

SOFTWARE QUALITY ASSESSMENT TECHNOLOGY

Toshihiko Sunazuka, Motoei Azuma, Noriko Yamagishi

Software Product Engineering Laboratory, NEC Corporation
7-15, Shiba 5-chome, Minato-ku Tokyo 108, JAPAN

Abstract

Necessities for Software Quality Measurement and Assurance Technology have been increased.

B. Boehm and McCall proposed software evaluation criteria. Based on these studies, G. Murine developed Software Quality Metrics (SQM).

SQM was applied to several projects in NEC experimentally. An outline of the experiment will be presented and the results discussed.

Software Quality Measurement and Assurance Technology (SQMAT) was developed in NEC as a total technology for applying to various types and size of software projects, throughout the software life cycle.

1. INTRODUCTION

There are increasing demands for technologies to develop high quality software. Quality metrics to evaluate various kinds of software from various viewpoints in each development phase, and the methodology to use the metrics, are especially required.

B. Boehm proposed over 60 quality metrics [1] in 1976, and showed how to evaluate total software quality.

In 1977, Walters and McCall reduced quality factors to 11 candidates in on RADC (Rome Air Development Center) report [2].

Based on these studies, SQM (Software Quality Metrics) was developed by G. Murine (METRIQS Incorporated) as a quantitative software quality assessment technology. SQM has been applied to several sites in America already, and good results have been reported [3][4][5].

SQMAT was developed in NEC, taking SQM experimental use results into consideration. More than 500 persons have already been trained in its use.

This paper describes SQM outline, experimental use, SQMAT outline and its technology transfer.

2. SQM OUTLINE

SQM is a rigorous, precise software quality methodology consisting of

measurable quality factors, criteria and elements. The SQM objective is to produce cost-effective quality software. Figure 1 shows the quality metrics structure.

SQM takes a deductive measurement approach to assessing both product quality and process quality. Therefore, it has a hierarchy structure.

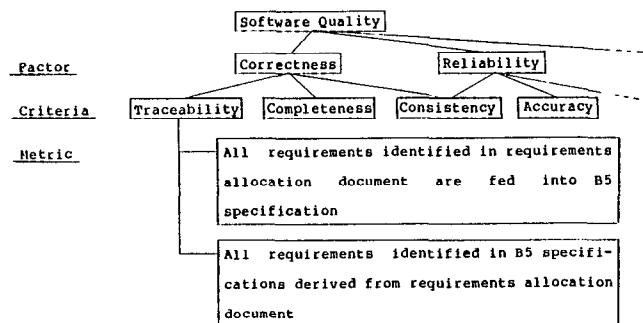


Figure 1. Quality Metrics Structure

The SQM measurement approach is as follow.

(1) Life-cycle properties (called Quality Factors) are identified and ordered.

SQM is customized by software category, development phase and user requirements. In using SQM, these conditions are taken into consideration and important Quality Factors are chosen from twelve candidates, which include one added to McCall's 11 Factors.

(2) Each Quality Factor is further defined by a set of attributes (called Criteria).

Necessary Criteria for each Quality Factor are chosen in this stage, referring to the relation matrix between Quality Factors and Criteria, whose size is a 12 x 23 digit matrix. 23 Criteria are based on McCall's 25 Criteria.

(3) Each Criteria is quantified by individual measurements (called Metrics).

In this stage, Metrics are decided in response to the necessity for each Criteria chosen. For individual Metrics, reviewers give a full account of parts to measure on the tally sheet. For each part,

if quality requirements are satisfied, this part's score is "1". If quality requirements are not met, the score is "0". Criteria score is a ratio of "1" and Quality Factors score is a weighted sum of Criteria scores.

3. SQM EXPERIMENTAL USE

3.1 Preliminary Study and Experiment Preparation

The "Software Productivity Committee" was organized to solve software problems and to increase software productivity and quality as a company-wide committee in NEC. The committee includes several task groups. The "Quality Assurance Task Group" was established in September 1982, to develop technologies, i.e., metrics, methodologies and tools, for software quality measurement and assurance.

As the first step to develop the technologies, Murine and his associates were invited and a one week SQM and SQA (Software Quality Assurance) seminar was held.

Some efforts were required for translating the materials, understanding the technology, discussing how to use it, and preparing the materials written in Japanese.

The "SQA/SQM Text" was issued six months later by the group, and was used at the first SQM pilot seminar. In October 1983, the text was revised. The "SQM Handbook", which contains instructions on how to measure software quality in detail, was also issued.

The second training course was held in November, aiming to transfer SQM technology to those who wanted to use it experimentally on actual projects.

3.2 Experiments Outline and Target Projects

SQM was applied experimentally to some actual projects, taking as much caution as possible to retain the original technology style. Data were collected for use in assessing the technology. The following describes an experiment outline and the results obtained from four projects. Figure 2 shows the applying phase and assessment object for each project.

Phase	Definition	Design	Manufacture	Test
Project				
A	↔		(c)	
B		HIPO	COBOL	
C		SPD	COBOL	
D		(SP chart)	(PL/I)	PL/I

Figure 2. Applying phase and assessment object for each project

Project A

- (1) Software category : Software development and maintenance software
- (2) Assessment object : Requirement Specifications in the requirements definition phase
- (3) Organization : 4 persons assessment team without the writer
- (4) Quality metrics : 28 metrics about "Usability" (Fig.3)
- (5) Assessment cost : 3 or 4 hours for individual assessment in advance and 1 or 2 hours for team assessment
- (6) Effect :

The number of pages for the specifications was reduced from 22 pages to 12. The specifications became easy to read and more understandable.

Only necessary functions, based on user needs, had been described.

Development costs, after the requirements definition phase, diminished by one third.

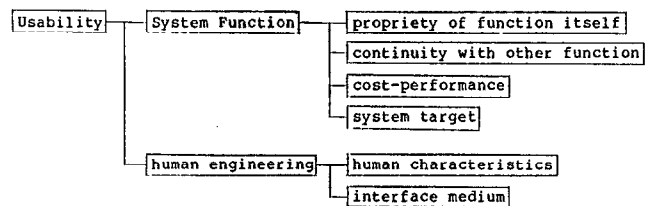


Figure 3. Quality metrics for project A

Project B

- (1) Software category : A cost control system
- (2) Assessment object : Detailed specifications written by HIPO and source list written by COBOL for 4 programs
- (3) Organization : 3 persons assessment group; the project leader and two from a third group
- (4) Quality metrics : 15 Metrics derived from 5 Criteria (cf. Fig.4) about "Correctness" and "Reliability" in the design phase, and 22 Metrics in the manufacturing phase
- (5) Assessment cost : 8.5 hours / kilo lines in design phase and 8.9 hours / kilo lines in manufacturing phase
Assessment cost ratio was 20.8%.
(Design cost, manufacturing cost and test cost were 36.9, 19.5, and 9.7 hours / kilo lines, respectively.)
- (6) Effects :

Effects were as shown in Fig.5.

SQM cut down the test cost from 19.1 hours / kilo lines to 9.7.

After release, no error was detected in the part of the system assessed by SQM, but 2 bugs / 19.167 kilo lines were detected in the rest of that part.

Total quality score in the design phase was .84 and that in the manufacturing phase was .98.

Total quality score in the design phase was .84 and that in the manufacturing phase was .98.

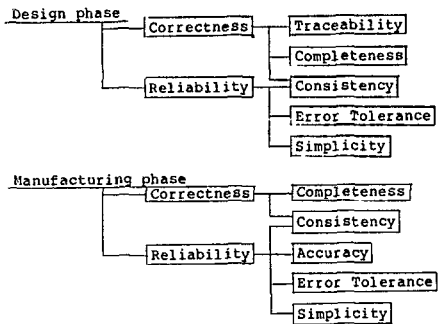


Figure 4. Quality metrics for project B

Quality	number of bugs detected during 3 month period after release 0 bugs/KL (SQM applied) 0.104 bugs/KL (SQM not applied)
Cost	test phase cost 49% cut down

Figure 5. SQM effect

Project C

- (1) Software category : Business application software (small scale program)
- (2) Assessment object : Detailed specifications written by SPD (Structured Programming Diagram) and source list written by COBOL
- (3) Organization : 1 assessor from a third group
- (4) Quality metrics : 9 Metrics in the design phase and 8 in the manufacturing phase about 2 Factors (Fig.6)
- (5) Assessment cost : 4.8 hours / kilo lines in the design phase and 4.6 hours / kilo lines in the manufacturing phase

Total assessment cost was 12.7% of the software development cost. (Design, manufacturing and test costs were 27.0, 23.9 and 13.9 hours / kilo lines, respectively.)

- (6) Effects :

SQM saved 46.2 % of the cost for the manufacturing phase (from 44.4 hours / kilo lines to 23.9) and 19.9 % for the test phase (from 17.4 hours / kilo lines to 13.9).

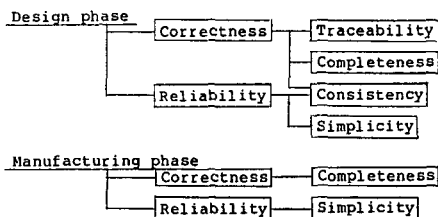


Figure 6. Quality metrics for project C

Project D

- (1) Software category : A part of an operating system for a telephone exchange
- (2) Assessment object : supplied software written by PL/I for 3 programs
- (3) Organization : 2 persons in NEC, as a part of the acceptance test
- (4) Quality metrics : 12 Metrics about 2 Factors (cf. Fig.7), in addition to acceptance test
- (5) Assessment cost : 6.7 hours / kilo lines (= 11.0 hours / 1.643 kilo lines).
- (6) Effects :
Subcontractor sales-points and weak-points become clear.
In subcontractor management, mainly quality guidance was carried out.

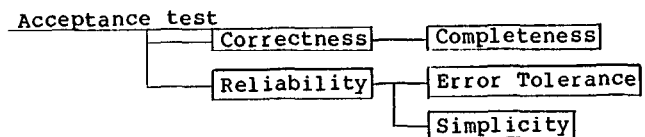


Figure 7. Quality metrics for project D

3.3 Results

Two sorts of data were gathered to assess the SQM technology.

- (1) Scores on the same object by different reviewers (cf. Table 1)
- (2) Number of errors in the test phase for each project (cf. Table 2)

Table 1. Scores on the same object by different reviewers

	Veteran	Newcomer-1	Newcomer-2
Score	.915	.957	.932
Number of measurement points	201	139	132
Number of detected errors	17	6	9
Measurement time (minutes)	150	180	185
Measurement time / point	.746	1.295	1.402

Table 2. Relationship between number of errors and SQM score

Comparison items Program	Number of errors in test phase (errors / KL)	SQM score	Detailed score	
			Correctness	Reliability
Prog-1	13.9	.944	.905	.986
Prog-2	21.9	.921	.943	.904
Prog-3	5.3	.974	.963	.988

The difference between scores by different reviewers is within 5% at most (Table 1). It was concluded that the score was objective and that anyone can use the SQM for measuring, because the reviewer group included personnel ranging from newcomers to veterans.

It was found, from Table 2, that the higher the SQM score is, the fewer the errors that could be detected in the test

phase. It was concluded that high correlation exists between SQM score and result of quality assessment in the test phase. Thus, the score is considered to be reliable.

4. SQMAT (Software Quality Measurement and Assurance Technology)

Taking the SQM experimental use result into consideration, SQMAT was developed as a technology which will satisfy the necessity for company-wide use. This necessity means that the technology should be applicable not only to a large software project, with sufficient quality assurance staff, but also to a very small project working on a tight schedule. The software category also varies, i.e., built in micro processor program, small business application, electronic telephone switch, large realtime application, and main frame computer operating system.

SQMAT is integrated technology which consist of;

- (1) Strategy to use the technology throughout the software life-cycle,
- (2) Metrics and methodology for measuring quality,
- (3) Methodology and tool for quality evaluation and assurance,
- (4) Documentations and training packages for transferring the technology.

4.1 Requirements

Requirements applicable to the technology, considering the above mentioned necessity for company-wide use, are as follows.

- (1) The technology should be goal oriented. This means that quality requirement should be defined quantitatively and clearly from a user viewpoint as well as a functional requirement.
- (2) Quality should be measured, evaluated and assured from all the quality aspects, based on software category.
- (3) A reasonable number of quality aspects, from the user's viewpoint, is between 5 to 9, considering human recognition and understanding capability.
- (4) Accurate and simple shortcut measurement methods should both be provided, because quality should be measured and assured at reasonable cost.
- (5) Technology should be easy to understand and use, and technology transfer facilities should be provided.

4.2 Strategy

The feed forward control principle is applied, in addition to the feed back control principle.

In many cases, software quality is

improved in the process by design review and code inspection. The number of errors will be detected and corrective action will be carried out by the final test. This is feed back control. It is good for quality, but a more positive control will contribute to quality improvement; that is, quality improvement action should be accomplished before errors are detected. This is feed forward control. The procedure is to establish the quality target first, then discuss how to accomplish achieving the desired high quality and how to include the data in the development process.

This procedure is shown in Fig.8.

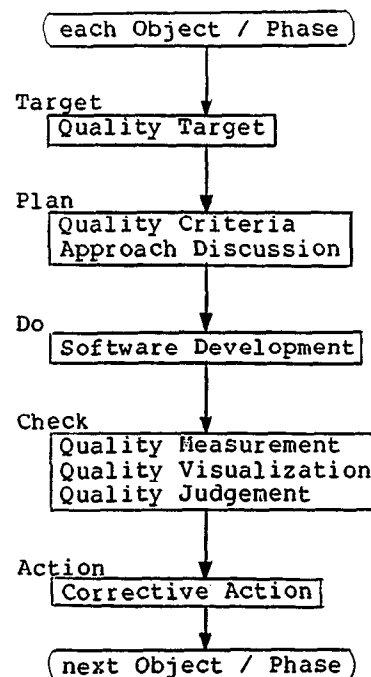


Figure 8. SQMAT procedure

4.3 Metrics

Metrics are segregated into three levels, i.e., Software Quality Requirement Criteria (SQRC), Software Quality Design Criteria (SQDC) and Software Quality Measurement Criteria (SQMC).

SQRC is the quality criteria, based on user requirements and software characteristics, which correspond to the Factor.

SQDC is the quality criteria, based on designer and implementor viewpoint, to satisfy the requirements, which correspond to Criteria.

SQMC is the review checklist, used in inspecting to determine whether the quality is good or not. This is corporate know-how. It can be used to measure results quantitatively.

Relationship between SQRC and SQDC is shown in Table 3.

Table 3. Relationship between Software Quality Requirements Criteria and Software Quality Design Criteria

Software Quality Requirements Criteria \ Software Quality Design Criteria	Correctness	Reliability	Maintainability	Flexibility	Usability	Efficiency	Security	Interoperability
Traceability	○							
Completeness	○							
Consistency	○	○	○					
Simplicity		○	○					
Accuracy		○						
Error Tolerance		○						
Modularity			○	○				○
Self-Descriptiveness			○	○				
Conciseness			○					
Instrumentation			○					
Generality				○				
Expandability				○				
Training					○			
Communicativeness					○			
Operability					○			
Machine Independence				○				
Software System Independence				○				
Execution Efficiency						○		
Storage Efficiency						○		
Access Control							○	
Access Audit							○	
Data Commonality								○
Communication Commonality								○

4.4 Methodology

It is suggested that software quality be measured and assured by all Software Quality Requirement Criteria. To avoid increasing measurement costs by this suggestion, SQRC importance ranking and simple measurement methods were considered.

"SQRC importance ranking"

SQRC are classified according to importance rank. Rank A means "very important", rank B is "important", and rank C is "ordinary". When the criteria is meaningless for the software, rank is "-", which means "not applicable". Quality to be satisfied prior to others differs for different software categories and user requirements. The difference in importance rank is reflected in the measurement method. That is, the higher the required quality is, the more detailed and precise the measurement is.

"Measurement methods"

There are three types of measurement method;

- a : Accurate method
- b : Comprehensive method
- c : Shorthand method

Accurate method

This measurement method corresponds to the rank A criteria. For each SQMC, list all points to be measured, and assess each point by YES-NO assessment or 4-stage assessment.

Assessment criteria is as follow. YES is "satisfactory", and NO is "unsatisfactory" for YES-NO assessment. 4 is "very satisfactory", 3 is "satisfactory", 2 is "almost satisfactory", and 1 is "unsatisfactory" for 4-stage assessment.

Comprehensive method

On each SQMC, where it is not necessary to measure in detail, the whole assessment object can be measured by 4-stage assessment.

This method takes less measurement cost than accurate method.

Shorthand method

On each SQDC, the whole assessment object can be measured by 4-stage assessment. Though the assessment viewpoint is too wide to measure quality accurately, this method is effective to measure multiple viewpoints at a small cost.

Guide-lines were set up on the relationship between importance rank for SQRC and measurement method (cf. Table 4). In Table 4, SQM corresponds to rank A and accurate measurement. Other cases are not considered here.

Table 4. Relationship between quality importance and measurement method

Quality importance \ Measurement method	(A) Very important	(B) Important	(C) Ordinary
(a) Accurate measurement	◎	○	○
(b) Comprehensive measurement	X	◎	○
(c) Shorthand measurement	X	X	◎

◎ : standard case

○ : case which has sufficient QA cost

X : case not desirable

This SQMAT methodology has been accepted by a growing number of site engineers.

4.5 SQMAT Technology Transfer

The following two actions have been taken to cause this methodology to become more practical.

- (1) Support tool development
- (2) Training course development and carrying it out

Support tool : The tool provides functions for quality metrics

establishment support and for display of a graph indicating quality, so a manager can determine the present quality state and can execute proper action. Figure 9 shows an example of the SQRC result. If there is a difference between the target and the result, the cause is investigated thoroughly by graphing the SQDC

Training course : A curriculum was developed for the software engineers and for the managers. Methodology, case study and practice are covered. These courses are held continuously for transferring the technology at a pace of a hundred persons a month. As the result, it has been penetrating into a number of software development sites in NEC. The necessity for assessment from a multiple point of view and the importance of quantitative control have become more widely understood.

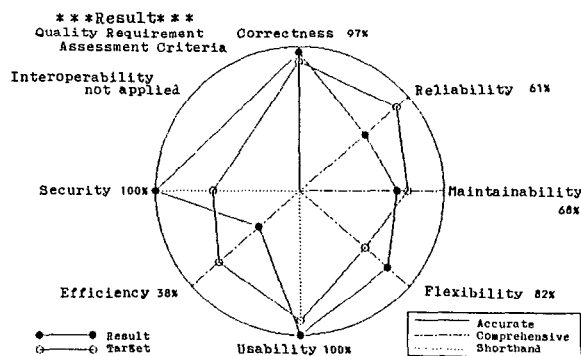


Figure 9. Example of displaying quality

5. CONCLUSION

Four items have been described. SQM provides ways to carry out visual management on software quality, through indicating the quality quantitatively. SQMAT can be applied to every scale of software, because of the economical measurement method development. Supplemental use of SQM and SQMAT provides more effective management methods. Further study on SQMAT assessment and quality control system development are needed.

The following results were obtained by using SQM.

- (1) The quality target can be concretely established, because of the quantifying quality.
- (2) Quality can be assessed objectively, because there are only small differences between reviewers' score.
- (3) "Visual management" can be put into practice through displaying quality graphically.

SQMAT, which matches the environment in NEC, brought about other effects.

- (1) It can be a common quality criteria, because of assessing from a fixed multiple point of view.
- (2) It can be applied to small projects, within a reasonable measurement cost, because of using the comprehensive or shorthand method.

SQM and SQMAT are necessary technologies to measure quality quantitatively for managing software quality, which is a subject of world-wide interest. Thus, international standards on quality criteria, especially SQRC and SQDC level, will be necessary. Metrics (METRIQS Inc.) and SQMC (NEC Corp.) should be integrated and classified by software category, requirements, etc.

SQMAT is being assessed now at several sites in NEC. The strategy is positive for producing high quality, using a measuring method reasonable in cost.

By using SQMAT, it is expected that high quality could be achieved and a good deal of the cost for test and maintenance phase could be saved.

Both SQM and SQMAT are self-contained technologies. They can be used supplementally in measuring software quality, because the second level quality criteria is the same (Criteria and SQDC) and measurement methods don't overlap each other.

Future themes are as follow.

- (1) Development of a quality control system, which can show on the monitor both present and future state of quality at each development phase.
- (2) Correlation analysis between the score by SQMAT and user satisfaction.

As the demand for software increases, the difference between demand and supply for software personnel is increasing and serious problems are arising. It is therefore necessary to develop both product engineering and management engineering. The authors are convinced that this methodology contributes to management engineering, from a standpoint of not only controlling quality but also managing software development by means of quality assessment.

ACKNOWLEDGEMENT

The authors thank Mr. Gerald E. Murine of METRIQS Incorporated, Mr. C.L. (Skip) Carpenter, Jr. of General Dynamics Corporation and Mr. Hiroyuki Ochi of YDK Consultants International for their helpful SQM guidance and support. They are grateful to Dr. Yukio Mizuno and Dr. Kiichi Fujino for their continuous encouragement and advice to the authors.

They also thank Mr.Hiroyuki Mori and other QAG colleagues for their contributions to this study.

<References>

- [1] Boehm,B.W. and others : "Quantitative Evaluation of Software Quality", 2nd ICSE, Vol.2, pp.592-605, 1976
- [2] RADC TR-77-369, 1977
- [3] Murine,G.E. : "Applying Software Quality Metrics in the Requirements Analysis Phase of a Distributive System", Proceedings from Minnow Brook Conference, 1980
- [4] Murine,G.E. and others : "Applying Software Quality Metrics", ASQC Quality Congress Transactions, 1983
- [5] Murine,G.E. and others : "Measuring Software Product Quality", Quality Progress, Vol.17, No.5, pp.16-20, 1984
- [6] Mizuno,Y. : "Software Quality Improvement", 6th compsoc 82, 1982
- [7] Azuma,M. and others : "Quantitative Assessment Technique of Software Quality (1) -- required conditions and operation --", Information Processing Society of Japan, 29th National Conference, p.681-682, 1984
- [8] Sunazuka,T. and others : "Quantitative Assessment Technique of Software Quality (2) -- metrics and measurement method --", Information Processing Society of Japan, 29th National Conference, p.683-684, 1984
- [9] Yamagishi,N. and others : "Quantitative Assessment Technique of Software Quality (3) -- real cases and assessment --", Information Processing Society of Japan, 29th National Conference, p.685-686, 1984

<Appendix> Quality Metrics Definitions

Software Quality Requirement Criteria Definitions

CORRECTNESS

Extent to which a program satisfies its specifications and fulfills the user's mission objectives.

RELIABILITY

Extent to which a program can be expected to perform its intended function with required precision.

EFFICIENCY

Extent to which a program performs a function with required computing resources and code.

SECURITY

Extent to which access to software or data by unauthorized persons can be controlled.

USABILITY

Effort required to learn, operate, prepare input, and interpret output of a program.

MAINTAINABILITY

Effort required to locate and fix an error in an operational program, and to test a program.

FLEXIBILITY

Effort required to modify and/or transfer an operational program, and/or to use a program in other applications.

INTEROPERABILITY

Effort required to couple one system, one sub-system or one module with another.