2013 International Conference on Communication Systems and Network Technologies

A Review on Software Quality Models

Brijendra Singh, Suresh Prasad Kannojia

drbri_singh@hotmail.com, spkannojia@gmail.com

Department of Computer Science, University of Lucknow, Lucknow

Abstract: The concern of quality of software product has been increasing in most industrial sectors. In addition, awareness of the central importance of the production process has been increasing. Process is important because industry cares about their intrinsic qualities, such as uniformity of performance across different projects and productivity, with the aim of improving time to market and reducing production costs. But processes are also important because experience has shown that they have a decisive influence on the quality of product that is, by controlling process, we can achieve better control of the required quality of products. The relationship between process and the quality of products holds especially in software production, because of the intrinsic nature of the software. In this paper we have gone through literature of software quality process and software quality models. We have studies some of important models and compare them. These models have been categorized into two i.e. Software development process models & Software product models. Software process models have been applied for many years in an effort to bring order and structure development. Each of these conventional models suggests a somewhat different process flow, but all perform the same set of generic framework activities: Communication, Planning, Modeling, Construction, and deployment. Review also provides the progress in software quality models and practices that can be applied today to achieve the quality of finished product.

Introduction: Software standards are needed to achieve acceptable level of quality in both software products and process. Software Quality [1]: The definition of software quality has evolved over time [2]. Initially, it was defined as conformance to a standard or a specification. Later, the definition was changed to adapt to highly dynamic business environments. In 1991, the International Organization for Standardization adopted ISO 9126 as the standard for evaluating software quality. This standard defines quality as "the totality of features and characteristics of a product or service that bears on its ability to satisfy given needs" [3]. ISO 9126 compliments ISO 9001, which deals with the quality assurance of the process used for developing products. A commonly used definition of software quality is the density of post release defects in a software program, which is measured as the number of defects per thousand lines of code [4, 5].

Gaffney [6] reported that the best estimator for the number of errors in a software module was the number of lines of code. Krishnan and Kellner [7] also confirmed this finding. Harter and Slaughter [8] found that product complexity significantly lowered software quality, which is somewhat contrary to [6], which did not find software complexity affecting error rates significantly.

Banker and Slaughter [9] found software volatility, defined as the frequency of enhancements per unit of functionality in a given time frame, to be a significant predictor of software errors. Data complexity, defined as the number of data elements per unit of application functionality, also increased the number of defects. They also found that structured programming techniques moderated the effects of volatility and data complexity on software errors. Using a game-theoretic model, Austin [10] suggested that under schedule pressures, developers were likely to compromise on quality. Krishnan et al. [11] found personnel quality, which is measured using peer and supervisor assessments, to be a significant estimator of software quality. They also found that front-end investments, which improved customer-requirements analysis, enhanced quality. A number of approaches have been proposed to improve software quality [12]. These include total quality management (TQM), Six Sigma [13], and CMM [14]. The basic idea behind all these approaches is to identify ways to improve quality in a given situation. The relationship between process improvements and quality has also been investigated [15]. The most significant development in this area has been the development of CMM [16, 17]. John Moses [8] argued that the problem of subjective measurement of quality attributes should not be ignored if quality is to be introduced into software in a controlled way. Further, it is argued that direct measurement of quality attributes should be encouraged and that in fact such measurement can be quantified to establish consistency using an existing approach. There are many experts who argue that the "Quality" of the development process is the best predictor of product quality; this issue, and the problems surrounding it, is discussed extensively in [9]

Software process modeling [20] has been recognized as an important topic since the early days of software engineering. Process modeling Language plays a crucial role when it is used to define and analyze the processes. Complex processes have many elements and there are many potential ways to improve them. Without a quantitative understanding of the process steps it is

978-0-7695-4958-3/13 $26.00 © 2013 IEEE
DOI 10.1109/CSNT.2013.171

difficult to tell which ones are effective. Processes can be measured for size, effort, schedule and cost under successful performance. A process should be defined based on organization's intents or business goals. Although current studies of software process improvement have aroused interest of many researchers in this field, software development strategies do not provide a clear indication for constituents of software processes.

Many researchers have recognized the need for software process improvement since the 1930s [21]. Software process improvement applied to process modification in software development it can improve product quality, decrease costs and increase profitability. In order to achieve software development process standardization, models for software process quality or standards for software quality assurance have been proposed by previous researchers, including CMMI [21], ISO15504 [22], ISO15939 [23]. The technology for software process enactment and control was also focused [24]. The goal of process evaluation focuses on continuous improvement of software process driven by the implementation of software process metrics. As a result, process metrics play a major role in the process capability assessment and its management [25]. Software process improvement [26] is a common term used for modifying processes in software development that aims to improve product quality and business value. Successful improvement of development process should take advantages of complementary use of both quantitative and qualitative methods. Complex processes have many elements and there are many potential ways to improve the processes. Without a quantitative understanding of the process steps, however, it is difficult to tell which ones are effective. Process measurements provide the data you need to objectively understand how your process works and to see what you can do to improve it [27].

The process measurement involves the collection of data about time, size and defects. Such analysis can improve software process quality [28]. Defining a process lies in organization's intents that are composed of business goals [29]. Yet, process definitions still suffer from the following drawbacks:

- Currently the processes in organization are closely related to the fixed forms. The changes to the forms and processes are considerably difficult so that the text actually advises against doing so [21] and [30].
- The process measurement method, forms and procedures are not concurrently designed to support a process enactment. For this reason, one must immediately redesign the forms and scripts embedded to process definition in order to apply the proper metrics to any other kind of work product [31].

From long history many researchers have recognized the need of Product Quality: Traditionally, the quality of product is defined in terms of its fitness of purpose. Although fitness of the purpose is satisfactory definition of quality for hardware products, but it is not satisfactory for software products. Number of models [32, 33, 34, 35, 36, 37] has been proposed to evaluate the quality of software product, based on various quality characteristics. McCall quality model [32] attempts to bridge the gap between users and developers by focusing, on a number of software quality factors. The evaluation of software has been done by Boehm's quality model [34, 35], uses a given set of attributes and metrics. More recently, model has been developed by Dromey's [36, 37], which is focusing on the relationship between the quality attributes and the sub attributes, as well as attempting to connect software product properties with software quality attributes.

In this paper, we concentrate on software quality models. Software standards are required to achieve acceptable level of quality of software product. In the Next section of this paper we will discuss the software quality Models and philosophies, it also describe some popular software quality models for comparison point of view.

Software quality models and philosophies: Why software quality models are used? Generally models provide the stepwise product development tracks. If any software development organization, wants to develop specified quality products, than they needs to follow the organized manner to produce the product in states phases, time, cost and environment. To achieve desired software quality, software quality models to identify high-risk program modules are used. A software quality model is a useful tool for meeting the objectives of software quality, reliability and software testing initiatives of different projects.

The purpose of this section is to provide an overview to popular quality models and presenting a more philosophical management view on software quality, because quality management philosophy can be sometimes a good alternative to the more formalized quality model.

Number of quality prediction models has been proposed in Literature. In this paper we have chosen McCall, Boehm's, Dromey's, ISO, for comparison purpose.

Mccall quality model: Quality model presented by Jim McCall [32] which is known as Mc Call Quality Model and is primarily aimed towards the system developers and the system development process. McCall quality model attempts to bridge the gap between users and developers by focusing on a number of software quality factors that reflect both the users' views and the developers' priorities [32]. The McCall quality model has three major perspectives for defining and identifying the quality of a software product as shown in fig.1

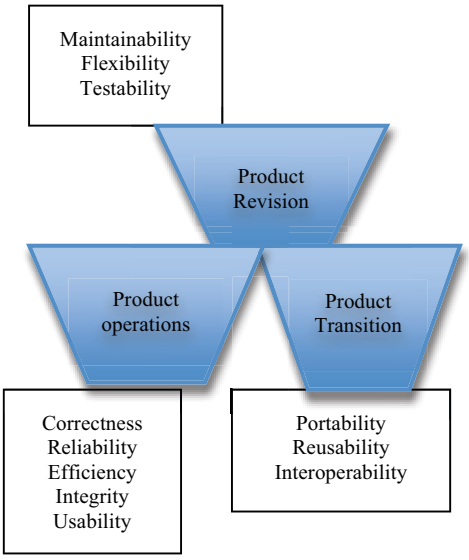

Fig. 1 Mc Call Quality (Triangle) Model

▪ Product revision: Includes maintainability (the effort required to locate and fix a fault in the program within its operating environment), flexibility (the ease of making changes required by changes in the operating environment) and testability (the ease of testing the program, to ensure that it is error-free and meets its specification).
▪ Product transition: It is all about portability (the effort required to transfer a program from one environment to another), reusability (the ease of reusing software in a different context) and interoperability (the effort required to couple the system to another system).
▪ Product operation: The Quality of product operations depends on correctness (the extent to which a program fulfils its specification), reliability (the system's ability not to fail), efficiency (further categorized into execution efficiency and storage efficiency and generally mean use of resources, e.g. processor time, storage), integrity (the protection of the program from unauthorized access) and usability (the ease of the software). In this model 11 attributes have been considered to define the product quality.

Boehm's quality model: In 1978 Barry W. Boehm presented a quality prediction model and addresses the contemporary shortcomings of models that automatically and quantitatively evaluate the quality of software [34, 35]. In essence his models attempts to qualitatively define software quality by a given set of attributes and metrics. Boehm's model is similar to the McCall Quality Model, it also presents a hierarchical quality model structured shown in fig.2. These are High level characteristic, Intermediate level characteristic and primitive characteristics each of which contributes to the overall quality level.

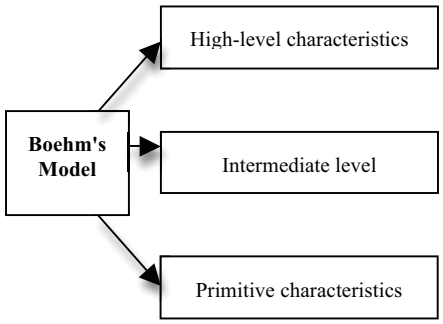

Fig. 2 Boehm's Quality Model Structure

The high level characteristics: It represents basic high-level requirements of actual use to which evaluation of software quality could be put the general utility of software. The high level characteristics address three main questions that a buyer of software has:
• As-is utility: How well (easily, reliably, efficiently) can I use it?
• Maintainability: How easy is it to understand, modify and retest?
• Portability: Can I still use it if I change my environment?
▪ **The intermediate level characteristic:** It represents Boehm's 7 quality factors that together represent the qualities expected from a software system: Portability, Reliability, Efficiency, Usability, Testability, Understandability, and Flexibility.
▪ **The primitive characteristics**: It provides the foundation for defining qualities metrics, which was one of the goals when Boehm constructed his quality model. In this model 17 attributes have been considered to define the product quality.

Dromey's quality model: In the year 1995 a quality model presented by R. Geoff Dromey is more recent model similar to the McCall's, Boehm's and the FURPS (functionality, Usability, Reliability, performance, supportability) quality model [36, 37]. Dromey proposes a product based quality model that recognizes that quality evaluation differs for each product and that a more dynamic idea for modeling the process is needed to be wide enough to apply for different systems. Dromey is focusing on the relationship between the quality attributes and the sub-attributes, as well as attempting to connect software product properties with software quality attributes. The principal of Dromey's Quality Model shown in fig. 3

Now after review of Dromey model, we will find that the reliability attribute is common, which can be, achieve through product properties: correctness, internal and contextual. Also maintainability attribute is common in internal, contextual, descriptive properties of the product.

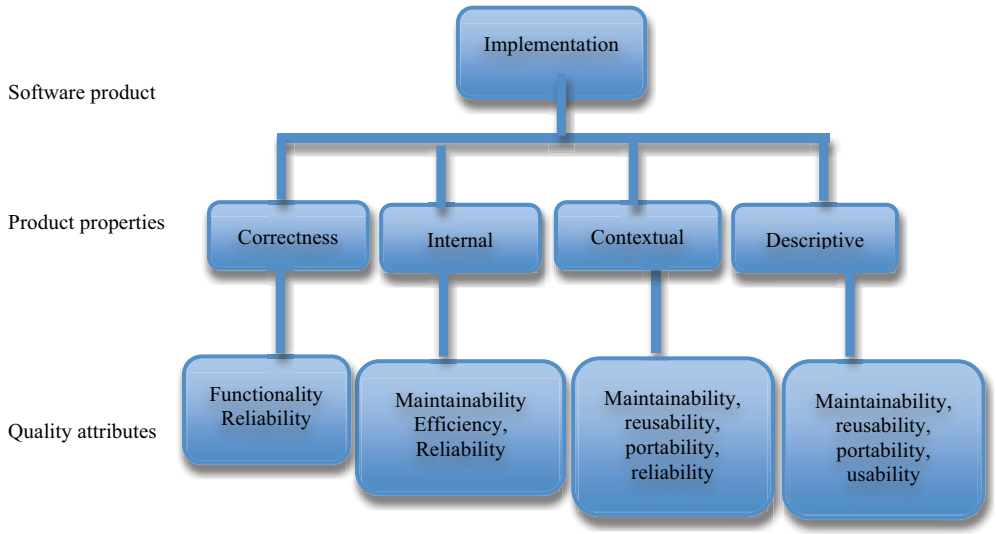

Fig. 3 Dromey's Quality Model

The quality attributes: reusability, portability affects the contextual, descriptive properties of the product. In this model 7 (Seven) attributes have been chosen to define software product quality.

Iso 9126 standard quality model: For a long time, the user community was looking for a single model for depicting and expressing quality. An universal model makes it easier to compare one product with another. In 1992, a derivation of the McCall model was proposed as the basis for an internal standard for software quality measurement. This is called "Software Product Evaluation: Quality Characteristics and Guidelines for their Use". This standard is more commonly referenced by its assigned standard, ISO 9126 [38]. The standard claims that these six attributes of product quality are comprehensive; i.e. any component of software quality can be described in terms of some aspect of one or more of the six factors. In turn, each of the six is defined as a "set of attributes that bear" on a relevant aspect of software, and each can be refined through multiple levels of sub-characteristics.
ISO/IEC then started work on SQuaRE (Software product Quality Requirements and Evaluation), A more extensive series of standards to replace ISO/IEC 9126, with numbers of the form ISO/IEC250mn. For instance, ISO/IEC 25000 was issued in 2005, and ISO/IEC 25010, which supersedes ISO/IEC 9126-1, was issued in March 2011. ISO 25010 has eight product quality characteristics (in contrast to ISO 9126's six) [39].

Analysis of software quality models: We have compare the Quality models McCall [32], Boehm [34], Dromey [37], ISO 9126 [38] and IOS 9126 [39] based on various attributes of quality.

Table 1 Comparison of attributes of various models

Criteria /goals	McCall [32]	Boehm [34]	Dromey [37]	ISO 9126 [38]	ISO 9126 [39]
Correctness	*	*			
Reliability	*	*	*	*	*
Integrity	*	*			
Usability	*	*	*	*	*
Efficiency	*	*	*	*	*
Maintainability	*	*	*	*	*
Testability	*				
Interoperability	*				
Flexibility	*	*			
Reusability	*	*	*		
Portability	*	*	*	*	*
Clarity		*			
Modifiability		*			
Documentation		*			
Resilience		*			
Understandability		*			
Validity		*			
Functionality			*	*	*
Generality		*			
Economy		*			
Compatibility					*
Security					*

The common attributes of various models such as Reliability, Usability, Efficiency, Maintainability and Portability has been shown in the Table 1. As we have seen that 'Testability', 'Interoperability', and 'Understandability' are used as attributes in some quality models. However, in ISO 9126-1 [39], these attributes are

defined as sub characteristics. More specifically, the 'Testability' is belonging to the 'Maintainability' is belonging to the 'Usability' characteristics and the 'Interoperability' is belonging to the 'Functionality' characteristics. One of the important security attribute has been added by ISO 9126 [39], which cover the Reliability, Confidentiality and Integrity. Without Reliability and Security, it is not possible to achieve the quality of software product. In literature we found that quality models attempted to evaluate several quality characteristics but failed to provide reasonable accuracy. We believe that quality models must evaluate high-level quality characteristics with accuracy in terms, well known to software engineers to help maintainers in assessing programs.

Conclusion: Traditionally a quality product is defined in terms of its fitness of purpose that is, a quality product does exactly what the user want it to do. For Software products, the fitness of purpose is usually interpreted in terms of satisfaction of the requirement laid down in software requirement specification document. As we have seen during Literature survey of software quality that the modern view of quality associates a software product with several quality factors such as Portability, Usability, Reliability, Correctness, and Maintainability, etc. Software Reliability, unlike many other quality factors, Can be measured directly and estimated historical and development data. Software reliability is defined in statistical terms as "The probability of failure- free operation of software in a specified environment for a specified time".

The focus of reliability for software is on preserving predictable, correct execution despite the presence of unintentional defects and other weakness and unpredictable environment stage change. Reliability attribute is common with product properties: Correctness, Internal and contextual as used in Dromey Model [37]. This model has chosen seven attributes to define quality of software product. In ISO 9126 [39], Compatibility and Security attributes have been added for software product quality model. Software security extends the requirements of reliability and safety to the need to preserve predictable, correct execution even in the face of malicious attack on defects or weaknesses and environmental stage changes. Without security architecture and features, adequate levels of confidentiality, Integrity, accountability, and non-repudiation may be unattainable. Are all sub characteristics equal in affecting software characteristics: in the Literature quality models define the relation between quality characteristics and sub characteristics. However, the impacts of quality sub characteristics on characteristics are not equivalent. For example: Adoptability and Install-ability are two sub characteristics related to Portability, the question is: If we asses the value of Adaptability as p and the value of Install-ability as q then the value of Portability equals to the p + q.

Table 1 shows the Comparisons of various models based on various attributes. We analyze that five attributes such as software Reliability, Usability, Efficiency, Maintainability and Portability are very useful for common man of the software user because reliability attributes itself cover the performance of functions of software as well as specified environment for software product. In certain cases, if cost of security is increasing more than the cost of software in that case question is that why security is required.

We conclude that any software product model may be useful for software developers to satisfy their need and thereby gaining the optimum quality within specified environment.

References:

[2.1] M. Agrawal and K Chari, "Software Effort, Quality, and Cycle Time: A Study of CMM Level 5 Projects," IEEE Trans. On Software Engineering, Vol.33, no.3, March 2007

[2.2] C. K. Prahalad and M.S. Krishnan, "The New Meaning of Quality in the Information Age," Harvard Business Rev., Vol. 1999, pp. 109-118, 1999

[2.3] ISO/IEC 9126-1, 2001, Int'l Standards Organization, 1991

[2.4] M. Diaz and J. Sligo, "How Software Process Improvement Helped Motorola," IEEE Software, Vol. 14, no. 5, pp. 75-81, Sept.-Oct., 1997

[2.5] C. Fox and W. Frakes, "The Quality Approach: Is It Delivering?," Comm. ACM, Vol. 40, pp. 25-29, 1997

[2.6] J. E. J. Gaffney, "Estimating the Number of Faults in Code," IEEE Trans. Software Eng., Vol. 10, no. 4, pp. 459-464, July 1984

[2.7] M.S. Krishnan and M.I. Kellner, "Measuring Process Consistency: Implications Reducing Software Defects," Management Science, Vol. 25, pp. 800-815, 1999

[2.8] D. E. Harter and S.A. Slaughter, "The Cascading Effect of Process Maturity on Software Quality," Proc. Int'l Conf. Information Systems, 2000

[2.9] R. D. Banker and S.A. Slaughter, "The Moderating Effects of Structure on Volatility and Complexity in Software Enhancement," Information Systems Research, Vol. 11, pp. 219-240, 2000

[2.10] R. D. Austin, "The Effects of Time Pressure on Quality in Software Development: An Agency Model," Information Systems Research, Vol. 12, pp. 195-207, 2001

[2.11] M. S. Krishnan et al., "An Empirical Analysis of Productivity and Quality in Software Products," Management Science, Vol. 46, pp. 745-759, 2000

[2.12] V. Basili, “The Experience Factory and Its Relationship to Other Quality Approaches,” Advances in Computers, Vol. 41, pp. 65-82, 1995

[2.13] T. Pyzdek, “The Six Sigma Handbook: The Complete Guide for Greenbelts, Blackbelts, and Managers at All Levels,” McGraw-Hill, 2003

[2.14] P. Jalote, “CMM in Practice: Processes for Executing Software Projects at Infosys,” Addison Wesley Longman, 2000

[2.15] W.S. Humphrey, “Characterizing the Software Process: A Maturity Framework,” IEEE Software, Vol. 5, no. 3, pp. 73-79, Mar. 1988

[2.16] M. C. Paulk, “How ISO 9001 Compares with the CMM,” IEEE Software, Vol. 12, no. 1, pp. 74-83, Jan. 1995

[2.17] M. C. Paulk et al., “Capability Maturity Model,Version 1.1,” IEEE Software, Vol. 10, no. 4, pp. 18-27, July 1993

[2.18] J. Moses, “Should we try Measure Software Quality attributes directly,” Software Quality Journal 17:203-213,2009

[2.19] N. E. Fenton, S. Lawrence Pfluger and R. Colass, ”Science and Substance: A Challenge to Software Engineers,” IEEE Software, pp. 86-95 July1994

[2.20] W. H. Shen, N. L. Hsueh and P.H Chu, ”Measurement-Based Software Process Modeling,” Journal of Software Engineering 5 (1): 20-37, 2011

[2.21] M. B. Chrissis, M. Konrad and S. Shrum, “CMMI: Guidelines for Process Integration and Product Improvement,” 2nd Edn., Addison-Wesley Professional, Reading, MA., USA., ISBN-13: 978-0-321-15496-5, 2003

[2.22] H. V. Loon, “Process Assessment and ISO 15504,” Springer, New York, ISBN: 9-780387-300481, 2007

[2.23] ISO/IEC 15939, “Software Measurement Process Model,” http://segoldmine.ppi-int.com/content/standard-isoiec-15939-software-measurementy-process, 2002

[2.24] A. Fuggetta, “Software Process: A Roadmap,” Proceedings of the Conference on The Future of Software Engineering, June 4-11, New York, NY, USA., pp: 25-34, 2000

[2.25] G. M. Muketha, A. Ghani, M. H. Selamat and R. Atan, “A Survey of Business Process Complexity Metrics,” Inform. Technol. J., 9:1336-1344, 2010

[2.26] B. McFeeley, “IDEAL: A User’s Guide for Software Process Improvement,” Software Engineering Institute/Carnegie Mellon University, Pittsburgh,PA.,CMU/SEI-96-HB-001,1996 http://www.sei.cmu.edu/reports/96hb001.pdf.

[2.27] A. Bobkowska, “Quantitative and Qualitative Methods in Process Improvement and Product Quality Assessment,” Proceedings of the 12th Conference on European Software Control and Metrics, April 2-4, UK., pp: 1-10, 2001

[2.28] W.S. Humphrey, “A Discipline for Software Engineering,” Addison-Wesley, Reading MA., USA., ISBN-13:9-780201-546101,1995

[2.29] T. M. Koulopoulos, “The Workflow Imperatives: Building Real World Business Solution Van Nostrand Reinghold,” New York, 1995

[2.30] A. Fadila, and G. said, “A New Textual Description for Software Process Modeling,” Inform. Techno. J., 5: 1146-1148, 2006.

[2.31] P. Failer and W.S. Humphrey, “Software Process Development and Enactment: Concepts and Definitions,” Technical Report, CMU/SEI-92-TR-004. http://www.sei.cmu.edu/reports/92tr 004.pdf, 1992

[2.32] J. A. McCall, P. K. Richards, and G. F. Walters, "Factors in Software Quality", Nat'l Tech. Information Service, no. Vol. 1, 2 and 3, 1977

[2.33] B. Kitchenham, and S. L. Pfleeger, "Software Quality: The Elusive Target [special issues section]", IEEE Software, no. 1, pp. 12-21, 1996

[2.34] B. W. Boehm, J. R. Brown, H. Kaspar, M. Lipow, G. McLeod, and M. Merritt, “Characteristics of Software Quality,” North Holland, 1978

[2.35] B. W. Boehm, J. R. Brown, and M. Lipow, “Quantitative Evaluation of Software Quality,” International Conference on Software Engineering, Proceedings of the 2nd international conference on Software engineering, 1976

[2.36] R. G. Dromey, "Concerning the Chimera [software quality]", IEEE Software, no. 1, pp. 33-43, 1996

[2.37] R. G. Dromey, "A Model for Software Product Quality", IEEE Transactions on Software Engineering, no. 2, pp. 146-163, 1995

[2.38] ISO/IEC 9126-1 Software Engineering - Product quality, Part 1: Quality model", 2001,

http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=22749

[2.39] ISO/IEC 25010 Systems and Software Engineering -- Systems and Software Quality Requirements and Evaluation (SQuaRE)—System and Software Quality Models, 2011,

http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=35733