Measuring the Unmeasurable Characteristics of Software Product Quality

[1]Jamaiah H. Yahaya,[2]Aziz Deraman
*[1,Corresponding Author]College of Arts and Sciences, University of Northern Malaysia (UUM),
jhyahaya@yahoo.com; jamaiah@uum.edu.my
[2]Vice Chancellor Office, University of Malaysia, Terengganu, a.d@umt.edu.my
doi: 10.4156/ijact.vol2.issue4.10

Abstract

Software quality is evolving beyond static measurement to a wider scope of quality definition. Previous studies have indicated the importance of human aspect in software quality. But software quality models have not included comprehensively this aspect together with the behavioural aspect of the quality. This research has proposed a Pragmatic Quality Factors (or PQF) as a software quality measurement and metrics that includes the human aspects as well as behavioural aspect of quality. The two aspects of quality are essential and necessary as to balance between technical and non-technical (human) facet. In addition, this model provides flexibility by giving priorities and weights to the quality attributes. The priority and weight are essential to reflect business requirement in the real business environment. Therefore, it is more practical that suits with different users and purposes. As for our research, PQF is used for assessment toward software product certification. It is implemented through collaborative perspective approach between users, developers and independent assessor. This paper explains in detail the PQF model and its applications.

Keywords*: software quality model, Pragmatic Quality Factors, Behavioural Attributes, Impact Attributes*

1. Introduction

In the era of 1990s, software quality has been realized as an important element. This era was also known as quality era, which software quality has been quantified and brought to the center of development process. The business's software significant impact to today economy generates consideration in producing good quality software with cost effective development process [22]. At the same time, companies are competing to produce software which are claimed to be good and fulfill user's expectation and requirements. Furthermore, companies unable to provide any justification on the quality of their products to the users and users are left with uncertainties on the standard and quality of the software [7][26]. This raises legal and moral questions: how do we determine the quality of the software being developed? What are the mechanisms to assess and evaluate software products? To what extent is an organisation that develops software responsible for its result?

General expression of how quality is realized in software dealing with "fitness for use" and "conformance to requirements". The term "fitness of use" usually means characteristic such as reliability, functionality, reusability and etc. On the other hand, "conformance to requirements" means that software has value to the users [24]. ISO defines quality as "the totality of features and characteristics of a product or services that bear on its ability to satisfy stated or implied needs" [14]. Peter J. Denning presented his argument that "software quality is more likely to be attained by giving much greater emphasis to customer satisfaction. Program correctness is essential but is not sufficient to earn the assessment that the software is of quality and is dependable" [9]. Software quality and evaluation not only deal with technical aspects but also in dimensions of economic (managers' viewpoint), social (users' viewpoint) and as well as technical (developers' viewpoint) [6].

In many organizations, software is considered as one of the main asset with which the organization can enhance its competitive global position in this global economy era. To remain competitive, software firms must delivers high quality products on time and within budget. At the same time, many complaints have been reported regarding quality of the software. They claim that software quality is not getting improved but deterioration steadily and worsening [28]. Software Engineering Institute's Capability Maturity Model (CMM) (cited in [23]) reports the following quote from a software manager: "I'd rather have it wrong than have it late. We can always fix it later". Therefore, users report

and claim that software is being delivered with bugs that need to be fixed and dissatisfied with the product [9][26].

This paper presents the measurement of software quality based on collaborative perspective approach. It presents the background of this research which includes the state-of-the-art of software quality models and the measurement and metrics in software quality. The discussion moves on to the methodology of this research and follows with the proposed quality factor, Pragmatic Quality Factors, its application and conclusion.

2. Background

2. 1. Software Quality Models

Software and quality are among the most common topic of discussion about computers. Fenton and Pfleeger suggest, "Without an accompanying assessment of product quality, speed of production is meaningless"[11]. This observation has led to the development of software quality model that measure and combine with productivity models. Thus, several quality models are available from literature and the most well know models are McCall, Boehm, FURPS, ISO 9126, Dromey and Systemic.

The McCall quality model is one of the earliest models. The model is usually constructed in a tree-like fashion. The upper branches hold important high-level quality attributes, such as reliability and usability, which want to quantify. Each quality attribute is composed of lower-level criteria [15]. The factors associated with this model are: Correctness, reliability, efficiency, integrity, usability, maintainability, testability, flexibility, portability, reusability and interoperability. In this model factors are not directly measured and therefore a set of metrics are defined to develop the relationship. McCall defines metrics in form of checklist that is used to grade attributes of the software. It is interesting to notice that some of the factors are still relevant and as fresh today as they were in 70's. However McCall model does not cover functionality.

The Boehm model is similar to McCall model that it represents a hierarchical structure of characteristics, each of which contributes to total quality [3]. Boehm model sees the view of software with general utility. It looks at utility from various dimensions, considering the types of user expected to work with the software once it is delivered. General utility is broken down into portability, utility and maintainability. Utility is further broken down into reliability, efficiency and human engineering. Maintainability is in turn broken down into testability, understandability and modifiability. This model is presented in levels called primary uses, intermediate construct and primitive constructs.

Hewlett-Packard developed a set of software quality factors that make up its name FURPS. The FURPS model takes five characteristics of quality attributes - Functionality, Usability, Reliability, Performance, and Supportability. When the FURPS model is used, two steps are considered: setting priorities and defining quality attributes that can be measured [15]. One disadvantage of this model is that it does not take into account the software product's portability [17].

ISO 9126 defines product quality as a set of product characteristics. The characteristics that govern how the product works in its environment are called external quality characteristics. The characteristics relating to how the product is developed are called internal quality characteristics. ISO 9126 indicates six main quality characteristics which associated with several subcharacteristics. ISO 9126 model can be used as a practical approach for defining quality and the questionnaire based method [13]. It has been invented since 1991 and today, it is still being accepted and used in researchers that deal with software quality [1]. However, at the same time it has the disadvantage of not showing clearly how these aspects can be measured [20] and the model only focusing on developer view of the software [19].

Dromey proposes a working framework for building and using a practical quality model to evaluate requirement determination, design and implementation phases. Dromey points out that high level quality attributes, such as maintainability, functionality and reliability, cannot be built into the system. The alternative way to input quality into software is by identifying a set of properties and build them up consistently, harmoniously and fully to provide high level quality [10].

The systemic model is developed by identify the relationship between product-process, efficiency-effectiveness and user-customer to obtain global systemic quality [17]. The disadvantages of this model are that it does not cover the user requirements and conformation aspects.

Analysis from previous quality models have demonstrated that there is different quality characteristics associated with different models. It shows that the main quality characteristics found in majority of the models are: efficiency, reliability, maintainability, portability, usability and functionality, which are presented in more recent models. These characteristics appear in all models and therefore, are considered as essential and vital.

Beside the quality models discussed above, there are several studies that investigate quality measures and related issues. Verner et al. investigates the definition of quality among IS professionals in Hong Kong [27]. The study shows that key characteristics identified by the practitioners are reliability, maintainability, and functionality that are well-documented, efficient and easy to use. Other characteristics supported by Boehm model are seen less important. The latest work is carried out by Peng et. al on evaluation of classifiers for software risk management [34].

Even though there are several models of quality available from literature, it still believes that quality is a complex concept. Quality is the eye of the beholder and it means different things to different people and highly context dependent [16]. Therefore, "Software quality is nothing more than a recipe. Some like it hot, sweet, salty or greasy" [29]. Thus, there can be no single simple measure of software quality acceptable to everyone**.** Literature on software quality shows that there are numbers of characteristics that contributes to the behavioral aspects of quality. A classification of characteristics might be necessary to group characteristics according to importance.

As observed from existing quality models for software product assessment, available identified quality attributes is difficult to meet current requirement and specification. Current and available quality models are much dependent on the usage of the assessment process and development requirement. The earliest models of quality such as McCall, Boehm, FURPS and ISO 9126 are limited to measure of external software characteristics such as reliability, maintainability, portability and functionality which do not consider other necessities needs such as conformance of user requirements and expectation. Software quality was more on customer satisfaction and software correctness was not sufficient to be declaring as good quality without satisfaction by the users [9]. This means that there is a requirement to include measurements of human aspects and the quality impact in the proposed quality model. Integrity as one of the vital attribute in current situation is not considered in previous models.

2. 2. Software Quality Measurement and Metrics

The ultimate goal of software engineering is to produce a high-quality software, application, or product. Measurement is used to assess the quality of the software. Software metric is defined by Gaffney [12] as "objective, mathematical measure of software that is sensitive to differences in software characteristics. It provides a quantitative measure of an attribute which the body of software exhibits". The quantitative assessment is the assessment on certain attributes, which we can measure. Measurements can be used to assist in estimation quality of software product. Without measurement, judgment can be based on subjective assessment. Indicators or metrics provide insight into the product and measure quality indirectly [30].

Software measurement can be categorized into direct measurement and indirect measurement. Direct measurement of the software engineering process includes lines of code (LOC) produced, execution speed, memory size, and defect reported over some period of time. Indirect measurement of products includes functionality, complexity, efficiency, reliability, and many other "-abilities". These characteristics are unmeasurable software quality characteristics. The unmeasurable characteristics are decomposed into several subcharacteristics and metrics to generate a measureable metrics.

IEEE Standard Glossary of Software Engineering Term defines metrics as "a quantitative measure of the degree to which system, component, or process possesses a given attribute". Measure provides a quantitative indication of the extent, amount, dimension, capacity, or size of some attribute of a product. Quality factor defined in this framework is a management-oriented attribute of software that contributes to its quality. In ISO model it is called quality characteristics. In this framework, quality aspect of software is broken down into several factors or characteristics, and further decomposed into

subfactors or also known as subcharacteristics and metrics. Subfactor is a decomposition of a quality factor to its technical components. The metrics, which are at the third level of the software quality framework, are the direct metrics that are used to estimate quality factor of software.

The following discussion aspires to demonstrate the software quality framework with focusing on unmeasurable and measurable aspects of quality characteristics. For an example, functionality in ISO 9126 model is broken down into subfactors which are suitability, accuracy and interoperability. The decomposition of subfactor is at level two of hierarchy. Functionality is considered as unmeasurable characteristics and involved undirect measurement. In order to convert this unmeasurable to a measurable characteristic, subfactors of functionality is decomposed into the higher level in the hierarchy viz the third level. At the third level of the hierarchy the subfactors are decomposed into metrics used to measure software products. The decomposition is shown in Figure 1 and Table 1.

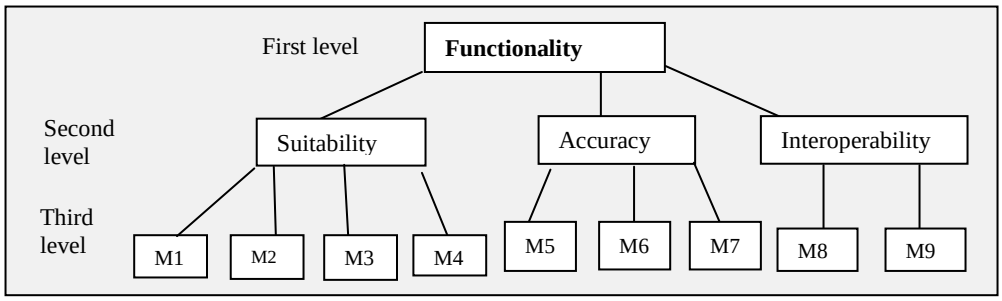

Figure 1. Decomposition of functionality

Table 1. Example of quality factor, subfactors and metrics

Factor: Functionality		
Subfactors	**Metric**	**Measure**
Suitability	M1: Functional Implementation coverage	Are functions described in specification confirmed with functions in executing testing?
	M2: Functional specification stability	Are functions obliged to be changed after testing?
	M3: Functional implementation completeness	Are functions described in specification confirmed with implementation and complete?
	M4: Functional implementation correctness	Are functions described in specification confirmed with implementation and correct?
Accuracy	M5: Incomplete result	Number of incomplete results obtains from the software.
	M6: Incorrect result	Number of incorrect results obtains from the software.
	M7: Unexpected results issued	Are unexpected results issued during running the software?
Interoperability	M8: Data format based for data exchangeability	Number of data format which are approved to be exchanged with other software or system during testing on data exchange
	M9: User's success attempt based for data exchange	Number of turns which user fail to exchange data formats with other software or system

In this example, functionality (unmeasurable characteristic) is decomposed into suitability, accuracy and interoperability. These three subfactors are decomposed to the third level, metrics, which are named as M1, M2, M3, M4, M5, M6, M7, M8, and M9 (refer to Table 1). The decomposition is as follow:

Subfactor -> {metrics}
Suitability -> {M1, M2, M3, M4}

Accuracy ->{M5, M6, M7}
Interoperability ->{M8, M9}

The metrics are measurable and provide measurement to estimate quality. External metrics uses measures of a software product derived from measures of the behaviour of the system of which it is a part, by testing, operating and observing the executable software or system [4]. Thus, data is gathered and required to arrive an indication of quality [11]. Eventually metrics gathered can cost a lot of money [8] and therefore it is suggested to collect practical target data that will produce meaningful inferences.

Evaluating or assessing the quality of software is very important, not only from the perspective of software engineers to determine the quality level of their products but also from a business point of view, such as to make a choice between two similar products. Assessment of product means judging to which the software product meets the quality characteristics. There are three methods of software product assessment: -

- First party assessment. This assessment involves internal product and process evaluation during and after development process. The testing and assessment are done by the developers themselves.
- Second party assessment. Second party assessment method is also referring to acceptance test on product delivery. The assessment and testing is conducted by the developers and users.
- Third party assessment. This method involves independent body assessment and evaluation for example, a testing laboratory or any appointed body.

3. Methodology

The research approach used in this study was deductive approach. Deductive approach was used during the development of the model where theory and concepts of software quality were derived from the literature and empirical findings before the model was applied and tested in real case studies. This consideration of deductive approach was mainly referred to suggestions by Page & Meyer [18] and Trochim [25]. The research approach involved four phases:-

3. 1. Theoretical Study

In this phase current state-of-the-art in the development of software quality and assessment were being reviewed in depth. Based on literature findings in issues and factors affecting software quality, the research continued with questionnaire designed and tested it through pilot study. The data from pilot test was analysed to produce pilot reports and modifications on the items in questionnaires were implemented before the real survey was conducted.

3. 2. Empirical Study

The empirical study involved preparing requirement studies within intended environment, conducting a survey and analyzing data. The survey was conducted to gather data and information from various agencies involved in software development and acquisition in Malaysia. Findings from this phase were used as the basis for producing specification and requirements for proposed software quality model.

3. 3. Model Construction

Based on the empirical and literature findings, an initial software quality model was constructed. The concept, definition and contributing factors to PQF initiatives were used to identify attributes that were required in the assessment of software. This has led to the development of a software quality model, which met current software assessment requirements. The suggested quality model was named pragmatic quality model, which described the relationships between attributes (which mostly unmeasurable) and the measurable metrics. As suggested by Svahnberg et al. [21] in model construction, all variables in the model need to be defined and weighted according to their importance in relation to their influence in software assessment. The formulation of the weight factors that

classified attributes into different levels was also being provided.

3. 4. Application and evaluation

The application of the model was carried out to evaluate the model. These involved conducting and collaborating three case studies in three large organizations in Malaysia. The applications on the case studies tested and validated the proposed model by assessing three systems operating in their environments. As the model evaluation was carried out by the case study, a model refinement was conducted as necessary.

4. Pragmatic Quality Factor (PQF): A Practical Software Assessment Model

The PQF consists of four main components: behavioural attributes, impact attributes, responsibility, and weight. The components are explained in the following sections.

4. 1. The Behavioural Attributes

The behavioural attribute is defined as the external quality characteristic of specific software and how it behaves in the actual operating environment. The behavioural attributes include efficiency, functionality, maintainability, portability, reliability, integrity and usability. The behavioural attributes are derived from ISO 9126 attributes with the integrity aspect included. ISO 9126 model is a generic quality model for any software product but requires some customization and enhancement for particular case [5]. In the age of hackers and firewalls, the importance of integrity aspect has increased. This attribute measure the ability to with-stand attack on its security that comprises of program, data and document. It covers threat and security aspects. Findings from previous survey [31] indicated the importance of integrity in software quality attributes.

In PQF, attributes are decomposed into several subattributes and then a further level of decompositions to associate with directs measurable metrics. Each of the subattributes and metrics comprises of information on interviewees. Examples of the decompositions of attributes are shown in Table 1 and Table 2. These examples show that functionality can be broken down into three subattributes: suitability, accuracy and interoperability. The subattributes are later decomposed to several metrics associated with them and the metrics are measurable to the users or developer of the software. They are also measurable to the external assessor who will also be the independent assessor in the assessment.

Table 2. A decomposition of functionality

Attribute : Functionality		
Subattributes	**Metric**	**Interviewee**
Suitability	Functional Implementation coverage	*User*
	Functional specification stability	*User*
	Functional implementation correctness	*User*
	Functional implementation completeness	*User*
Accuracy	Incomplete result	*User*
	Incorrect result	*User*
	Unexpected results issued	*User*
Interoperability	Data format based for data exchangeability	*User, Developer*
	User's success attempt based for data exchange	*User, Developer*

4. 2. The Impact Attribute

The impact attribute defined in PQF refers to the human aspect of quality toward the product. It illustrates the impact of the software in term of quality to the users and also measures the conformity of software to the user requirement. This attribute is important to balance the quality model between technical measurement of software and human factor [8]. Similar to behavioural attributes, the impact attribute is made up of several subattributes and metrics that show the measurement of the attributes. The impact attribute is decomposed into two distinct subattributes which by means of user perceptions and user requirements. The metrics include measures of popularity, performance, trustworthiness, law and regulation, recommendation, environmental adaptability, satisfaction and user acceptance. Table 3 shows the subattributes and their associated metrics.

Table 3. A decomposition of impact attributes

Attribute : Impact		
Subattributes	**Metric**	**Interviewee**
User Perceptions	Popularity	*User*
	Performance	*User*
	Law & Regulation	*User*
	Recommendation	*User*
	Trustworthiness	*User*
	Requirement & Expectation	*User*
	Environmental adaptability	*User*
User Requirement	*User acceptance*	*User*
	Satisfaction	*User*

4. 3. Responsibility and Measurement of Metrics

The third component in PQF is the responsibility. It is defined as the responsibility person to answer the questions related to metrics. It is also named as the interviewee in this model. The PQF has identified specific interviewee to responsible in giving the assessment score of each metrics.

The measurements used are Likert scale of 1 to 5 based on collaborative perspective among assessment team members. Likert scale is defined as something that is the satisfaction measured based on perception. The Likert technique presents a set of attitude statements. Subjects are asked to express agreement or disagreement of a five-point scale. Each degree of agreement is given a numerical value from one to five. Thus a total numerical value can be calculated from all the responses. The scale used in this approach is recommended as 1 = unacceptable, 2 = below average, 3 = average and 4 = good, 5= excellent.

4. 4. Classification of Attributes and Weight Factors

The weighting factors defined in PQF is based on findings from previous survey [32]. From this analysis, the function point approach is used to group and classify attributes into three distinct classifications namely low, moderate and high. Then, the attributes are sorted into these classifications according to the calculated weight score. The analysis shows that functionality is 14.29% more important compared to other quality attributes defined in this model. It obtains the highest weight in this analysis. Reliability is considered 12.34% more important and integrity is considered 11.69% important. These three attributes (functionality, reliability and integrity) are classified in the classification group of high. This finding is consistent with survey done by Bazzana, Andersen and Jokela [2]. Second group of classification defined as moderate includes safety (8.44), efficiency (9.09%), maintainability (7.79%) and usability (7.79%). The third group of classification defined as low includes flexibility (5.84), interoperability (6.49), intraoperability (5.84), portability (5.19%) and survivability (5.19). See also the previous publication for detail [32].

For the purpose of assessment and certification, weight factor is therefore assigned to each group accordingly. This is consistent with the requirements of having different weights for attributes (refer to Table 4) and are also considered in other studies for different domain such as e-commerce applications [36][37].

Table 4. Classifications of attributes and its weight factor

Levels	Attributes	Weight Factor
Low	Flexibility Intraoperability Interoperability Portability Survivability	1-4
Moderate	Safety Efficiency Maintainability Usability	5-7
High	Functionality Reliability Integrity	8-10

5. Application and Evaluation

PQF has been applied in software certification model developed by our research group. The certification process requires a software quality model as a benchmark and standard of the assessment. The quality model must suits with the certification specifications and requirements. Thus, PQF is suitable and fulfill certification requirements with customisation. The whole process of assessment has been implemented and tested in real case studies which involves collaboratively with three large organizations in Malaysia. In these case studies three main systems operated in their environment have been selected and assessed. The exercises completed in less than a week depending on the numbers of main users of the system and the availability of the users and other respondents. The assessment of the system was done through collaborative discussion and evaluation between the three different assessment members which by means are users, developers and independent assessor. The independent assessor led the assessment team. One of the case studies will be presenting in this paper and is referred as product *ABC*.

Table 5 shows an example of a result showing the scores obtained by product *ABC*. It illustrates the scores of the behavioural attributes and the impact attributes (human aspects) defined in PQF of this product. In this example, product *ABC* is a staff information system operating in higher institution in Malaysia. It was developed by internal experts in the organization and was operating for more than 10 years in the environment. The table shows the final analysis of Product *ABC*. Column 1 of this table refers to the maximum value of each score by respondents. Column 2 refers to the weight values given by the owner of the software or any appointed individuals, column 3 is the average score obtained by this assessment. Based on the weights assigned, scores are computed as shown in column 4. Final computed values as in column 5 are the computed values of quality scores obtained according to attributes. For this case, the final computed quality score for the behavioural attributes is 85.6% and for the impact attribute is 94.7%. The final computed overall quality score of product *ABC* is 90.1%. The detail algorithm is explained in [33].

In order to investigate the individual quality attributes of this product, the results are tabulated in the summary table as displayed in Table 6. These scores can be plotted into a kiviat chart to easily realise the result. Each attribute in Figure 2 is represented by axis and scores are plotted at the limits between 0-100%. Attribute that fall on the limit's outer layer is considered better quality compared to attributes at inner layers of this graph. In Case *ABC*, functionality, efficiency, reliability, integrity and the impact attribute, user conformity fall in better quality level compared to maintainability, portability, and usability (refer to Figure 2).

Table 5. Analysis of Product *ABC* using PQF

Behavioural Attributes	**Maximum Value**	**Weight**	**Score Obtained**	**Score**	**Quality Score (%)**
	(1)	**(2)**	**(3)**	**(4)**	**(5)**
Efficiency	5	7	4.70	0.609	12.2
Functionality	5	10	4.96	0.919	18.4
Maintainability	5	7	3.58	0.464	9.3
Portability	5	3	3.50	0.194	3.9
Reliability	5	10	4.50	0.833	16.7
Usability	5	7	3.41	0.442	8.8
Integrity	5	10	4.33	0.817	16.3
TOTAL		**54**		**4.279**	**85.6**
The Impact					
User Conformity					94.7
Total Product					**90.1**

Table 6. Quality score by attributes and subattributes

Attribute	***Score***	***Attribute***	***Score***
Efficiency	*4.7*	*Functionality*	*4.96 (99.3%)*
Time behavior	*(94.0%)*	*Suitability*	*4.88*
Resource	*4.5*	*Accuracy*	*5.0*
Utilization	*5.00*	*Interoperability*	*5.0*
Maintainability	*3.58*	*Portability*	*3.50 (70.0%)*
Analysability	*(71.6%)*	*Adaptability*	*4.75*
Changeability	*3.05*	*Installability*	*2.60*
Testability	*3.25*	*Conformance*	*5.00*
	2.00	*Replaceability*	*5.00*
Reliability	*4.5*	*Integrity*	*4.33 (86.7%)*
Maturity	*(90.0%)*	*Security*	*4.33*
Fault Tolerance	*4.75*	*Data Protection*	*3.00*
Recoverability	*4.38*		
	4.33		
Usability	*3.41*	*User*	*4.73 (94.7%)*
Understandability	*(68.2%)*	*Conformity*	*4.67*
	2.72	*User's*	*4.83*
Learnability	*3.40*	*perception*	
Operability	*4.61*	*User requirement*	

Upon completion of the assessment exercise, a meeting was conducted with the assessment team, the management and stakeholders. The meeting had agreed with this assessment results presented by the assessment team and the committee also agreed that the model was beneficial and valuable for assessment of software in real operating business environment. This model could be used multiple times to monitor the performance of the system during its life cycle. It provided beneficial information to the developers, owners as well as the stakeholders on the quality status of the system. A second outcome from this meeting concerned with the attribute's weight used in this model. The weights assigned for individual behavioural attributes in this model were useful to reflect the business requirements. The committee approved that the validity of the weight values associated with quality attributes was depended on the maturity of the person in the development process and overall information planning of the organization.

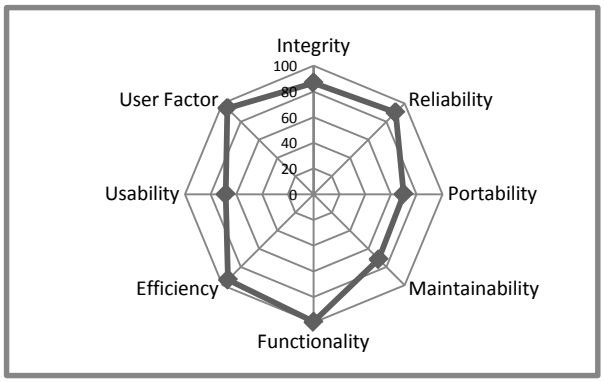

Figure 2. Kiviat chart of product *ABC*

Similar studies were conducted but assessment done on different systems. One of the system assessed was hospital information system (product *XYZ*) which was operating in main hospital in Kuala Lumpur, Malaysia. The detail results of this assessment and certification was discussed and explained in a separate publication (see [35]). In summary, the hospital information system or product *XYZ* achieved final computed overall quality score as 74.5% which was equivalent to basic and acceptable. In this case, the computed quality score for the behavioural attributes was 65.6% and for the impact attribute is 83.5%. Table 7 demonstrates the result.

Table 7. Analysis of product *XYZ* using PQF

Behavioural Factors	Max Value	Weight	Score Obtained	Score	Quality Score (%)
	(1)	(2)	(3)	(4)	(5)
Efficiency	5	7	3.05	0.403	8.1
Functionality	5	9	3.33	0.566	11.3
Maintainability	5	7	3.35	0.442	8.8
Portability	5	4	3.47	0.262	5.2
Reliability	5	9	3.14	0.533	10.7
Usability	5	7	3.71	0.490	9.8
Integrity	5	10	3.08	0.582	11.6
TOTAL		**53**		**3.278**	**65.6**
Impact Factors					
User Factor					83.5
Total Product					**74.5**

6. Future Work

Our previous work participated in solving problem in ensuring and determining quality of software product. The candidate software in the assessment is the software product that is already operating in an actual environment. Current work in this research is the development of support tool that automate the process efficiently and enables users to do the assessment at any time throughout its life cycle. The tool will be installed in computers at user sites and the software under assessment will be assessed by the users at their own convenient times.

PQF as explained in this paper consists of static model of quality. Even though it provides certain level of flexibility to the organization in the assessment by allowing to choose weight factors but this model unable to improve its components according to current and future requirements. This research will be further enhanced to produce a more comprehensive and intelligent model of software quality

that capable to learn in the environment. This can be done by applying artificial intelligence technique in the quality model. With this new intelligent model, new attributes associated with quality will be included when the system suggests and recommends to the environment.

7. Conclusion

Pragmatic quality factor (PQF) is a pragmatic software quality model which could be used in assessment of software operating in certain environment. It focuses on measuring the quality in-use in the actual environment. PQF consists of four main components: 1) behavioural attributes, 2) impact attribute, 3) responsibility and measurement of metrics and 4) classification of attributes and weight factors. Weighted Scoring Method applied in this model is beneficial and valuable to the organizations as the weight factors of each attribute are defined separately. As suggested in literature stakeholders are more interested in the overall quality and therefore, assigning weights to reflect business requirements is essential. It allows the owner of the product to tailor and customise weight factors of individual attributes but guided by the weight defined in this model. This model shows how the unmeasurable characteristics can be measured indirectly using measures and metrics approach. It has been tested involving assessment and certification exercises in real case studies in Malaysia. This model is supported by our developed tool named SoCfeS in the assessment which will also support continuous assessment throughout the software life cycle.

8. References

[1] R. Adnan & M. Bassem, "A new software quality model for evaluating COTS components", Journal of Computer Science 2(4), pp. 373-381, 2006.
[2] G. Bazzana, O. Andersen & T. Jokela, "ISO 9126 and ISO 9000: friends or foes?" In the proceedings of the IEEE Software Engineering Standards Symposium, 1993.
[3] M.F. Bertoa, J.M. Troya & A. Vallecillo, "A survey on the quality information provided by software component vendors", In proceedings of the 7th ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering (QAOOSE 2003), pp. 25-30, 2003
[4] N. Bevan, "Quality in use: Meeting user needs for quality", 1999, Retrieved 14 October 2007 from http://www.usabilitynet.org/papers/qiuse.pdf.
[5] J. Boegh, "Certifying software component attributes", IEEE Software, May/June, pp.74-81, 2006.
[6] L. Buglione & A. Abran, "A quality factor for software", In proceeding of QUALITA99, 3rd International Conference on Quality and Reliability, pp. 335-344, 1999.
[7] Compuware, "Application quality and its business impact- a view from the top (White paper)", 2003, Retrieved 13 January 2004 from http://www.compuware.com/whitepapers/ok.asp.
[8] C.A. Dekkers & P.A. McQuaid, "The dangers of using software metrics to (Mis) Manage", IT Pro, March/April, pp. 24-30, 2002.
[9] P.J. Denning, "What is software quality?" A Commentary from Communications of ACM, January 1992.
[10] G.R. Dromey, "Cornering the chimera", IEEE Software, January, pp. 33-43, 1999.
[11] N.E. Fenton. & S.L. Pfleeger, "Software Metric: A rigorous & practical approach", London: Thompson Computer Press, 1996.
[12] J.E. Gaffney, "Metrics in software quality assurance", ACM, Nov, 126-130, 1981.
[13] R. Hendriks, R.v. Vonderen & E.v.Veenendaal, "Measuring software product quality during testing", In proceedings of the European Software Quality Week Conference, 2000.
[14] ISO/IEC 9126. "Software quality characteristics and metrics-Part2: External metrics", Technical Report, ISO/IEC JTC1/SC7/WG6, 1996.
[15] K. Khosravi & Y.G. Gueheneuc, "A quality model for design patterns", Retrieved 26 October 2005 from http://www.yann_gael.gueheneuc.net/work/Tutoring/Documents/041021+Khosravi+Technical+Report.doc.pdf, 2004.
[16] B. Kitchenham & S.L. Pfleeger, "Software quality: The elusive target", IEEE Software,

January, pp. 12-21, 1996.
[17] M. Ortega, M. Perez & T. Rojas, "Construction of a systemic quality model for evaluating a software product", Software Quality Journal, vol. 11, pp. 219-242, 2003.
[18] Page, C & Meyer, D., "Applied Research Design for Business and Management", Sydney:McGraw-Hill, 2000.
[19] Pfleeger, S.L., "Software Engineering: Theory and Practice", 2nd ed. Upper Saddle River, N.J: Prentice Hall, 2001.
[20] A.K. Rae, H.L. Hausen & P. Robert, Software Evaluation for Certification: Principles, Practice and Legal Liability. Middlesex, UK: McGraw-Hill, 1995.
[21] M. Svahnberg, C. Wohlin, L. Lundberg, & M. Mattsson, "A method for understanding quality attributes in software architecture structures." ACM (online), 2002.
[22] F. Shull, C. Seaman & M. Zelkowitz, "Quality time: Victor R. Basili's Contributions to Software Quality", IEEE Software, Jan/Feb, pp. 16-18, 2006.
[23] S.A. Slaughter, D.E. Harter & M.S. Krishnan, "Evaluating the cost of software quality", Communications of The ACM 41**(**8), pp. 67-73, 1998.
[24] Tervonen, I. "Support for quality-based design and inspection". IEEE Software, January, pp. 44-54, 1996.
[25] W.M.K. Trochim, W.M.K., "Deduction & Induction thinking", Retrieved 25 July 2007 from http://www.socialresearchmethods.net/kb/dedind.php, 2006.
[26] J.A. Whittaker & J.M. Voas, "50 years of software: Key principles for quality", IEEE IT Pro, Nov/Dec, pp. 28-35, 2002.
[27] J. Verner, T. Moores & A.R. Barrett, "Software quality: perceptions and practices in Hong Kong", Proceedings of the Third IFIP International Conference on Achieving Quality in Software, 1996.
[28] J. Voas, "Limited software warranties", In proceedings of the Engineering of Computer Based Systems (ECBS2000), pp. 56-61, 2000.
[29] J. Voas, "Software's secret sauce: The "-ilities". IEEE Computer, November/December, pp. 14-15, 2004.
[30] T.E. Vollman, "Software quality assessment and standards", Computer, June, pp. 118-120, 1993
[31] J.H. Yahaya, A. Deraman, A.R. Hamdan, "Software Quality and Certification: Perception and practices in Malaysia", Journal of ICT (JICT), vol. 5, Dec, pp. 63-82, 2006.
[32] J.H. Yahaya, A. Deraman, A.R. Hamdan, "Software product certification model: Classification of quality attributes", In Proceedings of The First Regional Conference of Computational Science and Technology (RCCST 07), Kota Kinabalu, pp. 436-440, 2007.
[33] J.H. Yahaya, A. Deraman, A.R. Hamdan, **"**Software certification model based on product quality approach", Journal of Sustainability Science and Management, vol. 3(2), December, pp. 14-29, 2008.
[34] Peng, Y., Kou, G., Wang, G., Wang, H. and Ko, F., "Empirical evaluation of classifiers for software risk management", International Journal of Information Technology and Decision Making, Vol. 8, Issue: 4, Page 749 – 768, 2009.
[35] J.H. Yahaya, A. Deraman & A.R. Hamdan, "Continuosly ensuring quality through software product certification: A case study", Proceedings of the International Conference on Information Society (i-Society 2010) London, UK, pp193-198, 2010.
[36] B. Behkamal, M. Kahani & M.K. Akbari, "Customizing ISO9126 quality model for evaluation of B2B applications", Information and Software Technology Journal, 51(2009), pp 599-609, 2009.
[37] Ahmad Rababah & F.A. Masoud, "Key factors for developing a successful e-commerce website", Communication of the IBIMA, vol. 2010, Article 10763461, 2010.