

# A pragmatic approach to software quality for information systems in small and medium sized organisation

O. Hopker

*Faculty of Computing, Swansea Institute of Higher Education, Mount Pleasant, Swansea, SA1 6ED, Wales*

## Abstract

Quality can be defined as fitness for purpose. This paper explores the criteria for software quality in small and medium-sized organisations where the needs and expectations from information systems, and the environment in which the systems are developed, may be quite different to that found in the larger organisations. The paper concludes that in these circumstances, the more customary features associated with quality issues may be inappropriate. It goes on to develop guidelines for an approach to systems development which promote quality by emphasising steps which can be most easily and cheaply incorporated alongside current practices.

## Introduction

Much effort has been expended in devising means to ensure quality of large, complex or safety-critical systems developed in large or specialised organisations. With the increasingly competitive and turbulent environment, all organisations are becoming more dependent on information and information systems for healthy survival. Quality issues in small and medium-sized organisations, and where software may be less specialised, have received less attention.

Research carried out in Wales by Hopker [5,6], where the profile of organisations is characterised by a high proportion of small and medium-sized organisations and (apparently outlying) branches of multi-national or national organisations, has shown that of the 70 postal survey respondents who develop software in-house, 42 (60%) had less than 10 programming/systems analysis staff. A high proportion of these organisations indicated however, that they

## 76 Software Quality Management

expect their information systems to provide them with information for competitive advantage as well as to support administration and daily operations and aid management decision-making. Their dependence on these systems though apparently less direct or immediate than that of larger organisations, is nevertheless increasing. These aspirations imply a need for quality software and systems.

In the large or specialised organisations, the ever-present dependence on the information systems for a wide range of activities from the most mundane of everyday tasks to corporate decision-making necessitates software which can be relied upon to continue operating under a variety of conditions; in safety-critical systems, the over-riding factor in defining quality is likely to be the rate of system failure. Such factors, coupled with the size and volume of projects under development and the remoteness of the users, dictates that the quality system and quality assurance processes and procedures must be comprehensive. The presence of a sizable in-house systems development team whilst bringing its own quality problems, provides the expertise and scope to apply a formal quality system rigorously. Software houses needing to meet the specified requirements of their customers and promote their reputation, are in much the same position.

The criteria for software quality in small and medium-sized organisations and the means to achieve quality may be quite different to that found elsewhere.

### The Constraints

Following up the postal survey, interviews with systems managers in ten small or medium organisations identified the typical development context as that of restricted resources, short time-scales and limited staff expertise. The smaller organisation has less computing and systems staff resources not only in numbers, but also in terms of specialisation, expertise, training opportunities and role separation. It may be at an earlier stage of maturity with respect to development and use of computer-based systems, reflected in less competent user skills from requirements definition onwards.

Although historical influences and traditional methods were found to dominate approaches to system development, there is frequently not time to undertake all or even most stages of a life-cycle development model. Questionnaire responses had suggested that traditional, prototyping and structured methods were used by 75%, 43% and 33% of organisations respectively for system development. The subsequent interviews showed a less clear-cut situation, and indeed it seems that a 'Pick-n-Mix' methodology is common when selecting tools and techniques [6]. Structured methodologies are used selectively - the manager selecting those tools and techniques

considered useful or appropriate to the project in hand. Prototyping is popular because it facilitates speedy development, addressing the constraint of short time-scales. Documentation, frequently a casualty of the need for speed of development, is further reduced by the application of prototyping. Systems managers were generally aware of the short-comings of the approaches the constraints forced them to adopt; some commented that they would "like to do things better" or that greater emphasis placed at the front-end of development when possible, seemed to produce better results.

## Quality in the Context of SMEs

Before attempts can be made to improve software quality in SMEs, it is necessary to determine what quality means in this context. There have been many different definitions of quality in respect of information systems and software engineering [10], but nearly all of them include reference to use or user thus giving rise to a generic definition, 'fitness for use' or 'fitness for purpose'. Musa [8] suggests that a shift will take place in software engineering from developer-orientated quality measures (faults) to user-orientated ones (failures) and that users are increasingly demanding information systems with an optimum balance of cost, timely availability of new features and quality level as experienced in operation. This observation, which draws attention to aspects of software quality other than 'correctness', is in accord with the constraints of budget and time-scales identified in practice in the Welsh survey. An important consideration here is how the software will be used and which functions are most frequently used; it is the failure rate (or perhaps we should say success rate) of these that will be identified as the practical level of quality.

The wider perspective of quality of information (rather than the software which produces it) may be more pertinent in the context of the declared intentions of organisations for their information systems, that is, to provide information for competitive advantage as well as to support administration and daily operations and aid management decision-making. Van der Pijl [13] identified this as the degree to which the information supports the goals and targets of the organisation in which it is used, with a quality assessment made by comparing the required and available information (deduced from the organisation's environment, goals, targets, structure and culture) and the required and actually installed information systems.

The cost of non-conformance to quality requirements is recognized by Voss [14] as made up of the cost of:

- fixing problems during testing,
- changing functionality to meet misunderstood requirements,
- carrying out maintenance, for example, fixing problems in released systems,

## 78 Software Quality Management

- supporting users who have inadequate documentation,
  - trying to understand how the system works so that problems can be fixed.
- An important component that can be added to this list is that of opportunity cost. Voss reports that the biggest problems are encountered in the specification of requirements, where many users find it difficult to express what they want in clear, unambiguous terms. He advocates fostering a quality culture in which there is wide staff involvement and setting up a documented quality system. The investigation in Wales collected no data in regard to the cost of quality, but showed that prototyping is seen as an appropriate means to establish user requirements.

### Improving quality

Project management, quality assurance and software engineering standards, methods and tools are deployed to produce good software. Research efforts to improve quality have focused on the processes and products of software development. Rijsenbrij and Bauer [11] outlined a modular quality system made up of conditions for quality, a project specific quality system and quality audits. Techniques suggested by IEEE P1061 [7] are: testing, simulation and formal proofs (for product assurance), management review (for process assurance), and technical review, software inspection, walkthrough and audit (for process and product assurance). Product assurance is more appropriate for ensuring quantifiable factors such as correctness, whereas process assurance may be more appropriate for factors such as usability, flexibility and maintainability which are more difficult to quantify. For complex software and safety-critical systems, much effort is expended on verification and validation of software, and calculating the 'fault' rate of the code. For non-critical systems however the requirement may be rather different.

Smith [12] outlined an approach to software quality, which he defined as "meeting the user's requirements first time"; this meets the aims of the organisations investigated in Wales. The approach integrates cultural issues with the introduction of techniques, tools and methods to promote quality. It includes:

- management commitment,
- organising for quality,
- measuring quality,
- communication,
- provision of tools,
- recognition,
- continuous improvement.

The use of fourth generation languages, formal methods, structured methods (e.g. SSADM) and object-orientated paradigms have each in turn been extolled by their proponents as enhancing productivity and quality.

Advocacy of structured methodologies has pointed to the increased emphasis on the early stages of development fostering a more rigorous definition of requirements, intermediate products during development enabling quality control and verification, a sound system structure which can be evaluated, and independence of modules and improved documentation which aid maintenance. For example, Avison et al [2] identified information system development methodologies as providing rigorous ways to capture user requirements and structured approaches as facilitating quality improvement by their functional decomposition of a complex problem into manageable units. However, they reported problems associated with introducing development methodologies and the limitation of some such methodologies with respect to verifying correct communication of ideas. Alavi and Wetherbe [1] demonstrated that incorporation of prototyping into a methodology could lead to improved quality by using the methodologies to capture initial requirements and fourth generation languages to maintain currency of those requirements. Dekleva [3] has found some evidence that systems developed using modern methodologies are more reliable than those developed by the traditional approach. His study showed that time spent on emergency error correction and the number of system failures decreased significantly with the application of a modern methodology. He also showed that system size and complexity influenced maintenance problems.

Developing a framework to guide development for individual information systems within an organisation, Ward et al [15] suggested that the method, tools and techniques should suit the nature of the specific information system's role with regard to the organisation's activities. Thus they recommended that the purchase of standard packages is appropriate for 'support' systems, applications which improve management and performance but are not critical to the business. 'High potential' systems, applications which may be of future strategic importance, require low-cost rapid development with high user involvement; characteristics of a prototyping approach. On the otherhand, 'factory' systems, applications which are critical to sustaining existing business, should be robust, reliable and easily maintained, and thus should utilise lifecycle methodologies.

Controlling process alone is unlikely to result in quality software products; product assessment is also necessary. Neil [9] identified two approaches to software quality, organisational (control and structure), and technical (notation and methods), but he and Grady [4] argue for measurement and assessment in order to evaluate improvement. Neil described reliability modelling, checklist assessment, textual measurement, structural measurement, and test coverage measurement as product assessment methods which directly affect quality. Of these, reliability modelling (the collection of data on failure and usage times when testing or operating), and checklist assessment (the manual inspection to determine the presence or absence of product attributes), are achievable by the

## 80 Software Quality Management

small or medium sized organisation. Grady suggested that inspection is a technique which can be started by anyone from developer to company president and which prevents errors as well as detecting them.

### Conclusions

The issue of software quality management in small and medium enterprises should, as elsewhere, be addressed by attention to process and product. However, there will be constraints on the achievement of quality in both respects imposed by budget, timescale and staff expertise. It is necessary to bear in mind these constraints when suggesting any changes or additions to the information systems development processes. Attractive though it may be to argue that the apparent ad hoc selection of development processes should be discouraged and that more time and money committed to improve software quality, such arguments are unhelpful if they do not reach the threshold of feasibility.

It is important to bear in mind the concept of quality of information and its role in supporting the goals and objectives of the organisation. The definition of user requirements which meet these needs is paramount, particularly if the mistakes over the last thirty years of larger organisations producing the wrong systems are not to be repeated. This is not an easy task for the less experienced organisation.

The small numbers of development staff lead to greater reliance on individuals with consequent implications for continued development or maintenance should staff leave. They also restrict the opportunities for structuring to accommodate specific quality roles; even role separation within a project development may not be possible. Thus quality control must be incorporated into the normal development activities. Fostering a quality culture within the organisation may help to provide the space for this.

### Guidelines for quality in SMEs

By considering the various suggestions that have been made for improving quality, it is possible to draw up steps which can feasibly be undertaken by the small or medium enterprise without requiring a dramatic culture change.

These are:

- foster management commitment, quality culture, recognition of quality issues;
- communicate;
- strive for continuous improvement;
- apply an appropriate development approach: purchase software packages, prototype with 4GLs, use tools and techniques of structured methods;
- carry out a technical review;

- test the software and the whole system;
- inspect the product;
- walkthrough each major stage;
- carry out reliability modelling;
- carry out checklist assessment.

Some of these measures will already be applied to a greater or lesser extent in some if not all organisations. This is best illustrated by the use of information systems development methods. Many of the proclaimed quality benefits of using structured methodologies may not be realised; the selective use of tools and techniques may preclude the production of staged deliverables, the methodology may not be exploited to ensure user requirements are rigorously defined or to ensure a sound system structure. Prototyping, although useful for identifying user requirements, reduces the production of documentation.

Modifying or augmenting a chosen 'Pick-n-mix' methodology with procedures suggested by the list above would lead to an improvement in quality of information and long-term reduction of software development costs.

A final recommendation is that taking care to use only reputable suppliers and to ensure appropriate criteria for such aspects as reliability, documentation, flexibility, and backup support, standard packages should be purchased where possible. Provided quality products are selected, this is likely to result in a cost saving and free resources allowing more time and hopefully greater attention to quality for other systems.

## Final thoughts

It should be possible for the small or medium sized enterprise to achieve improved software quality by incorporating some simple procedures into its normal information systems development practices. Such steps would produce for the organisation, long-term benefits with regard to its competitive position.

## References

1. Alavi, M. and Wetherbe, J. C., (1991) 'Mixing prototyping and data modelling for information systems design', *IEEE Software*, 8(3), 86-91.
2. Avison, D. E., Shah, H. U., and Wilson, D. N., (1994) 'Software quality standards in practice: the limitations of using ISO-9001 to support software development', *Software Quality Journal*, 3, June 1994.
3. Dekleva, S.M., (1992) 'The Influence of the Information Systems Approach on Maintenance', *MIS Quarterly*, 6(3), September 1992.

## 82 Software Quality Management

4. Grady, R.B., 'Practical results from measuring software quality', CACM, 36(11), November 1993.
5. Hopker, O. M., (1993), 'Systems Methodologies for Change - Findings of a Welsh Survey', Proc. 3rd BIT Conference, Manchester.
6. Hopker, O.M., (1994) 'Evaluation and Choice of Information Systems Methodologies - A Welsh Perspective', Proceedings of 2nd Conference on Information Systems Methodologies, Edinburgh, 1994.
7. IEEE P1061, (1988), Standard for a software quality metrics methodology.
8. Musa, J.D., (1989) 'Faults, Failures, and a Metrics Revolution', IEEE Software, 6(2), March 1989.
9. Neil, M., (1994) 'Measurement as an alternative to bureaucracy for the achievement of software quality', Software Quality Journal 3, June 1994.
10. Rowe, J. M. and Neal, R. D., (1993) 'TQM in system development', J. of System Management, 44(5), May 1993.
11. Rijsenbrij, D. B. and Bauer, A. H., (1993) 'A quality system for a software house', J. Systems Software 23(3).
12. Smith, A., (1991) 'Quality - an integrated approach', UNICOM - Applied Information Technology Reports 6, Software Quality and Reliability 'Tools and Methods'.
13. Van der Pijl, G. J., (1994) 'Measuring the strategic dimensions of the quality of information', J. of Strategic Information Systems, 3(3), March 1994.
14. Voss, A., (1991) 'The real importance of software quality and the role of quality systems', UNICOM - Applied Information Technology Reports 6, Software Quality and Reliability 'Tools and Methods'.
15. Ward J., Griffiths P., and Whitmore P. (1989), 'Strategic Planning for Information Systems', Wiley.