A Novel Method for Quantitative Assessment of Software Quality

Neelam Bawane neelambawane@yahoo.com
Assistant Prof., Dept. of MCA
PES Institute Technology
Bangalore, 560085, India

C. V. Srikrishna cvsrikrishna@yahoo.co.in
Prof. & Head, Dept. of MCA
PES Institute Technology
Bangalore, 560085, India

Abstract

This paper deals with quantitative quality model that needs to be practiced formally through out the software development life cycle at each phase. Proposed quality model emphasizes that various stakeholders need to be consulted for quality requirements. The quality goals are set through various measurements and metrics. Software under development is evaluated against expected value of set of metrics. The use of proposed quantitative model is illustrated through a simple case study. The unaddressed quality attribute reusability in ISO 9126 is also discussed.

Keywords- quality assurance, quality metrics, quality model, stakeholders

1. INTRODUCTION

Quality can not be added to the system as an afterthought, instead it must be built into the system from the beginning. This paper proposes a quantitative software quality model which can evaluate requirements engineering phase of system development rigorously. The objective of this paper is to identify the need of a software quality model to be used as a foundation to Software Quality Engineering. The paper illustrates the use of the proposed model during system analysis & design and demonstrates usefulness through a case study. Software quality related literature in section II and a brief description of existing quality models in section III presents background knowledge about the topic considered in this paper. In Section IV, authors propose a quality model where need of early quality analysis and importance of collecting the requirements from various stakeholders are shown. A case study to demonstrate the application of proposed model is presented in Section V.

2. RELATED LITERATURE

According to Gordon [12], software quality is an important characteristic affecting overall system development lifecycle cost, performance and useful life. Increasing demands from the marketplace for a greater emphasis on quality in software products are promising to revolutionize good practice of software engineering [4].

It is now well established that production is meaningless without assessment of product quality. "Quality is a complex and multifaceted concept." Garvin [10] describes quality from five different perspectives: transcendental view, user view, manufacturers view, product view, and value-based view.

Kitchenham and Pfleeger [7] have recently discussed Garvin's [9] approach in the context of software product quality, accordingly Garvin's model is a useful starting point not as a quality model in its own right but rather as a specification of a set of requirements for quality models or alternatively as a set criteria for evaluating product quality models.

The fact "quality must be monitored from the early phase such as requirements analysis and design" provides need of Software Quality Assurance (SQA) [5]. The aim of the SQA organization is to assure that the standards, procedures and policies used during software development are adequate to provide the level of confidence required for the process or product. SQA is defined as "a planned and systematic pattern of all actions necessary to provide adequate confidence that the item or product conforms to established technical requirements".

With increasing importance placed on standard quality assurance methodologies by large companies and government organizations, software companies have implemented rigorous quality assurance (QA) processes to ensure that these standards are met [20]. Various software quality assurance models have been developed by different organizations to ensure that specific standards are met and to give guidelines on achieving these standards [15].

Bourque [27] also suggests that the implementation of quality in a software product is an effort that should be formally managed throughout the Software Engineering lifecycle. Such an approach to the implementation of quality leads to Software Quality Engineering (SQE). Suryn [34] has suggested that SQE is an application of a continuous, systematic, disciplined, quantifiable approach to the development and maintenance of quality of software products and systems. Georgiadou [10] demonstrated that more mature the process and its underlying lifecycle model, earlier the identification of errors in the deliverables.

Thus, to achieve quality in software processes and products, it is necessary to develop systematic measurement programs, compatible with the organizational objectives and tailored to the quality aspects that are being considered [30]. There is a need to establish baselines of performance for quality, productivity and customer satisfaction by the organizations. These baselines are used to document current performance and improvements by showing deviations from the baseline. In order to establish a baseline, a model must be established [26]. A quality model is a schema for better explanation of our view on quality. Some existing quality models can predict fault-proneness with reasonable accuracy in certain contexts. Few standard models are discussed in next section.

3. STANDARD MODELS

In the past years the scientific and industrial communities have proposed many QA standards and models. According to Moore [20] "there are more than 300 standards developed and maintained by more than 50 different organizations." Most popular model is ISO/IEC 9126 [16], which specify requirements for a quality management system within an organization.

The metrics listed in ISO/IEC TR 9126-2 are not intended to be an exhaustive set. Users can select or modify and apply metrics and measures from ISO/IEC TR 9126-2:2003 or may define application-specific metrics for their individual application domain. Software metrics are the only mechanized tools for assessing the value of internal attributes [17]. Software metrics are defined as "standard measurements, used to judge the attributes of something being measured, such as quality or complexity, in an objective manner" [24].

ISO/IEC 9000:2005 [13] provides guidance for the use of the series of International Standards named Software product Quality Requirements and Evaluation (SQuaRE). Software Quality in the Development Process (SQUID) [18] allows the specification, planning, evaluation and control of software quality through the software development process. SQUID uses external and internal quality measures defined in ISO 9126. Although the existence of documentation is a key requirement of a functional ISO 9001 Quality Management System (QMS), it is not in itself sufficient. To develop and implement a fully functional ISO 9001 QMS, it is essential that a small/medium-sized enterprises correctly identifies the initial state of its QMS and the path it will follow to achieve the desired state [1].

Capability Maturity Model (CMM) proposed by Software Engineering Institute (SEI) provides a framework for continuous software process improvement [31]. The key notion is that CMM provides guidelines for conducting audits, testing activities, and for process improvement. The CMM approach classifies the maturity of the software organization and practices into five levels describing an evolutionary process from chaos to discipline [31] as Initial (chaotic), Repeatable (project management), Defined (institutionalized), Managed (quantified), Optimizing (process improvement).

McCall's model [19] of software quality incorporates 11 criteria encompassing three main perspectives for characterizing the quality attributes of a software product. These perspectives are Product revision (ability to change), Product transition (adaptability to new environments), and Product operations (basic operational characteristics)

Boehm's model [8] is based on a wider range of characteristics and incorporates 19 criteria. At the highest level of his model, Boehm defined three primary uses (basic software requirements), these three primary uses are as-is utility, the extent to which the as-is software can be used (i.e. ease of use, reliability and efficiency), Maintainability, ease of identifying what needs to be changed as well as ease of modification and retesting, Portability, ease of changing software to accommodate a new environment

FURPS developed by Hewlett-Packard 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 [21]. One disadvantage of this model is that it does not take into account the software product’s portability [25].

Dromey [29] proposes a working framework for building and using a practical quality model to evaluate requirement determination, design and implementation phases. Dromey includes high-level quality attributes: functionality, reliability, efficiency, usability, maintainability, portability, reusability and process mature. In comparing to ISO 9126, additional characteristics like process maturity and reusability are noticeable.

Georgiadou [11] developed a generic, customizable quality model (GEQUAMO) which enables any stakeholder to construct their own model depending on their requirements. In a further attempt to differentiate between stakeholders, Siaka et al [22] studied the viewpoints of users, sponsors and developers as three important constituencies/stakeholders and suggested attributes of interest to each constituency as well as level of interest. More recently, Siaka and Georgiadou [23] reported the results of a survey amongst practitioners on the importance placed on product quality characteristics. Using their empirical results they extended ISO 9126 by adding two new characteristics namely Extensibility and Security which have gained in importance in today’s global and inter-connected environment.

Basili and Rombach [33] define a goal-based measurement program. The concept of the goal/question/metric paradigm is to identify goals, translate into the questions that need to be answered to determine if one is meeting or moving towards these goals, and then selecting metrics that provide information to help answer these questions.

The criteria in all above models are not independent. They interact with each other and often cause conflict, especially when software providers try to incorporate them into the software development process. There are a number of difficulties in the direct application of any of the above models. The models are static since they do not describe how to project the metrics from current values to values at subsequent project milestones. It is important to relate software metrics to progress and to expected values at the time of delivery of the software [3].

To formulate the requirements of a quantitative quality model for software development, three issues must be addressed: the different interest groups need to be identified; the intended applications of the model need to be spelled out; and it is necessary to establish the quality needs or perspectives/views of the different interest groups [28].

4. PROPOSED QUALITY MODEL

According to Dromey [28] “The first task in building a software product quality model is to identify what the intended applications of the model are and to address the needs of the different interest groups that will use the model in different applications.” The attributes of a quality model should be sufficient to meet the needs of all interest groups associated with the software.

Proposed quantitative model (Figure 1) keeps quality attributes a central consideration during application analysis and design. There are often a number of stakeholders involved in the design process with each having different quality goals. The model suggests the method of analyzing the stakeholders’ quality requirements and computes the relative priorities among the quality attributes and their subcharacteristics.

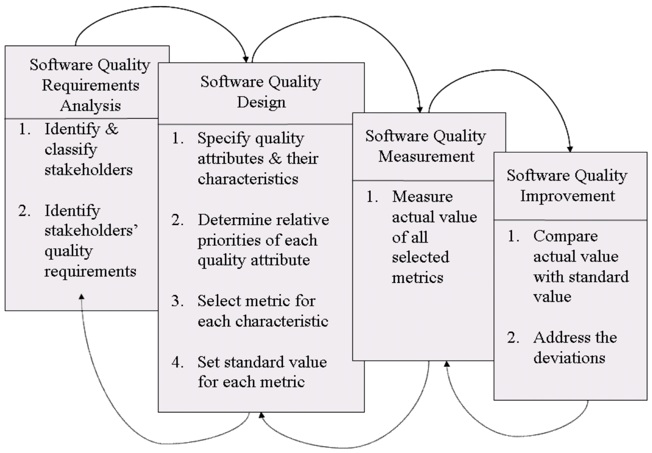
Software Quality Requirements Analysis
1. Identify & classify stakeholders
2. Identify stakeholders’ quality requirements
Software Quality Design
1. Specify quality attributes & their characteristics
2. Determine relative priorities of each quality attribute
3. Select metric for each characteristic
4. Set standard value for each metric
Software Quality Measurement
1. Measure actual value of all selected metrics
Software Quality Improvement
1. Compare actual value with standard value
2. Address the deviations

FIGURE1: Quantitative Quality Model

An integral part of an effective user-centered development is the collection of requirements and the identification of common user tasks. A number of methods can be used to gather the requirements and to identify the task groups. For quantifying the relative priorities of quality attributes and their subcharacteristics of software design, the constant sum pair wise comparison method of Analytical Hierarchy Process (AHP) [2, 32] is employed. Proposed quality model has four major phases which are discussed in following sections:

4.1 Software Quality Requirements Analysis (SQRA)

In managing customer’s quality expectations, relevant views and attributes need to be reviewed first, because different quality attribute may have different levels of importance to different customers and users [16]. For example, reliability is a prime concern for the business and commercial software systems because of people reliance on them and the substantial financial loss if they malfunction. SQRA is customized by software category, development phase and users’ requirements.

Considering all requirements, quality factors are chosen from the set of factors given by McCall quality model [19] and ISO 9126. Each quality factor is further defined by a set of attributes, called criteria, which provide the qualitative knowledge about customers’ quality expectations. This qualitative knowledge helps to quantify the goals in the software quality design (SQD). SQRA can be carried out as follows:

1) Identification & classification of stakeholders

Authors emphasize focus on identification of important stakeholders. Interview is a common method that can be employed in requirements analysis. Each stakeholder will have an idea of their expectation and visualization of their requirements. Various stakeholders may be grouped based on similar characteristics. The proposed model classifies stakeholders based on their jobs as these are directly related to quality preferences.

2) Identification of stakeholders' quality requirements
Stakeholders' quality requirements can be gathered through existing requirements gathering techniques such as Win Win requirement gathering method [6] and the goal oriented requirement gathering method [30]. External review team reviews the requirements and may add a set of quality requirements. Requirements of various groups are tabulated (see case study).

4.2 Software Quality Design (SQD)

Once all the quality requirements are listed, each attribute is quantified by individual metric as measurement is essential if quality is to be achieved. Various metrics are available for different quality characteristics. Requirements are customized to products and users, thus expected values of the corresponding metrics are determined for the product under development. Necessary steps can be followed to determine priorities of quality characteristics, related metrics and expected values which are as follows:

1) Specify quality attributes and their corresponding characteristics to satisfy stakeholders' quality requirements
Quality attributes and their subcharacteristics are specified for each stakeholder group based on ISO 9126. It is not easy to translate a user requirement (informal quality requirements) into a set of quality characteristics (formal specification of the software quality as defined in ISO/IEC 9126)

2) Determine the relative priorities of subcharacteristics of each quality attribute
The relative priorities of quality characteristics are computed for each stakeholders group. 100 points are allocated between each pair of quality attribute at a time based on their preferences for the attributes to make comparative judgment. The AHP [32] is applied to transform judgment values to normalized measures. The use of AHP to obtain the initial values is systematic quantification of the stakeholders' expectations, as it is the subjective judgments of the stakeholders.

3) Select the metrics for each characteristic
Metrics can be selected for each of the characteristics based on the concept of goal/question/metric.

4) Set the standard value for each metric
Based on the priorities and standards defined by ISO/IEC TR 9126, a standard value is associated with each metric to achieve the characteristic expected by stakeholders' groups. Once specific quality goals, expectations, measurements and metrics are set, related tools, techniques and procedures to achieve these goals can be selected.

4.3 Software Quality Measurement

The measurement offers visibility into the ways in which the processes, products, resources, methods, and technologies of software development relate to one another. According to the ISO/IEC 15939 Software Engineering – Software Measurement Standard decision criteria are the "thresholds, targets, or patterns used to determine the need for action or further investigation or to describe the level of confidence in a given result".

1) Measure the actual value
During development life cycle, in each phase, the values of selected metrics can be measured, compared and analyzed with respect to standard set values.

4.4 Software Quality Improvement

The feed back on measurement step enables the software engineers to assess the quality at each development stage and in turn helps to improve whenever violation from set goals is found.

1) Compare the actual value with standard value for each metric
Deviations may be encountered in the project plan, process description, applicable standards or technical work. Recommendations derived from the interpretation of actual value and established of the metrics are provided to the development engineers for improvement.

2) Address the deviations

The product can be adjusted to make things better and improved. It is important to establish a follow up procedure to ensure that items on the issues list have been properly corrected.

5. CASE STUDY

The case study considered in this paper is related to an automated application for job consultancy firm. A survey is carried out for various stakeholders based on their knowledge, interest and job responsibilities. Stakeholders are classified in three groups as given in table-1.

No.	Group
1	Manager
2	Developer
3	User

TABLE 1: STAKEHOLDERS GROUPS

The quality requirements for the system under development, identified by different stakeholders' groups are tabulated in table-2.

No	Group	Quality requirements		
1	Manager	Team size	Cost	Delivery time
2	Developer	Maintainability	Portability	Reusability
3	User	Reliability	Usability	Efficiency

TABLE 2: QUALITY REQUIREMENTS OF STAKEHOLDERS GROUPS

Quality attributes listed in table-2 under group developer are maintainability, portability, reusability that is further divided into subcharacteristics. This division follows quality analysis framework given by ISO 9126. Quality attributes and their subcharacteristics for the developer group are shown in table-3 as a sample. However reusability is not part of quality framework given by ISO 9126 and is addressed in this paper.

Maintainability (QA1)	Analyzability (SC11)
	Changeability (SC12)
	Stability (SC13)
	Testability (SC14)
Portability (QA2)	Adaptability (SC21)
	Installability (SC22)
	Conformance (SC23)
	Replaceability (SC24)
Reusability (QA3)	Coupling (SC31)
	Comprehensibility (SC32)
	Interoperability (SC33)

TABLE 3: QUALITY ATTRIBUTES AND SUBCHARACTERISTICS FOR DEVELOPER GROUP

A total of 100 points is distributed between each two attributes. This distribution shows their ratio scale for prioritizing quality attributes and their corresponding subcharacteristics. The relative priorities are computed by AHP constant sum method. Three experts from the developer group are identified for prioritization considering three related attributes and corresponding subcharacteristics (refer appendix A). Average and relative of the priorities of quality attributes and their subcharacteristics are computed.

5.1 Average Priorities

QA1 = Maintainability, QA2 = Portability, QA3=Reusability

QA1	60	QA1	45	QA2	45
QA2	40	QA3	55	QA3	55

SC11 = Analyzability, SC12 = Changeability, SC13 = Stability, SC14 = Testability

SC11	60	SC11	60	SC11	60
SC12	40	SC13	40	SC14	40
SC12	40	SC12	40	SC13	50
SC13	60	SC14	60	SC14	50

SC21 = Adaptability, SC22 = Installability, SC23 = Conformance, SC24 = Replaceability

SC21	55	SC21	60	SC21	55
SC22	45	SC23	40	SC24	45
SC22	50	SC22	45	SC23	40
SC23	50	SC24	55	SC24	60

SC31 = Coupling, SC32 = Comprehensibility, SC33 = Interoperability

SC31	55	SC32	45	SC31	60
SC32	45	SC33	55	SC33	40

5.2 Relative Priorities

QA = {0.354, 0.271, 0.375}
SC1 = {0.332, 0.180, 0.244, 0.244,
SC2 = {0.302, 0.223, 0.202, 0.273}
SC3 = {0.403, 0.289, 0.308}

For each required characteristic, appropriate metrics are chosen and their expected values are set based on the priorities calculated. Few relevant metrics are shown in table-4 (Source: ISO9126).

Maintainability (QA1)	Analyzability (SC11)	Reliability index
		Comment percentage
		Cyclomatic complexity
	Changeability (SC12)	Code duplication
		Maximum number of references violation
	Stability (SC13)	Correlation of complexity / size
		Global variables usage
	Testability (SC14)	Cyclomatic complexity
Portability (QA2)	Adaptability(SC21)	Mean efforts to adapt
	Installability (SC22)	Installation efforts in Man-months
		Parameter change ratio
	Conformance (SC23)	Standard conformance ration
	Replaceability (SC24)	Function change ratio
		Source code change ratio
Reusability (QA3)	Coupling (SC31)	Cohesion and coupling metrics (Fan-in & Fan-out)
	Comprehensibility (SC32)	Comment percentage
	Interoperability (SC33)	Size of domain independent part

TABLE 4: METRICS AND THEIR EXPECTED VALUES FOR QUALITY SUBCHARACTERISTICS

Similarly, other users' quality requirements are analyzed and specified. All the metrics are measured during development as and when required, and compared with the expected values for the conformance of quality characteristics expected by various stakeholders. On occurrence of any deviation, desired changes are made that shows improvement in product quality.

6. CONCLUSION

Aim of this research paper is to provide the model to establish the quality requirements expected by various stakeholders and to incorporate these requirements in the product under development. Proposed quantitative quality model takes a set of quality requirements as input for the development of a software application. The model is dynamic and allows product deliverables to be compared with set goals by various stakeholders through measurements and metrics throughout the development life cycle. The case study validates the suitability and usefulness of the proposed model. The quality attribute reusability is discussed in addition to other quality attributes of ISO 9126.

ACKNOWLEDGEMENT

The authors acknowledge the support extended by the management of PES Institute during the course of research.

7. REFERENCES

[1] Andres Sousa-Poza, Mert Altinkilinc, Cory Searcy, “Implementing a Functional ISO 9001 Quality Management System in Small and Medium-Sized Enterprises”, International Journal of Engineering, v. 3 Issue 3, 2009

[2] Arun Sharma, Rajesh Kumar, P.S. Grover, "Dependency Analysis for Component-Based Software Systems", ACM SIGSOFT Software Engineering Notes, v.34 n.4, July 2009 [DOI: 10.1145/1543405.1543424]

[3] Arun Sharma, Rajesh Kumar, P.S. Grover, “Managing Component-Based Systems With Reusable Components”, International Journal of Computer Science and Security, v. 1 Issue 2, 2007

[4] Ashley Williams, “The documentation of quality engineering: applying use cases to drive change in software engineering models”, SIGDOC '04: Proceedings of the 22nd annual international conference on Design of communication: The engineering of quality documentation, October 2004

[5] Avadhesh Kumar, Rajesh Kumar, P.S. Grover, “An Evaluation of Maintainability of Aspect-Oriented Systems: a Practical Approach”, International Journal of Computer Science and Security, v. 1 Issue 2, 2007

[6] B. Boehm et al., “Win Win requirements negotiation process: A multi project analysis”, Proceedings of the International Conference on Software Process, Chicago, 1998

[7] B. Kitchenham, S. L. Pfleeger, “Software Quality: The Elusive Target”, IEEE Software, vol. 13(1), pp.12-21, 1996

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

[9] D. Garvin, "What Does 'Product Quality' Really Mean?" Sloan Management Review, Fall, pp 25-45, 1984

[10] E. Georgiadou “Software Process and Product Improvement: A Historical Perspective”, International Journal of Cybernetics, Volume 1, No1, pp172-197, Jan 2003

[11] E. Georgiadou, “GEQUAMO– A Generic, Multilayered, Customisable, Software Quality Model”, International Journal of Cybernetics, Volume 11, Number 4 , pp 313-323 November 2003

[12] Gordon W. Skelton, “Integrating total quality management with software engineering education”, ACM SIGCSE Bulletin, Volume 25 Issue 2, June 1993

[13] ISO 9000:2005 Quality management systems Fundamentals and vocabulary, 2005

[14] ISO 9001:2000 Quality management systems Requirements, 2001

[15] ISO 9004:2000 Quality management systems Guidelines for performance improvement, 2000

[16] ISO/IEC, IS 9126-1, “Software Engineering – Product Quality – Part 1: Quality Model”, Geneva Switzerland: International Organization for Standardization, 2001

[17] ISO: ISO/IEC 14598-1. International Standard Information technology software product evaluation, 1999

[18] J. Bøegh, S. DePanfilis, B. Kitchenham, A. Pasquini, “A Method for Software Quality Planning, Control and Evaluation”. IEEE Software, 69-77, March/April 1999

[19] J.A. McCall, P.K. Richards and G. F. Walters, “Factors in software quality”, Griffiths Air Force Base, N.Y. : Rome Air Development Center Air Force Systems Command, 1977
[20] J.W. Moore, “Software Engineering Standards: A User’s Road Map”, IEEE Computer Society, Los Alamitos, CA, 1998
[21] K. Khosravi, & Y.G. Gueheneuc, “A quality model for design patterns”, http://www.yann_gael.gueheneuc.net/work/Tutoring/Documents/041021+Khosravi+Technical+Report. doc.pdf, 2004
[22] K.V. Siaka, E. Berki, E. Georgiadou, C. Sadler, “The Complete Alphabet of Quality Software Systems: Conflicts and Compromises”, 7th World Congress on Total Quality & Qualex 97, New Delhi, India, 17-19, February 1997
[23] K.V. Siaka, E. Georgiadou, “PERFUMES: A Scent of Product Quality Characteristics”, SQM, UK, March 2005
[24] M. Lorenz, J. Kidd, “Object-Oriented Software Metrics”, 1st Ed. Prentice Hall,1994
[25] M. Ortega, M. Perez, & T. Rojas, “Construction of a systemic quality model for evaluating a software product”, Software Quality Journal 11: 219-242, 2003
[26] N. E. Fenton, and M. Neil, “A Critique of Software Defect Prediction Models”, IEEE Transactions on Software Engineering (25:5), pp.675-689, September/October 1999
[27] P. Bourque, R. Dupuis, A. Abran, J.W. Moore, L.L. Trippet S. Wolff, “Fundamental Principles of Software Engineering - A Journey”, Journal of Systems and Software, 2000
[28] R. G. Dromey, “A Model for Software Product Quality”, IEEE Trans. Soft. Eng., pp 146-162, 1995
[29] R. G. Dromey, “Software product quality: Theory, model and practice. Software Quality Institute,” Griffith University, Brisbane, Technical Report, 1999
[30] S. Godbole, “Software Quality Assurance: Principles and Practices”, Alpha Science International Ltd., 2004
[31] Software Engineering Institute, “The Capability Maturity Model: Guidelines for Improving the Software Process”, MA: Addison-Wesley, 1994
[32] T. L. Saaty, “The Analytic Hierarchy Process”, McGraw Hill, Inc., New York NY, 1980
[33] V.R. Basili and H.D. Rombach, “The TAME projects: Towards improvement-oriented software environment”, IEEE Transactions in Software Engineering, 14, no.6, Nov 1988
[34] W. Suryn, “Course notes SYS861”, École de Technologie Supérieure, Montréal, 2003

APPENDIX

Developer group

a) Preferences of stakeholder 1

QA1 = Maintainability, QA2 = Portability, QA3 = Reusability

QA_1	60	QA_1	50	QA_2	45
QA_2	40	QA3	50	QA3	55

SC11 = Analyzability, SC12 = Changeability, SC13 = Stability, SC14 = Testability

SC_{11}	65	SC_{11}	60	SC_{11}	55
SC_{12}	35	SC_{13}	40	SC_{14}	45
SC_{12}	40	SC_{12}	35	SC_{13}	50
SC_{13}	60	SC_{14}	65	SC_{14}	50

SC21 = Adaptability, SC22 = Installability, SC23 = Conformance, SC24 = Replaceability

SC_{21}	55	SC_{21}	60	SC_{21}	50
SC_{22}	45	SC_{23}	40	SC_{24}	50
SC_{22}	50	SC_{22}	45	SC_{23}	45
SC_{23}	50	SC_{24}	55	SC_{24}	55

SC31 = Coupling, SC32 = Comprehensibility, SC33 = Interoperability

SC_{31}	60	SC_{32}	45	SC_{31}	60

SC_{32}	40	SC_{33}	55	SC_{33}	40

b) Preferences of stakeholder 2

QA1 = Maintainability, QA2 = Portability, QA3 = Reusability

QA_1	65	QA_1	55	QA_2	45
QA_2	35	QA3	45	QA3	55

SC11 = Analyzability, SC12 = Changeability, SC13 = Stability, SC14 = Testability

SC_{11}	60	SC_{11}	65	SC_{11}	55
SC_{12}	40	SC_{13}	35	SC_{14}	35
SC_{12}	45	SC_{12}	40	SC_{13}	55
SC_{13}	55	SC_{14}	60	SC_{14}	45

SC21 = Adaptability, SC22 = Installability, SC23 = Conformance, SC24 = Replaceability

SC_{21}	60	SC_{21}	55	SC_{21}	55
SC_{22}	40	SC_{23}	45	SC_{24}	45
SC_{22}	50	SC_{22}	50	SC_{23}	40
SC_{23}	50	SC_{24}	50	SC_{24}	60

SC31 = Coupling, SC32 = Comprehensibility, SC33 = Interoperability

SC_{31}	55	SC_{32}	50	SC_{31}	55
SC_{32}	45	SC_{33}	50	SC_{33}	45

c) Preferences of stakeholder 3

QA1 = Maintainability, QA2 = Portability, QA3 = Reusability

QA_1	55	QA_1	40	QA_2	45
QA_2	45	QA3	60	QA3	55

SC11 = Analyzability, SC12 = Changeability, SC13 = Stability, SC14 = Testability

SC_{11}	55	SC_{11}	55	SC_{11}	60
SC_{12}	45	SC_{13}	45	SC_{14}	40
SC_{12}	45	SC_{12}	40	SC_{13}	50
SC_{13}	55	SC_{14}	60	SC_{14}	50

SC21 = Adaptability, SC22 = Installability, SC23 = Conformance, SC24 = Replaceability

SC_{21}	50	SC_{21}	65	SC_{21}	50
SC_{22}	50	SC_{23}	35	SC_{24}	50
SC_{22}	55	SC_{22}	40	SC_{23}	40
SC_{23}	45	SC_{24}	60	SC_{24}	60

SC31 = Coupling, SC32 = Comprehensibility, SC33 = Interoperability

SC_{31}	55	SC_{32}	40	SC_{31}	60
SC_{32}	45	SC_{33}	60	SC_{33}	40