In practice, quality requirements of software products are often described in vague and general terms. As a consequence, it is difficult for test engineers to verify the quality of the software product against these requirements. This article describes a method used to identify the most important quality characteristics by means of a risk assessment, to define completion criteria, and to subsequently measure software quality characteristics using the ISO 9126 standard. The authors have applied this method, among others, during testing of copier/printer controller software at Océ Technologies, a Dutch developer and manufacturer of copying and printing equipment.

Key words: completion criteria, ISO 9126, metrics, product quality, quality profile, risk analysis, testing

Measuring Software Product Quality

ERIK VAN VEENENDAAL AND ROB HENDRIKS
Improve Quality Services BV
ROBERT VAN VONDEREN
Océ Technologies

PROJECT CONTEXT

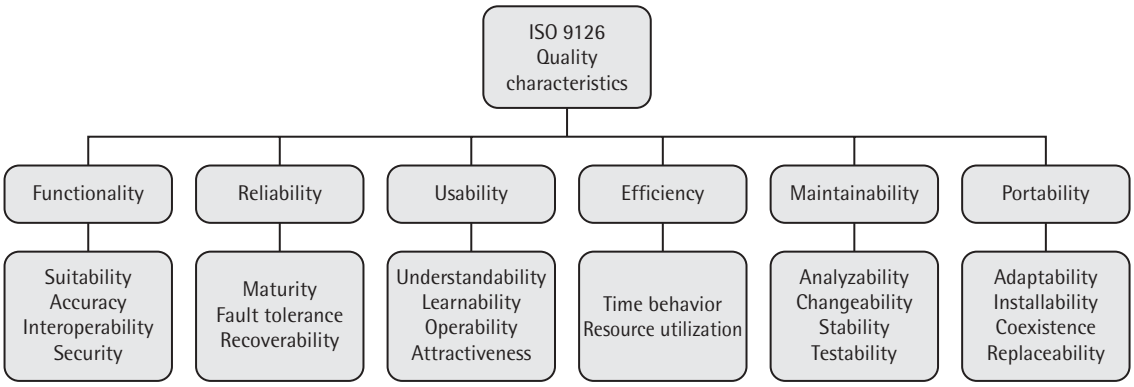
In 1997, Océ Technologies began developing a new line of copier/printer controllers to be used in a new family of monochrome high-volume hybrid copier/printers. In April 1999 this development entered the engineering phase. The engineering of the software for this controller line took place at three sites: two in The Netherlands and one in France. Approximately 60 software engineers were involved in developing this software product, hereafter referred to as the "controller software." The controller software was developed in an incremental way, with each development cycle conforming to the V-model (Myers 1979). Each increment took between three and five months.

At the start of the engineering phase a test and quality assurance team was formed with the assignment to verify correct functional behavior and to determine the quality of the controller software. One problem that the test team encountered was that the required quality level of the controller software was not specified. The only quality requirements available referred to the copier/printer product as a whole and not to its software components. However, the available requirements were easy to measure and easy to understand, for example, the mean number of copies between failures (MCBF, which pertains to all system errors) and the average copies per repair (ACPR, which is the average number of copies made between visits of a service engineer).

QUALITY MODEL: ISO 9126

The objective of the test team was to get a clear baseline of the quality requirements before beginning the testing phase.

FIGURE 1 ISO 9126 quality characteristics overview



Therefore, a quality model had to be found that could help define and measure software product quality. In 1977 McCall, Richards, and Walters (1977) proposed the idea of breaking down the concept of quality into a number of quality factors. The idea has been followed by many other authors who have tried to capture software product quality in a collection of characteristics and their depending subcharacteristics, which, in turn, are connected to indicators and metrics. By doing so, every author imposes his or her own hierarchically layered model of software product quality. In these varying models some elementary characteristics keep reappearing, although their place in the hierarchy can differ. In recent years the International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC) have defined a standard set of quality characteristics. This set reflects a big step toward consensus in the software industry and thereby addresses the general notion of software quality. This ISO 9126 standard (ISO 9126-1 2001) defines six quality characteristics and the subdivision of each quality characteristic into subcharacteristics (see Figure 1). The six quality characteristics defined in ISO 9126 are: 1) functionality, 2) reliability, 3) usability, 4) efficiency, 5) maintainability, and 6) portability.

SELECTION OF PRODUCT QUALITY CHARACTERISTICS

Terminology

Not all quality characteristics are of equal importance to a software product. Portability may be unimportant when the development aims at a dedicated platform and maintainability may not be an issue when it is a product with a short life cycle. Therefore, the most important quality (sub) characteristics must be identified and selected by means of a risk assessment. This can be done by interviewing stakeholders inside the project (such as the product manager, the project manager, and the software architect) and outside the project (the various types of users are important). Copier users are not just the people operating the copier; often forgotten are stakeholders such as the service engineer and the system administrator.

The quality characteristics as defined by the ISO 9126 standard are not always easy to interpret. What does maintainability, or even worse, coexistence, mean? It is difficult to communicate regarding these quality characteristics in an unambiguous way. As it will be hard to understand for information technology (IT) professionals, it will be even harder for users of the copier, who, in general, have no or limited IT knowledge. Most of the users do not even perceive that the product contains software. The quality characteristics are usually not part of the stakeholders' terminology. It will therefore be difficult to determine those quality characteristics that are important for the software component of the product just by asking, "Do you think resource behavior is important?" The people interviewed have their own definition of the characteristics and their own view of what the system should do and what is important.

Structured Questionnaire

To overcome these problems, a questionnaire risk-based method has been developed and empirically

FIGURE 2 Sample questions structured questionnaire

Question	Quality characteristic
What is the number of products to be sold in a certain market area? (see Trienekens and van Veenendaal 1997) 1. 1 – 1,000 2. 1,000 – 10,000 3. More than 10,000	Suitability Maturity
Does the product belong to a product family? 1. No 2. Yes	Analyzability Changeability
Can software product failures cause damage? 1. Some economic loss 2. Serious economic loss 3. Safety damage or environmental disaster	Maturity Operability
What is the average experience of the recognized user groups with regard to the product? 1. More than one-year experience 2. Less than one-year experience 3. No experience	Understandability
How many users can be identified per product? 1. 1 – 10 2. 10 – 100 3. More than 100	Maintainability Reliability Usability
Are there any alternatives to carry on with the activities when the software fails? 1. Yes 2. No	Reliability
Are certain parts of the software available on more than one platform? 1. No 2. Yes	Portability

validated in the European ESPRIT project SPACE-UFO (Van Veenendaal and McMullan 1997; Trienekens and Van Veenendaal 1997). The method relates the wishes and needs of stakeholders regarding the software product to software quality characteristics. The stakeholders' needs and requirements are explored and identified using a structured questionnaire. The questionnaire consists of a number of questions regarding so-called generic product characteristics. These product characteristics are intended to be stated in the stakeholders' language. Instead of asking whether usability is important one asks questions about the characteristics of the users that influence the usability requirements, for example, the number of users, their experience with the product (or a similar one), and their educational level. Similar questions are asked for the other quality characteristics. The answers given are used to deduct the most relevant quality (sub) characteristics by means of two-dimensional matrices. The SPACE-UFO questionnaire is a standardized questionnaire and was used without major customization for the Océ controller project. A part of the structured questionnaire is shown in Figure 2. For each question the related quality (sub) characteristics are indicated.

Quality Profile

Within the copier/printer project four key people were selected. In addition, three representatives of the different types of users were selected. Thus, seven people were interviewed, each having his or her typical view on the copier/printer product. From the participants' answers, a list of important quality characteristics can be deducted. Furthermore, importance levels can be identified (from level D to level A). This whole process results in a so-called quality profile (ISO/IEC 14598-5 1996). A quality profile reflects the notion of quality for a certain software product and makes quality more tangible for all stakeholders (including testing). Once the initial quality profile has been defined, a consensus meeting is organized with all stakeholders to discuss the quality profile, the results of the interviews, and the observed differences. Figure 3 shows the resulting quality profile (on a quality characteristic level) for the controller software.

As can be observed in the quality profile, the quality characteristics functionality, reliability, usability, and

FIGURE 3 Quality profile

	Quality Level				
	–	D	C	B	A
Functionality				X	
Reliability				X	
Usability			X		
Efficiency		X			
Maintainability			X		
Portability	X				

maintainability were considered the most important for the controller software. This can be explained for the type of copier the project is developing: It is expected to run in a highly professional document production environment, where the uptime (highly related to "reliability" and "maintainability") is of utmost importance. Also, the copier/printer will be the successor product to an analog high-volume copier model, where a clear functionality demand (being compatible with existing products as well as providing an extension) is expressed. Usability scores high but was not considered further for testing the controller software. This is because usability is a property of the user interface, which was not part of the controller software development project, as it is developed in a separate project.

As a result, the test team decided to focus on three quality characteristics (functionality, reliability, and maintainability), and to develop a set of completion criteria (metrics) for these quality characteristics.

IDENTIFICATION OF QUALITY METRICS

Once the quality profile has been determined, it is time to start focusing on how to evaluate and measure the relevant quality (sub) characteristics. The ISO 9126 standard defines in part two (External metrics) and part three (Internal metrics) a large number of possible metrics per characteristic, including measurement scales and application guidelines (ISO 9126-2 2002; ISO 9126-3 2002). A selection must be made of the metrics provided and, if necessary, extended with self-defined metrics. Within the project for each quality characteristic, a selection of metrics was made based on either ISO 9126 or on the product requirements (for example, MCBF). The selection of metrics to be used was mainly based on whether it was possible and easy to measure them, since this was considered to be a pilot project within Océ for measuring software product quality.

Most of the metrics were used directly from the ISO standard, but some had to be fine-tuned to the project's definitions. An example is the ISO 9126 metric mean time between failures (MTBF). This metric is often used to give an indication of the maturity of the system. For copier/printers the maturity is often indicated by means of the metric MCBF, as the number of copies made is leading instead of the up-time. Therefore a slight change for some metrics was desirable. Furthermore, the product requirements often include quality statements applicable to the product as a whole. As only the controller software was taken into account, these quality statements needed to be translated to the controller software. For example, the MCBF is higher for the controller software than for the product, because only failures caused by the controller software are considered. For example, paper jams are not important when evaluating the software maturity.

In total 16 metrics were defined, one of which was in addition to the internal and external metrics provided in the ISO 9126 standard. The availability of design documentation was considered to be a good indication of the maintainability. Therefore a metric was added to verify the amount of design documentation that was available vs. the amount of design documentation planned. An overview of some of the metrics defined can be found in Figure 4.

DEFINING COMPLETION CRITERIA

It was decided that completion criteria, as a hypothetical baseline, had to be defined before starting actual measurement. The goal was not to measure relative improvement with each test cycle but to compare the measured quality against the completion criteria, which must be reached before the product can be released. Defining completion criteria in advance forces people to start discussion when quality targets are not met. If no baseline is defined one is tempted to accept the quality as is because no tangible reference exists. The product is easily considered to be "good enough."

These completion criteria were defined by asking some experienced engineers within the project and on comparable software projects for an estimate on each metric. These estimates were then compared and discussed in a team meeting. Each estimate needed to state the "minimum but sufficient quality level" for release. For example, the metric MCBF was defined to be 100,000. This means that only one failure, caused by the controller software, may occur every 100,000 copies. It was decided to measure all criteria (when applicable) during the system test phase of each incremental development cycle. The result of each development cycle could thus be scored against the defined baseline. In this way, both the improvements per

FIGURE 4 Examples of product quality metrics

Quality characteristic	Subcharacteristic	Metric	Explanation
Functionality	Suitability	*Functional implementation completeness:* Number of missing functions detected during system testing / Number of functions described in requirement specifications.	How many functions have been implemented in relation to the number of functions specified in the requirement specifications?
		Functional implementation correctness: Number of correctly implemented functions confirmed during system testing / Number of functions described in requirement specifications.	How many functions have been implemented according to the requirement specifications?
Reliability	Maturity	*Mean copies between failures:* Total number of copies during system testing / Number of defects, caused by controller software, detected during operation time.	How frequent are the defects of the controller software in operation?
		Defect detection: Absolute number of defects detected during all test phases / Number of estimated defects to be detected, measured for each release.	What is the proportion of the defects found?
		Test completeness: Number of actually executed test cases / Number of test cases to be executed.	How reliable is the test process for product quality statements?
Maintainability	Analyzability	*Availability of design documentation:* Available design documentation (that is, software architecture, top-level design, analysis views, design views, and interface specifications) / Identified design documentation.	What is the proportion of design documentation available?
		Inspected design documentation: Inspected design documentation / Available design documentation.	How reliable and correct is the content of design documentation?

development cycle, as well as the discrepancy with the "minimum but sufficient quality level," could be depicted and reacted upon.

Collection of the measurement data during the system test phases implied that thorough administration was necessary during test execution. In addition to defects found, it was important to record the number of copies made, the total down time of the copier/printer, the number of failures successfully restored by the system, and so on. It was important to clearly instruct the test engineers on what data should be recorded, otherwise important information necessary to calculate the metrics could be missing. The test engineers were given instructions on how to collect the metrics, and a set of supporting forms was established.

After each system test phase the measured values were evaluated to see whether the product was approaching its completion criteria. If necessary, the completion criteria can be modified, but not after a thorough discussion within the steering committee regarding the desired quality levels. Management should always formally approve a change in the completion criteria.

REAL-LIFE MEASUREMENTS

Until the first commercial release of the controller software, five development increments were evaluated and measured. For one increment, an additional system test phase was added, leading to a sixth measurement (C2 patch). Some measurement results of the increments are presented in Figure 5.

From Figure 5 one can see that the MCBF was far below the completion criteria defined. Still the completion criterion was not adjusted. The low number for MCBF for increments C1, C2, and C2 patch was

FIGURE 5 Measurements results

Quality characteristic	Sub-characteristic	Metric	Completion criteria	Value C1	Value C2	Value C2 patch	Value C3	Value C4	Value C5
Functionality	Suitability	Functional implementation completeness	0.90	0.91	1.0	1.0	**0.82**	0.90	0.91
		Functional implementation correctness	0.80	**0.45**	0.80	0.84	**0.61**	**0.70**	**0.64**
Reliability	Maturity	Defect detection[1]	0.75	**0.46**	**0.63**	**0.63**	1.08	1.19	1.43
		Mean copies between failures	100000	not available	**93**	**175**	**4880**	**11204**	**6560**
		Test completeness	0.90	**0.75**	**0.78**	**0.33**	0.92	0.97	0.95
Maintainability	Analyzability	Available design documentation[1]	not defined	0.70	0.62	0.62	0.66	0.73	0.70
		Inspected design documentation[1]	0.75	**0.71**	**0.67**	**0.67**	**0.59**	**0.49**	**0.49**

[1]These measurement values are cumulative.

mainly because the controller software was not yet robust for failures in the scanner or printer. For example, a paper jam also resulted in a failure in the controller software. From increment C3 onward the software was robust for paper jams, and the MCBF increased tremendously. Still the value was far below the initially defined quality target. It appeared that during the initial estimate it was not taken into account that the tests were aiming at finding functional defects in the software and were not defined with the objective of finding defects by producing large numbers of copies.

It is interesting to compare the measurements from increment C2 patch to those from increment C2. The software test on increment C2 was hindered by major instability and performance problems. These problems were resolved, after which measurement C2 patch took place. Although the difference between the metrics of increment C2 and increment C2 patch is relatively small (only nine major defects were solved, approximately 5 percent of the total number of defects solved), the users perceived increment C2 patch as a much "better" system. This example shows that the metrics and values indicated in Figure 5 should be interpreted with great care. For instance, when a system only shows a few critical defects in the most important part of that system, it will be of an unacceptable quality level, however the metrics may show otherwise. The severity of a defect and its location in the system are not taken into account. This means that an experience-based quantitative evaluation is needed in addition to the metrics.

From increment C3 onward one can see that the defect detection rate is greater than 1, which shows that more defects were found than initially expected. The estimate for the defects expected to be found may be incorrect—this needs further investigation. Literature (for example, Hatton 1995 and Pfleeger, Hatton, and Howel 2001) and historical metrics from previous new products of the Océ organization were used to estimate the number of defects expected to be found. The main assumption was that on average six defects could be found per thousand lines of code (KLOC). For reused code this value was adapted to two defects per KLOC. Furthermore, automatically generated code was seen as newly developed code, and, therefore, also six defects per KLOC were expected.

The number of defects found vs. the number expected should be approximately one for each increment. Note that this is a cumulative value. In the project, however, defects were found rather late over the increments. At increment C3 the integration test process was strengthened, which may explain the rise of defect detection from that moment on.

After increment C3 a major shift of responsibilities between the test team of the controller software and the system test team of the copier/printer occurred. From that point on the functional tests were the responsibility of the copier/printer test team, while integration testing remained the responsibility of the controller software test team. Because of this shift a different test strategy was applied to system testing. The printer/copier system test team did not only focus on functionality, but also on performance and duration. Therefore, more copies were made with a minimum number of user interactions, which affected the total number of copies made during a system test phase. This affected the metric MCBF. The types of tests that have been executed are important to consider when evaluating the metrics.

Some of the measurement values show a decrease over the various increments. This can be explained by the fact that measurements for a certain increment only take into account the documentation and code that has been developed within the particular increment. For example, the metric "functional implementation completeness" is determined for the functionality developed within each increment individually. It is then possible that a measured value decreases compared to that of a previous increment. For cumulative measurements all increments are taken into account. For example, when during a particular increment many design documents are written but not inspected, this will have a negative effect on the metric "inspected design documentation."

THE RELEASE DECISION PROCESS

The software quality metrics were measured within the project for more than two years. During that period much experience was gained on the metrics themselves, how to interpret them, and how they could be used, such as for setting project priorities. The metrics were initially only applied to provide visibility on software product quality. The next step was to submit changes in the development process to improve the quality of the software and the process. For example, if the defect detection rate is lower than expected, as can be seen for releases C1 to C2 patch, but the test completeness is high (75 percent to 78 percent), it is necessary to evaluate the effectiveness of the test process. It is, of course, also possible that the engineering team makes fewer errors than expected.

Of course, if the completion criteria are not met, additional tests may be required. However, the tension between time-to-market and product quality will always remain. The metrics provided by using this method are a major input for management to support the release decision process. They allow a thorough discussion with all stakeholders on product quality and the identification of outstanding risks. As a result, the test plan may be revised to permit the relaxation (or strengthening) of the test completion criteria.

The reporting on product quality has been received well within the project, both by people involved in the quality profile definition and those only receiving the test results. The discussions on product quality are now based on more than "just" the number of defects and the subjective perception of stakeholders.

REFERENCES

Hatton, L. 1995. *Safer C: Developing software for high-integrity and safety-critical systems.* New York: Mc-Graw-Hill.

ISO/IEC 9126-1. 2001. *Software engineering-Software product quality-Part 1: Quality model.* Geneva, Switzerland: International Organization for Standardization.

ISO/IEC DTR 9126-2. 2001. *Software engineering-Software product quality-Part 2: External metrics.* Geneva, Switzerland: International Organization for Standardization.

ISO/IEC DTR 9126-3. 2000. *Software engineering-Software product quality-Part 3: Internal metric.* Geneva, Switzerland: International Organization for Standardization.

ISO/IEC CD 14598-5. 1996. *Information technology-Software product evaluation-Part 5: Process for evaluators.* Geneva, Switzerland: International Organization for Standardization.

McCall, J. A., P. K. Richards, and G. F. Walters. 1977. *Factors in software quality* (RADC-TR-77-363). Rome, New York: Rome Air Development Center, Griffiss Air Force Base.

Myers, G. J. 1979. *The art of software testing.* New York: Wiley-Interscience.

Pfleeger S. L., L. Hatton, and Howel. 2001. *Solid software.* New York: Pearson Education Inc.

Trienekens J. J. M., and E. P. W. M. van Veenendaal. 1997. *Software quality from a business perspective.* Deventer, The Netherlands: Kluwer Bedrijfsinformatie.

Van Veenendaal, E. P. W. M., and J. McMullan, eds. 1997. *Achieving software product quality.* S-Hertogenbosch, The Netherlands: UTN Publishing.

BIOGRAPHIES

Erik van Veenendaal has been working as a practitioner and manager in the IT industry since 1987. As a test manager and quality consultant he has been involved in many projects, has implemented structured testing, and carried out test process improvements activities in a large number of organizations. He is the author of numerous papers and books on software quality and testing, including the best-seller *Testing According to TMap*. He is a regular speaker at both national and international testing conferences and a leading international (ISEB accredited) trainer in the field of software testing. At EuroStar'99 he received the best tutorial award for a tutorial on usability testing. van Veenendaal is the founder and managing director of Improve Quality Services Ltd., a company that provides consultancy and training services in the area of quality management, usability, inspection, and testing. He is also a senior lecturer at the Eindhoven University of Technology, Faculty of Technology Management. He is on the Dutch ISO standards committee for software quality. He can be reached by e-mail at eve@improveqs.nl .

Rob Hendriks has worked in the field of software quality for technical systems, with a specialization in software testing, since 1996. Currently he works as a quality and test consultant for Improve Quality Services BV. Prior to that he worked as a test coordinator and consultant within projects for consumer electronics and professional systems. He has been involved in the definition and deployment of the Software Engineering Institute's Capability Maturity Model (CMM) key process areas Software Quality Assurance within a large organization. On a regular basis Hendriks is a trainer on inspections and software testing, and he is an accredited trainer for the ISEB Foundation Certificate in Software Testing. Hendriks is a certified moderator and a member of the ISEB Software Testing Examiners Panel.

Robert van Vonderen has worked for Océ Technologies B.V. since his graduation from the Eindhoven University of Technology. He conducted a range of applied research studies in the areas of embedded software, geographical information systems, and printer controller software. In recent years he has worked as a project manager and project leader for the Océ Printing Systems division in the area of printer controllers and maintenance. He is currently responsible for the integration, testing, and quality assurance activities of the newest line of controller development.

How helpful did you find this article? Please provide feedback at our online reader survey, http://www.asq.org/mr/sqp5_issue1.html .

STATEMENT OF OWNERSHIP, MANAGEMENT, AND CIRCULATION

1. Title of Publication: *Software Quality Professional*
2. Publication Number: 1522-0540
3. Date of Filing: October 7, 2002
4. Frequency of Issues: Quarterly
5. Number of Issues Published Annually: 4
6. Annual subscription price: $40.00
7. Location of Known Office of Publication: ASQ, 600 N. Plankinton Ave., Milwaukee, WI
8. Location of Headquarters or General Business Offices of Publisher: Same
9. Name and Address of Publisher: William A. Tony, ASQ, 600 N. Plankinton Ave., Milwaukee, WI 53201-3005; Editor: Taz Daughtrey, 736 Sanhill Dr., Lynchburg, VA 24502-4924
10. Owner: ASQ, 600 N. Plankinton Ave., Milwaukee, WI
11. Known Bondholders, Mortgagees, and Other Security Holders Owning or Holding 1% or More of Total Amount of Bonds, Mortgages, or Other Securities: Not Applicable
12. FOR COMPLETION BY NONPROFIT ORGANIZATIONS AUTHORIZED TO MAIL AT SPECIAL RATES. The purpose, function, and nonprofit status of this organization and the exempt status for Federal income tax purposes: has not changed during the preceding 12 months
13. Publication Title: *Software Quality Professional*
14. Issue date for Circulation Data below: September 2002

15. Extent and nature of circulation	Average no. of copies of each issue during preceding 12 months	Actual no. of copies of single issue published nearest to filing date
A. Total No. Copies Printed (Net Press Run)	4306	3220
B. Paid Circulation		
1. Paid/Requested Outside-County Mail Subscriptions Stated on Form 3541	2579	2417
2. Paid In-county Subscriptions	0	0
3. Sales through dealers and carriers, street vendors, counter sales, and other non-USPS paid distribution	0	0
4. Other Classes Mailed Through the USPS	369	329
C. Total Paid Circulation	2948	2746
D. Free Distribution by Mail (Samples, Complimentary, and Other Free)		
1. Outside-County as Stated on Form 3541		
2. In-county as Stated on Form 3541		
3. Other Classes Mailed through the USPS	20	20
E. Free Distribution Outside the Mail (Carriers or other means)	10	0
F. Total Free Distribution (Sum of 15d and 15e)	30	20
G. Total Distribution (Sum of 15c and 15f)	2978	2766
H. Copies Not Distributed	1328	454
I. Total (Sum of 15g and 15h)	4306	3220
J. Percent Paid and/or Requested Circulation (15c divided by 15g times 100)	100	100

16. Publication of Statement of Ownership is printed in the December 2002 issue of this publication.
17. I certify that the statements made by me above are correct and complete.

William A. Tony
Publisher