Establishing and managing a relevant notion of quality


Department of Mathematics and Computer Science, University of Salford, M5 4WT, UK

ABSTRACT

Quality is considered to exist in the relationship between customers and products, rather than as an intrinsic characteristic of a product. Quality is concerned with customer needs and expectations and, with respect to software, is contingent upon the organizational context. Having identified the product to be supplied, the customer and other affected parties, an accommodation concerning a relevant notion of quality is established. A quality management framework is proposed which addresses four quality elements from the supplier perspective: product performance, process management, resource performance, and customer expectation management. Traditional software engineering is considered to be working within a functionalist paradigm with a production view of quality. Information system development is more concerned with the organizational context and a use view of quality. In defining a relevant notion of quality the professional role and responsibility of the developer is considered, which requires the tension between the customer and supplier notions of quality to be recognised.

1 INTRODUCTION

The quality of software and the quality of the supporting process of software engineering are fundamental issues for organizational effectiveness. However, much of the effort expended on software quality has focused on achieving more effective control of the software engineering process [Humphrey 1989; Haddley and Sommerville 1990], emphasising the need for improved project management and the introduction and subsequent enforcement of standards. The desire to establish control over the software engineering process has lead to the search for software metrics to monitor attributes that include software errors, software reliability, software usability, and software maintainability [Metkit].
However, there is also recognition that process engineering by itself is insufficient. Developing large software systems must be treated partly 'as a learning, communication, and negotiation process' [Curtis, Krasner & Iscoe 1988]. Software quality is problematic and it is not sufficient to reduce it to metrics, quality assurance and standardisation. Fenton recognises the 'philosophical debate about the meaning of software quality' [Fenton 1991]:

Quality is perceived in terms of those external product attributes which are relevant for particular types of users (page 222).

This paper is concerned with the management of quality, which we believe is not synonymous with process engineering. In order to gain an insight into the quality management requirement we first consider the nature of quality and the product of the development process. We then ask who is the customer and through what means might a relevant notion of quality be agreed. Having argued for the need to address explicitly the identification of the product, the customer, and the customer's quality requirement, a quality management framework is proposed.

2 THE NATURE OF QUALITY

A commonly accepted definition of quality is that supplied by the International Standards Organization [ISO 1986]:

The totality of features and characteristics of a product or service that bear on its ability to satisfy stated or implied needs

Another prototypical definition of quality is [Gitlow et al 1989]:

Quality is a judgement by customers or users of a product or service; it is the extent to which the customers or users believe the product or service surpasses their needs and expectations

Quality has also been defined as 'fitness for purpose' [Juran 1979]. All of these definitions of quality require the presence of a customer (a person or group of people) to perceive the quality of a product.

Thus, quality should not be considered to be intrinsic to a product; quality exists in the relationship between 'customer' and 'product' (where customer can be any stakeholder or affected party, and product can include goods and services). In the language of data modelling, quality exists in the relationship between the entities 'customer' and 'product' and is not solely an attribute of either entity. For quality to be purely an attribute of a product requires the acceptance of a metaphysical belief that quality can exist at a transcendental level (i.e., independently of human beings). For the practical purposes of quality management it is the subject/object view of quality that will be explored; the subject is the customer (who perceives quality) and the object is the product (which has attributes which may or may not be perceived by the customer as constituting quality). The notion of quality is considered to be a socially constructed phenomenon that changes over time in response to
changes in the environment and to changes in values, where values govern the beliefs of the customer, the supplier and other affected parties concerning what is 'good' or 'bad'.

It is not meaningful to discuss the notion of software quality without reference to a subject, i.e., it is necessary to consider whose quality expectation it is that the software is expected to satisfy, in what organizational context, and at what time. The term software quality only takes on meaning when a user, or some group of users considered to have sufficiently homogeneous requirements, has been identified. Software can be developed through a process of system development with a greater or lesser regard for the organizational situation in which that software will be used and become part of. If the objective is to improve quality then it is essential to clarify what product, or service, is being offered to the customer.

3 THE PRODUCT OF SYSTEM DEVELOPMENT

It is possible to consider the product of system development within a range that is delimited at one extreme by traditional software engineering and at the other by information system development. These stereotypical extremes are contrasted in order to gain an insight into the nature of software and information system quality.

Software Engineering

The traditional definition of software engineering is a 'discipline for developing a complete, consistent, unambiguous specification' [Boehm 1976]. The software engineering view of system development is concerned with the design and control of a process that results in an artefact or product which meets or surpasses a requirement, as defined by a user requirement specification. The software engineering perspective is deterministic, founded on the assumption that there is some definable, true and real set of requirements that can be elicited and formally specified [Halpin and McCormack 1992, Roland and Proix 1992]. The objective is the production of a piece of software (a technical artefact) that conforms to a specification, has zero defects and is produced as efficiently as possible. It is this concern with the process of engineering technical artefacts that we refer to as a production view of quality.

Information Systems Development

In considering the nature of an information system it is necessary to consider what is meant by information. If information is considered to be data that have meaning within a specific context, then data can only have meaning when considered by a human being, or a group of human beings. The information system is thus concerned with giving people access to data, supporting them in interpreting that data (i.e., transforming it into
information), and assisting them in carrying out their organisational activity [Checkland 1988]. The technical artefacts produced by software engineering become part of an information system that incorporates formal and informal mechanisms for creating, communicating and using information [Land 1992]. Information system quality is concerned partly with how well technical artefacts perform in the organisational setting [Miles 1985]. The product of the system development process is a service rather than a product, that service being to assist organizations in the construction of information systems [Swanson and Beath 1989]. Such a service will routinely, but not necessarily, involve the use of technical artefacts which result from the process of software engineering. It is the concern with the effectiveness of software artefacts within an organizational context and the provision of a service that we refer to as a use view of quality.

<table>
<thead>
<tr>
<th>regulation/consensus</th>
</tr>
</thead>
<tbody>
<tr>
<td>Functionalist</td>
</tr>
<tr>
<td>technical expert</td>
</tr>
<tr>
<td>warrior</td>
</tr>
<tr>
<td>Radical structuralist</td>
</tr>
</tbody>
</table>

Figure 1
Roles and Metaphors Assumed in System Development

Is there a fundamental difference between the notion of quality prevalent in software engineering, which emphasises the production view of quality, and the information system notion of quality, which emphasises the use view of quality? It would be a misrepresentation to see the software engineering paradigm as one that ignores the organizational context and the individual user, particularly given the current level of interest in human-computer interaction. The differences between software engineering and information system development are not reducible purely to a question of production versus use. The differences are perhaps more subtle and can be explored by considering the different paradigms within which developers work.
4 PARADIGMS EMPLOYED IN SYSTEMS DEVELOPMENT

Figure 1 identifies four paradigms [Hirschheim and Klein 1989; Avison and Wood-Harper 1990] by categorizing assumptions about the way in which developers gain knowledge of requirements, ranging from subjective to objective, and assumptions about the environment in which they work, ranging from consensus to conflict. Working within an objective epistemology one attempts to treat the social aspects of information systems in a scientistic manner, where the human and social aspects are just more objects to be dealt with. The subjective view considers such an approach to be inappropriate where people are concerned. The developer who perceives the situation as one of consensus sees order and stability; the developer who perceives the situation as one of conflict sees stress, coercion and change.

The different combinations give rise to four paradigms, for each of which we expect developers to perceive differently the nature of quality. For a discussion of the contingent aspects of the roles adopted by developers see [Bell & Wood-Harper 1992].

![Diagram](image)

**Figure 2**

Technical Artefacts and Information Systems

The 'expert' sees quality as a rational process of engineering where the objective of an information system is not considered to be problematical once it has been enshrined in an unambiguous specification. Quality is concerned with meeting a specification, implying the need for rigorous control of the
122 Software Quality Management

process. The developer chooses from the potentially infinite combinations of product attributes those that are felt to reflect the customer's quality requirement (possibly on the basis of the developer's experience of supporting customers who use the product in the field). There is a tendency for the developer to perceive quality solely as an attribute of the product rather than in the relationship of the customer and the product. Much of software engineering practice today follows this functionalist paradigm in its use of reductionist methods.

<table>
<thead>
<tr>
<th>Software Engineering</th>
<th>Information Systems Development</th>
</tr>
</thead>
<tbody>
<tr>
<td>Produces a product - the technical artefact</td>
<td>Supplies a service to assist the organization in constructing information systems</td>
</tr>
<tr>
<td>The emphasis is on controlling the process (QA) - the production view</td>
<td>The emphasis is on how technical artefacts are used - the use view</td>
</tr>
<tr>
<td>Quantitative techniques (metrics) are the preferred method of assessment</td>
<td>Qualitative techniques and judgement are the preferred method of assessment</td>
</tr>
<tr>
<td>Quality becomes perceived as intrinsic to products (reinforced through standards)</td>
<td>Quality is contingent and resides in the customer perception of a product</td>
</tr>
<tr>
<td>The process is based on an engineering paradigm, where the means-ends approach can lead to a preference for the waterfall model of development</td>
<td>The ends (purpose) of an information system are considered to be problematical, leading to a preference for incremental and evolutionary development</td>
</tr>
<tr>
<td>The notion of quality is given</td>
<td>The notion of quality is socially-constructed</td>
</tr>
<tr>
<td>The notion of quality endures</td>
<td>The notion of quality is continually changing</td>
</tr>
<tr>
<td>Quality is dependent upon the deep aspects of the product (e.g., the technical design)</td>
<td>Quality is dependent upon the customer's view of the product (the superficial aspects)</td>
</tr>
</tbody>
</table>

Table 1
Software Engineering and Information System Development Contrast

The 'warrior' also perceives quality as objective and is thus still concerned with the 'means-ends' approach. However, developers working within this paradigm assume that there are conflicting interests and place greater emphasis on the question of whose quality requirement is being addressed, for example, is it management's perception of quality or the workers' view of quality as represented by a trade union. There is recognition that it will not be possible to satisfy fully everybody's quality requirement.

The 'teacher' considers quality to be socially constructed. Quality is whatever it is inter-subjectively agreed to be. The overriding need is to involve as many different groups as possible in order to include multiple perspectives and interests. The resultant definition of quality will reflect a consensus of the views of the involved parties (e.g., senior managers,
developers, end-users etc.). One approach that incorporates these multiple perspectives is Soft Systems Methodology (SSM) [Checkland 1981; Checkland and Scholes 1990].

The 'emancipator' also considers quality to be formed from multiple perspectives and interests, but assumes that there will be conflict and the exercise of power. The emancipator is concerned with constructing a notion of quality that takes into account the interests of the less powerful with the aim of human advancement through emancipation. SSM recognises that conflicts of interests are to be expected and thus expects the end result to be an accommodation rather than a consensus. However, SSM is considered to be insufficient for addressing situations typified by conflict and a significantly unequal distribution of power. In such situations, Critical Systems Thinking has been suggested [Flood and Jackson 1991].

Currently, system development is working predominantly within a functionalist paradigm with an accordant emphasis on the production view of quality. An organizational view of system development reflects the social construction of information systems and relates quality to how technical artefacts are used within an organizational context. If quality products are produced by quality processes then software quality is essential if information system quality is to be achieved, although 'software quality' is no guarantee of information system quality. It is, therefore, not a matter of choosing between the different paradigms but a question of how to balance the dialectic nature of the production and use views of quality. In some situations the production view of quality may dominate (where requirements can be agreed upon and are perceived as being relatively straightforward); in others, the organizational context may dominate where discussion and exploration of the problem domain are needed before the purpose and form of the information system can be agreed, perhaps involving the use of prototyping [Budde et al].

The relationship between information system quality and software quality is shown in figure 2 and software engineering and information system development are contrasted in table 1. Having considered the product of system development we now address the question of who is the customer and what attributes of the product reflect the customer's quality requirements.

5 Establishing a Relevant Notion of Information System Quality - The Customer View

A modified form of SSM is proposed as the basis for incorporating multiple perspectives in establishing a relevant notion of quality [Vidgen, Wood-Harper & Wood 1992b]. The notion of quality adopted will be contingent upon a specific organizational context and the values held by those involved in the process of establishing a notion of quality. The process of defining an in-context notion of quality involves exploring the organizational quality domain, identifying quality factors, the trade-off of quality factors, and the establishment of quality assessment criteria.
Of particular significance is the exploration of the quality domain. One of the purposes of this activity is to identify who is the principal customer for the information system and who are the affected parties, where affected parties may have requirements that differ from those of the customer. Affected parties are those affected by, or who have an interest in, the proposed information system, such as the computer operations staff who are expected to run the computerized system (an internal party) and regulatory bodies (an external party). The need to identify a principal customer reflects the requirement for someone to 'own' and take responsibility for a process. This is not to say that the primary customer necessarily has additional rights and privileges compared with other affected parties.

The development staff are not considered to be a dispassionate observer of this process. The development staff are one of the affected parties and are inextricably bound up in the process of defining a relevant notion of information system quality. Partly, this is due to pragmatic reasons: it is not possible for developers to adopt the role of a value-free observer; the interests of the developer affect the process of development and the form of the information system. It also recognizes the role of the professional developer who has a duty to advise customers of the implications of different courses of action and to recognize where new techniques might be advantageously employed. It is not sufficient to say that quality is whatever the customer says it is - the professional developer must share the responsibility for and be involved in, the process of establishing a relevant notion of quality [Dahlbom & Mathiassen 1991].

6 THE ROLE OF METRICS, QUALITY ASSURANCE, AND STANDARDS

If quality is to be defined by the relationship of the customer to the product then there are implications for the role of metrics, quality assurance (QA) and standards with respect to quality management.

Metrics

Metrics may be used to quantify some attributes of the object of quality but cannot be considered to define quality unless the attributes being quantified are relevant to the needs and expectations of the customer. Quality has been considered within the context of 'some very general models of software quality which have gained acceptance within the software engineering community' (page 223) [Fenton 1991]. However, these quality models are only relevant to the extent to which they reflect the needs and expectations of specific customers in specific situations. It is necessary to explicitly consider whose quality requirement is to be satisfied. Once the customer and the quality requirements are identified then there is a context for the use and interpretation of metrics.
Quality Assurance (QA)

QA is concerned primarily with controlling the process that generates products and services. QA is, therefore, two steps removed from quality since QA does not guarantee a quality process and a quality process does not guarantee a quality product. At best, QA is a 'good house-keeping' seal of approval that the process is documented and under control (to a greater or lesser extent). Indeed, it seems reasonable to expect that, without some element of cultural change, the implementation of a process-based standard (such as BS5750) might lead to greater bureaucracy and a diminution of organizational effectiveness [Braa & Øgrim 1992].

Standards

Standards can be seen as an attempt to totalize and institutionalize quality and to provide product consistency. There is a danger of a standard being thought to directly represent quality, where quality is perceived as intrinsic to any product that conforms to the standard. It is essential to ask whose quality the standard represents and whose interest it is serving. If quality is a local and contingent phenomenon then standards can only be successful to the extent that they reflect the individual's quality requirement in a specific situation at a specific time. If the standard does not change quickly enough to incorporate changing circumstances and values it will become peripheral to quality. Standards tend to reinforce a production view of quality and the status quo, which is not appropriate when an organizational (use) view of quality is needed.

Metrics, QA and standards all have a role to play in the improvement of quality. However, all three can degenerate into an 'object orientation' that concentrates on measuring and controlling the processes that generate products and services with insufficient reference to the customer ('subject-orientation').

7 THE REQUIREMENTS OF QUALITY MANAGEMENT - THE SUPPLIER VIEW

The need to assess the performance of technical artefacts in an organizational context reflects the use view of quality. The need to have a controlled and repeatable process reflects the production view of quality. But, a strictly controlled process restricts the essentially adaptive nature of information systems, which must continually respond to changes in the environment. Paradoxically, quality requires both a use and a production perspective, despite the two views often having conflicting requirements. Over and above the need to address the use and production aspects of quality, the quality management system should also be concerned with expectation and image management, and resource performance assessment.
Expectations and Image Management

If quality is concerned with the expectations of customers, quality can be achieved not only by monitoring how customers use the product but by active management of customer expectations. Setting expectations at an unrealistic level means that the supplier can only be perceived by the customer as failing to deliver a quality product. Setting the expectations at an appropriate level, one that can be achieved, or exceeded, gives the supplier the opportunity of supplying a quality product. The fact that the product is unchanged is not relevant, although, should the supplier attempt to lower expectations to an unacceptable point, then competitors can be expected to enter the market with superior products. For example, if a project manager quotes four weeks to deliver a software module, but misses the deadline by 2 days, he may be perceived by the customer as failing with respect to a significant quality factor. If the same manager quotes 5 weeks and delivers 3 days early, the customer may perceive this as an instance of a good quality service. The absolute time taken has not changed, but the customers expectations have been managed. This should not be seen as a cynical exercise in manipulation - customers make business plans based on delivery dates given by suppliers and organizations are caused considerable disruption by suppliers not meeting the expectations of customers.

Linked with the management of customer expectations is image. The customer will, in part, judge the product on the attributes which are immediately visible, for example the format of the documentation, the screen design, the physical characteristics of the hardware, even the appearance of the support staff. If the objective is to improve quality as perceived by the customer then these superficial aspects of the information system cannot be considered to be peripheral and should be treated with at least as much concern as the deeper attributes such as the quality of the technical design.

Resource performance assessment

If information system development is considered as a transformation process, then the process and the product need to be supplemented by giving consideration to the inputs to the process. The inputs are the resources needed to perform the process, such as technical skills, management skills, methods, CASE, and funding. Of these inputs, perhaps the most important is the quality of the people involved in the process. Curtis [Curtis 1992] has reported Boehm's finding that [Boehm 1981]:

'a project staffed with uniformly very low rated personnel on all capability and experience factors would require 10.53 times as much effort to complete the project as would a project team with the highest rating in all the above factors'
Curtis also quotes NASA's Software Engineering Laboratory (NASA-SEL) which found differences of 20+ to 1 among individuals [Curtis 1981].

If a quality process is needed to produce quality products it is reasonable to assume that a quality process requires quality inputs (resources). It is suggested that the fourth element of quality management should be the assessment of resource performance.

In summary, the four quality elements are:

- process management (quality assurance);
- product performance assessment (marketing);
- customer expectation and image management (sales and customer service);
- resource performance assessment (supplier evaluation).

Figure 3 shows these four quality elements and identifies the dialectics of the customer/supplier notion of quality and the production/use nature of quality. The tensions between the components of a dialectic relationship should be addressed, not with the aim of eradicating one or other of the components, but rather with the objective of striking a balance between the components that is appropriate to the situation.

Quality can also be seen as comprising chains of supplier/customer relationships, some of which will be reciprocal (e.g., personnel supplying staff to system development and system development supplying an information system to personnel). Because the product of one process is often a resource for another process it is important to establish and manage relationships with both suppliers and customers. These relationships are expected to change over
time and their maintenance is a fundamental aspect of quality management [Lewis 1991; Vickers 1984].

8 A QUALITY MANAGEMENT FRAMEWORK

The quality management framework in figure 4 is a generic model that could be applied to information system development, traditional software engineering, or, as is the case in practice, a hybrid. The rectangles and straight lines represent the more mechanistic and 'objective' aspects of quality management, as described by the four elements of quality. The irregular forms represent the subjective aspects of quality, which includes the definition of a relevant notion of quality.

Figure 4
The Quality Management Framework
In the quality management framework customers are seen as having requirements that are specified and re-specified (a reflective dialogue between the supplier and the customer [Schön 1991]). The requirements are subject to constraints imposed by other affected parties, such as operators, executive management, other departments within the organization, consumer groups and regulatory bodies. The performance of the products and services supplied are assessed for conformance to specification and to identify design/redesign opportunities. It is not necessary for requirements to change for redesign to take place - redesign is concerned with improving the way existing requirements are implemented.

Defining a notion of quality is not a once-off exercise - the quality requirement changes over time as the environment in which the organization is operating changes and as organizational values change. The dotted lines in figure 4 are an application of double-loop learning [Argyris & Schön 1978, Argyris 1982] which emphasises the need to question assumptions made in the planning process. The quality management framework learns in a single loop by monitoring performance against quality criteria and learns in a second loop by continuously investigating what constitutes a relevant notion of quality.

This framework is proposed as a means of identifying how and where quality is lost. In different situations the different quality elements will be perceived as having different degrees of importance. It is argued that the organization should at least be aware of the potential dangers of completely ignoring individual elements of the quality framework. For example, if there is no process established to review over time what constitutes a relevant notion of quality, or if there is no attempt to manage customer expectations, or no assessment of supplier performance, then it is likely that quality will be lost.

9 CONCLUSIONS

Before the notion of software quality can be defined and managed the customer and the product must be identified. Quality exists in the relationship between the customer and the product and not solely in the product. Developers' perceptions of the product of system development, and hence their view of what constitutes an appropriate process for making that product, can be understood by questioning the assumptions (the paradigm) developers adopt in building systems. Developers may perceive the product as ranging from a technical artefact, with an emphasis on the production view of the nature of quality, through to an information system where the developer is concerned with how the technical artefact performs in the organizational setting (the use view of the nature of quality).

Although it is the foundation upon which all other efforts to improve quality rest, the activity of identifying the customers and affected parties who are involved in developing a relevant notion of quality has been given insufficient attention. However, we do not accept that quality can be reduced to 'whatever the customer wants'. In arriving at a relevant notion of quality it may be necessary to take into account the views of a number of affected
Software Quality Management

parties, one of which is the system development group which has a professional interest in and a shared responsibility for the quality of the resultant information system. The supplier should also be aware of the different ways in which quality can be improved, or lost, and should consider not only process management but also expectation management, resource assessment and, product performance. Metrics, QA, and standards can only improve quality in a non-random way once a relevant notion of quality has been established.

Two significant dialectics have been identified, customer/supplier notion of quality and the production/use nature of quality. Rather than expend effort in attempting to eradicate the tensions in these relationships, the developer should aim instead to play off the components one against the other. To focus on one component to the virtual exclusion of the other is likely to result in a loss of quality - a dialectic view of quality requires that the production/use and the customer/supplier relationships are explicitly considered and an appropriate balance found. Unless a dialectic approach is taken the result is likely to be software products with 'quality' that is not relevant to specific users, and information systems that are full of good intentions but lacking in the technical robustness to make them effective.

We see no simple and general solution to the customer/supplier notion of quality and the production/use nature of quality. In recognizing that there is no once and forever solution to quality the developer will see these dialectic relationships as subject to continual change and be open to new ways of looking at quality as new descriptions and new circumstances emerge. For example, if the Object-Oriented paradigm allows organizations to take a component approach to system development by buying artefacts rather than manufacturing them [Vidgen & Wood-Harper 1992a], then we expect to see software engineering become more rigorous and concentrated within the hands of a smaller number of truly expert designers who will be concerned with a production view of quality. This would free development staff in organizations to concentrate on a use view of quality and how those technical artefacts can be used to make their organizations more effective. This is not to say that quality problems will be cured by Object-Orientation, merely that one dialectic (the production/use view of quality) may become less relevant in some situations, only to be replaced by another, as yet unthought of, dialectical relationship. It is in this sense that quality improvement is a never ending process.
REFERENCES


BRAA K and ØGRIM L (1992), *Quality Assurance - An Assurance of Quality?*, proceedings of the 15th International systems Research seminar in Scandinavia IRIS), 9 - 12 August, University of Oslo


CHECKLAND P and SCHOLES J (1990), *Soft Systems Methodology in Action*, Wiley, Chichester


DAHLBOM B and MATHIASSEN L (1991), *Struggling with Quality, the Philosophy of Developing Computer Systems*, Gothenburg Studies in Information Systems, Sweden


GITLOW H, GITLOW S, OPPENHEIM A and OPPENHEIM R (1989), *Tools and Methods for the Improvement of Quality*, Irwin, USA


METKIT, Esprit II, Project 2384


VIDGEN RT, WOOD-HARPER AT and WOOD JRG (1992b), *Towards Multiple Perspectives of Information Systems Quality*, proceedings of the 15th International systems Research seminar in Scandinavia IRIS), 9 - 12 August, University of Oslo