Determination of quality characteristics of software products : concepts and case study experiences

DETERMINATION OF QUALITY CHARACTERISTICS OF SOFTWARE PRODUCTS; CONCEPTS AND CASE STUDY EXPERIENCES

Peter J. Eisinga,
IT Consultant
KEMA Arnhem
The Netherlands

Jos J.M. Trienekens,
Manager of IT projects, Senior Researcher
Frits Philips Institute for Quality Management
Eindhoven University of Technology
The Netherlands

Mark van der Zwan,
Graduate student
Industrial Engineering and Management Science
Eindhoven University of Technology
The Netherlands

Arnhem/Eindhoven, April 1995

Abstract

Quality of software products has been a rather intangible concept for both customers and developers over the years. In spite of interesting results in research and practice, software developers are hardly able to specify the quality characteristics of their software products. As a consequence, customers do not know what realistic requirements and expectations are with respect to software quality.

This paper aims at the determination of software quality characteristics from the point of view of both customers and developers. After an introduction of concepts of software product quality this paper will focus on a current research project of KEMA, a Dutch certification body, and the Eindhoven University of Technology in the Netherlands. The ultimate goal of this project is to arrive at a methodology to determine, i.e. to assess and to evaluate, the quality characteristics of software products. A well-defined basis has to provide independent evaluators with methods and tools to objectively assess what developers (can) promise and what customers can expect with respect to software quality.

Introduction

Many specialists in the software industry, both researchers and practitioners, have depicted the 1990s as the quality era [2]. Customers 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. Customers of software products ask for clear, objective and quantifiable software quality.

The software industry has been trying to fill these needs. Three approaches of software quality can currently be distinguished. We call them the process approach, the resource approach, and the product approach, respectively.

Examples of process orientation are ISO9000-3 [12], the SEI's Capability Maturity Model [10], and the SPICE project (Software Process Improvement and Capability Determination) [8]. Total Quality Management is more or less an umbrella concept for all kinds of quality improvement activities [19]. TQM aims in particular at raising awareness, management commitment and teamwork. It is often called an integrated approach. Because of its strong orientation on the human factor, we would like to consider it as a (human) resource approach.

Both the process and the integrated approach can be considered as indirect approaches to achieve quality improvement of software products. They focus on the definition, the structuring and the evaluation of software processes, respectively, and on increasing the quality awareness of software engineers, the organization of team-work, and the creation of management commitment. Customers of software products are hardly involved in either approach. In spite of interesting results in these approaches there are still questions such as:
- how can quality be made explicit and concrete by developers?

- how can customers be convinced of the quality characteristics of software products (why should they trust suppliers)?
- who can decide if software quality promises and expectations are realistic?

To answer these questions, a product approach of software quality is needed. 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 assessment and evaluation of quality characteristics. Examples of quality characteristics of software products are (in accordance with ISO 9126) reliability, usability, efficiency, maintainability. Examples of research projects in this area are on the Dutch national level the QUINT project[20] and on the international European level the development of ISO 9126-standards[12] and the current ESPRIT project SQUID (P8436). 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 SQUID project addresses in a structured way the identification and determination of the needs of the customer or the requirements of the business situations in which a software product is or will be used.

For several years the determination of software quality characteristics has been an important research theme at the Eindhoven University of Technology (TUE) in the Netherlands. Concepts for determination were developed and validated in practice, see e.g.[9, 23]. Some of the main insights resulting from this research are:
- quality needs of customers are related to business system characteristics;
- software quality characteristics can be identified by analyzing business situations and their requirements;
- software quality characteristics have to be assessed and evaluated by independent software product evaluators.

These insights are currently used in a research project of KEMA, a Dutch certification body. This research project is called the Software Product Quality (SPQ) project. The key objective is a methodology for objective and independent assessment and evaluation of software products. Before 1998 KEMA wants to provide a new service to the IT-market: a fully developed assessment laboratory that applies scientifically founded methods and tools and that is able to evaluate and certificate software products in accordance with international standards (such as ISO 9126). Through evaluation of software products quality has to become visible and tangible. The ultimate goal of KEMA is to improve the quality of software products distributed to the market and thus the improvement of an aspect of the quality of our working life and life in general.

This paper introduces on the one hand software quality concepts and a model for software quality assessment and evaluation and on the other hand reports the experiences that were recently gained in two realistic case studies of KEMA.
The structure of the paper is as follows. Section 1 introduces the main concepts of software product quality determination. Section 2 presents and discusses the application of the concepts in two case studies. Section 3 discusses the progress that has been made and points out further work to be done.

1 Concepts for software product quality

In order to make the case studies and their results easier to read, the most important concepts and terms will be introduced and explained in this section. We will make a distinction between concepts of software product quality and concepts for determining software product quality.

1.1 Software product quality concepts

This section respectively presents the concept of the quality characteristic, the concept of the software quality profile and the concept of the software quality level.

1.1.1 Software quality characteristics

In 1977 McCall et al.[17] 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 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[16, 18]. 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 Electrotechnical Commission (IEC) have defined a set of quality characteristics. This set reflects a first step towards consensus in the IT industry and thereby addresses the global 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, interoperability, compliance and security;
- 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 metrics for each of the (sub-)characteristics [14].

However, a question that has hardly been addressed until now is how the needs for software quality of customers can be translated into a 'preferred' set of well specified software quality characteristics. Of course it will be clear that customers 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 customers. However, until now no systematic approach has been available that supports or guides these activities. The value of a well structured and scientifically based evaluation of software products now becomes clear: to give a customer confidence in the quality of a software product and to confirm the efforts of a developer in implementing quality in a software product.

1.1.2 Quality profiles and quality levels

The confidence that a customer, i.e. purchaser, user, operator etc., may place in a product grows with the thoroughness of the evaluation of the software product. Increasing effort in evaluation has as a consequence increasing time and costs. However, it will be clear that not all (sub-)characteristics need to be evaluated in every software product with the same amount of thoroughness. For example the usability of a word processor needs to be evaluated 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. And the reliability of the software in a nuclear power plant is of more importance than its usability, and of course this does not mean that the usability is of no interest. These rather intuitive assumptions were studied in several international research projects, such as the ESPRIT project SCOPE (Software CertificatiOn Programme in Europe) [1]. SCOPE has developed guidelines for the determination of the amount of thoroughness of evaluation that is needed for a certain software product in a certain business environment. Furthermore, SCOPE has introduced four *levels of evaluation*, 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 of the product or the type of interface used in the software product.

The SCOPE guidelines are rough directions for evaluators to determine a *software quality profile*. In a quality profile the relevant quality (sub-)characteristics and their accessory levels of evaluation are established. A quality profile reflects the notion of quality for a certain software product and makes quality tangible for both developers and customers.

The paragraphs above point out that there are some concepts for software product quality. But until now there have been no systematic approaches, methods and tools for software evaluators to determine the (required) quality characteristics of a software product. The next section focuses on the development of 'determination concepts' for software product quality.

1.2 Concepts for determining software product quality

In this section we make a distinction between an upper and a lower phase for the determination of software product quality. The first phase is the phase of the identification and specification of software quality characteristics that are required by a customer. In the lower phase the developer takes the software quality specifications as a starting point to realize quality, or to 'build quality' into a software product. Because this lower phase is the most elaborate one, both in theory and in practice, we will briefly describe it first.

1.2.1 Realization of software quality

The realization process of software quality has often been described as a software product quality loop, see e.g. [7]. In this loop software quality specifications are linked to engineering measures. The execution of these measures leads to quality attributes of a software product that can be compared with the original quality specifications. The comparison closes the loop.

The software quality loop focuses on the implementation of software quality specifications, i.e. the specified (sub-)characteristics. The loop does not address the identification of software quality characteristics from the needs and the wishes of a business environment. In contrast to the strong emphasis on the specification of functional requirements in the upper phase of the life-cycle of software products, hardly any attention is given, both in literature and in practice, to the specification of software quality characteristics.

1.2.2 Identification and specification of software quality characteristics

Recently in research projects of the Eindhoven University of Technology (TUE) an approach was developed for the identification and specification of software quality characteristics [21]. The proposed TUE approach relates wishes and needs of business situations to software quality (sub-)characteristics. The approach is based on the so-called Process-Control-Information (PCI) paradigm [3]. The PCI paradigm states that the requirements of an information system or software product (i.e. the need for information) can be determined by an analysis of the business processes (i.e. the system to be controlled) and the business control mechanisms (i.e. the controlling system).

The TUE approach proposes that software developer and customer should communicate about requirements and needs with respect to software quality. Communication should be focused on the characteristics of a business situation and subsequently on the type of software product or information system that is needed. Whereas the software quality loop only focused on realization of quality characteristics, and the European SCOPE project mainly took a 'quick' look at business situation aspects, the TUE approach puts analysis of the business situation in a central place. Based on an analysis of the essentials of a business situation, the relevant software quality characteristics can be identified and specified.

1.3 From business situation characteristics to software quality profiles

In accordance with literature, e.g.[6], we use a distinction between three types of business situation characteristics: the business system characteristics, the software product characteristics, and the customer characteristics.

1.3.1 The business system characteristics

The previously mentioned PCI paradigm has been used to investigate the characteristics of business systems. The study of system science has given us several useful concepts which have been translated into business system characteristics and these characteristics are associated with relevant software quality characteristics as described in two examples. The business system characteristics are:

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

A first example of a relationship between these characteristics and software quality characteristics; if the sensitiveness of a controlled system grows and the stability of that system decreases, the need for efficiency of software products, i.e. time behaviour, increases. Another example: the more precarious the reaction pattern of the controlled system is, the more need there is for operability and understandability of a software product, and the more important the quality characteristic usability of the software product becomes.

It must be clear that the above-mentioned relationships are mainly intuitive and hypothetical. They have to be elaborated and validated in practice. This is also the case in the following relationships between the three dimensions of a software product and relevant software quality characteristics.

1.3.2 The software product characteristics

Software products can be characterized for three dimensions, the dimension of the scope of a product, the dimension of the life-cycle phases a developer is responsible for, and the dimension of the uniqueness of a product, respectively[21].

- The **scope** of a product: is it just the source code or are user's guides or even the organizational fit also under consideration?
- The **phases** of the life-cycle: will the developer be held responsible for his or her work before delivery or throughout the lifetime of a product?
- The **uniqueness**: is a product a standard application or a one-of-a-kind custom-made software product.

An example of the relationships between software product dimensions and quality characteristics: a very specific custom-made product will concentrate on the usability whereas a standard product will be focused especially on software quality characteristics such as maintainability and portability. Another example; if the developer is only responsible for the life-cycle phase before delivery, then he or she will concentrate on the functionality and reliability of the software product. If the developer is responsible for the entire life-cycle, then the developer will put great effort into the maintainability and portability of the software product, as well as in the quality characteristics suitability (sub-characteristic of functionality), and usability.

A third type of business situation characteristics is needed for identification and determination of the relevant software quality characteristics. This is the human aspect of the business situation: the customer of the software product.

1.3.3 The customer characteristics

A customer is directly or indirectly involved in the purchase, the use or the management of the software product. The following classes of customers can be distinguished:

- **Purchasers**: the persons or organizations that purchase a system, product, or service from a supplier.
- **Users**: four types of users can be distinguished: a) those who mainly feed the software product with data (input-orientated), b) those who mainly use the provided information (output-orientated), c) the management of the users and d) other users (this type depends on the software product considered).
- **Operators**: mainly two types of operators can be distinguished: a) those who are responsible for the operation of the software product and b) those who are responsible for the adaptation of the software product to changing wishes of the users (apdatation that can be conducted by the operator without applying others means than provided by the software product).

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

An example of the customer characteristics and their relationships with software quality characteristics: a software product intended for many unexperienced, input-orientated users with a low level of education will need to be highly understandable and learnable (both are sub-characteristics of the quality characteristic usability). A product intended for a few experienced operators with a high level of education, will need to be maintainable and portable.

1.3.4 The theoretical model

In this section a theoretical model has been proposed for the determination of quality characteristics of software products. With this model we are trying to make the process of determining the quality of a software product less expert-based. The main objective is to create a repeatable process in which the stated needs of the customer are translated into the various quality characteristics and their accessory evaluation levels. The model is visualized in figure 1. The three relevant parties are involved;

the product itself, people who will use the product and the business system where the product has to fit in.

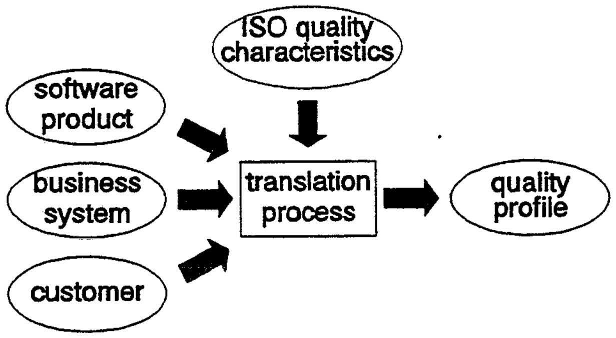

Figure 1 The proposed model.

In the next section the model is applied to two case studies to gain insight into the practical applicability of the model and to be able to further develop this first idea.

2 Practical applications

In this section the theoretical model that was developed in the preceding section will be applied in two pilot projects (or case studies). Those studies were started in order to obtain more field experience in the determination of quality characteristics. The names of the products described and the clients of KEMA involved are fictitious.

2.1 Pilot project 1: The DCD case

PROD is the producer of the product X. PROD sells X to big transporters who sell it to several smaller distributors. The price which is paid for the product X is directly related to the demand for X. The software product in this pilot project is called a 'data-collector device' (DCD) and provides the information needed to calculate the signal of the demand for X by the market. The payments throughout the chain of the producer, transporter, distributor and consumer, are based upon this signal. PROD passes the signal on to this chain so that the market has the means to regulate the demand for X. PROD is responsible for the correctness of the signal of the demand for X.

2.1.1 The software product

The software product DCD is part of the system that calculates the signal of the demand for X (the DFX system) and can be characterized as follows.

Scope of the product: The product is part of the DFX system and has no direct contact with the business system. The only thing considered is the software product itself (source code, test plans, test coverage etc.).
Phases of the life-cycle: Adaptive maintenance and training sessions are not relevant in this case. The supplier of the DCD product will only be held accountable for the life-cycle phases before delivery.

Uniqueness: The software product can be called specific because it is tailor-made for PROD.

2.1.2 The business system

A business system consists of a system to be controlled and a controlling system. In this case study the controlled system is designed to measure the demand for X and provide the signal based on these measurements. The controlling system is supposed to take care of these functions of the controlled system. The DFX system can be seen as the controlling system and the considered software product DCD is a part of the DFX system.

As mentioned earlier the information provided by the DFX system is also used to indirectly regulate the demand for X. This control of the demand for X falls outside the boundaries of the business system. However, the function of the information provided by the DFX system stresses the importance of the data collected and processed by the DCD software product.

Number and **interdependence** of system variables: There is just one kind of variable which has to be processed (the input) and there is just one kind of variable that forms the output of the DCD product for the DFX system.
Controllability and **predictability** of system variables: The variables that influence the business situation are easy to control and predict within the system boundaries because there is only one kind of input and output variables.
Sensitiveness and **stability** of the controlled system: The DFX system, and thus the DCD product, has to be sensitive and although the business situation is very stable if the signal is available, the environment of the business system will become uncontrollable and unstable if the software product fails.
Reaction pattern of the controlled system: The reaction pattern of the market may be precarious but this lies outside the boundaries of the business situation. The reaction pattern of the controlled system itself is very straightforward.

2.1.3 The Customer

There are no direct users, only people who are indirectly affected by the output of the software product DCD and the operators of the DFX system. These operators are experienced technicians who know what the product does, how it works and what the importance of the output is.

The indirect customers of the product depend on the signal which the software product helps to compose. Their interests are already covered by the characterization of the business system.

2.1.4 Relevant quality characteristics

From the characterization of the product, business situation and customer (completed with the expertise of a quality consultant or information analyst), it could be concluded that the following ISO 9126 quality characteristics are the most relevant.

Functionality (accuracy, security)
Because of the specificity of the DCD product and because of the importance of the actions that are based upon the provided information.

Reliability
Because of the uncontrollable and unstable situation that will occur if the product fails.

Efficiency
Because of the need for control by the environment of the business situation, the DFX system is a real-time system.

2.1.5 Determining the quality profile

The relevant quality characteristics are now known, so the next step is to determine the evaluation levels to complete the quality profile for the software product considered. This is done with the criteria proposed by SCOPE. The description of customer, product and business system is thereby complemented and the following levels are set. Per level the most important complementary reasons are mentioned.

Functionality: level C
Because of the economic loss which will result from failing of the product (because of the importance of the information provided) and the security level needed (defined by ITSEC [11]).

Reliability: level A
Because of the desired availability (24 hours a day) due to the importance of the signal.

Efficiency: level A
Because the product is a real-time system.

2.1.6 Some remarks

In this first case study we have applied a model to guide the determination of a quality profile. The product, the business system and the customer are characterized in order to determine the relevant ISO quality characteristics. Then the evaluation levels are established by applying the SCOPE guidelines.

The model had to be supported by the expertise of a software product quality consultant. The expert had to set the business system boundaries and translate the intuitive relationships to this specific situation. In section 3 we will return to the conclusions of the different case studies. First we will present the second pilot project, the NetPlanner case.

2.2 Pilot project 2: The NetPlanner case

NetPlanner is a software product which is developed to aid the planning, design and control of distribution networks. The product aims not only at specialists but also at generalists. It can be seen as a product which has been adapted to the present way of knowledge transfer; everybody has access to all kind of knowledge at different levels of detail.

2.2.1 The software product

The software product is a DOS application with a graphical user-interface. The source code is made in PASCAL. NetPlanner can carry out three kinds of network-calculations. Customers of NetPlanner who are registered users of the program are provided with updates of the program. The software product can be characterized as follows:

Scope of the product:In this case not only the software is considered but also the users' guides, the service provided by the supplier of the product and the 'fit' in the business situation. Although NetPlanner is a tool, it looks a bit like an information system.
Phases of the life-cycle: The whole life-cycle of the product is considered.
Uniqueness: The product is not very unique, because substitutes are available; however, those programs are not as advanced as NetPlanner.

2.2.2 The business system

The product is intended for energy distribution companies, energy production companies, industry, engineers and for educational purposes. The product is not the only product that provides decision-makers with the information needed. It is a tool and not a complete information system. Failing of the tool can, however, cause significant economical damage to the company which uses the product. The business system can be characterized as follows:

Number and **interdependence** of system variables: The number of variables that are relevant to the business system (and the program processes) can be scaled from small to medium. The variables are interdependent; if one variable is altered, the others will change too.
Controllability and **predictability** of the system variables: Not every variable that influences the business system which the product needs to support can be controlled or predicted easily. The business system is affected by so-called irregular variables.
Sensitiveness and **stability** of the controlled system: The planning, design and control of electricity distribution networks is not a sensitive or unstable activity.
The **reaction-pattern** of the controlled system: As can be concluded from the remarks made on the other dimensions, the controlled system is not precarious.

2.2.3 The customer

The customers of the product are companies in the energy distribution and production industry. The users and operators are trained engineers (except of course those users who use the program as a training facility) who are orientated on the output of the software product.

2.2.4 Relevant quality characteristics

By characterization of the product, business system and customer and with the expertise of a consultant, the following ISO quality characteristics could be marked as the most relevant.

Functionality (suitability, interoperability and compliance)
Because of the number and interdependence of the relevant variables and the need to fit into the specific business situation of electricity networks.
Usability
Because the product looks like an information system with just less 'responsibility'.
Maintainability
Because of the lifetime responsibility of the developer for the software product.

2.2.5 Determining the quality profile

Which level of evaluation could be assigned to the relevant quality characteristics? The same approach has been used as in the first pilot study. The most important complementary reasons are described.

Functionality: level C
Because of the risk of significant economical loss if the product gives incorrect information.
Usability: level C
Because the product looks like an information system and because of its graphical interface. And because the users are experts or trained people.
Maintainability: level C
Because of the lifetime (8 - 15 years) and the responsibility of the developer for the product.

2.2.6 Some remarks

The same approach is applied in this case as in the first DCD case. The use of the model was more straightforward than in the first case. However, the expertise of an software consultant was indispensable.

The reason why the model was of more help in this NetPlanner case than it was in the first DCD case, is to be found in the literature on which the model is based. Most literature published on software product quality concentrates on information systems. Although NetPlanner cannot be called an information system, it certainly looks more like one than the software product in the DCD case. If embedded software and information system software are the two ends of a continuum, the DCD product must be placed on the 'embedded' side and NetPlanner on the 'information system' side.

As in the first case the two steps (first determining the relevant quality characteristics and then the evaluation levels) show some overlap and it looks as if the later step makes the first one unnecessary. In the following section the results will be discussed.

3 Discussion on results of pilot projects

In the two cases the model was a helpful guideline for the software consultant in determining the quality profile. It helped the consultant to think in the characteristics of the product, the situation the product will work in and the people who will use the product. The model combines different insights described in literature on the subject of software product quality and proposes a translation process that partly formalizes the determination of the quality characteristics. However, the present model cannot provide the information which is needed by a non-experienced consultant. In its present form the model does not add to the knowledge of the consultant.

The proposed model focuses on products resembling information systems. The determining dimensions sometimes become almost irrelevant if the product concerned does not look like an information system at all. The dimensions in general do not all have the same power of distinction and not every dimension is applicable to every sort of software product, business system or customer.

A third remark that must be made concerns the application of evaluation levels. The strict line between the two steps —is a quality characteristic relevant and what evaluation level should be applied— leads to confusion. The first step is taken on the basis of more general terms and the second step is taken on the basis of norms and limits. Although this seems very logical, the practical application of the concrete last step makes the first step look unnecessary. The general terms in the first step are however essential for the determination of quality characteristics because the guidelines used in the second step will never be complete enough for application to all software products. On the other hand, there is a need for variations in the thoroughness with which the quality characteristics are evaluated.

Further work to be done

From the remarks made we can conclude that a model with more expertise is needed. It must be able to characterize a software product by the use of dimensions with great distinguishing power. This model must also be able to appoint the right level of thoroughness to the different quality characteristics of the software product. A logical next step in the development of the model seems to be an integration of the level concept and the use of determining dimensions. Thus the relevant level of thoroughness is implicitly determined by the detailed characterization of the sort of product, customer and environment.

The dimensions upon which the characterization has been based until now must be taken in careful and detailed consideration. The model must be applicable to all kinds of software products, for all kinds of customers in all kinds of business systems. Further confrontations with the field are necessary to enhance the applicability of the proposed model and to make it more than a guideline, to make it a real tool in the determination of quality characteristics of software products.

References

1. Bache R. and G. Bazzana, *Software metrics for product assessment.* McGraw-Hill, London, 1994.

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

3. Bemelmans T.M.A., *Bedrijfskundig Ontwerpen van Bestuurlijke Informatiesystemen*, in: P.A. Cornelis, J.M. van Oorschot (eds.), Automatisering met een menselijk gezicht (in Dutch), Kluwer, 1986.

4. Berg van den R.J., J.J.M. Trienekens, *Setting Priorities in Software Product Quality; towards a CASE based instrument*, in: Proceedings of the Sixth workshop on CASE: Software Improvement with Case, Singapore, Singapore, 1993.

5. Boehm B.W. et al, *Characteristics of Software Quality*, TRW Series of Software Technology, Vol. 1, North Holland Publishing Company, 1978.

6. Davis G.B., M.H. Olsen, *Management Information Systems,* McGraw-Hill, London, 1984.

7. Delen G.P.A.J., D.B.B. Rijsenbrij, *The Specification, Engineering and Measurement of Information Systems Quality*, in: Journal System Software. Vol. 17, 1992.

8. Dorling A., *SPICE: Software Process Improvement and Capability dEtermination,* in: Software Quality Journal, Vol. 2, 1993.

9. Heemstra F.J., R.J. Kusters, J.J.M. Trienekens, *Defining Systems Quality: involving end-users*, in: Proceedings of the European Function Point User Group Conference, Bristol, England, 1993.

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

11. Information Technology Security Evaluation Criteria (ITSEC), *Provisional harmonised criteria, version 1.2.* Commission of the EC, 1991.

12. ISO 9000-3, *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, 1991.

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

14. ISO/IEC JTC1/SC7 N1201, *Working Draft; Information Technology: Indicators and Measures,* ISO/IEC, 1994.

15. Jarke M., K. Pohl, *Information Systems Quality and Quality Information Systems,* Kendall, Lyytinen, DeGross (eds.), in: Proceedings of the IFIP 8.2 Working Conference on The Impact of Computer Supported Technologies on Information Systems Development, Minnesota, USA, 1992.

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

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

18. Mellor P., *Critique of ISO/IEC 9126.* Centre for Software Reliability, City University, London, 1992.

19. Oakland J.S., *Total Quality Management,* in: Proceedings of the Second International Conference ont Total Quality Management, Cotswold Press Ltd., 1989.

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

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

22. Trienekens J.J.M., *Quality Management in Software Production*, A Customer Oriented Approach, in: Proceedings of the IFIP 5.7 Working Conference on Integration in Production Management, H.J. Pels and J.C. Wortmann (eds.), The Netherlands, North-Holland, 1992.

23. Trienekens J.J.M., P. Thoma, *Production Characteristics as a Basis for Effective Quality Systems, learning form industrial manufacturing*, in: Proceedings of the Second International Conference on Software Quality Management (SQM'94), Computational Mechanics Publications, UK, Edinburgh, Scotland, 1994.

24. Vliet J.C. van, *Software engineering, Principles and Practice*, John Wiley & Sons, 1993.