of reality, enabling details to be eliminated and an entity or concept to be viewed from a particular perspective. This is one reason why a quality model has become essential for ensuring that a firm product and process meets customers' needs [1]

As software becomes more and more pervasive, there has been a growing concern in the academic community about software quality. Software Quality is the degree to which a system, component, or process meets specified requirements and customer or user needs or expectations [2]. In the last 15 years, the software industry has created many new different markets, such as open source

software and commerce over the Internet. With these new markets, customers of programs now have very high expectations on quality and use quality as a major drive in choosing programs [3].

Developing high quality software is hard [1]; It is a complex concept, due to the large number of quality attributes that can be addressed [4, 5]. In order to know if quality has been achieved, or degraded, these attributes have to be measured. From this viewpoint, it is necessary to define a model that considers quality requirements for software system [5, 6]. However, defining the most important attributes to measure and how to measure is the difficult part.

Different authors and organizations have proposed different models intended to evaluate the quality of software in general [2, 7]. A difficulty is that different quality models tend to differ with respect to classification and definition of attributes [7]. Due to this

difficultly, ISO and IEEE have tried to standardize software quality by defining models that combine and relate software quality characteristics and sub characteristics [8].

In this context, this paper describes the problems related to defining a software quality framework, describing the quality attributes and related metrics for the system evaluation. The new proposed classification addresses the following questions: 1. which quality criteria should be considered for the evaluation of specified system characteristic, 2. how we can evaluate them by using different quality metrics and, 3. which stakeholder should be responsible for such an evaluation.

In order to answer the above questions, the following work has been done:

1. A list of the most common and most frequent dimensions included in the most well known models has been defined as a hierarchy of factors, criteria and metrics. Discovering the relationship and interaction between them. This hierarchy will be used as a cross reference manual for determining the influence of each factor, criteria and metric on each other.
2. Proposing a new model based on the new classification and the common aggregation approach (weighted average or sum) for the evaluation of software quality attributes.

2. Quality models

In order to understand and measure quality, scientists have often built models of how quality characteristics relate to each other [9]. So far, scientists have prepared many models intending to cover the entire software development. In this paper, a number of important quality models are mentioned [10]. The quality models are categorized in two categories: Primitive Models and related Models.

2.1. Primitive models

Primitive models are the most standardized and the best known. Some authors consider them as the most used quality model and the basics for all related research [11-15].

Those models are categorized into hierarchical and non hierarchical models.

2.1.1. Hierarchical models

In this section, the most important hierarchical models are mentioned because of their importance and usefulness in the classification of the quality model attributes. Moreover, these models will be utilized as the basis for the new classification perspective represented in this paper.

2.1.1.1. McCall model (1977): McCall proposes one of the first structured quality models. Some authors consider McCall's model as the first and the most used quality model [9]. Fig. 1 illustrates the model framework which is divided to the following levels:

1. Highest level: specifies the major aspects or factors which represent the management or customer view of product quality.
2. Middle level: specifies the attributes that provide the characteristics for the factors. Few criteria are defined for each factor.
3. Lowest level: specifies the software quality metrics that measure the software attributes. Further, these factors are categorized based on the uses of a software product as follows:
 - *Product Operation:* refers to the product's ability to be understood, to be stable and functional.
 - *Product Revision:* is related to error correction and system adaptation.
 - *Product transition:* is related to portability characteristics assuming rapidly changing hardware.

One of its major contributions 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 [4, 8 and 9].

2.1.1.2. Boehm model (1978): Boehm defines software quality in terms of qualitative attributes and measures it using metrics. The model is based on a wider range of characteristics and incorporates 19 criteria. Fig. 2 presents the hierarchical levels as follows [16]:



Fig. 1. McCall quality model.

- *High level*: defines basic functionalities which can assist in measurement of software quality. It represents basic high-level requirements of actual use. It addresses three main questions that a buyer of software has:
 - *As-is utility*: How well (easily, reliably, efficiently) can I use it as-is?
 - *Maintainability*: How easy is it to understand, modify and retest?
 - *Portability*: Can I still use it if I change my environment?
- *Intermediate level*: Contains the same quality factor proposed by McCall as well as these factors constitute the overall expectations which customer hoping from the product.
- *Lowest level*: exhibits the metrics hierarchy and defines the factors which help measuring and ensuring the certain level of hierarchy.

The Boehm model is similar to the Mc Call model in that it represents a hierarchical structure of characteristics, each of which contributes to total quality but it has a few notable differences. 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 [11].

One obvious difference is its factoring of quality characteristics. As an example, in McCall's model maintainability and testability are at the same level which means in McCall's model a product can be maintainable without being testable and vice versa. In Boehm's model, testability is a sub factor of maintainability, which means an increase in testability implies an increase in maintainability. By itself this isn't necessary an important distinction but it does point out the difficulty of having a definitive quality model.

Another way Boehm's model is different from McCall's model is that Boehm's model applies only to code and the fact that Boehm is the strong believer that quality has much more to do with maintainability and related terms [11, 15]. However, Boehm's model contains only a diagram without any suggestion about measuring the quality characteristics. [15].

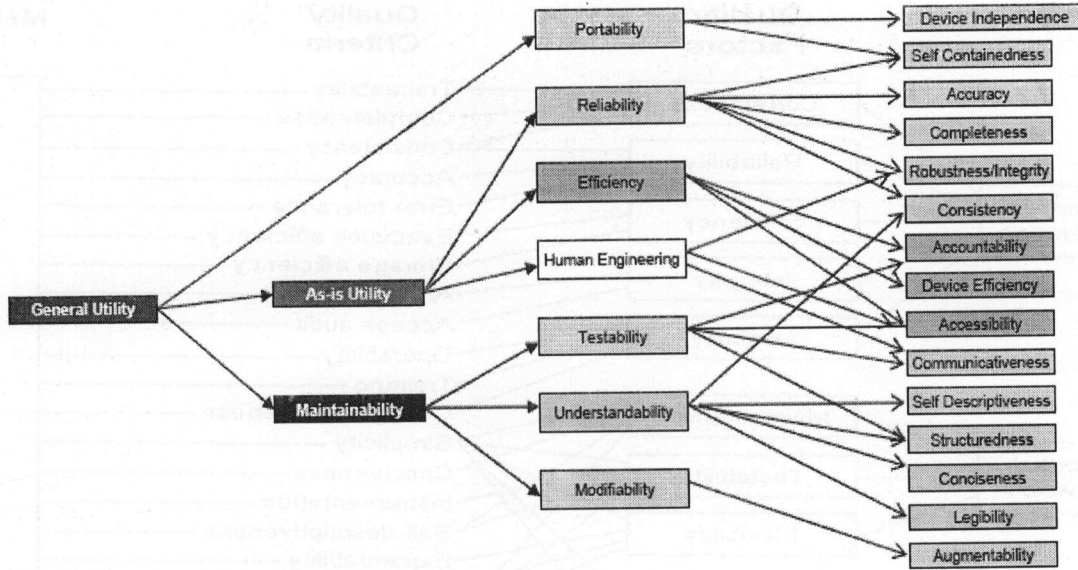


Fig. 2. Boehm quality model.

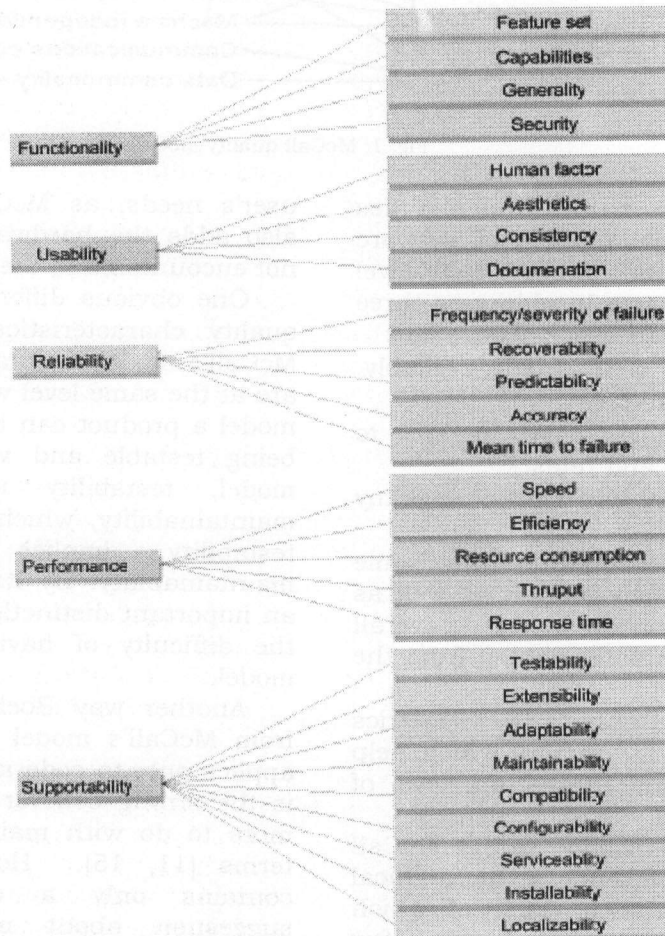


Fig. 3. FURPS quality model.

2.1.1.3. *FURPS model (1987)*: FURPS model presented by Robert Grady is used to simplify the process of defining the appropriate measurements in each life cycle phase (Specification / Design / Implementation/ Testing/Support), through focusing on specific quality criteria. It stands for five quality criteria which are divided into 27 sub-characteristics as shown in fig. 3. These criteria are decomposed into two different categories of requirements:

- *Functional (F)*: input and expected output.
- *Non-functional (URPS)*: Usability, Reliability, Performance and Supportability.

Each of the quality criteria maps to one or more metrics. Using the FURPS model involves two steps: 1. Establishing priorities, 2. making quality attributes measurable.

Establishing priorities are important because of the trade-offs involved between quality factors. Thus the decision has to be made, what type of quality is relevant in a given project and a prioritized list of quality factors has to be defined. Once priorities have been established, measurable goals for each quality factor have to be defined. These measures are, as the priorities, project specific and also depend on the phase of the life cycle.

FURPS itself does not contain a technique to derive the metrics for the quality criteria. In general this derivation is context and project (customer) dependent, unless a static definition of quality is given. Goal oriented measurement approaches can help in finding the appropriate metrics. In practice some metrics will be reused from project to project, though this is not recommended.

The perspective of the quality factors and the quality criteria is product and customer oriented. But, there is a loose integration of the user and the product view. In the context of embedded systems, FURPS model has been used in the development for software and for hardware/software systems like Boehm model. Experience showed that aspects of reliability and supportability were heavily influenced by hardware design decisions. One disadvantage of this model is that it fails to take account of the software product's portability.

2.1.1.4. *ISO 9126 (1991)*: ISO 9126 is an international standard for the evaluation of software. It is an extension of previous work done by McCall 1977, Boehm 1978. It represents the latest (and ongoing) research into characterizing software for the purposes of software quality control, software quality assurance and software process improvement [14, 15]. ISO/IEC 9126-1 defines a quality model in terms of:

- *Internal quality*: is evaluated using internal attributes of the software (design modularity and compliance). How the product was developed e.g. size, test and failure rate.
- *External quality*: is evaluated when the software is executed, typically during formal testing activities. How the product works in its environment.
- *Quality in use*: refers to the user's view of the software quality when they use it in a particular environmental context. In other words, quality in use is evaluated after the software is deployed to the operational environment. The quality model for quality in use is categorized into four characteristics: effectiveness, productivity, safety and satisfaction.

The set of ISO 9126 quality views in fig. 4 is based on the belief that internal quality has an impact on external quality, which in turn has an impact on quality in use. Therefore, the achievement of quality in use depends to some extent on the achievement of external quality, which in turn depends on the achievement of the internal quality of the software product itself.

As shown in fig. 5, the internal and external quality models share the same hierarchical structure, with two levels. The first level has six characteristics, which are broken down into 27 sub characteristics in the second level.

The advantages, or in other words, the most important features of this model are that:

- This approach to quality evaluation decomposes the concept of quality into a set of lower level quality characteristics that are recognizable properties of a product or service which refine "quality" into something more concrete and measurable [16].

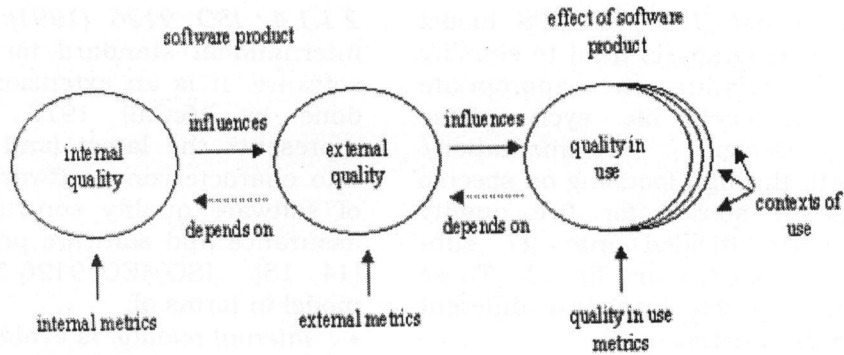


Fig. 4. Quality along the software life cycle [15].

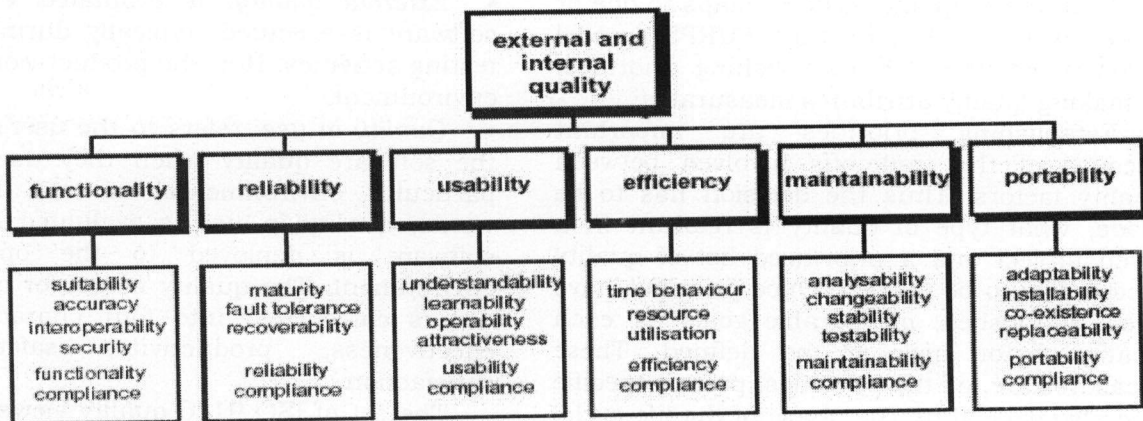


Fig. 5. ISO 9126 quality model.

- It defines the internal and external quality characteristics of a system, where the internal attributes influence or determine the external attributes obtained by the end-user [11].
- It is generic so it can be applied to any software product by tailoring to a specific purpose [10].
- It defines a three-level (strict) hierarchical structure of quality concepts [17].
- Familiar labels or single words are used to identify each characteristic and sub characteristic, using terms that are commonly understood in practice [17].
- It consists of concise definitions, where each characteristic and sub characteristic is defined using a single sentence [17].
- Even evaluation procedures are defined in a separate standard to illustrate procedures for conducting product evaluations [18].
- It is preferable because it represents a broad consensus among researchers and

practitioners and is widely accepted and used in practice [17].

- It provides a common vocabulary to express quality of user needs.

But its major disadvantage is that it does not provide a clear way to measure these quality aspects [9]. Pfleeger [19] reports some important problems associated with ISO 9126: 1. There are no guidelines on how to provide an overall assessment of quality and rather than focusing on the user view of software, 2. The model's characteristics reflect a developer view of software. For the most part, the overall structure of ISO 9126 is similar to past models, McCall (1977) and Boehm (1978) since they use hierarchical frameworks, but they use different quality frameworks and terminology. ISO 9126 model reflects a user view since sub characteristics relate to quality aspects that are visible to the user; McCall's model reflects a product view since criteria relate to internal software properties.

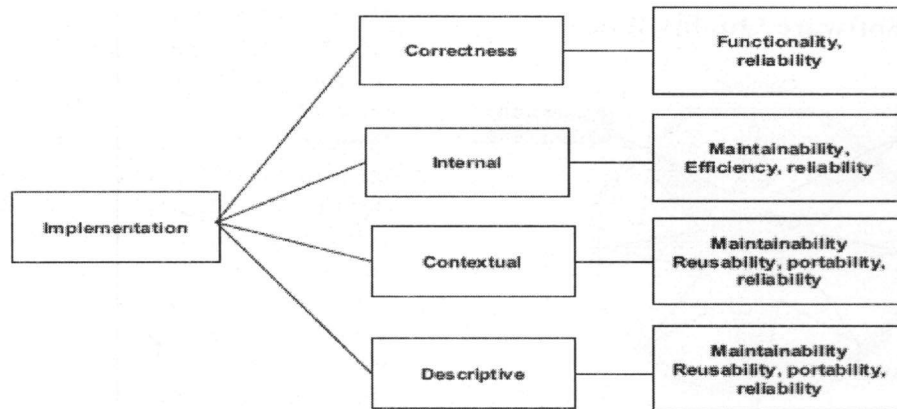


Fig. 6. Dromey model.

2.1.1.5. Dromey model (1996): Unlike the previous model, Dromey takes a different approach to software quality model. The model is based upon the product perspective of quality. He has built a quality evaluation framework that analyzes the quality of software components through the measurement of tangible quality properties. As illustrated in fig. 5, these components all possess intrinsic properties that can be classified into four categories:

- *Correctness:* Evaluates if some basic principles are violated.
- *Internal:* Measure how well a component has been deployed according to its intended use.
- *Contextual:* Deals with the external influences by and on the use of a component.
- *Descriptive:* Measure the descriptiveness of a component.

This model doesn't consider the efficiency of software to determine the quality of software [20]. The disadvantage of the Dromey model is associated with the high level quality attributes (maintainability, functionality, reliability) which can't be built into the system. The alternative way to input quality is identifying a set of properties and build them up consistently, harmoniously and fully to provide high level quality [11]. It is not feasible to judge both attributes Reliability and Maintainability of a system before it is actually operational in the production area [5]. Links must be established between tangible product properties and intangible quality attributes.

2.1.2. Non-hierarchical models

The non-hierarchical models can be considered as sequential models. Unlike hierarchical models, they have the lack of identifying sub attributes for the associated high level attributes. In addition, it is not clear how to measure the quality of those attributes [15]. The most known of them are summarized into the next subsections.

2.1.2.1. Bayesian Belief Networks (BBN): BBN are powerful models for modeling causes and effects in a wide variety of domains. They are compact networks of probabilities that capture the probabilistic relationship between variables, as well as historical information about their relationships effects via an intuitive graphical representation. Each of the variables in the BBN is represented by nodes. Each node has states, or a set of probable values for each variable. Nodes are connected to show causality with an arrow indicating the direction of influence. These arrows are called edges. It is very effective for modeling situations where some information is already known and incoming data is uncertain or partially unavailable. Because of all of these capabilities, BBN are being increasingly used in a wide variety of domains where automated reasoning is needed [20].

2.1.2.2. Star model: The star model is a conceptual model based on the acquirer and supplier as defined in ISO/IEC 12207 (1995) [20]. As shown in fig. 7, there are three elements: Procurer, producer and product.



Fig. 7. Star model.

- Procurer (acquirer):
- Enters in a contract with the producer to create software product and specify its quality characteristics.
- The lead party in any contractual arrangement because it is the acquirer's users and technical support professionals.
- Dictates profile and maturity of the supplier organization who dictate the success or failure of the software product. The procurer's perspective of the:
 - Producer: they use the best project management techniques available and that they engage in first-rate processes to create a quality product.
 - Product: it must be acceptable by the user community and that it can be serviced and maintained by their professionals.
 - Producer (supplier): The model focuses on its maturity as software developers and the development processes that they used to create quality software products" [20].

2.2. Related models

This section gives a quick highlight on some of the quality models in the literature. These models were taking the one or two of the hierarchal models as the basis for these working mechanisms.

M. Goulão's work (2002) [15] proposed the component quality model. However, this work only proposed the quality attribute without definition of metrics.

Martin-Albo's work (2003) [14] proposed the component quality model. This work is based on ISO 9126. However, this work defined the only characteristics and sub characteristics. Proposed metrics by this work might not measure the value. This work only defined the definition of metrics. Also, this work was not reflecting of component specific features since the work intends to apply all characteristics of ISO 9126 to component quality model.

C-QM (2003) [12]: This work was based on the features of COTS components and we defined its quality model which consists of characteristics, sub characteristics, and metrics for evaluating COTS components. The work derived 4 characteristics (functionality, reusability, maintainability, and conformance), derived 12 sub characteristics and metrics for measuring the characteristics. However, scope of this work is limited as COTS components. Therefore, quality model of extended business component are needed

Losavio F (2003) [18]: proposes an ISO 9126-1 based technique to specify the relevant quality characteristics, refined until the attribute level or measurable items, involved in the architectural design process.

OSMM Model (2004) [17]: Decomposes the quality model into six constituents (*Product software, Support, Documentation, Training, Product integrations and Professional services*), each one having some weight. The evaluator assigns a score to each element and the final evaluation mark is the weighted sum of the scores. It is simple and thus easy to apply, it is often criticized for not taking into account some important software artifacts, such as the source code itself.

Victor B (2005) [18]: presents a top-down approach to establish a goal-driven measurement system: 1. the team starts with organizational goals, 2. defines measurement goals 3. Poses questions to address the goals 4. Identifies metrics that provide answers to the questions.

Knight and Burn (2006) [13]: summaries 12 widely accepted Information Quality (IQ) Frameworks collated from the last decade of Information System research. The frameworks share a number of characteristics regarding their classifications of the dimensions of quality. Their analysis reveals the common

elements and shows that the most well known models and the basis of the different frameworks are McCall, Boehm, FURPS, ISO 9126 and Dromey.

The main difference between these works and the work represented in this paper is the size of the domain for which the quality attributes are defined.

3. Proposed classification

In spite of the differences and the similarities between the different models discussed in the previous section; it is found that the common objectives between all the models are to find the main factors essential to describe them and the way to quantify useful attributes. But the problem is that it isn't possible to provide matrices on a quality attribute level (at least not for many of the attributes) [10, 6]. Instead, lower level attributes are introduced. They are not exclusive for each attribute but could influence any attribute positively, negatively or not at all. For example, how the changeability of a system can be measured or in other words, what are the metrics needed to measure it. By considering the previous example, the main objective is to decompose the concept of quality into a set of lower level quality characteristics that are recognizable properties of a product or service which refine "quality" into something more concrete and measurable.

In order to reach this goal, table 1 has been constructed. The table aggregates the most common dimensions and the most frequently included in the different model and classifies them into lower level as a hierarchy of factors, criteria and metrics to let the factors needed to be measured too abstract and directly measured. Also, the table presents all the relationships between the attributes to identify the influence of them on





each other. The table is divided into many parts as follows:

Part 1: Quality Characteristics: presents the different attributes in the previous models by decomposing them into more specific attributes (factors, criteria, and metrics) as presented in the lower part of the table:

1. Factors: the main factors represent the behavioral characteristics of the system are represented in fig. 8.
2. Criteria: is a group of attributes for a quality factor related to software production and design as presented in fig. 9.
3. Metrics: is a measure that captures some aspect of quality criteria. The metrics used in this model are described in fig. 10.

The relationships published in [8, 9, 11, 12, 19, 21 and 23 to 33] were summarized, considering many types of interactions as follow:

- "+" good value of one attributes result in good value of the other (synergistic goals); If character A is enhanced, then characteristic B is likely to be enhanced
- "-" good value of one attributes result in bad value of the other (conflicting goals); If characteristic A is enhanced, then characteristic B is likely to be degraded
- "0" the attributes do not affect each other; if a characteristic A is enhanced, then characteristic B is unlikely to be affected.
- "x" the attributes affect each other, If a characteristic A is enhanced, then characteristic B is likely to be affected
- "H" relation between the attributes is high
- "L" relation between the attributes is low.
- "M" relation between the attributes is medium

-  attribute is identified by Dromey
-  attribute is identified by McCall
-  attribute is identified in ISO Model
-  attribute is identified by Boehm

Correctness
Reliability
Efficiency
Integrity
Usability
Maintainability
Testability
Flexibility
Portability
Reusability
Interoperability
Understandability
Modifiability
Suitability
Accuracy
Security
Maturity
Error Tolerance
Recoverability
Learnability
Operability
Attractiveness
Adaptability
Installability
Co-existence
Replaceability
Analysability
Changeability
Stability
(1) Factors

Fig. 8. Star model.

Criteria
Traceability
Completeness
Consistency
Execution Efficiency
Storage Efficiency
Accessibility
Operability
Training
Communicativeness
Simplicity
Conscienceless
Instrumentation
Self-Descriptiveness
Expandability
Generality
Modularity
Independence
Commonality
Self-Containedness
Robustness
Accountability
Structuredness
Legibility
Augmentability
Documentation
Anomaly
Time Behavior
Resource Behavior
Product Requirements
Product Architecture
Design & Implementation
Integration & Testing
Product Maintenance
Product Improvement
Baseline
Auditability
Distributedness

Fig. 9. Criteria presented in the model.

Metrics
Cross Reference
Completeness checklist
Procedure and data Consistency
Processing / data Usage Effectiveness measure
Storage effectiveness measure
Access control and audit
User Input / output Communicativeness
Training checklist
Communication effectiveness
Design Structure, Control Flow, Specificity
Quantity / Effectiveness of comments
Unit Referencing and Implementation
Modular Design and Implementation
SW / machine Independence
Communication and Data Commonality
Dependability
Data Storage, Computation / Design Extensibility
Access and well structured
Device, Communication, HW, Computation failure
Response time, throughput
Memory / disk Utilization
Argument, Feature, Inspections based design
Architecture, control flow, ERD
Algorithm, Source code, SW design
Code coverage metrication, complexity
On line SW upload, Maintainability index
User documentation, usability inspection
Simplex architecture, log files, system manuals
Design Structure

Fig. 10. Metrics used in the model.

Part 2: It contains the relationships between the factors to consider the effect that one factor has on another when there is a change in it occurs. It is gluey directly to part 1.
 Part 3: identifies the main models used, represents the levels in which each model is

divided to and the factors related to each model as shown in the following figure.
 Part 4: it contains many properties that must be taken into consideration when using each factor. These properties shown in the following figures are as follows:

Internal					X	X	X	X	X											(6) Type				
External	X	X	X	X	X				X											(5) Stakeholders				
User	+	H	L		H	L	X	+	H	X	X	X	X	X	X	X	X	X	+	X				X
Analyst / Developers	+	H	M		M	H	X	0	H	X	X	X	X	X	H	X	X	0	X					X
Manager	+	H	H		L	H	X	+	L				X	H					+				X	
Life Cycle				X		X		X	X	X	X			X				X	X				X	
Run Time				X					X				X	X	X	X	X						X	

Fig. 14. The framework properties.

1. Factor Types:

- External: refers to software execution. It is measured and evaluated during tests in a simulated environment. In general, Software users only care about the external qualities because they affect them
- Internal: is the quality needed from the internal perspective of the product, and it is in the habit of remaining without alterations unless the product is re-designed.

2. Cycle Processes: The model is classified into two classes; the quality characteristics that can be observable at *runtime* (that are discernable at component execution time) and the quality characteristics that can be observable during the product *life-cycle* (that are discernable at component and component-based systems development).

3. Stakeholders involved: In order to know the role of each stakeholders and the impact of each one on the quality model components, the stakeholders that have a relationship with the system throughout its life cycle, are identified. Also to solve the problem of who should be responsible for such evaluation. The weighted sum of the relations is used in establishing the aggregation of the score for the evaluation of software quality attributes. The aggregation score is done for 2 levels:

- Criteria Level: 37 metrics are used with 168 influences on the factors.
- Factor Level: 29 factors are used.

The table is designed in this way to be considered as a software manual to evaluate each factor by decomposing it into many criteria's which are defined using the metrics. Also, using this table, at the design phase of any model, can be used as a cross reference manual to indicate the effect of each factor on the others. So, the effect of any update on one factor on the others factors can easily be indicated. By combining the previous part, the

following framework is obtained as shown in the above figure.

4. Conclusions

Several models, specializing in measuring the quality of software products have been described. The features of these models have been studied, analyzed and their limitations outlined. Specifically, Functionality of a software product was not considered directly by McCall's model. No suggestion about measuring the quality characteristics has been found in Boehm's model. FURPS model fails to take account of the software product's Portability. ISO 9126 has the limitation of not showing very clearly how certain quality aspects can be measured. The disadvantage of Dromey's model is associated with Reliability and Maintainability. It is not feasible to judge these two attributes of a system before it is actually operational in the production area.

To overcome these limitations, all the attributes that have been found in the surveyed models have been presented from a new perspective. They are classified into a sub levels associated with metrics to facilitate measuring them and the relationships among them are also defined as shown in table 1. This table can be used as a software manual to describe and evaluate any software model.

In this work, effort has been done to establish a classification framework for software quality attributes. The relation between all parts of this framework has been explained and most of the quality attributes that exist in the literature have been included. The future work proposed is to apply this framework in the Software applications models according to these factors and these criteria to propose the optimum software metrics to enhance and evaluate any software application.

References

- [1] Adnan Rawashdeh and Bassem Matakah, "A New Software Quality Model for Evaluating COTS Components", *Journal of Computer Science*, Vol. 2 (4), pp. 373-381, ISSN 1549-3636 (2006).
- [2] Hongyu Pei Breivold, Ivica Crnkovic and Peter Eriksson, "Evaluating Software Evolvability", *SERPS*, 24-25 October, Göteborg (2007).
- [3] Khashayar Khosravi, "A Quality Model considering Program Architecture", Montreal University. Khashayar Khosravi, Aout (2005).
- [4] Charles River analytics, "About Bayesian Belief Networks", for BNet Version 1.0 Last Updated April 22 (2008).
- [5] Parastoo Mohagheghi, Vegard Dehlen, "A Metamodel for Specifying Quality Models in Model-Driven Engineering", *SINTEF*, P.O. Box 124 Blindern N-0314 Oslo, Norway (2009).
- [6] Julien Dormoy, Jean-Michel Hufflen, Olga Kouchnarenko and et al., "A Synthesis of Existing Approaches to Specify Non-Functional Properties", (2008).
- [7] Yoonjung Choi, Sungwook Lee and HOup Song, "Practical S/W Component Quality Evaluation Model", ISBN 978-89-5519-136-3 *ICACT* (2008)
- [8] Kevin K.F. Yuen and C.W. Henry "Software Vendor Selection using Fuzzy Analytic Hierarchy Process with ISO/IEC 9126", *International Journal of computer Science*, 21 August (2008).
- [9] Joaquina Martin-Albo, Manuel F. Bertoa, Coral Calero, Antonio Vallecillo, Alejandra Cechich and Mario Piattini, "A Software Component Metric Classification Model," *IEEE Transactions on Journal Name*, Manuscript ID.
- [10] Dipl.ing. Gregor Panovski. *Product Software Quality*. Eindhoven, February (2008).
- [11] B. Vinayagasundaram and S. K. Srivatsa. *Software Quality in Artificial Intelligence System*. *Information Technology Journal*, Vol. 6 (6), ISSN 1812 - 5638, pp. 835 - 842 (2007).
- [12] Sherif M. Tawfik, Marwa M. Abd-Elghany, and Stewart Green, "A Software Cost Estimation Model Based on Quality Characteristics," In *MeReP: Workshop on Measuring Requirements for Project and Product Success*, *IWSM-Mensura*, *IWSM (International Workshop in Software Measurement)* and *MENSURA (International Conference on Software Process and Product Measurement)*, Palma de Mallorca, Spain, 5-8 November (2007).
- [13] Jamaiah Haji Yahaya, Aziz Deraman and Abdul Razak Hamdan. *Software Quality from Behavioural and Human Perspectives*. *IJCSNS International Journal of Computer Science and Network Security*, Vol. 8 (8), August (2008).
- [14] Asif Javed. *Metric-Oriented Quality Model*, Department of Computer Science Faculty of Applied Sciences International Islamic University, Islamabad. Thu, 28 Sep. (2006).
- [15] Ioannis Samoladas, Georgios Gousios, Diomidis Spinellis and Ioannis Stamelos, "The SQO-OSS Quality Model: Measurement Based Open Source Software Evaluation", *The European Community's Sixth Framework Programme Under the Contract IST-2005-033331* (2005).
- [16] Shirlee-ann Knight and Janice Burn, "Developing a Framework for Assessing Information Quality on the World Wide Web," *Informing Science Journal* Vol. 8 (2005).
- [17] [http://www.bth.se/tek/besq.nsf/\(WebFiles\)/BFEFBED4E650D690C125706900324572/\\$FILE/chapter_4.pdf](http://www.bth.se/tek/besq.nsf/(WebFiles)/BFEFBED4E650D690C125706900324572/$FILE/chapter_4.pdf)
- [18] Marc-Alexis Cote, Witold Suryn, *Software Quality Model Requirements for Software Quality Engineering*.
- [19] Ronan Fitzpatrick, Catherine Higgins, *Usable Software and its Attributes: A synthesis of Software Quality*, *European Community Law and Human-Computer Interaction*.
- [20] Hazura Zulzalil, Abdul Azim Abd Ghani, Mohd Hasan Selamat, Ramlan Mahmood, "A Case Study to Identify Quality Attributes Relationships for Webbased Applications", *IJCSNS International Journal of Computer Science and*

- Network Security, Vol. 8 (11), November (2008).
- [21] Alexandre Alvaro, Eduardo Santana de Almeida, Silvio Romero de Lemos Meira. Quality Attributes for a Component Quality Model. Federal University of Pernambuco and C.E.S.A.R - Recife Center for Advanced Studies and Systems, Brazil. January (2009).
- [22] Parvinder Singh Sandhu and Gurdev Singh, Dynamic Metrics for Polymorphism in Object Oriented Systems. Proceedings of World Academy of Science, Engineering and Technology, Vol. 29, ISSN 1307 - 6884, May (2008).
- [23] K.K. Aggarwal, Yogesh Singh, Arvinder Kaur and et al., "Investigating Effect of Design Metrics on Fault Proneness in Object-Oriented Systems", Journal of Object Technology, Vol. 6 (10), pp. 127 - 141, November-December (2007).
- [24] Jihad AL Dallah, "A Design-Based Cohesion Metric for Object-Oriented Classes", International Journal of Computer Science and Engineering Vol. 1 (3) (2007).
- [25] K.K. Aggarwal, Yogesh Singh, Arvinder Kaur and et al., "Application of Artificial Neural Network for predicting Maintainability using Object Oriented Metrics", Proceedings of World Academy of Science, Engineering and Technology, Vol. 15, ISSN 1307 - 6884, October (2006).
- [26] Mahmoud O. Elish, "A Case Study on Structural Characteristics of Object Oriented and it's Stability", Proceedings of the IASTED International Conference on Software Engineering, pp. 89-93, Austria, Feb. (2005).
- [27] G. Canfora, F. Garcia, M. Piattini and et al., "A Family of Experiments to Validate Metrics for Software Process Models", Journal of Systems And Software, Vol. 77, pp. 113 - 129 (2005).
- [28] Marc-Alexis, Witold Suryn and Elli Georgiadou, Software Quality Model Requirements for Software Quality Engineering (2004).
- [29] David Rine and Mahmoud Elish, "Investigation of Metrics for Object-Oriented Design Logical stability", Proceedings of the 7th IEEE European Conference on software Maintenance and reengineering (CSMR 03), pp. 193 - 200, Italy, Mar. (2003).
- [30] Rajecndra K. Bandi, Vijay K. Vaishnavi et al., "Predicting Maintenance Performance Using Object Oriented Design Complexity Metrics", IEEE Transactions on Software Engineering, Vol. 29 (1), January (2003).
- [31] Javier Garzas and Mario Piattini, "Analyzability and Changeability in Design Patterns", SugarloafPloP (2002).
- [32] Khaled El Emam, Saida Benlarbi, Nishith Goel and et al., "The Confounding Effect of Class Size on the Validity of Object Oriented Metrics", IEEE Transactions on Software Engineering, Vol. 27 (7), July (2001).
- [33] Lionel C. Briand, Jurgen Wust, John W. Daly and et al., Exploring the Relationships Between Design Measures and Software Quality in Object Oriented Systems (2000).
- [34] Khaled El Emam, Saida Belarbi and Shesh Rai, A Validation of Object Oriented Metrics. V 10 - 18 / 10 / 1999 (1999).

Received April 16, 2009
Accepted August 18, 2009