Published in *Reliability, Quality and Safety of Software-Intensive systems*, by D. Gritzalis (ed.) (1997),
Chapman Hall, ISBN 0-412-80280-5

Testing based on users' quality needs

Drs. E. P.W.M. van Veenendaal CISA and Dr. ir. J. J.M. Trienekens

Abstract

Quality of software products has been a rather intangible concept for both users and software testers over the years. In spite of interesting results in research and practice, most users can not describe the quality characteristics of the software product that they need. As a consequence, software testers do not know what software quality requirements are desirable or realistic, which makes it almost inpossible to test a software product from a user's point of view.

Determination of software quality characteristics starts by listening to "the user" and his or her quality needs. This can be done by characterization of (a part of) the business situation the user wants to support, using a software product. The attributes by which a business situation can be characterized are clustered into three categories: business process characteristics, user characteristics and software product characteristics.

Starting from the various business characteristics the software quality characteristics can be derived in a deterministic way. Software quality characteristics can be expressed in sub-characteristics, which are related to indicators and metrics. These characteristics, initially derived from the users' quality needs, can then form a solid basis for a user-oriented testing process.

Several (European) research projects are gathering empirical data to validate and calibrate the relationships between quality needs, business characteristics, software quality characteristics and metrics. The ultimate objective is to develop a practical basis for third-party testing of software quality. The various concepts will also be implemented as instruments for both developers and software testers to realize, verify and validate software quality.

Keywords

Software quality, testing, user/customer orientation, evaluation model and European projects

Improve Quality Services, Waalreseweg 17, 5554 HA Valkenswaard, The Netherlands
Tel 040 – 2089283 • Fax 040 – 2021450 • E-mail info@improveqs.nl

USERS ASK FOR QUALITY

Many specialists in the software industry, both researchers and practitioners, have depicted the l990s as the quality era (Bassile,1991). The increasingly intensive use of software products in the society as a whole, but especially in business environments makes the IT user accustomed with the possibilities of software products. Users of software products have pushed quality to the forefront of their wishes. They can no longer be satisfied with software products that “only” meet the defined functional specifications, that are delivered in time and at reasonable costs. Users of software products ask for clear, objective and quantifiable software quality. Too often users experience unsatisfactory quality of software products in view of their needs, although they are convincingly - but not in their own language - told that the product meets all kinds of (technical) standards: the product has been tested with the most sophisticated tools, complexity measures are low and no GOTO's are used. However, these qualities do not bother the user directly during the usage of the software products. Only if the technical qualities and characteristics of the product are derived from the users' needs and priorities user's can have confidence in the fact that the product meets their quality standards, ensuring satisfactory and comfortable use. The software industry has been trying to fulfill these needs. Two main approaches of software quality can currently be distinguished. We call them the process approach and the product approach.

Process approach

The process approach can be considered as an indirect approach to achieve quality improvement of software products. It focuses on the definition, the structuring and the evaluation of software processes, the organization of teamwork and the creation of management commitment. Users of software products are hardly involved in the process approach. Examples of process orientation are ISO9000-3 (ISO,1991), TickIT (TickIT,1991), the SEI's Capability Maturity Model (Humprey,1989), and the SPICE project (Software Process Improvement and Capability Determination) (Dorling,1993). In spite of interesting results there are still questions such as:

- how can quality be made explicit and concrete by developers?
- how can users be convinced of the quality characteristics of software products?
- who can decide if software quality promises and expectations are realistic?

Product approach

To answer these questions, a product approach of software quality is needed. Product orientation is often considered to be the opposite of process orientation, but they are in fact complementary. A product approach provides measurements that enable the software industry to improve the processes in an effective and efficient way. A product approach reflects the idea that software quality can be gained on the one hand by identification and specification of quality characteristics of software products and on the other hand by assessments and evaluations of quality characteristics. Examples of quality characteristics of software products are (in accordance with ISO 9126) reliability, usability, efficiency and maintainability. Examples of research projects in this area are on the Dutch national level the QUINT project

(SERC,1992), on the international level the development of the ISO 9126 standard (ISO/IEC,1991), and ESPRIT projects such as METKIT, SCOPE and SQUID. In these projects quality characteristics are elaborated as external and internal product attributes, measures and metrics are defined etc. Although a certain level of objectivity and quantification is reached, neither the QUINT project nor the ESPRIT projects addresses in a structured way the identification and determination of the needs of the user or the requirements of the business situation in which a software product is or will be used. Therefore, assessments and evaluations of software product quality are hardly possible from a user's point of view. New ESPRIT projects like SPACE-UFO (user focus) and MultiSpace, which adressen quality within multimedia systems, both try to bridge the gap between the technical and the user's point of view regarding software product quality.

In this paper we will focus on testing, being the most important evaluation concept. A new approach is presented to test software quality starting from explicit specifications of users' quality needs and business requirements.

SOFTWARE TESTING

State-of-the-practice

In the software industry an unambiguous, universally used definition of the testing concept does not yet exist. The many types of testing for a large variety of purposes, the vague descriptions of test domains and the sometimes informal application of the test procedures make the definition of testing difficult. Testing is in practice often interpreted simply as measurement, comparison, trying out etc. But this pragmatical way of thinking is not sufficient in current professionel business environments. ISO 8402 (ISO,1994) defines the testing process as: “activities such as measuring, examining, investigating, sampling, inspecting or gauging one or more characteristics of a product or service and comparing the results with specified requirements in order to establish whether conformity is achieved for each characteristic.” Testing is all about detecting faults: testing will indicate the lack of quality. A manageable definition of testing, which will be used throughout this paper is: “testing is a process of planning, preparation and measuring, which has the purpose of determining the characteristics of an information system and demonstrating the differences between actual and required quality” (Pol,1995).

Looking at the state-of-the-practice of software testing today, it can be noted that a lot of improvements have taken place during the last few years. Many organizations are using the V-model often accompanied by a detailed phasing model for black-box testing, a separate physical testing environment has become common practice and testing techniques are available to support different kind of coverages. Also, CAST (Computer Aided Software Testing) has improved over the years providing the testers with many useful features. Most important is perhaps the fact that organizations are now willing to pay for testing activities and that testing is finally being considered to be a profession. The statement on the improvement of the testing process is supported by an investigation on testing carried out in the Netherlands by NGGO.

Table 1 The testing process (source: NGGO investigation report [NGGO,1991])

Average percentage on test effort related to total project effort	20%
Uses specific guidelines or standards for the testing process	49%
The testing activities are separately planned in each phase	75%
Test data is a part of milestone products	52%
Recognizes the effectiveness of test preparation	91%
Executes tests	96%
Recognizes the effectiveness of re-testing	84%

It must be stated however, that most of the improvements that have been made focus on the testing process (“doing the things right”), rather than on the effectiveness of the process (“doing the right things”). Also, most of the improvements that have been made are rather technically oriented and do not put much emphasis on user-orientation. The key questions that have to asked are:

- does the improved testing process also provide the users with better information on the extent to which the software product satisfies their needs?
- are we testing the right things from a user's point of view, i.e. are we testing “fitness for use”?

The key activity in the testing process when adressing the subject of “doing the right things” is defining the testing strategy. The testing strategy describes what is going to be tested and how thoroughly it is going to be tested.

Testing strategy

One of the most important ativities in the testing process is defining the testing strategy. The definition of the testing strategy takes place during the planning phase of the testing process. During the planning phase the basis is laid for a manageable and high-quality testing process. Defining the testing strategy is an instrument for test management to communicate with the principal, users, developers and other parties involved about the organization and strategic choices of the testing process. The testing strategy defines what is going to be tested and how it is going to be tested. Choices have to be made because it is impossible to test the system completely; 100 % coverage on all quality characteristics is perhaps possible in theory, but no organization has the time and money for it. The testing strategy is determined via a process of communication by all parties involved, trying to define the most important parts and quality characteristics of the software product. The aim is to have the most feasible coverage on the right part and quality characteristics of the software product.

Within the process of defining a testing strategy a number of steps regarding quality characteristics can be distinguised:

- **Selection of quality characteristics**: in consultation with all parties involved a number of quality characteristics are selected that are relevant to the test of the underlying software product. Ideally, the relevant quality characteristics are already defined in a quality requirements specification. To support the selection process a risk analysis can be performed to be able to identify the most important testing areas more thoroughly.
  The selection process can be qualified as informal and is rather technically oriented because of the nature of most quality characteristics. Users do not really know what is meant by characteristics like portability, analysability or recoverability.
- Determination of the **relative importance of quality characteristics**: the relative importance of each of the selected quality characteristics has to be defined. The relative importance is expressed as a percentage. Determining the relative importance is not about getting precise percentages, which is meaningless at this stage. The objective of this step is to achieve in consultation with the principal and other parties involved a general view of the importance of the selected characteristics. By having to give percentages to each of the characteristics, a group of users and test specialist will establish consensus.
- Determination of **acceptance criteria** and **testing techniques**: the relevant characteristics are specified by linking indicators to them along with economic, reliable and objective measurement protocols, i.e. testing techniques. Also, the acceptance criteria, being the values these indicators should have on delivery of the product, should be defined. Within the process of software testing it is then relatively easy to evaluate the delivered product according to agreed measurement protocols and acceptance criteria.

The results of the first two steps are often reflected in a so-called strategy matrix. The two-dimensional strategy matrix shows which quality characteristics are relevant and their relative importance. In the example given below, the information system consists of three subsystems (S1, S2 and S3). The relationships between the quality characterics and the three subsystems are also visualized in the matrix. “x” means that the quality characteristic is applicable to the subsystem, and “xx” means the combination quality characteristic/subsystem needs special attention, i.e. more thorough testing. Again we emphasize that there is no sense in trying to derive mathematical precision from this matrix.

	S1	S2	S3	Relative importance
Functionality	xx	x	xx	40%
Reliability	xx	x	xx	20%
Usability	x			10%
Efficiency	x			10%
Maintainabilit	x	x	x	20%
Portability				-
				100%

Figure 1 An example of a strategy matrix

Although the above described process of defining a testing strategy is a structured one that has been succesfully applied in practice in a great number a projects, it still has some important shortcomings. The process is very informal and therefore not repeatable and reproducible which is a prerequisite for third-party testing in a certification context. The quality of the results depends very strongly on the quality of the people performing the definition of the testing strategy. The major shortcomings, however, is based on the fact that the communication is solely about quality characteristics and not about the users' quality needs. Therefore, the process of defining a testing strategy is not very user-oriented or business-oriented; nevertheless we are trying to test “fitness for use”. If on top of this no standards on quality characteristics are being used, which is often the case, the users become even more confused.

Within the product approach of software quality a number of concepts have recently been developed that may contribute towards more user-oriented and business-oriented software testing. Subsequently, the concepts of software quality characteristics, quality profile and quality levels, an approach for the identification and specification of quality characteristics based on a business situation, and an evaluation model are described.

CONCEPTS FOR SOFTWARE PRODUCT QUALITY

Software quality characteristics

In 1977 McCall et al. (McCall,1977) proposed the idea of breaking down the concept of quality into a number of quality factors. This idea has been followed by many other authors who have tried to capture software product quality in a collection of characteristics and their depending sub-characteristics 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 on reappearing, although their place in the hierarchy can differ.

Some writers argue that there is no justification for these implied models because there are no universal concepts and corresponding base units to build a model of software quality upon (Kaposi,1994). But despite this critique on the scientific appropriateness of quality characteristics, the practical applicability is evident; in the growing number of publications on software product quality there appears to be consensus on the quality characteristics to be used.

The International Organization for Standardization (ISO) and the International Electro-technical Commission (IEC) have defined a set of quality characteristics. This set reflects a first step towards consensus in the software industry and thereby addresses the general notion of software quality. The ISO 9126 standard defines six quality characteristics and proposes, in an appendix to the standard, the subdivision of each quality characteristic in a number of sub-characteristics.

The characteristics and their proposed sub-characteristics are, respectively:

- functionality, which consists of five sub-characteristics: suitability, accuracy, security interoperability and compliance;
- reliability, which can be defined further into the sub-characteristics maturity, fault tolerance and recoverability;
- usability, which can be divided into the sub-characteristics understandability, learnability and operability;
- efficiency, which can be divided into time behaviour and resource behaviour;
- maintainability, which consists of four sub-characteristics: analysability, changeability, stability and testability;
- portability, which also consists of four sub-characteristics: adaptability, installability, conformance and replaceability.

In working group six (WG6) of ISO/JTC1/SC7 ongoing research takes place to determine indicators and metrics for each of the (sub-)characteristics.

Quality profiles and quality levels

The confidence that a user, i.e. purchaser, end-user, operator etc., may place in a product grows with the thoroughness of the test of the software product. Increasing effort in testing has as a consequence increasing time and costs. However, as stated before when describing testing strategy, not all (sub-) characteristics need to be tested in every software product with the same amount of thoroughness. For example, the usability of a word processor needs to be tested more thoroughly than its reliability, but of course this does not mean that the reliability of a word processor is of no interest at all. The reliability of the software in a nuclear power plant is of more importance than it's usability, which of course does not inevitably mean that usability is of no interest. These rather intuitive assumptions were studied in several international research projects and in working groups of international standards organisations. Guidelines (ISO/IEC,1994) have been developed for the determination of the amount of thoroughness of testing that is needed for a certain software product in a certain business environment. Furthermore, levels of testing have been developed with an increasing amount of thoroughness (from level D to level A). The levels are established by looking for instance at the economical risks of a failure or the safety and security aspects of the software product. These concepts and guidelines are rough directions for testers to determine a software quality profile. In a quality profile the relevant quality (sub-)characteristics and their accessory levels of testing are established based on certain business-related characteristics. A quality profile (see figure 2) reflects the notion of quality for a certain software product and makes quality more tangible for both developers and users.

	Evaluation level				
	-	D	C	B	A
Functionality				X	
Reliability			X		
Usability		X			
Efficiency		X			
Maintainabilit			X		
Portability	X				

Figure 2 An example of a quality profile

A question that has hardly been addressed until now is how the needs for software quality of users can be translated into a "preferred" set of well specified software quality characteristics, i.e. a quality profile. Of course it will be clear that users should try to specify, or should be supported in specifying, their needs and wishes in terms of software quality characteristics. And it will also be clear that developers should try to characterize the quality of their products in terms that are clear to users. However, until now no systematic approach, methods and tools for both developers and software testers to determine the (required) quality characteristics of a software product has been available. The next section focuses on the development of "determination concepts" for software quality profiles from business characteristics.

Deriving quality profiles from business characteristics

In research projects of the Eindhoven University of Technology (TUE) an approach was developed for the identification and specification of software quality characteristics (Trienekens,1994). This approach, which has been adopted by KEMA and is currently being elaborated in the ESPRIT projects SPACE-UFO and MultiSpace, relates the wishes and needs of users in their business situations to software quality (sub-)characteristics. The users' needs and business requirements are explored and specified by using questionnaires. Answers from users to questions (regarding their needs for quality) are related to required software quality characteristics by means of two-dimensional matrices. The approach is based on the so-called Process-Control-Information (PCI) paradigm (Bemelmans,1986). The PCI paradigm states that the (quality) requirements of a software product can be determined by an analysis of the characteristics of a business situation. Based on an analysis of the essentials of a business situation, the relevant software quality characteristics can be identified and specified. In accordance with literature (Davies,1984), a distinction between three types of business situation characteristics has been made: the business process characteristics, the user characte-ristics and the software product characteristics.

Business process characteristics

The previously mentioned PCI paradigm has been used to investigate the characteristics that influence the control of the business process or business system. The study of system science has given us several useful concepts which have been translated into business process characteristics that have subsequently been associated with relevant software quality characteristics. The business process characteristics are:

- **Number** of process variables and their **interdependence**.
- The **controllability** and **predictability** of process variables.
- The **sensitiveness** and **stability** of the controlled process.
- The **reaction pattern** of the controlled process.

Two examples are given of relations between these business process characteristics and software quality characteristics:

- when the number of process variables is high and there is a lot of interdependence the process is difficult to control, the need for usability, i.e. understandability and operatability, is high.
- if the stability of the process is low, there is a need to be able to change the supporting software product accordingly. In terms of quality characteristics this means that maintainability, i.e. changeability, and perhaps portability are important.

User characteristics

A user is directly or indirectly involved in the purchase, the use or the management of the software product. Therefore, human aspects play an important role in the identification and specification of software quality characteristics. The following types of users can be distinguished:

- **input-oriented** end-users;
- users that use the information from the system (**output users**);
- management (**remote users**) that gets it's information from intermediates.

Those user types can be refined by looking at three dimensions: the experience the users have with similar software products or with software products in general, the educational background of the user and the number of users.

We will give two examples:

- a software product intended for many inexperienced users with a low level of education will need to be highly understandable and learnable (both are sub-characteristics of the quality characteristic usability);
- if many input-oriented end-users are involved they ask for a software product that is easy-to-use, e.g. has a high level of operability, and good performance, e.g. has a good level of time-behaviour.

Software product characteristics

Software products can be characterized by three dimensions, the type of software product, it's profile of use and the required infrastructure, respectively (Trienekens,1994).

- The **type** of software product: how is the product characterized, as a transaction processing system, a decision support system, a knowlegde based system, a word processing system or something else? Will the system be a standard application or a one-of-a-kind custom-made software product.
- The **profile-of-use**: how many hours a day is the system in operation, can it be characterized as an on-line or a batch system and do batch-jobs also run during day-time?
- The **infrastructure**: does the software product aim at running on diferent kinds of hardware and in a networking environment?

Again, we will give two examples:

- a transaction processing system with a lot of large batch-jobs recoverability (a sub-characteristic of the characteristic reliability) is likely to be important.
- if the software product is a standard software package, the portability of the software product will be considered to be very important.

Most of the above-mentioned relationships are still mainly intuitive and hypothetical. Although some pilot projects have taken place with encouraging results, they have to be further elaborated and validated in practice.

The evaluation model

This section proposes an evaluation model for the determination of software quality profiles. This model will be used to make the process of determining the quality of a software product less expert-based. The main objective is to create a repeatable and reproducible process in which the user's quality needs or business requirements are translated into a set of prioritized quality characteristics and their accessory evaluation levels. The evaluation model is visualized in Figure 3.

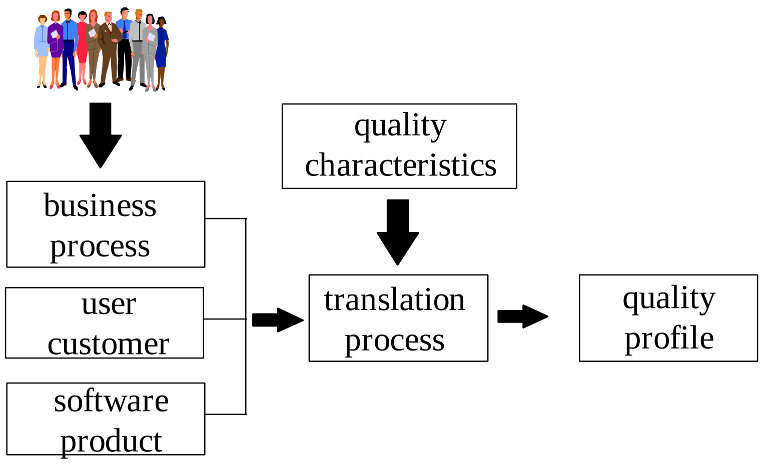

Figure 3 The evaluation model

The translation process is a step-wise process in which the needs of the users are translated into a set of well-defined software quality characteristics. Communication with users plays a central role. In the first step a structured questionnaire is used to identify the quality requirements of the users regarding a software product. In the second step the answers of users to the questions will be related in a (semi-) deterministic way to software quality characteristics by means of two-dimensional matrices. In the third and final step, the meaning of the quality characteristics from the business context point of view and their relative importance are determined in co-operation with the users. During this step the quality profile is enriched with a lot of valuable information on the business context.

A QUALITY PROFILE AS A BASIS FOR USER BASED TESTING

Within the evaluation model the quality profile has become a user-oriented quality profile. If we compare a quality profile to a strategy matrix we can see a lot of similarities. Both show which quality characteristics are relevant in the specific situation and in both the importance of the selected quality characteristics is expressed. In the quality profile this is done by using the evaluation levels and in the strategy matrix this is done by means of the concept of “relative importance”. In the process of determination they both put emphasis on thorough communication with users. The main difference between the two lies in the fact that the quality profile, when used within the evaluation model, is really user-oriented. Also the quality profile does not explicitly make use of a risk analysis, although a thorough context analysis based on business characteristics takes place. This means that if software testing uses the quality profile it has to make some (minor) changes based on a cost/benefit analysis, a risk analysis and perhaps due to time constraints, to determine the testing strategy.

Summarizing, we conclude that a quality profile is a perfect and solid basis for a user-oriented testing strategy. Subsequently, this leads to a more user-oriented testing process and the ability to test “fitness for use”.

Quality profile not only for testing

It must be stated that the use of the evaluation model and the quality profile is not only applicable to software testing, but should also be integrated into the software development process. The quality profile along with the information on the business characteristics can be used to produce well- founded quality requirement specifications. The process of determining a quality profile through the use of the evaluation model should ideally be performed jointly by developers, software testers and users.

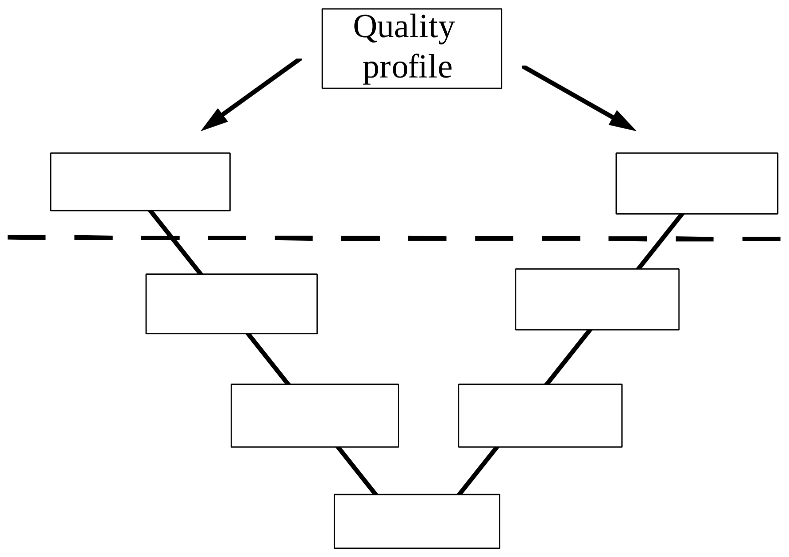

Figure 4 Quality profile and the V-model

CONCLUSIONS AND FURTHER RESEARCH

Pilot projects have shown that the evaluation model is a useful guideline in determining a quality profile. Based on the characteristics of the business process, the information system and the user, most of the quality characteristics can be determined. Subsequently, a quality profile can form a solid basis for a testing strategy, which is used to evaluate the quality of the software product. In this way the users' quality needs are explicitly incorporated into the testing process, resulting in a more user-oriented testing or evaluation process.

Further research at KEMA and in the ESPRIT projects SPACE-UFO and MultiSpace will be directed towards a more thorough elaboration of the evaluation model, the evaluation levels and further concretization of the evaluation process of a software product. The research directions are briefly explained:

Elaboration of the evaluation model

For each of the three clusters of characteristics: business process, user and software product the questionaires will have to be further elaborated and validated in practice. Based on the answers given by the users to the questions one should be able to establish in a deterministic way a prioritized set of quality characteristics for a specific business situation.

Standards on evaluation levels

Working with evaluation levels requires for standards and related metrics. Without these standards the detemination of an evaluation level is limited to subjective judgement, which of course is not desirable when evaluating in a certification context. Further research will be directed towards the development of objective standards for evaluation levels.

Concretization of the evaluation of software quality

There still is limited understanding of the possibilities of evaluation of software quality. Especially in the relationships between types of software products, quality characteristics, indicators and metrics a lot of research has yet to be done. The research aims at developing a typology of software products and with each software product type a list of relevant quality characteristics accompanied by indicators, metrics and measurement protocols.

In the short term the above-mentioned research activities along with pilot projects will take place, aiming at a thorough and practical approach to user-oriented software product evaluation. The ultimate objective for KEMA is to have an objective and efficient evaluation process for software product quality in a certification context.

REFERENCES

Basili V.R., J.D. Musa (1991), The future engineering of software: a management perspective, *IEEE Computer*, Vol. 24-9.

Bemelmans T.M.A. (1986), Business oriented design of information systems, in: P.A. Cornelis, J.M. van Oorschot (eds.), *Automation with a human face* (in Dutch), Kluwer.

Davies G.B., M.H. Olsen (1984), *Management Information Systems*, McGraw-Hill, London.

Dorling A. (1993), *SPICE: Software Process Improvement and Capability dEtermination,Software Quality Journal*, 1993-2.

Humphrey W.S. (1989), *Managing the Software Process*, Addison-Wesley.

ISO 8402 (1994), *Quality management and quality assurance - Vocabulary*, International Organization of Standardization.

ISO 9000-3 (1991), *Quality management and quality assurance standards, Part 3: Guidelines for the application of ISO 9001 to the development, supply and maintenance of software,* International Organization of Standardization.

ISO/IEC 9126 (1991), *Information Technology - Software product evaluation - Quality characteristics and guidelines for their use*, International Organization of Standardization.

ISO/IEC/JTC1/SC7 (1994), CD 14598-5, *Information Technology - Software product evaluation - Part 5: Evaluators guide, International Organization of Standardization*.

Kaposi A. and M. Myers (1994), *Systems, models and measures*, Springer-Verlag, London.

McCall, J.A., P.K. Richards and G.F. Walters (1977), *Factors in Software Quality*, RADC-TR-77-363 Rome Air Development Center, Griffis Air Force, Rome, New York.

NGGO (1991), *About the testing of Information Systems* (in Dutch), Tutein Nolthenius.

Pol, M, R.A.P. Teunissen and E.P.W.M. van Veenendaal (1995), *Testing according to TMap* (in Dutch), Tutein Nolthenius, ‘s Hertogenbosch.

SERC-Quint (1992), *The specification of software quality, a practical guide* (in Dutch), Kluwer Bedrijfswetenschappen.

TickIT (1991), *Guide to Software Quality Management Systems, Construction and Certification using EN 29001*, TickIT Office, London.

Trienekens J.J.M. (1994), *Time for Quality, working towards better information systems* (in Dutch), Thesis Publishers, Amsterdam.

BIBLIOGRAPHY

Drs. E.P.W.M. van Veenendaal CISA has been working in the IT-industry since 1987 carrying out assignments in the field of quality assurance, project control, EDP-auditing and software testing. He has been responsible for and involved in a number of European ESPRIT and ESSI projects within the area of Software Product Quality. Erik van Veenendaal is the founder and managing director of Improve Quality Services. Improve Quality Services provides services in the area of quality management, usability and testing. He is also liased to the Eindhoven University of Technology carrying out Ph.D. research on software testing and a joint author of “Testing according to TMap”.

Dr. ir. J.J.M. Trienekens is an associate professor at the Eindhoven University of Technology. He is responsible for a number of practice-oriented IT-research projects at the Frits Philips Institute for Quality Management (FPIQM) and the Eindhoven University of Technology, faculty Technology Management. He is also the chairman of the Dutch ENCRESS club and as a consultant to KEMA involved in the European ESPRIT project SPACE-UFO.