A comparison of Software Quality Characteristics and Software Sustainability Characteristics

Sulaiman Aljarallah
Department of computer science
Loughborough University
Loughborough, UK

Dr. Russell Lock
Department of computer science
Loughborough University
Loughborough, UK

ABSTRACT

Software sustainability has generated much interest in the software engineering field in recent times, and has been widely investigated across different fields and from different standpoints. The relationship between software quality and software sustainability is still an open question. In this study, a literature survey and comparison was conducted using three-phases, having as a starting point the comparison of basic models for software quality. A follow-up study, conducted at a more comprehensive level to cover both basic models and the most cited tailored models. , Software sustainability literature is investigated to find the most frequent characteristics. Finally, data gathered from these studies and a comparison shows a similarity in the top level of these characteristics between software sustainability and software quality, and the emphasis on sustainability, maintainability and portability. The study suggests that ISO 25010 can be utilised by software sustainability. As a future work, the findings will be investigated empirically to support designing software sustainability framework identifying the most important criteria in the technical dimension.

CCS CONCEPTS

• General and reference~Surveys and overviews • Social and professional topics~Sustainability

KEYWORDS

Software sustainability, Software quality, Software sustainability characteristics, Quality characteristics. Quality models

1 Introduction

There is a growing interest with regard to sustainability and the claims for sustainable development are escalating [1]. Sustainability is widely considered to have three main dimensions which called the triple bottom line: social, economic and environmental [2]. The most widely accepted definition of sustainable development is that of the UN Commission on Economic Development [3] [2]; whose 1987 Brundtland Report defines it as *"development that meets the needs of the present generation without compromising the ability of future generations to meet their own needs"* [4] .

Similarly, there is no agreed definition of software sustainability [2] [5] however, many authors, including [6] cite [7] definition. Dick et al. focused their definition of software sustainability on the impact (positive or negative) of software on various dimensions (economy, society, human beings, and environment) when using, deploying or developing software. Considering the previous argument, this study aims to understand the difference between software quality and sustainability in terms of characteristics and priority based on data collection from literature to support the design of a future software sustainability framework.

2 Software sustainability and software quality models

Some authors believe sustainability is a Non-Functional Requirement, or NFR [8]. Software quality characteristics correlate with each other, including sustainability as NFRs. Software sustainability is an umbrella term that covers various software quality characteristics; however, there is no agreement about how to relate sustainability to software quality [9].

It is essential to consider NFRs not just in isolation but in combination due to their criticality in software development [10]. Sommerville [11] states that failing to meet NFRs can cause instability in an entire system, affecting sustainability. However, NFRs are controversial for various reasons. According to Adams [10] there is no single agreed definition of NFRs, there is no complete list and there is no single universal classification, framework or taxonomy to support them. [12] states that it is very difficult to model: NFRs usually contradict each other, are hard to enforce throughout development, and are hard to appraise. [13] identify 252 NFRs, whereas [10] states that there are more than 200. [14] report 530 NFRs collected from 11 industrial requirements. As a result, there is no agreement about how many exist.

According to [15] the quality of software could be best described through NFRs. This necessitates a way to identify the most important software characteristics for a piece of software. If we claim that characteristic X is more important than characteristic Y, we need evidence to support this claim. As a result, software quality models have provided a first step in defining the vital characteristics of software quality, which reflect the stability and consistency of software. Poor-quality software could have serious consequences, such as financial loss and mission failure [16]; such effects directly influence software sustainability. Therefore, consideration of software quality models in this study leads to a

specification of which characteristics most important for sustainability.

3 Quality models Types

Many quality models have been defined and published [17], and each of these models contains various characteristics or factors [18][19]. Dubey et al.[20] state that software quality can be categorised into two areas, namely software procedure (process) quality and software product quality. Software process quality is described as software engineering-related components such as technology, people, tools and organisation, whereas software product quality comprises many aspects, such as the clarity of documentation and integrity, design traceability, application reliability and test integrity as its basic characteristics. [15] and [21] categorise quality models into two types, namely basic (1977–2001) and tailored (2002–2011) models.

The purpose of basic models is to define quality in a new way that covers various aspects of software quality, taking into consideration different points of view regarding the evaluation of software products [15]. Both [21] and [15] argue that basic quality models are hierarchical in structure. The former argues that the basic models are McCall, Boehm, FURPS, Dromey and ISO 9126. However, [21] and [22] point out that the FURPS model is a private model since it was developed by Hewlett Packard for quality control of their own software.

Basic quality models are generic and can be adjusted for any software [21], whereas tailored models are designed for specific applications and domains [15]. Tailored models started to appear in 2001 with the Bertoa model [21]. It has been argued that most of the basic quality models, especially ISO 9126, are the base of tailored models [21]. [23] argue that tailoring reference models is an important way of reducing their size, which in turn leads to a reduction in the complexity of the model.

4 Methodology

The sustainability stakeholders for software development encompass a wide range of perspectives and interests [24]. In addition, the scope of software development is itself broad, with tailored models only existing for a small subset of domains. As a result in order to provide a fair comparison this paper explores basic quality models rather than tailored domain specific ones. In order to provide a more detailed understanding of the evolution of these model's comparisons are provided on historic as well as more recent models. This paper follows the approach of [21]which defines the timeline from 1977–2013 for software quality models.

Searches for quality models were conducted to define timelines, model types and model content. Google Scholar, Springer, Web of Science, Scopus, Google Books, IEEE and Copac were used for this. The key words used in this search were 'software quality', 'software quality models', 'comparing software quality models', 'basic models', 'software product quality models' and 'software quality models review'.

The main objective of this research is to determine the most important characteristics of software sustainability. To that end the criteria outlined in Table 1 were used to identify appropriate models for comparison.

Table 1: Methodology criteria

No	Criteria	Justification
1	All are well-known quality models.	They should be complete and fully documented as well as discussed and critiqued by many authors.
2	All are quality models, or are standards containing reference models for quality.	IEEE 1061 / IEEE730 for example contain a software quality model in their appendices.
3	All are hierarchical.	All basic models are hierarchal in structure, enabling comparison of factors (high-level) with each other.
4	All are from different sections of the timeline (history).	This shows how the focus & scope has shifted between software characteristics over time.
5	The comparison is conducted in two dimensions: characteristics and models.	The study focuses on the most important characteristics (factors) in each quality model, so the criteria and metrics are discarded in this comparison.
6	The analysis is conducted according to the frequency of each characteristic.	Only characteristics which appear in half the models examined are put forward, for scoping reasons.
7	The findings from the basic models study are compared to those of other studies.	To verify and support our findings.

5 Software quality models

In order to determine software quality, software quality models have been developed and published [25]. These models differ in terms of characteristics, sub-characteristics and metrics. There are many software quality models, which will be discussed in the following section according to the date of their appearance.

5.1 McCall's software quality model

McCall's quality model was published in 1977 [17]. Developed by the US Air Force, it was the first model for software quality [25]. It is called an FCM (factor criteria metric) model [25]; its structure is hierarchical, and it defines quality attributes (quality factors, quality criteria and quality metrics) [17]. The model bridges the gap between users and software developers, concentrating on a collection of factors that express users' views and developers' priorities[10][26]. McCall's has three main viewpoints for defining and identifying software quality: product operation,

product revision and product transition. These viewpoints contain 11 quality factors[25][26]. McCall's quality model has its criteria, which are connected to quality factors, and they are both reflected in tree-style [25].

[25] argue against McCall's quality model; they state that defining accurate quality requirements for software products may be problematic, as some metrics are subjectively measured. The quality measurement is based on a Boolean Yes or No [21]. McCall's quality model is one of the leading models, since it has been used as a base for creating other quality models. Its measurement method ensures there is no halfway measurement, i.e. there is not 50%, or even 100% or 0%, which could make the judgment biased or inaccurate. However, it is arguable that some software factors may not be objectively measured, such as usability. [27] argues that in spite of its age, McCall's model is still relevant today.

5.2 Boehm's software quality model

In 1978, Boehm presented his quality prediction model [25] [26], which was the result of a study conducted by Boehm and two of his colleagues [10]. Boehm and his colleagues identified 23 non-functional requirements, or software quality characteristics, [10]. It has been argued that while there is a similarity between Boehm's and McCall's quality models in terms of structure [28], Boehm's model represents an improvement over McCall's model [25]. There is increased emphasis on the needs of stakeholders within Boehm, including users, developers, testers and maintainers [25]. According to [28], Boehm addressed defects within existing quality models, which provided a quantitative evaluation of software quality [26]. [28] point out that a set of attributes and metrics are given in order to define software. The structure of this model is divided into three sections, namely high-level, intermediate and primitive characteristics[28][25].

The high-level characteristics show software as a general utility which contains three sub-characteristics: utility as-is, maintainability and portability; the intermediate level contains seven quality factors: portability, reliability, efficiency, usability, testability, understandability and flexibility; and the primitive level defines the metrics and is the base of defining quality metrics [25].

5.3 Evans and Marciniak's quality model

Evans and Marciniak's quality model, developed in 1987, is an enhancement to McCall's model [29]. This model covers 12 factors that are categorised into three categories, namely design, performance and adaptation [30][19]. This model has removed testability from McCall's model [29] and added two new characteristics, namely verifiability and expandability [30]. For [27], testability can be seen as an element of maintainability, so its mere elimination distinguishes neither the Evans and Marciniak model nor that of Deutsch and Wills (discussed next) from McCall's model; they are distinct only in the new factors added to them.

5.4 Deutsch and Wills' quality model

Deutsch and Wills' quality model emerged in 1988 as an enhanced version for McCall's quality model [29]. The relationship between main characteristics and sub-characteristics is many-to-many. This model contains 15 characteristics that are grouped into four categories: operating (functional), performance, change and management [30].

Deutsch and Wills eliminated testability from McCall's model [29] and added five new characteristics, namely verifiability, expandability, safety, manageability and survivability [30]. Verifiability and expandability appear in Evans and Marciniak's quality model; therefore, Deutsch and Wills' model is essentially an Evans and Marciniak quality model with three added characteristics: safety, manageability and survivability. In other words, it is a version of McCall's model with five new characteristics. Both Evans and Marciniak's quality model and Deutsch and Wills' quality model are alternative quality models [30][19] because they have been derived from McCall's model.

5.5 FURPS

The FURPS quality model was introduced by Robert Grady and Hewlett-Packard Co. in 1992, and was later extended to become FURPS+ by IBM Rational Software [31]. The characteristics of FURPS are classified into two categories, functional (F) and non-functional, according to user requirements (URPS); the model focuses on user requirements and disregards the perspective of the developer [31]. The aforementioned categories are used to assess product quality and product requirements[25]. The structure of this model is the same as that of McCall's and Boehm's quality models, but it is less well-known and less regarded than those models [28][25]. According to [32], the FURPS criteria are categorised into five sections, namely functionality, usability, reliability, performance and supportability; each category has its attributes. [10] has stated that FURPS+ covers 218 non-functional requirements. However, this model does not cover architectural integrity [22]. [19] explain that the FURPS+ model covers four characteristics, namely design requirements, implementation requirements, interface requirements and physical requirements.

[22] points out that FURPS has some drawbacks; for example, domain-specific attributes are not addressed in this model. Similarly, portability is not taken into consideration [25]. However, [32] add portability to their list of FURPS+ attributes as it is part of supportability. Moreover, [31] argue that maintainability is also not taken into account as a key characteristic of this model. In addition, [18] points out that FURPS is a quality model for a specific purpose, since it was built by IBM Rational Software Company for the company's own use. In others words, it is not a general basic model, but one built for a specific company and use. However, arguing that FURPS+ is not a basic model bias appears untenable given it is used widely within the software industry even if it is less known compared to McCall and Boehm quality models.

5.6 Dromey's quality model

Dromey's model, introduced in 1995, is similar to McCall's, Boehm's and the FURPS+ quality models [28] but is more recent [26]. [25] state that it was designed to work generically for a wide range of systems. Therefore, the idea behind the model is to focus on product quality because each software product has different attributes in terms of quality evaluation from other software products [26]. According to [33] the proposed model set up a linkage between tangible product characteristics and less tangible quality attributes[26]. It has been argued that this model concentrates on the relationship between quality attributes and sub-attributes [26].

[22] argues that Dromey proposed a framework for evaluating three things: definition of requirements, design, and implementation; as a result, the framework consists of three models: a requirement quality model, a design quality model and an implementation quality model. Dromey's quality model is divided into two layers, namely high-level attributes and subordinate attributes [31]; it defines seven quality attributes [26]. Moreover, [22] states that high-level product attributes for the implementation model cover four attributes, namely correctness (functionality, reliability); internal attributes (maintainability, efficiency, reliability); contextual attributes (maintainability, reusability, portability, reliability); and descriptive attributes (maintainability, reusability, portability, usability). [22] mentions one disadvantage of this model: maintainability and reliability cannot be evaluated before the software is in operational. Furthermore, domain-specific attributes are not addressed in this model.

5.7 ISO/IEC 9126 quality model

ISO/IEC 9126 is a standard developed by Subcommittee SC7 - Software and Systems Engineering of International Organization for Standardization [34]. This standard was designed in order to build a model for software by providing collections of guidelines and measurements for software characteristics[25].

ISO 9126 is a respected model among other software quality standards[25]. It is derived from McCall's quality model [26], Boehm's quality model [28] and the FURPS model [22].

According to [25], ISO 9126, Version 4 (revised version) defines three aspects, namely quality in use and internal quality and process quality [32].

The model contains six characteristics of product quality [26], and each one of them is divided into sub-characteristics [17]. The high-level of this model contains quality characteristics (six characteristics) and the low-level contains software quality criteria (21 sub-characteristics) [31][34]. The main characteristics of this model are portability, maintainability, efficiency, functionality, reliability and usability [34].

[34] state that the ability to use ISO/IEC 9126 across many systems is a major strength of this model. [35] point out that a comparison between this model with other models shows that ISO/IEC 9126 is more complete than other models.

Even though ISO/IEC 9126 is the best available model to date [37], it does have some drawbacks. [34] outline the main drawbacks of ISO 9126. In their opinion, the major disadvantage of this model is generality; moreover, an important shortcoming of this model is that it does not describe business manager needs that are defined in the return investment and sustainability quality factors. Traceability of software and consistency of data are not outlined in this model, and the measurement method is not included. Furthermore, [37] argued that there is inconsistency in this model between classical terminology in measurement and the computing domain, which leads to increased concern about the validity of suggested measures. Moreover, there is a conflict between ISO/IEC 15939 measurements information model with regard to several cases that have arisen with ISO/IEC 9126: some measures are not identified and/or are classified as base or derived measures. Furthermore, ISO/IEC 9126 does not provide guidelines for setting up a good predication system [17].

5.8 ISO/IEC 25000 quality model

ISO/IEC 25000 is a standard for SQuaRE which an acronym for System and software Quality Requirements and Evaluation [38]. The first version was published in 2005 and was replaced by the second version in 2014 [38].

According to [38][39], SQuaRE is divided into five standards; our focus in this study in the quality model division ISO/IEC 2501n.

ISO/IEC 25010 is the replacement for ISO/IEC 9126 and was officially introduced in 2010 [40]. The goal of ISO/IEC 25010 quality model is to provide guidance during the development of software products with a specification and evaluation of quality requirement [21]. [39] developed this model with the mandate of providing a quality model for computer systems and software products, based on quality in use, data and the provision of guidance on the use of the quality models.

Since ISO/IEC 25010 is a revised version of ISO/IEC 9126, it contains many changes and additions. Firstly, internal and external quality factors have been combined into a product quality factor in ISO/IEC 25010. Secondly, security has become a main characteristic rather than the sub-characteristic it was in ISO/IEC 9126. Thirdly, compatibility has been added as a new main characteristic [40]. This brings the total number of main characteristics to eight, compared to six for ISO/IEC 9126. These eight characteristics are further divided into several sub-characteristics [41].

According to [42], ISO/IEC 25010 is based on two models, namely the quality in use model (five characteristics) and the product quality model (eight characteristics). The quality in use model comprises effectiveness, efficiency, satisfaction, freedom from risk and context coverage. The product quality model contains the following characteristics: functional suitability, performance efficiency, compatibility, usability, reliability, security, maintainability and portability [41].

5.9 IEEE Std. 1061–1992, IEEE Std. 1061–1998, IEEE Std. 730–2014

IEEE Std. 1061 was approved in 1992 as a standard dealing with software quality metrics [43]; the IEEE Std. 1061–1998 is a revision of the IEEE Std. 1061–1992 [44]. According to the IEEE Standards

Association [45], the IEEE Std. 1061–1998 is the active version and it was reaffirmed in 2009, whereas the IEEE Std. 1061–1992 is now an inactive version. The IEEE Std 1061–1998 is a standard for software quality metrics methodology [46]; it is a framework, and its methodology is applicable to all types of software in all phases of the software life cycle [47]rather than being prescribed for particular metrics [44]. According to [43], it is vital for users to be aware that this is a process standard and does not mandate specific metrics for use. It is a quality model with associated framework [48].

Using this standard enables an organisation to carry out the following tasks [43]:

- Assess achievement of quality goals.
- Establish quality requirements for a system at its outset.
- Establish acceptance criteria and standards.
- Evaluate the level of quality achieved against the established requirements.
- Detect anomalies, and highlight potential problems
- Predict future quality levels.
- Monitor changes in quality when software is modified.
- Assess the ease of evolutionary change to the system.
- Validate a metric set.

According to [47], the main goal for this standard is to 'establish software quality requirements, identify software quality metrics, implement the software quality metrics, analyze the software metrics results and validate the software quality metrics' (p. 811). Three categories are covered by this standard, namely measures, process and target and goals; however, standard coverage is not complete or comprehensive since this is not the main purpose for this standard [44].

The metrics framework has been designed to be flexible. Addition, deletion and modification of quality factors and sub-factors are allowed in this framework, thereby allowing expansion of various sub-levels [43]. [49] states that using metrics does not preclude the need for human judgment in many cases.

The IEEE Std. 1061–1992 provides a usable list of definitions for factors and sub-factors; however, this illustration is mentioned in the Annex section, which is not part of the standard. It is nonetheless interesting to examine this list of definitions, since it mentions that these factors and sub-factors are not exhaustive and that other possible metrics, factors and sub-factors can be used. The main factors here are efficiency, functionality, maintainability, portability, reliability and usability[49][50,p.20]. The list of definitions for Std. 1061–1992 should be taken into consideration as it is included in the IEEE's official documentation; [51] also state that in order to identify general quality characteristics and related criteria, the classic models of McCall, Boehm, ISO 9126 and the IEEE 1061 should be considered. [44] argue that there is an overlap between this standard and the IEEE 610.12 due to the different definitions for terminologies; even though both models are produced by the IEEE, the definition for metrics is different in these two standards.

A further standard that should be taken into consideration is IEEE 730–2014. This is an active standard for the Software Quality Assurance Plans Working Group [52]. Based on the IEEE 1219 [29], it can be described an iterative process for the management and execution of software maintenance activities [20]. Extra supplementary standards are defined, such as software quality assurance, verification and validation, and software configuration management which contains associated processes (external processes) [29]. This standard contains six major characteristics, namely efficiency, reliability, functionality, maintainability (supportability), portability and usability [20]. A closer look at the IEEE 730 Std. and its characteristics and sub-characteristics shows that it shares the same structure as the IEEE 1061, which is an example for that standard in its appendixes.

5.10 Discussion of quality models

In his discussion of hierarchical quality models, [53] points out several issues that have been raised by critics, such as the fact that decomposition principles for quality models, which are used for quality characteristics, are usually vague and ambiguous. The accuracy for measuring software quality characteristics is usually not high enough for them to be measured directly. In a study of the adoption of these models among companies, Wagner found that 28 percent of companies had used these models, whereas 71 percent had developed their own models. This suggests that there is a real need for customisation in the use of quality models.

Many studies have been conducted to compare software quality models; one example is provided by [21], which compares basic software quality models. Another study, conducted by [46], compares the criteria of three models, namely McCall's, Boehm's and ISO 9126. Further comparisons have been conducted by [18] and by [25].

An analysis of these comparative studies shows that the selection of characteristics varies from study to study. For example, in Miguel, et al.'s study[21], 27 characteristics of ISO 25010 are selected, whereas this model has eight characteristics. For Boehm's model, six factors in Miguel, et al.'s study, whereas 17 characteristics in Milicic's study [46], and eight in Davuluru's study[25]. However, as shown in Al-Qutaish's study[18], Boehm shows seven main characteristics for this model.

These studies, therefore, appear contradictory as a result of differences in the understanding and selection of criteria. This may be caused by different definitions of the terminologies used for defining these characteristics. [31] discuss this issue in their comparison of various studies. [31] explain that the comparisons are not accurate due to various perspectives at the factor level. These occur as a result of inconsistency in definitions of quality factors; this, in turn, leads to poor understanding and subsequently low adoption rates of these models.

Other studies that compare quality models have been conducted; however, some of them do not state their methodology for these comparisons. Examples are [18],[20],[54], [25] and [16].

6 Results of the studies

After an investigation of these models, a comparative study was conducted to enumerate all the main characteristics of the best-known quality models; this is seen in Table 2 (in appendixes). The

table shows that reliability and usability are characteristics that exist in all models, and portability and maintainability exist in nearly all (eight of nine models). The total number of characteristics in the nine models under investigation is 23. The ranking for these characteristics can be seen in Figure 1.

As Figure 1 shows, the most common characteristics in these models are reliability, usability, portability, maintainability, efficiency, functionality and reusability; these appear in nearly half of the models. This does not imply that security is a marginal characteristic; it simply suggests that security in some models is a sub-characteristic of functionality or is another characteristic. According to [55], one of the problems related to software quality models is that some are now quite old; given recent breakthroughs in technology, the level of importance for some of the characteristics featured in the models may change as a result of changing priorities. The fact that priorities within these models have changed is a reflection of the demands placed by users and industry for certain characteristics. ISO 25010 considers security to be a major characteristic as it is an essential feature today. Quality cannot be achieved without security; for example, an insecure bank application will be not adopted by customers if it is considered too risky and a potential threat to their personal security. Likewise, if security is violated, the bank itself risks becoming a victim of theft, which in turn leads to higher bank fees and possible bankruptcy. It is, therefore, impossible to achieve software quality without considering reliability and security [26]. [56]state that security has attracted increased attention in the software engineering field as a result of an increase in security threats which have led to global identity theft and financial fraud. Moreover, national security is also a key consideration; this leads to increased efforts to protect the community against harm and consequently to an increased involvement of software in each facility.

Other key characteristics in some models are testability, interoperability and understandability; testability is a sub-characteristic of maintainability, and understandability is a sub-characteristic of usability in ISO 9126, and it has been renamed to be "Appropriateness recognizability" in ISO 25010 which also falls under usability [54]. Interoperability is a sub-characteristic of functionality in ISO 9126, whereas it is a sub-characteristic of compatibility in ISO 25010. As a result, the coverage for ISO 25010 seems to be more extensive than for other models due to its consideration of security as a main characteristic. Indeed, [21] state in their quality models comparison that ISO 25010 is the most complete basic model due to its coverage of characteristics.

The most common characteristics in our study are reliability, usability, portability, maintainability, efficiency, functionality and reusability. Reliability and usability are found in all models, whereas portability and maintainability are found in all models except FURPS+; maintainability is a sub-characteristic of supportability, and FURPS was criticised for not considering portability [25]. Efficiency appears in seven out of nine models. It is considered a major characteristic in ISO 25010 on quality-in-use model and a sub-characteristic of performance in FURPS. Functionality is absent from only four of the nine models: those of McCall, Boehm, Deutsch and Willis, and Evans and Marciniak. Reusability is found in almost half of the quality models in this study. FURPS, Boehm's model, ISO 9126, ISO 25010 and the IEEE 1061 and 730 do not address reusability as major characteristic; however, in the IEEE 1061 and 730, it is included as a sub-characteristic of portability. In other models, it is addressed as a major characteristic. A further study was conducted, which supports our findings by providing the same results. This study comprises a combination of four comparative studies of quality models. In this study, we collected as much as we could from quality models, regardless of their classification. In other words, a combination of all types of models – basic, tailored and commercial – was analysed to discover the frequency of each characteristic in the study. The first and second study covered 27 models; this is outlined by Dreheeb et al. in two articles [57][16]. A further two models, those of Sharma and Kumar, have been added to the 27 models from [20]. [19] carried out a comparative study of basic, well-known software quality models; their report adds a model that was not included in previous studies, namely the Software Engineering Institute's 1995 model.

Finally, we added the SPARDAT commercial banking model to the set of quality models because it was designed for software development in the banking environment [31] [55]. The total number of these models is 31, which cover 35 software quality characteristics. Results are seen in Table 4 (in appendixes).

It can be seen from Figure 2 that usability is the most frequent characteristic, appearing in almost all models; this is followed by reliability, as seen in Figure 2.

The result of this study support the findings of the former study but in a different order. However, if we apply the same conditions for choosing the nominated characteristics, reusability should be excluded because the total number of models in this study is 31,

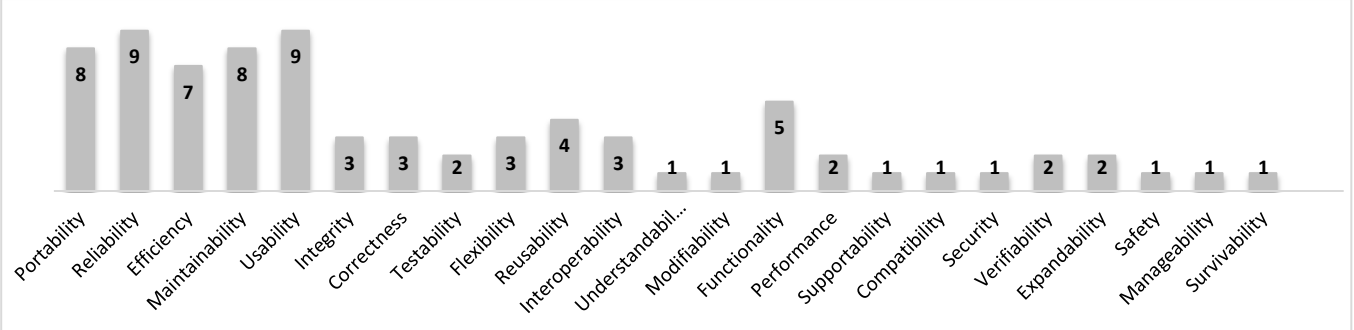

Figure 1: Frequency of occurrence of characteristics over nine SQMs.

and the frequency of reusability across the models is less than half. As a result, the most frequent characteristics of quality models, in order of frequency, are as follows: usability (28), reliability (25), maintainability (24), efficiency (23), portability (23) and functionality (16). Moreover, reusability (12) and flexibility (11) can be considered candidates for major characteristics of software quality since they appear in over a third of these models.

[13] conducted an investigation of the most common NFR characteristics, finding that performance, reliability, usability, security and maintainability are the most frequent NFRs in quality models. [10] outlines the results of a study conducted to analyse criteria in eight historical models with regard to NFRs. These models are Boehm's model (1976), Cavano and McCall (1978), McCall and Masumoto (1980), Bowen (1985), Grady and Caswell (1987), Grady (1992), Blundell (1997), Somerville (2007) and ISO/IEC Std. 25010 (2011). The total collected NFRs criteria is 96 NFRs. The result shows that reliability, maintainability and usability are most frequent in these models, followed by efficiency, interoperability and portability. This result supports our study but with minor differences in the order for the top three characteristics. Half of these models include accuracy, completeness, consistency, correctness, integrity, testability, modularity, operability and reusability.

Considering the characteristics identified by Adams' study [10], it can be seen that testability and modularity belong to maintainability according to ISO 9126, ISO 25010 and McCall's model. Operability belongs to usability according to ISO 9126, Deutsch and Willis' model, McCall's model and ISO 25010. Moreover, accuracy belongs to reliability according to Boehm's, McCall's and Deutsch and Willis' models, whereas it is a sub-characteristic of functionality in ISO 9126 and IEEE 730 and 1061. Furthermore, integrity belongs to reliability according to Boehm's model, whereas in the SEI model and ISO 25010, it belongs to security. Interoperability is a sub-characteristic of functionality according to ISO 9126, and IEEE 730 and 1061, whereas it appears as a sub-characteristic of compatibility in ISO 25010. Completeness is a sub-characteristic of functionality in IEEE 730 and1061 and ISO 25010, whereas it belongs to correctness in Deutsch and Willis' model. Moreover, correctness is a sub-characteristic of functionality in ISO 25010, whereas it is a sub-characteristic of maintainability in IEEE 730 and 1061. Consistency is a sub-characteristic of reliability, verifiability and correctness in Evans and Marciniak's and Deutsch and Willis' model, whereas it is a sub-characteristics of reliability and correctness in McCall's model.

As a result, Adams's study [10] has mixed factors (major characteristics at the high level in the models) with criteria (sub-characteristics at the lower level); after refining them according to various models as argued previously, it is clear that the main characteristics are reliability, usability, maintainability, portability, efficiency and reusability. Functionality is also a main characteristic that covers several sub-characteristics (accuracy, completeness, interoperability and correctness). Therefore, since these sub-characteristics exist in the majority of the models discussed in Adams' study[10], and as explained before, these characteristics are the criteria for functionality, it is necessary to add functionality as a major characteristic that covers all these sub-characteristics in many models. Security also should be considered a major characteristic since integrity has been accounted for, as one of its sub-characteristics.

As a result, the major characteristics that should be taken into consideration based on Adams' study [10] are reliability, usability, maintainability, portability, efficiency, reusability, functionality and security.

Looking at the results of previous studies, it is worth mentioning that there are similarities between the results of these studies and ISO 25010 quality model. All characteristics that have been suggested by the two studies to be the most frequent characteristics appear as main characteristics in ISO 25010. The only characteristic that is not frequent in comparative studies of quality models and that is a main characteristic of ISO 25010 is compatibility, which covers two sub-characteristics, namely interoperability and co-existence. Reusability is a sub-characteristics of maintainability. Figure 3 shows the mapping between our results in previous studies and ISO 25010 quality model.

As a result, it is strongly suggested that ISO 25010 be considered the most complete quality model. It is the most recent quality model, it has gained widespread acceptance in the field of software engineering, and it is a substitute for ISO 9126, a successful quality model.

7 Software Sustainability study results

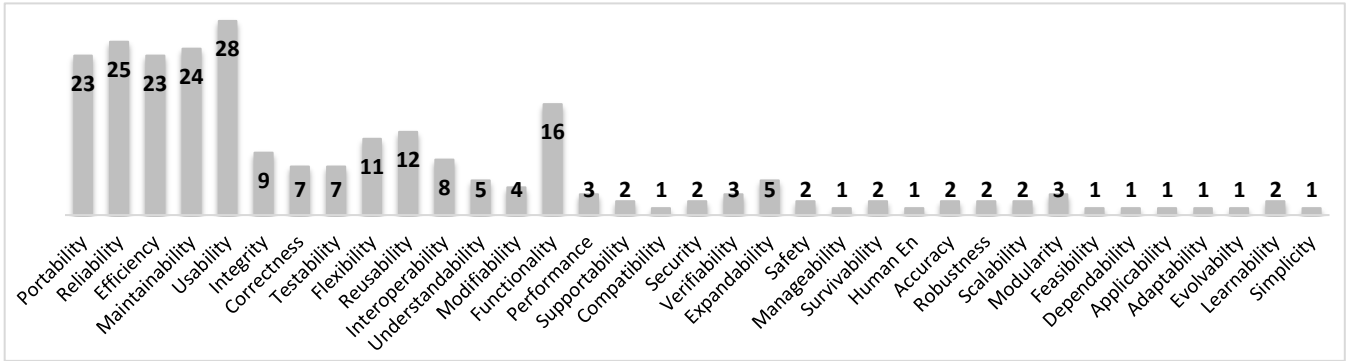

Figure 2: Frequency of occurrence of software quality characteristics in four studies.

Moving from quality models to what the software sustainability literature has stated with regard to the characteristics of software sustainability; [56] (2014) argue that sustainability in software engineering not only encompasses energy efficiency and green IT, but must also consider the second- and third-order impacts of software systems. To do so, sustainability must be considered as a first-class quality attribute and specified as a nonfunctional requirement of IT systems.

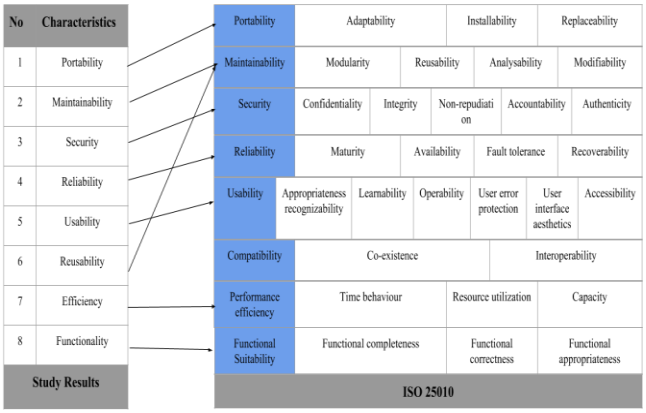

Figure 3: Mapping between study results and ISO 25010

Software sustainability is considered a first-class quality attribute, and it should be taken into consideration like other quality attributes such as security and safety[56]. However, software sustainability is not mentioned in all the basic quality models which have been discussed, having attracted relatively little attention in the past compared to other major attributes such as reliability. The closest similarity is security, which rarely existed in the criteria of older quality models but which is present as a major attribute in ISO 25010, which was developed in 2005. Security has moved from a sub-attribute in ISO 9126 for functionality to a main attribute in ISO 25010, and one which includes five sub-attributes.

The same situation is happening now for sustainability as a result of its importance in the field of software engineering. Cost and time are the most critical attributes in industry, which means that sustainability has become an economic necessity. Socialability (ability to comply with social values) and its major values play an important role in expressing the importance of sustainability, which is reflected in many fields – software engineering is just one of them. Sustainability has been described as a non-functional requirement by [58], which means it is not a new characteristic but it has not gained more attention in the field of software engineering. [56] explain that the consideration of software sustainability in software engineering started in 2010. Therefore, a study was conducted to determine which characteristics of software sustainability have been most frequently mentioned in the literature.

Looking at Table 3 (appendixes), it can be seen that the most frequent characteristics are portability and maintainability, followed by usability. This result is nearly the same with regard to characteristics compared in the previous studies, which have been discussed before; however, the order is different. This reflects the interest of sustainability authors in maintainability and portability more than in usability, as seen in Figure 4.

In previous studies, the results suggest the usability and reliability are the most frequent characteristics because of their importance for software quality; from a sustainability point of view, however, maintainability and portability have attracted more attention than other characteristics due to the changing focus from quality to sustainability, which considers other factors such as cost. There are other characteristics that have attracted little attention for software sustainability, namely efficiency, performance efficiency, reliability, reusability, perdurability and extensibility. Moreover, some of the aforementioned characteristics are sub-characteristics in quality models and have existed in many models; for example extensibility. Extensibility is a sub-characteristic of expandability in Deutsch and Willis' model and Evans and Marciniak's model. Moreover, the FURPS quality model addresses extensibility as a sub-characteristic of supportability, while in the IEEE 730 and 1061, extensibility is a sub-characteristic of maintainability (supportability). This is an indication of the existence of this sub-characteristic in many quality models and of the fact that it belongs to various major characteristics.

Poor quality leads to poor software, usually leading to low adoption rates [59], which is against the aims of sustainability. Quality is an important aspect of software sustainability, and software quality characteristics have to be considered in order to sustain software. The most important characteristics for sustainability are different from that of quality characteristics. From the previous studies, we can conclude that the major characteristics of software quality are usability, reliability,

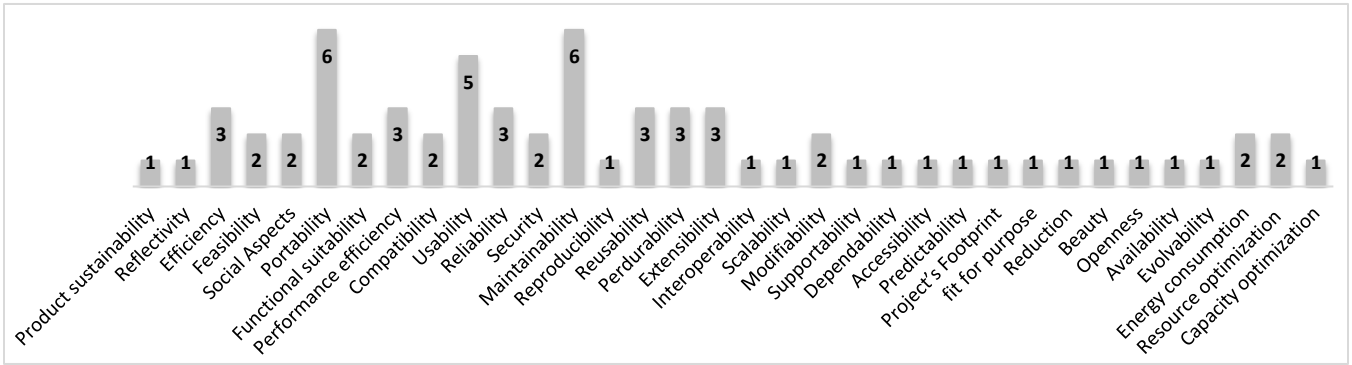

Figure 4: Software sustainability characteristics: frequency of occurrence from literature.

maintainability, portability, efficiency, functionality, reusability and security, whereas the most frequent characteristics of sustainability are maintainability, portability, usability and efficiency, followed by reliability, reusability, perdurability and extensibility.

In the same context, security cannot be ignored as a characteristic for software sustainability. Sustainability of any system is threatened when security is not taken into consideration. Despite the rank of security in the comparison, it is still a valuable characteristic which should be considered as a main characteristic for software sustainability. The overall characteristics for software sustainability are maintainability, portability, usability and efficiency, followed by reliability, reusability, security, perdurability and extensibility.

It can be argued that reusability and extensibility are sub-characteristics of maintainability as ISO 25010 shows that reusability is a sub-characteristic of maintainability. Perdurability is defined by [60] as 'the degree to which a software product can be modified, adapted and reused in order to perform specified functions under specified conditions for a long period of time'. According to this definition, perdurability covers three characteristics, namely reusability, modifiability and adaptability, but with respect to time or software age (durability). Making a connection between durability and sustainability is important; however, some software is designed for a short lifespan. In other words, software can be reusable, modifiable and adaptable, and can take into account cost and time, but its age is very limited so a trade-off should be considered at this stage. This approach overlaps with the durability approach for software sustainability, which is a narrow approach to consider sustainability in software industry.

As a result, two approaches are suggested to cover characteristics of technical dimensions for software sustainability. The first approach is to consider reusability and extensibility sub-characteristics that could be covered by maintainability in ISO 25010. Perdurability could also be covered by maintainability and portability since it consists of three characteristics as mentioned by [60]. Reusability and modifiability are covered by maintainability while adaptability is covered by portability. ISO 25010 can cover all software sustainability characteristics with two more vital characteristics, namely functional suitability and compatibility. Functionality is a stakeholder demand. If the software is not functional, it can be argued that its adoption rate will drop accordingly. Compatibility can be a candidate for software sustainability.

However, some of these characteristics can be covered by others from the same list. This brings us to another argument, which is how the software sustainability ranking of these characteristics reflects how the software sustainability scientific community considers these characteristics. An illustration for that is reusability, which is usually considered within maintainability; however, some software sustainability authors consider it a main characteristic, not a sub-characteristic of another characteristic. Considering the aforementioned argument, adopting a standard which provides full coverage for software sustainability characteristics could be beneficial, to avoid reinventing the wheel. As a result, ISO 25010 can provide abbreviated coverage for software sustainability characteristics.

8 Conclusion

This paper has taken an in-depth look into the literature regarding the differences between software quality characteristics and software sustainability characteristics. An investigation and comparison have been conducted for basic and tailored software quality models, concluding that the most frequent characteristics are reliability, usability, portability, maintainability, efficiency, functionality, reusability and security.

A further investigation followed analysing the most cited paper regarding software sustainability characteristics. The study concludes that the most frequent characteristics are maintainability, portability, usability and efficiency, followed by reliability, reusability, security, durability and extensibility. The order of the most important characteristics for sustainability is different from that of quality characteristics. Therefore, the technical dimension of software sustainability can utilise ISO 25010.

ACKNOWLEDGMENTS

The authors acknowledge the award of Ministry of Education (MoE) and Technical and Vocational Training Corporation (TVTC) in Saudi Arabia to Sulaiman Aljarallah to allow this research to be undertaken.

REFERENCES

[1] C. Leyh, M. Rossetto, and M. Demez, "Sustainability management and its software support in selected Italian enterprises," Comput. Ind., vol. 65, no. 3, pp. 386–392, Apr. 2014.

[2] C. Venters, C. Jay, L. M. S. Lau, M. K. Griffiths, V. Holmes, R. R. Ward, J. Austin, C. E. Dibsdale, and J. Xu, "Software sustainability: The modern tower of babel," in 3rd Workshop RE4SuSy, 2014, pp. 7–12.

[3] R. W. Kates, "Readings in Sustainability Science and Technology," Cambridge, MA, 213, 2010.

[4] L. Sheldrick, "Designing Ubiquitous Sustainability into Product Design Processes," Loughborough University, 2015.

[5] S. S. Mahmoud and I. Ahmad, "A green model for sustainable software engineering," Int. J. Softw. Eng. its Appl., vol. 7, no. 4, pp. 55–74, 2013.

[6] B. Penzenstadler, V. Bauer, and A. Fleischmann, "Seminar: Sustainability in Software Engineering." Technische Universit¨, munchen, pp. 1–5, 2011.

[7] M. Dick, S. Naumann, and N. Kuhn, "A Model and Selected Instances of Green and Sustainable Software," in What Kind of Information Society? Governance, Virtuality, Surveillance, Sustainability, Resilience, Springer, 2010, pp. 248–259.

[8] G. Rodriguez-Navas, L. Duboc, S. Betz, R. Chitchyan, B. Penzenstadler, and C. C. Venters, "Safety vs. Sustainability Design: Analogies, Differences and Potential Synergies," in workshop RE4SuSy, 2015, pp. 25–34.

[9] S. A. Koçak, G. I. Alptekin, and A. B. Bener, "Integrating Environmental Sustainability in Software Product Quality," in Re4SuSy, 2015, pp. 17–24.

[10] K. M. Adams, Nonfunctional Requirements in Systems Analysis and Design, vol. 28. Cham: Springer International Publishing, 2015.

[11] I. Sommerville, Software Engineering, 8th. Boston: Addison-Wesley, 2007.

[12] S. Easterbrook, "Lecture 16: Non-Functional Requirements (NFRs)," Toronto University, CS Dept., 2004. [Online]. Available: www.cs.toronto.edu/~sme/CSC340F/slides/16-NFRs.pdf. [Accessed: 29-8-2016].

[13] D. Mairiza, D. Zowghi, and N. Nurmuliani, "An investigation into the notion of non-functional requirements," in SAC '10, 2010, vol. 1, no. 1, p. 311.

[14] J. Eckhardt, A. Vogelsang, and D. M. Fernández, "Are 'non-functional' requirements really non-functional?," in Proceedings of the 38th International Conference on Software Engineering - ICSE '16, 2016, no. MAY, pp. 832–842.

[15] S. Thapar, P. Singh, and S. Rani, "Challenges to the Development of Standard Software Quality Model.," Int. J. Comput. Appl., vol. 49, no. 10, pp. 1–7, 2012.

[16] A. E. Dreheeb, N. Basir, and N. Fabil, "Comparative Study of Quality Models," Int. J. Comput. Sci. Electron. Eng., vol. 4, no. 1, pp. 35–39, 2016.
[17] J. Tate, "Software Process Quality Models: A comparative evaluation," University of Durham, 2003.
[18] R. Al-Qutaish, "Quality models in software engineering literature: an analytical and comparative study," J. Am. Sci., vol. 6, no. 3, pp. 166–175, 2010.
[19] B. Youness, M. Abdelaziz, B. Habib, and M. Hicham, "A Comparative Study of Software Quality Models," IJCSI Int. J. Comput. Sci. Issues, vol. 10, no. 6, pp. 309–314, 2013.
[20] S. K. Dubey, S. Ghosh, and A. Rana, "Comparison of Software Quality Models: An Analytical Approach," Int. J. Emerg. Technol. Adv. Eng., vol. 2, no. 2, pp. 111–119, 2012.
[21] J. Miguel, D. Mauricio, and G. Rodríguez, "A Review of Software Quality Models for the Evaluation of Software Products," Int. J. Softw. Eng. Appl., vol. 5, no. 6, pp. 31–53, Nov. 2014.
[22] D. Jamwal, "Analysis of Software Quality Models for Organizations," Int. J. Latest Trends Comput, vol. 1, no. 2, pp. 19–23, 2010.
[23] M. Klas, C. Lampasona, and J. Munch, "Adapting Software Quality Models: Practical Challenges, Approach, and First Empirical Results," in 2011 37th EUROMICRO-SEAA 2011, Oulu, 2011, pp. 341–348.
[24] S. Sandhu, S. McKenzie, and H. Harris, Linking Local and Global Sustainability, 1st., vol. 4. Dordrecht: Springer Netherlands, 2014.
[25] T. Davuluru, J. Medida, and V. S. . Reddy, "A study of software quality models," in ICAETR , 2014, pp. 1–8.
[26] B. Singh and S. P. Kannojia, "A Review on Software Quality Models," in CSNT, 2013, pp. 801–806.
[27] D. Galin, Software Quality Assurance - From theory to implementation, 1st. Harlow, Essex: Pearson Education Limited, 2004.
[28] P. Berander, L. Damm, J. Eriksson, T. Gorschek, K. Henningsson, P. Jönsson, S. Kågström, D. Milicic, F. Mårtensson, K. Rönkkö, and P. Tomaszewski, Software quality attributes and trade-offs, 1st,Blekinge Institute of Technology, 2005.
[29] S. Awan, F. Malik, and A. Javed, "An Efficient and Objective Generalized Comparison technique for Software Quality Models," Int. J. Mod. Educ. Comput. Sci., vol. 7, no. 12, pp. 57–64, Dec. 2015.
[30] B. Vintila, "Quality Standards in Open Source Lifecycle," Open Source Sci. J., vol. 2, no. 1, pp. 46–55, 2010.
[31] A. B. AL-Badareen, M. H. Selamat, M. A. Jabar, J. Din, and S. Turaev, "Software Quality Models: A Comparative Study," in Communications in Computer and Information Science, Berlin: Springer Berlin Heidelberg, 2011, pp. 46–55.
[32] D. Hutchison and J. C. Mitchell, Lecture Notes in Computer Science. 2012.
[33] R. G. Dromey, "A model for software product quality," IEEE Trans. Softw. Eng., vol. 21, no. 2, pp. 146–162, 1995.
[34] R. Djouab and M. Bari, "An ISO 9126 Based Quality Model for the e-Learning Systems," Int. J. Inf. Educ. Technol., vol. 6, no. 5, pp. 370–375, 2016.
[35] B. Behkamal, M. Kahani, and M. K. Akbari, "Customizing ISO 9126 quality model for evaluation of B2B applications," Inf. Softw. Technol., vol. 51, no. 3, pp. 599–609, Mar. 2009.
[36] ISO/IEC, "ISO/IEC 9126-1:2001," 2016. [Online]. Available: www.iso.org/iso/catalogue_detail.htm?csnumber=22749. [Accessed:30-8-2016].
[37] D. St-Louis and W. Suryn, "Enhancing ISO/IEC 25021 quality measure elements for wider application within ISO 25000 series," in IECON 2012 - 38th Annual Conference on IEEE Industrial Electronics Society, 2012, pp. 3120–3125.
[38] ISO/IEC, "ISO/IEC 25000:2014(en)," ISO/IEC, 2014. [Online]. Available: www.iso.org/obp/ui/#iso:std:iso-iec:25000:ed-2:v1:en. [Accessed: 25-5-2016].
[39] ISO/IEC, "ISO/IEC 25010:2011(en)," ISO/IEC, 2011. [Online]. Available: www.iso.org/obp/ui/#iso:std:iso-iec:25010:ed-1:v1:en. [Accessed: 25-5-2016].
[40] A. Adewumi, S. Misra, and N. Omoregbe, "Evaluating Open Source Software Quality Models Against ISO 25010," in 2015 IEEE International Conference on (CIT/IUCC/DASC/PICOM), Liverpool, 2015, pp. 872–877.
[41] T. L. Alves, P. Silva, and M. S. Dias, "Applying ISO/IEC 25010 Standard to Prioritize and Solve Quality Issues of Automatic ETL Processes," 2014 IEEE Int. Conf. Softw. Maint. Evol., no. JANUARY, pp. 573–576, 2014.
[42] ISO/IEC, "Systems and software engineering -- Systems and software Quality Requirements and Evaluation (SQuaRE) -- System and software quality models," 2011. [Online]. Available: www.iso.org/iso/catalogue_detail.htm?csnumber=35733.[Accessed: 31-8-2016].
[43] IEEE SA, "IEEE Standard for a Software Quality Metrics Methodology," IEEE Std 1061-1998, vol. no, 1998.
[44] F. García, M. F. Bertoa, C. Calero, A. Vallecillo, F. Ruíz, M. Piattini, and M. Genero, "Towards a consistent terminology for software measurement," Inf. Softw. Technol., vol. 48, no. 8, pp. 631–644, Aug. 2006.
[45] IEEE-SA, "1061-1998 - IEEE Standard for a Software Quality Metrics Methodology," 2016. [Online]. Available: www.standards.ieee.org/findstds/standard/1061-1998.html.[Accessed:01-9-2016].
[46] D. Milicic, "Software Quality Models and Philosophies," in Software Quality Attributes and Trade-Offs, L. Lundberg, M. Mattsson, and C. Wohlin, Eds. Blekinge Institute of Technology, 2005.
[47] J. De Oliveira, K. De Oliveira, and A. Belchior, "Measurement Process: A Mapping Among CMMI-SW, ISO/IEC 15939, IEEE Std 1061, Six Sigma and PSM," in 2006 International Conference on Service Systems and Service Management, 2006, vol. 1, no. 2002, pp. 810–815.
[48] Y. Choi, S. Lee, H. Song, J. Park, and S. Kim, "Practical S/W Component Quality Evaluation Model," in 2008 10th International Conference on Advanced Communication Technology, 2008, pp. 259–264.
[49] M. Barbacci, "Software Quality Attributes: Modifiability and Usability," Software Engineering Institute, Carnegie Mellon ..., 2004. [Online]. Available: www.ieee.org.ar/downloads/Barbacci-05-notas1.pdf.[Accessed: 01-9-2016].
[50] IEEE SA, "IEEE Standard for a Software Quality Metrics Methodology," IEEE Std 1061-1992. IEEE, 1993.
[51] J. J. Dujmović and H. Nagashima, "LSP method and its use for evaluation of Java IDEs," Int. J. Approx. Reason., vol. 41, no. 1, pp. 3–22, Jan. 2006.
[52] IEEE, "730-1998 - Standard for Software Quality Assurance Plans," 2014. [Online]. Available: https://standards.ieee.org/findstds/standard/730-1998.html. [Accessed: 12-9-2016].
[53] S. Wagner, Software Product Quality Control: Springer Berlin, 2013.
[54] Suman and M. Wadhwa, "A Comparative Study of Software Quality Models," Int. J. Comput. Sci. Inf. Technol., vol. 5, no. 4, pp. 5634–5638, 2014.
[55] R. Fitzpatrick, "Software quality: definitions and strategic issues," 1996.
[56] B. Penzenstadler, A. Raturi, D. Richardson, and B. Tomlinson, "Safety, Security, Now Sustainability: The Nonfunctional Requirement for the 21st Century," IEEE Softw., vol. 31, no. 3, pp. 40–47, May 2014.
[57] A. E. Dreheeb, N. Basir, and N. Fabil, "Impact of System Quality on Users' Satisfaction in Continuation of the Use of e-Learning System," Int. J. e-Education, e-Business, e-Management e-Learning, vol. 6, no. 1, pp. 13–20, 2016.
[58] L. Chung, B. A. Nixon, E. Yu, and J. Mylopoulos, Non-Functional Requirements in Software Engineering, vol. 5600, no. March 1999. Boston: Springer US, 2000.
[59] X. Zhang, T. F. Stafford, J. S. Dhaliwal, M. L. Gillenson, and G. Moeller, "Sources of conflict between developers and testers in software development," Inf. Manag., vol. 51, no. 1, pp. 13–26, Jan. 2014.
[60] C. Calero, M. A. Moraga, and M. F. Bertoa, "Towards a Software Product Sustainability Model," CoRR abs/1309.1640.

Appendixes

Table 2: Comparison of various software quality models

Software Quality Models											
No	Characteristics	McCall's	Boehm's	Evans & Marciniak	Deutsch & Wills	FURPS (+)	Dromey's	ISO 9126	ISO 25010	IEEE 1061	Total
1	Portability	✓	✓	✓	✓		✓	✓	✓	✓	8
2	Reliability	✓	✓	✓	✓	✓	✓	✓	✓	✓	9
3	Efficiency	✓	✓	✓	✓		✓	✓		✓	7
4	Maintainability	✓	✓	✓	✓		✓	✓	✓	✓	8
5	Usability	✓	✓	✓	✓	✓	✓	✓	✓	✓	9
6	Integrity	✓		✓	✓						3
7	Correctness	✓		✓	✓						3
8	Testability	✓	✓								2
9	Flexibility	✓		✓	✓						3
10	Reusability	✓		✓	✓		✓				4
11	Interoperability	✓		✓	✓						3
12	Understandability		✓								1
13	Modifiability		✓								1
14	Functionality					✓	✓	✓	✓	✓	5
15	Performance					✓			✓		2
16	Supportability					✓					1
17	Compatibility								✓		1
18	Security								✓		1
19	Verifiability			✓	✓						2
20	Expandability			✓	✓						2
21	Safety				✓						1
22	Manageability				✓						1
23	Survivability				✓						1
# characteristics		11	8	12	15	5	7	6	8	6	

Table 3 The presence of software sustainability characteristics in literature

software sustainability characteristics												
No	Characteristics	Naumann model	Calero ISO 25010+S	Calero green model	Hettrick	Souza and Koçak	Kern	Venters	Albertao	Taina	Khamis	Koziolek	SSI	Total
1	Product sustainability	✓												1
2	Reflectivity	✓												1
3	Efficiency	✓					✓		✓					3
4	Feasibility	✓					✓							2
5	Social Aspects	✓					✓							2
6	Portability	✓	✓	✓				✓	✓			✓		6
7	Functional suitability		✓	✓										2
8	Performance efficiency		✓	✓					✓					3
9	Compatibility		✓	✓										2
10	Usability		✓	✓		✓		✓	✓					5
11	Reliability		✓	✓	✓									3
12	Security		✓	✓										2
13	Maintainability		✓	✓		✓		✓				✓	✓	6
14	Reproducibility				✓									1
15	Reusability				✓			✓	✓					3
16	Perdurability		✓	✓			✓							3
17	Extensibility							✓				✓	✓	3
18	Interoperability							✓						1
19	Scalability							✓						1
20	Modifiability								✓			✓		2
21	Supportability								✓					1
22	Dependability								✓					1
23	Accessibility								✓					1
24	Predictability								✓					1
25	Project's Footprint								✓					1
26	fit for purpose									✓				1
27	Reduction									✓				1
28	Beauty									✓				1
29	Openness										✓			1
30	Availability												✓	1
31	Evolvability											✓		1
32	Energy consumption		✓	✓										2
33	Resource optimization		✓	✓										2
34	Capacity optimization			✓										1
Number of characteristics		6	9	9	3	2	4	7	11	3	1	5	3	

Table 4 Comparison of four studies of quality models

Software Quality Models																																		
Sources		(Dreheeb et al., 2016) , (Dreheeb, Basir, & Fabil, 2016b)																										Dubey et al., 20		ounuess et al., 2(	Fitzpatrick, 199		
No	Characteristics	McCall's	Boehm's	Evans & Marc	Deutsch & Wi	FURPS (+)	Dromey's	ISO 9126	ISO 25010	IEEE	AOSQUAMO	Bowen	Murine	Kazman	Khosravi K.	Ghezzi	SATC's	QMOOD 2002	AOSQ 2012	CSDQ 2009	DEQULATTE	UML	Gillies	OOQM	Perry	Bertoa	Alvaro	Rawashdeh	Sharma A	Kumar	SEI 1995	SPARDAT	Total
1	Portability	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓			✓	✓		✓	✓		✓	✓	✓	✓	✓	✓	✓	✓					23
2	Reliability	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓		✓	✓		✓	✓	✓	✓	✓	✓	✓	✓					25
3	Efficiency	✓	✓	✓	✓		✓	✓		✓	✓	✓	✓	✓			✓		✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓				23
4	Maintainability	✓		✓	✓		✓	✓	✓	✓	✓	✓	✓	✓		✓	✓		✓	✓			✓	✓	✓	✓	✓	✓	✓	✓			✓	24
5	Usability	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓				28
6	Integrity	✓		✓	✓							✓	✓			✓						✓	✓		✓									9
7	Correctness	✓		✓	✓							✓	✓										✓		✓									7
8	Testability	✓	✓										✓								✓	✓	✓		✓									7
9	Flexibility	✓		✓	✓							✓	✓	✓	✓	✓		✓					✓		✓									11
10	Reusability	✓		✓	✓		✓					✓	✓		✓	✓		✓			✓		✓		✓									12
11	Interoperability	✓		✓	✓							✓	✓									✓	✓		✓									8
12	Understandability		✓												✓			✓			✓	✓												5
13	Modifiability		✓															✓			✓	✓												4
14	Functionality					✓	✓	✓	✓	✓	✓			✓			✓	✓	✓	✓	✓	✓		✓					✓	✓				16
15	Performance					✓			✓																						✓			3
16	Supportability					✓											✓																	2
17	Compatibility								✓																									1
18	Security								✓																						✓			2
19	Verifiability			✓	✓							✓																						3
20	Expandability			✓	✓							✓						✓			✓													5
21	Safety				✓																										✓			2
22	Manageability				✓																													1
23	Survivability				✓							✓																						2
24	Human En		✓																															1
25	Accuracy															✓						✓												2
26	Robustness														✓						✓													2
27	Scalability														✓						✓													2
28	Modularity														✓			✓			✓													3
29	Feasibility																✓																	1
30	Dependability																														✓			1
31	Applicability																															✓		1
32	Adaptability																															✓		1
33	Evolvabilty																		✓															1
34	Learnability																				✓	✓												2
35	Simplicity																				✓													1
Total Number		11	7	12	15	5	7	6	8	6	6	13	11	6	7	8	8	8	7	6	12	12	11	6	11	5	5	5	6	4	4	3		