Reliability, Performance, and Supportability[13]. One disadvantage of this model is that it does not take into account the software product's portability [15].

ISO 9126 defines product quality as a set of product characteristics that governs how the product works in its environment are called external quality characteristics. ISO 9126 indicates six main quality characteristics which associated with several subcharacteristics. 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 [18] and the model only focusing on developer view of the software [17].

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 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 [9].

The systemic model is developed by identify the relationship between product-process, efficiency-effectiveness and user-customer to obtain global systemic quality [15]. 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 and are considered as essential and vital.

Quality is believed as a complex concept. It is the eye of the beholder and it means different things to different people and highly context dependent [14]. Therefore, "software quality is nothing more than a recipe. Some like it hot, sweet, salty or greasy" [24]. Thus, there can be no single simple measure of software quality acceptable to everyone**.**

As observed from existing quality models for software product assessment, available identified quality attributes is difficult to meet current requirement and specification. Current 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 is more on customer satisfaction and software correctness is not sufficient to be declaring as good quality without satisfaction by the users [8]. Thus, there are requirements to include measurements of human aspects and the quality impact in the quality model. Integrity as one of the vital attribute in current situation is not considered in previous models.

B. Software Quality Measurement and Metrics

The ultimate goal is to produce a high-quality software, application, or product. Measurement is used to assess the quality of the software. Software metric is defined 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"[11]. 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 [25].

Software measurement can be categorized into direct measurement and indirect measurement. Direct measurement 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. Metric can be defined as a quantitative measures of software or processes for a given attributes. It can be used to estimate quality [4].

The following discussion aspires to demonstrate the software quality framework with focusing on unmeasurable and measurable aspects of quality characteristics. For an example, functionality 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 indirect 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 Fig. 1 and Table 1. 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 used metric 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 and documentations. Thus, data is gathered and required to arrive an indication of quality. Eventually metrics gathered can cost a lot of money and therefore it is suggested to collect practical target data that will produce meaningful result.

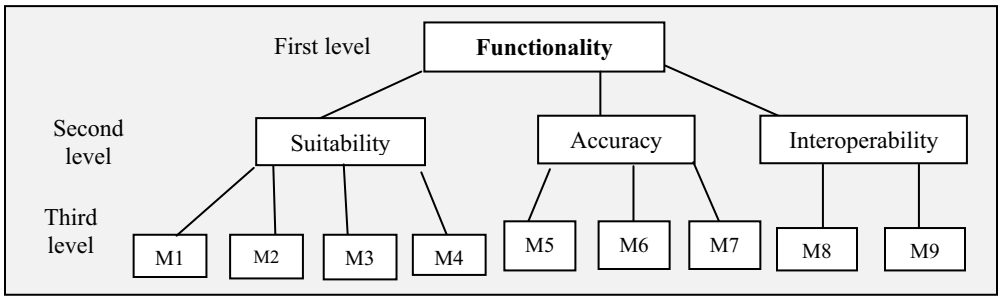

Figure 1. Decomposition of functionality

TABLE I. EXAMPLE OF QUALITY FACTOR, SUBFACTORS AND METRICS

Factor: Functionality		
Subfactors	Metric	Measure
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?

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.

III. METHODOLOGY

The research approach used in this study is deductive approach [16][21] where theory and concepts of software quality are derived from the literature and empirical findings before the model is applied and tested in real case studies. The research approach involves four phases:-

A. 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 proceeds with designing questionnaires and test it through pilot study.

B. Empirical Study

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.

C. Model Construction

Based on the empirical and literature findings, an initial software quality model is constructed. The concept, definition and contributing factors are used to identify attributes that are required in the assessment of software. This led to the development of a software quality model, which met current software assessment requirements. The proposed quality model is named Pragmatic Quality Factor (PQF), which describes the relationships between attributes (which mostly unmeasurable) and the measurable metrics. In model construction, all variables in the model are 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 provided.

D. Application and Validation

The application of the model is carried out to evaluate the model. These involved collaboratively with industry in Malaysia. The applications on the case studies test the proposed model by assessing systems operating in the actual environments. As the model evaluation is carried out by the case study, a model refinement is conducted as necessary.

IV. PRGAMATIC QUALITY FACTOR(PQF) : A PRACTICAL SOFTWARE QUALITY MODEL

The PQF consists of four main components: behavioural attributes, impact attributes, responsibility, and weight.

A. 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. 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. Our previous survey [26] 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. An example of the decompositions of attributes is shown in Table 1 and Table 2. In this example, it shows that functionality is 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 II. 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

B. 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 [7]. 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 (refer to Table 3).

C. 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. The Likert technique presents a set of attitude statements to measure satisfaction on perception. 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.

TABLE III. 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

D. Classification of Attributes and Weight Factors

The weighting factors defined in PQF is based on findings from previous survey [27]. 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 obtained 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 [27].

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.

V. APPLICATION AND VALIDATION

PQF has been applied in software certification model developed by our research group. The certification process requires a software quality model as the benchmark and standard of the assessment. The quality model must suit 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. 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 discussions and evaluation between the three different assessment members that includes users, developers and independent assessor. The independent assessor led the assessment team.

In order to study the individual quality attributes of this product, the results are tabulated in the summary table as demonstrated in Table 5. These scores can be plotted into a kiviat chart to easily realise the result. Each attribute in Fig. 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 this case (later is called product ABC), usability, portability and the impact attribute, user factor fall in better quality level compared to maintainability, functionality, efficiency, reliability and integrity. Refer to Fig. 2.

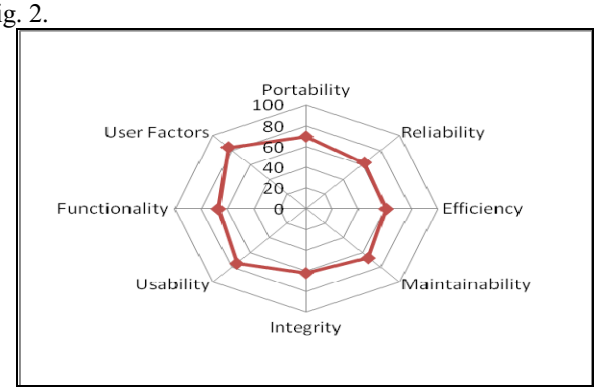

Figure 2. Kiviat chart of product ABC

TABLE V. QUALITY SCORE BY ATTRIBUTES AND SUBATTRIBUTES

Attribute	*Score*	*Attribute*	*Score*
Efficiency	*3.05 (61.0%)*	*Functionality*	*3.33 (66.7%)*
Time behavior	*3.25*	*Suitability*	*3.41*
Resource utilization	*2.75*	*Accuracy*	*3.38*
		Interoperability	*3.13*
Maintainability	*3.35 (66.9%)*	*Portability*	*3.47 (69.4%)*
Analysability	*3.43*	*Adaptability*	*3.67*
Changeability	*3.21*	*Installability*	*3.09*
Testability	*3.33*	*Conformance*	*3.50*
		Replaceability	*4.00*
Reliability	*3.14 (62.8%)*	*Integrity*	*3.08 (61.7%)*
Maturity	*3.44*	*Security*	*3.17*
Fault Tolerance	*3.03*	*Data Protection*	*3.00*
Recoverability	*2.89*		
Usability	*3.71 (74.3%)*	*User Factor*	*4.18 (83.5%)*
Understandability	*3.78*	*User's perception*	*4.25*
Learnability	*3.79*	*User requirement*	*4.06*
Operability	*3.63*		

Table 6 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) as defined in PQF. In this example, product ABC is a hospital information system operating in a large well-known hospital in Malaysia. It was developed by internal experts in the organization and was operating for more than 2 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 65.6% and for the impact attribute is 83.5%. The final computed overall quality score of product ABC is 74.5% (see [28] for detail of the algorithm).

TABLE VI. ANALYSIS OF PRODUCT ABC USING PQF

Behavioural Factors	Max Value (1)	Weight (2)	Score Obtained (3)	Score (4)	Quality Score (%) (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
a) Impact Factors					
User Factor					83.5
Total Product					**74.5**

VI. FUTURE WORK

This 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. A support tool that enables to automate the process efficiently at the users sides will be required for convenient assessment throughout its life cycle.

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 from its environment. With this new intelligent model, new attributes associated with quality will be included when the system suggests and recommends to the environment.

VII. 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 will be supported by a tool named SoCfeS in the future assessment which will also support continuous assessment throughout the software life cycle.

ACKNOWLEDGMENT

This project is funded by Malaysian Ministry of Science, Technology and Innovation.

REFERENCES

[1] R. Adnan & M. Bassem, "A new software quality model for evaluating COTS components", Journal of Computer Science 2(4)2006,pp. 373-381.

[2] G. Bazzana, O. Andersen & T. Jokela, "ISO 9126 and ISO 9000: friends or foes?" 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", Proceedings of 7th ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering (QAOOSE 2003), 2003, pp. 25-30.

[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] L. Buglione & A. Abran, "A quality factor for software", Proceeding of QUALITA99, 3rd International Conference on Quality and Reliability, 1999, pp. 335-344.

[6] 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.

[7] C.A. Dekkers & P.A. McQuaid, "The dangers of using software metrics to (Mis)Manage", IT Pro, March/April 2002, pp. 24-30.

[8] P.J. Denning, "What is software quality?" A Commentary from Communications of ACM, January 1992.

[9] G.R. Dromey, "Cornering the chimera", IEEE Software, January 1999, pp. 33-43.

[10] N.E. Fenton. & S.L. Pfleeger, "Software Metric: A rigorous & practical approach", London: Thompson Computer Press, 1996.

[11] J.E. Gaffney, "Metrics in software quality assurance", ACM , Nov 1981,126-130.

[12] ISO/IEC 9126. "Software quality characteristics and metrics-Part2: External metrics", Technical Report, ISO/IEC JTC1/SC7/WG6, 1996.

[13] 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.

[14] [14] B. Kitchenham & S.L. Pfleeger, "Software quality: The elusive target", IEEE Software, January 1996, pp. 12-21.

[15] M. Ortega, M. Perez & T. Rojas, "Construction of a systemic quality model for evaluating a software product", Software Quality Journal, vol. 11, 2003, pp. 219-242.

[16] C. Page & D. Meyer, "Applied Research Design for Business and Management", Sydney:McGraw-Hill, 2000.

[17] S.L. Pfleeger, "Software Engineering: Theory and Practice", 2nd ed. Upper Saddle River, N.J: Prentice Hall, 2001.

[18] A.K. Rae, H.L. Hausen & P. Robert, Software Evaluation for Certification : Principles, Practice and Legal Liability. Middlesex, UK: McGraw-Hill, 1995.

[19] F. Shull, C. Seaman & M. Zelkowitz, "Quality time: Victor R. Basili's Contributions to Software Quality", IEEE Software, Jan/Feb 2006, pp. 16-18.

[20] I. Tervonen, I. "Support for quality-based design and inspection". IEEE Software , January 1996, pp. 44-54.

[21] W.M.K. Trochim, W.M.K., "Deduction & Induction thinking", Retrieved 25 July 2007 from http://www.socialresearchmethods.net/kb/dedind.php, 2006.

[22] J.A. Whittaker & J.M. Voas, "50 years of software: Key principles for quality", IEEE IT Pro, Nov/Dec 2002, pp. 28-35.

[23] J. Voas, "Limited software warranties", Engineering of Computer Based Systems (ECBS2000) Proceeding, 2000, pp. 56-61.

[24] J. Voas, "Software's secret sauce: The "-ilities". IEEE Computer , November/December 2004, pp. 14-15.

[25] T.E. Vollman, "Software quality assessment and standards", Computer , June 1993, pp. 118-120.

[26] J.H. Yahaya, A. Deraman, A.R. Hamdan, "Software Quality and Certification: Perception and practices in Malaysia", Journal of ICT (JICT), vol. 5, Dec 2006, pp. 63-82.

[27] J.H. Yahaya, A. Deraman, A.R. Hamdan, "Software product certification model: Classification of quality attributes", Proc. The First Regional Conference of Computational Science and Technology (RCCST 07), Kota Kinabalu, 2007, pp. 436-440.

[28] 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 2008, pp. 14-29.