Defining systems quality: involving end users

Kusters, R.J.; Heemstra, F.J.; Trienekens, J.J.M.

Published in:
Proceedings of the profiting from software measurement conference, European Function Point User Group, 25-26 March 1993, Bristol, England

Published: 01/01/1993

Document Version
Accepted manuscript including changes made at the peer-review stage

Please check the document version of this publication:

• A submitted manuscript is the author's version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

Citation for published version (APA):
Defining systems quality: involving end-users

R. Kusters,
F. Heemstra,
J. Trienekens

Reprint Bdk/444


Reprinted for private circulation by the author(s).

Eindhoven University of Technology
Graduate School of Industrial Engineering and Management Science
Postbox 513
5600 MB Eindhoven
The Netherlands
Telephone +31 (40) 47 91 11
DEFINING SYSTEMS QUALITY: INVOLVING END-USERS

Dr. R. Kusters
Dr. ir. F. Heemstra
Ir. J. Trienekens

Section Information & Technology
Department of Industrial Engineering
Eindhoven University of Technology
P.O. Box 513
5600 MB Eindhoven
The Netherlands
Telephone: (..)3140472290
Telefax: (..)2140451275
Email: rku@bdk.tue.nl

ABSTRACT

Quality and quality control take an increasingly prominent place within the IS field. However, many organisations do not succeed in managing quality concepts successfully. Quality management fails to proceed beyond general procedures or gets bogged down in the application of a set of (isolated) metrics. In this article an approach will be described which is based on the notion of furthering the communication between developers and prospective users. The goal is to break the abstract notion of quality down into something which is understandable for prospective users as well as for system analysts. Such communication is in our opinion the basis for effective total quality management.

The approach described in this article was developed as part of a research project of the section Information & Technology of Eindhoven University of Technology. The project was carried out within a large Dutch governmental organisation. At present a follow-up project within the same organisation is aimed at further development and practical implementation of this approach.
1. INTRODUCTION

Quality and quality management more and more take a central place within the IS-community. The usual concepts of controlling time and effort no longer suffice to fulfil the requirements of customers and users (Basili and Musa, 1991; Card 1991). Principals require the assurance that, apart from meeting time and budget constraints, the final product will also meet the nebulous concept of 'quality'. Many definitions of quality exist. The framework of Garvin (1984) of different definitions, each hailing from a different point of view, illustrates this perfectly. In this paper we will adhere to a users viewpoint, in which 'fitness for use' is to be aimed at.

IS developers attempt to find means of providing this assurance. A currently popular approach is to try and gain the customers confidence by the development and certification of a quality management system. Based on international standards (ISO 9000-3, 1991) comprehensive sets of procedures aimed at quality control during the development process are being developed. The final goal here is to obtain a 'total quality management system'. Quality management in information systems development in this sense consists not only of the control of time, money and resources of IS projects, but also the identification, specification and realisation of functional and non-functional requirements is covered in this approach.

However, an important drawback of this approach is, that all too often it results in procedures and guidelines that are too abstract and too general in order to be applicable in a specific situation (Jarke and Pohl 1992). There are hardly any operational guidelines, either in literature or in practice, for the development of effective quality management within an organisation. International standards only offer a global framework aimed at defining quality management procedures.

In this paper an approach towards more directed development of quality management will be presented. The general assumption underlying the paper is that communication is a key factor in the introduction of an organisation wide quality management system. Lack of communication protocols is an important factor causing the high levels of abstraction and generality that were observed by Jarke and Pohl (1992).

In this paper we will first indicate why we assume that the lack of communication is an important factor inhibiting the successful introduction of quality management systems. Next we will look at existing methods for alleviating this problem. We will point out a problem that is inherent to these existing approaches and end the paper with the formulation of an addition to existing methods that may solve this problem.

2. PROBLEMS IN INTRODUCING QUALITY MANAGEMENT TOOLS.

For several years now the section of Information and Technology at Eindhoven University of Technology has been looking into the problem of designing and introducing management methods and tools into IS-development organisations (see e.g. Heemstra and Kusters, 1992;
Kusters and Heemstra 1992 and Van Genuchten et al. 1991). We found that IS-development
organisations exhibit many of the characteristics of the type of organisation that Mintzberg
described as 'professional bureaucracy' (Mintzberg, 1983).

This mostly shows itself in the degree of professional autonomy that is perceived by the
workers in the organisation in question. The members of a professional organisation
generally do not take the imposition of authority for granted. Only if this authority is based
on professional competence is any acceptance likely.

In order to assure the acceptance of any method, the problems caused by this attitude
somehow have to be avoided. We found that this is more likely if three design principles are
heed to (Bemelmans, 1991). These three design principles are:

- Local for local.
  In designing a method care has to be taken to adapt the method as much as possible to
  the specific circumstances of the target organization in such a way that it reflects the
  specific characteristics and jargon of this organisation.

- Internal control.
  Imposing control on a group of software developers by means of the obligatory use of
  one or more methods often has no or even adverse effects. Most professionals consider
  themselves capable of monitoring their own work. The design and introduction of a
  method will therefore have to rely on the appearance of the usefulness of the method to
  all parties involved.

- Closed loop.
  A closely related principle is the so called 'closed loop' principle. If care is taken that the
  people who gather the information can put this information into practical use themselves,
  high quality information gathering is more likely to happen. This is closing the loop, i.e.
  giving feedback to the suppliers of information.

One common theme underlies these three principles: that of active cooperation of all parties
involved based on mutual understanding. This brings us to the central theme of this paper.
Mutual understanding on a given subject is possible only if all people involved succeed in
communication on that subject. On the subject of quality this is by no means certain.

The notion of 'quality' has been referred to above as 'nebulous'. Its meaning is not obvious
from the outset, and much effort has already been spent in (re-)defining quality (e.g. Boehm
et al., 1978) but until now success has been patchy. The next section will look into this
aspect.

3. PRESENT APPROACHES AND PROBLEMS.

The distinction between functional and non-functional requirements is a common one in the
IS-field (see e.g. (Boehm et al. 1978; Cavano and McCall 1978; Deutsch and Willis 1978,
Bemelmans 1991; Delen and Rijsenbrij 1992)). Non-functional requirements, also called
quality requirements, are related to the way in which the functionality of the information
system is realised. Examples of these quality requirements are reliability, user friendliness,
maintainability and portability of an information system (Boehm et al, 1978).
Most of the literature in this area is based on the work of Boehm et al. (1978) and of Cavano and McCall (1978). In their view quality is expressed in a number of quality attributes of a software product. These attributes can be derived from a set of aggregated quality factors. The next step is to define metrics which can be used to test if the final product adheres to these attributes. Over the years this approach has been further refined by several authors (Willmer 1985; Deutsch and Willis 1987; Keller, Kahn and Panara 1990), and in particular the relation between the attributes have been described in a more realistic manner.

In general in this literature a hierarchical structure between quality factors and quality attributes is postulated. At the highest level are put strongly aggregated quality factors (e.g. maintainability, reliability, etc.). At the lowest level in this structure (measurable) quality attributes are placed which as 'properties' of an information system will have to be realised. Examples of these attributes are modularity, structuredness etc. (Boehm et al, 1978). Use of this hierarchy is made in a circular way as described in the more recent work of Delen and Rijsenbrij (1992) (see figure 1).

![Diagram](https://via.placeholder.com/150)

Figure 1: present approach to quality

This circular course consists of three steps which are titled 'factors', 'measures' and 'attributes'. The step 'factors' relates to the specification of quality factor put upon the system. 'Measures' deals with determining the activities that will have to be carried out in order to assure that these demands are met. 'Attributes' indicates those (preferably measurable) attributes that finally will characterise the system to be built. Of course, during project execution constant verification and adjustment will have to take place.

Further refinement has not taken away the most important objections to such an approach. Several of these objections may be distinguished.

- One objection is that too strong an emphasis is placed on the (technical) translation from aggregated demand in attributes and the rigid derivation methods that are prescribed.
Another problem is that no attention is paid to the prioritisation of these factors. Most of the time a trade-off between quality requirements and budget has to be made for which such priorities are needed as input. Present methods hardly give users the opportunity to indicate these priorities.

The greatest objection however is that principals and potential users are not supported in the formulation of their requirements towards quality (see e.g. (Kaposi and Kitchenham, 1987). Both aggregated quality factors as detailed quality attributes are described in technical terms. It is not clear how an organisation can determine those quality requirements that are relevant in its specific situation and in which language this can be formulated (Trienekens, 1992).

Consequences are clear; over optimistic user expectations, misunderstandings during the development process and dissatisfaction afterwards caused by unfulfilled expectations. The 'voice' of the user is not or only with difficulty be found in the final product. Another possible consequence is that it allows developers to deal with the quality factors in a technical way; as a result they are mainly focused at (standard) technical measures and at designing (standard) technical attributes;

Quality in this approach is not negotiable or manageable in this very important starting phase of system development. This means that quality management is limited to the introduction of a set of general procedures with relation to a limited set of (standard) attributes. User input is negligible, with all dangers this implies.

What is needed is a language which defines the rather vague notion of quality in terms that both system developers and potential users can relate to. Somewhat surprisingly such a language is not available. The literature referred to above describes quality requirements in terms of system properties, that is in a language that is particularly suited to IS-professionals. Potential users most of the time do not 'speak' this language, nor are they interested in learning it. This is not surprising, and given that users can be considered as customers, it is not realistic to assume such a willingness.

In a sellers market the seller can impose his methods on the customer. Thus is was possible for Henry Ford to state that his customers could buy any colour of car as long as it was black. However, the IS-field can not be characterised any more as a sellers market. All too many organisations are prepared to deliver information systems and even IS-departments within in most organisations no longer have sole claim on designing the systems required. So just as the Ford Motor Company nowadays sells cars in all colours, it is within the IS-field no longer feasible to require that customers be familiar with the arcane languages that IS-professionals use.

Based on these problems we conclude that an addition to the approach depicted in figure 1 is needed which explicitly involves potential users in the process of defining quality. Starting at quality factor and ending at quality attribute was shown to be insufficient to make quality manageable. An additional phase is needed in order to arrive at these quality factors. We will introduce the notion of quality requirements, as a user specific language which allows them to describe what they need.

Quality management, apart from dealing with project management phases, also involves the identification, specification and realisation of quality requirements. Insight into the quality that users require of an information system is of primary importance when aiming for an effective total quality management. Managing quality, when it is not known what is meant by quality can only by chance lead to the desired satisfaction of the user community.
It is not enough just to know the requirements on. One also needs to know how to achieve these requirements. This means that two distinct phases to quality management can be distinguished:

- first the need to identify quality requirements in a way that appeals to users,
- second, the need to embed the realisation of these requirements in the development process in a way that appeals to the developers.

The second phase, from quality requirement to specification, is covered extensively in literature (e.g. Boehm et al. 1978; Cavano and McCall 1978; Delen and Rijsenbrij 1992). In discussions with users and developers within the organisation we found that this part is often tendered to an outside contractor.

For the first phase, that always will have to take place within the organisation, no relevant concepts were found to exist, let alone that operational methods and techniques exist that support it. The main problems of users and developers in this first phase was found to be:

- how to identify quality requirements of users and user management,
- how to translate these requirements into the quality factors that can be handled by developers.

In the remainder of this paper a first step in the direction of a method that attempts to fill this gap will be described. Here we will loosely define 'developers' as all people involved in defining and developing an information system. As 'users' will be considered those people within the organisation who either direct (e.g. end users) or indirect (e.g. user management or system management) are involved in the use of an information system.

4. A NEW APPROACH FOR DEALING WITH QUALITY REQUIREMENTS.

In this section a quality control loop with respect to the identification of quality factors will be discussed, together with the building blocks of the approach and some examples.

4.1 A quality control loop in the starting phase of a project.

A first approach towards such a control loop is presented in figure 2. In this figure four main activities are described which will each be looked at.

Main activity 1.
Identifying quality requirements.
These requirements will have to be derived from the organisational context that the users in question operate in. This context, described in terms that are in every day use in the organisation, give information on the requirements that will be put on the system. A checklist will be provided to find out where the most important areas of concern are situated. It is quite likely that, without limits imposed by budgetary concerns, this will result in the description of a 'gold plated' system. In order to take this into account the next activity is added.

Main activity 2.
Setting priorities in the requirements.
Explicit criteria are set down. Based on these criteria priorities between the requirements are expressed. These priorities can be used as a basis for discussion in the balancing process between requirements and budget. Such a balancing process nearly always takes place (Bemelmans, 1991), but is rarely carried out in an explicit and accountable way so that budget and user requirements are directly balanced.

**Main activity 3.**
Relating quality requirements to quality factors.
The translation between the requirements set out by the users in their terminology, and the factors that will have to be in a language that developers are familiar with, will take place in this activity. To facilitate this translation a pre-defined set of relations will be provided.

**Main activity 4.**
The priorities that are set out in main activity 2 will have to be related to a similar priority setting of the factors. These priorities can be used as a guide during the development process. Given that projects are rarely executed according to plan, these priorities can function as a guide during a replanning process or, in case of larger problems, they can function as a guide in the re-execution of the balancing process described above.

This of course is not a one-way street. Feedback can take place at any time during project execution. Such a feedback will take place during any project, but it is often not discerned as an explicit activity. One of the side benefits of this approach is, that an explicit re-evaluation of the quality characteristics involving both users and developers is now possible.

One of the main results of this approach is that (main activity 3) the quality factors put upon the product are now defined. Using them as a starting point the second phase aimed at determining measures and attributes can now be started. This means that the approach fits in directly with the existing methods.
Quit often the budgetary consequences of the choices made during the first phase can only be evaluated after completion of the second phase. This means that the balancing process described above will have to be extended over the second phase as well. Feedback over both phases will have to be taken into account during this process. The two phases can be distinguished as being separate activities, however during execution it will most of the time not be possible to separate them.

4.2 Determining quality factors: tools and techniques.

Now some of the techniques and tools required in such an approach will be looked at. The main requirements are:

- checklist, based on a questionnaire;
- a list with relevant prioritisation criteria;
- a relation matrix (in order to link requirements and factors).

The checklist.
Apart from questions on purpose and benefit of the system also questions are posed on the organisational context within which the system will be used. Examples are questions regarding:

- the organisational position of the user (e.g. responsibilities and authority);
- user experience;
- tasks to be supported;
- connections with and dependencies on other tasks;
- volatility of the tasks.

The answers to these and other questions in essence form the quality requirements as perceived by the users. They will be used as a basis for deriving the quality factors. When formulating the questions in this list a start is made using terminology as presented in literature. For instance the notion of volatility can be found in Ghani (1992). When adapting the list to the specific set of circumstances that exists within the organisation a change of wording will have to take place.

A list with relevant priority criteria.
This list will serve as a guideline while setting priorities. It will be based on local conditions within the organisation and will contain a list of priority types that are in common use.

The relation matrix.
The idea for using a matrix depicting the relation between quality requirements on the one hand and quality factors on the other is taken from the method Quality Function Deployment (Hauser and Clausing 1988). We will illustrate this matrix and its use briefly.

4.3 The matrix depicting the relation between quality requirements and factors.

Goal of this matrix is the visualisation of the relationships between the quality requirements as vocalised by the users on the one hand and the quality factors needed by the developers on the other. The vertical axis of the matrix will contain the user requirements. These were acquired using the checklist described above.

The horizontal axis of the matrix will contain the (technical) quality factors required by the developers. The terms used here can be taken from literature (e.g. Boehm et.al. 1978; Delen and Rijsenbrij 1992) or from the locally used systems development method. Relations
between requirements and factors can now be entered in the matrix. The nature of the relationship can be indicated by using different symbols comparable to the symbols used in the method QFD. This information regarding relationships between requirements and factors exists independent of the type of project to be executed and can therefore be supplied as part of the method (see figure 3).

Quality factors

\[
\begin{array}{ccccccc}
& q_1 & q_2 & \ldots & q_{f1} & \ldots & q_{f4} & q_m \\
q_{r1} & & & & & & & \\
q_{r2} & & & & & & & \\
\vdots & \vdots & \ddots & \vdots & \vdots & \ldots & \ldots & \vdots \\
n & & & & & & & \\
q_{m1} & & & & & \square & & \\
q_{m2} & & & & & & \square & \\
q_m & & & & & \square & & \\
\end{array}
\]

symbols:

- weak positive
- strong positive
- weak negative
- strong negative

Figure 3: setup of the relationship matrix between quality factors and quality requirements

Some examples of the relations are:

- there is a relationship between the degree of user experience and the required degree of user friendliness;
- the number of users also has an impact on user friendliness;
- task volatility indicates high maintainability;
- high dependency on information indicates high reliability.

Information on priorities of quality requirements can be added to the vertical axis. From this priorities on the user demands, the horizontal axis can be derived. All information generated by the method is now available in this one matrix. This matrix which presents the link between the languages of users and developers can now be used as a means of communication between them.

This approach for determining and describing quality requirements is at this moment being expanded into a complete operational method. The method will offer support for determining quality factors, but the final decisions will have to be taken by the people involved. Such a method can only support this process, not supplant it.

This method offers an approach towards making quality discussable in the first phase of information system development and to set priorities. Use of the method in an actual situation will preferable take place by several developers and users. Different preferences and requirements will in this way be part of the discussion, which can only enhance the final
result. In this way a structured discussion is made possible in an early phase of development.

5. CONCLUSIONS

In this article an approach is described towards quality management with regard to information systems development. This approach differs from the one usually followed. Instead of developing a complete set of procedures according to the international ISO-9000 guidelines we aimed at developing an instrument that is to support the communication between user and developer when defining quality requirements of a prospective information system.

The method will have to be usable by all users in cooperation with developers on all levels of the organisation. The main reason for this is, that only this way users may participate actively and fruitfully in determining, realising and maintaining information systems quality. This was seen as an essential starting condition for the development of a total quality management system that is firmly rooted within the organisation.

It was found that sufficient methods are available to serve developers in their discussions on the subject of quality. However, users are not as well provided. This paper is a first attempt at developing a method to fill this gap. First on the basis of a checklist of questions regarding their ways of working, the tasks they perform and some characteristics of their environment an indication of user quality requirements is obtained. Priorities are set, and the link between these requirements and the quality factors needed by developers is made explicit. This method blends in well with existing methods aimed at detailing quality attributes of information systems and can thus be fitted in with the existing way of working.

Further work aimed at operationalisation of the method is under way and will early 1993 result in a proposal for implementation. With this method as a starting point a more profitable discussion between users and developers becomes possible. This we see as a first vital step in the direction of the implementation of a quality management system.

LITERATURE

Basili V.R., J.D. Musa,
The future engineering of software: a management perspective.

Bemelmans T.M.A.,
Bestuurlijke Informatiesystemen en Automatisering (Management Information Systems and Automation),

Boehm B.W. c.s,
Characteristics of Software Quality,
Card D.N.,
Measuring Software Design Quality,

Cavano J.P., J.A. McCall,
A framework for the measurement of software quality.


Deutsch M.S., R.R. Willis,
Software Quality Engineering,
Prentice Hall, 1987

Ghani J.A.
Task uncertainty and the use of computer technology.

Hauser J.R., D. Clausing.
The House of Quality.

Heemstra, F.J. and R.J. Kusters,
Software cost estimation and control; lessons learned,

Garvin, D.A.,
What does 'product quality' really mean,

Genuchten, M. v., F. Heemstra, M. Howard and R. Kusters,
Group Decision for Software Cost Estimation?

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 for Standardization, Switzerland, 1991

Jarke M., K. Pohl.
Information Systems Quality and Quality Information Systems.

Kaposi A., B. Kitchenham,
The architecture of system quality,

Keller S.E., L.G. Kahn, R.B. Panara.
Specifying Quality Software Requirements with Metrics.
Kusters, R.J. and F.J. Heemstra,
Cost benefit analysis implemented in a government organisation,

Mintzberg, H.,
Structure in fives: designing effective organizations,
Prentice-Hall, 1983.

Trienekens J.J.M.
Quality Management in Software Production, A Customer Oriented Approach.

Willmer H.,