

Quality management procedures in the development of scientific numerical software

M.F. Niven, D.F. Cowell & M.K. Patel

Centre for Numerical Modelling and Process Analysis, University of Greenwich, London, SE18 6PF, UK

ABSTRACT

This paper reviews the current situation relating to the use of software quality management systems (QMS) within the numerical software sector of the British software industry. The special features and constraints that characterise numerical programs are described together with a portrait of a typical company in this field.

A survey of numeric software developers has been conducted to gauge typical levels of quality awareness and QMS implementation compared to other sectors of the software industry. Based on the survey findings, perceived problems and misconceptions are addressed and possible ways forward are discussed. Wider issues such as product liability and secondary safety aspects are also raised, together with what may happen to developers who choose to ignore the quality initiatives that are already under way in other sectors of the software industry.

INTRODUCTION

Software quality management systems are currently receiving a lot of attention in industry, academia and in the press. This is not surprising as the software industry has had a long and undistinguished record of producing badly designed, over-priced, late and bug-ridden products. The term 'software crisis' was coined many years ago to identify this problem and it was thought that emerging design methodologies would dispel the prevailing ills and improve productivity dramatically. However, most of the problems have remained. The emergence of cheaper, ever more powerful computers has lead to a rapid increase in the demands for large, complex software packages to exploit this power and has brought the problems of lack of quality back into sharp focus.

Quality management systems (QMS) are seen as one possibility of enforcing the same type of discipline in the software development process as has been

126 Software Quality Management

practised in other branches of engineering for decades. However, the exact nature of software quality and how to measure it continues to generate much heated discussion and many research projects.

In 1987 the British Standards Institution (BSI) published BS 5750 on quality systems for design, development, production, installation and servicing. It was geared more towards conventional engineering and provided little guidance on the specific needs of software development. However, since then this guidance has been forthcoming (viz. the Department of Trade and Industry's TickIT initiative) and BS 5750 and its international counterpart ISO 9001/9002 have become increasingly the standard by which all types of quality systems should be assessed.

The fact that a QMS has been produced to comply with a standard does not guarantee that product quality will be satisfactory, or improve, and BS 5750 is currently subject to criticism for being too prescriptive, too expensive and too cumbersome. This can be particularly true for smaller companies but is something that can be minimised by implementation of a QMS appropriate to the size and style of business. However, the fact remains that quality systems and procedures are being demanded more and more by customers and BS 5750 is the best form of recognition that a QMS is itself of acceptable quality.

Professional software developers are increasingly implementing quality management systems to provide an overall framework for the definition and control of the development and maintenance process. These QMSs may represent the formal part of a Total Quality Management (TQM) philosophy and QMS certification to BS 5750 is often considered an important goal in the implementation of quality improvement. However, our findings indicate that the level of quality awareness and commitment varies significantly between different sectors of the British software industry, and that developers of numerical software are amongst the least active.

Numerical software is defined in this context as programs and packages that perform significant numeric mathematical processing on user data to produce quantifiable results. Examples of such software occur in the fields of Computational Fluid Dynamics (CFD) modelling of liquid and gas flow, Finite Element (FE) analyses of structures, and many others. These packages can be invaluable tools in scientific and engineering projects and confidence in their operation is very important. Hence, the low level of concern for quality in this area is worrying.

COMPOSITION OF A TYPICAL NUMERICAL SOFTWARE COMPANY

Most companies or organisations that specialise in scientific numerical software are of small size, typically 25 - 50 employees at one main site, with a turnover of around £5 million per annum. A number of small sales offices are supported in other countries. Often the business has grown out of academic research into mathematical modelling and algorithms, where the original developers of a numerical software package have left the research environment to concentrate on improving and marketing the product. This is quite likely the only product that the company supplies. The founder members have recruited further mathematics and computing graduates to develop the software as sales have increased and customers have asked for improvements and new capabilities. Sometimes the core software of the product was written many years ago to run on mainframe computers with batch processing and has been modified to allow better user interaction and to run on modern workstations and PCs.

The typical numerical software company is thus quite small in relation to those in many other sectors of the industry. It is also characterised by being made up of a large proportion of 'boffins' capable of researching and applying intellectually demanding modelling and storage techniques to upgrade the product's performance. Herein lies both the strength and the weakness of the organisation. The calibre of personnel implies that at least some of the programmers must have a good insight into the package's underlying algorithms and basic method of operation. However their original academic environment may not have emphasised the need for full adherence to a structured life-cycle.

QMS TAKE-UP IN THE NUMERICAL SOFTWARE SECTOR

A small survey was carried out by contacting a number of companies and organisations that develop software for in-house use or for external customers, to ascertain whether they operated a quality management system, and whether it was certified to ISO 9001. This exercise concentrated on the professional 'bespoke' software producer rather than the mass-market end of the industry where 'shrink-wrapped' software packages are the norm. No attempt was made to survey the various market sectors in depth or obtain quantitative numbers of companies using a QMS. However, it was hoped to get a qualitative impression of differences between the various software sectors.

The survey revealed a noticeable variation in attitudes to quality management between different sectors of the software industry. By far the highest application of a QMS occurred in companies that were involved in military or

128 Software Quality Management

defence contracts. This is not surprising since the UK Ministry of Defence (MoD) has demanded for many years that all suppliers be approved to the NATO quality standards AQAP1 and AQAP13 for hardware and software respectively. These NATO standards have now been replaced generally by BS 5750/ISO 9000 in current MoD procurement and hence the defence industry has necessarily updated their QMSs to obtain these approvals.

The IT sector appears to come next in implementing ISO 9000 compatible quality systems. Most of the leading software houses in this field have compliant quality management systems in operation in most of their divisions, with some actually certified to the above standard. They have recognised the future advantages in being amongst the first to implement formal quality systems both in terms of improved product/service quality, and for marketing reasons. Smaller suppliers of financial and business software solutions have been slower to implement formal quality procedures, possibly because of short term costs, and risk being overtaken by those companies that have taken quality on board.

The numerical software sector appears to be one of the last to appreciate the need for explicit quality management. Very few developers admit to operating proper quality assurance procedures, or intend to implement them in the near future. This is disappointing as many scientific and engineering software packages have secondary safety implications in their normal usage and could, from experience, benefit from improved operation. However, a number of the major CAE vendors have put much more effort into quality matters, probably as a direct result of customer disenchantment with previous generations of CAE tools.

PROBLEMS 'UNIQUE' TO SCIENTIFIC NUMERICAL SOFTWARE

When questioned on software quality control and management, the developer(s) of a scientific package may cite a number of 'unique' problems peculiar to this particular type of software that militates against the sensible implementation of a certified QMS. Some of these obstacles are listed as follows:

- a) The main computational part of the code is many years old and made up of various dialects of FORTRAN.
- b) The original developers have left and did not produce proper documentation. Re-engineering to better standards would be too difficult and costly.
- c) Modules have to be tightly coupled to optimise speed.

- d) Performance and storage requirements are so critical that all other factors are subservient to algorithmic efficiency.
- e) It is impossible to formally prove the validity of some of the algorithms. Convergence criteria can only be shown to work in practice on given data sets.
- f) Hardware dependencies necessitate different implementations on different hardware platforms to maintain accuracy.
- g) There is no point introducing a full structured QMS on a product that contains such a high percentage of elderly code.

A number of 'standard' complaints may also be raised to back up the special pleading stated above:

- a) People are too busy fixing bugs and introducing improvements to have time for a QMS.
- b) A QMS, and particularly its certification, are too costly for a small company. The customer would not be willing to pay extra for quality.
- c) ISO 9000 series standards are too procedural, prescriptive and bureaucratic and do not necessarily result in a better quality product or cost savings.

Looking at these comments objectively, we see that most of them are not really unique to this branch of software development at all. Many of the same constraints, for example, exist in real-time software and data processing programs but do not inhibit the application of quality management. Similarly, small company size and certification cost considerations are nothing new in terms of uniqueness. What is apparent is that a number of basic misconceptions exist in the applicability and usefulness of a quality system.

The first error occurs in the assumption that a QMS is not relevant where a high proportion of the total code is elderly. Companies do not seem to realise that a QMS can be implemented and certified to ISO 9001 purely for software maintenance and for future product development. At least then all future changes to old code would be handled in a properly controlled manner and the overall quality of the software package would gradually improve.

Secondly, the fact that an algorithm or convergence criterion may not be formally provable does not imply that the software implementation should be left to chance. In many ways thorough testing strategies become even more important and these can be aided by various software tools.

130 Software Quality Management

Regarding the general complaints, the BS/ISO standards quote minimum requirements for an acceptable QMS but do not dictate exactly how this is to be implemented. This can be both an advantage and a drawback. It means that a sensible QMS can be tailored to a given type and size of organisation and its style of working without excessive overheads, but it also implies that the organisation must be smart enough to produce the 'right' QMS. A quality system should be more than just a set of procedures; it should involve everybody and instill a 'quality culture' in an organisation. Product quality will not improve dramatically purely by paying lip service to the control systems. Further discussion on setting up a workable QMS is given below.

THE COST OF QUALITY

An organisation spends money on quality, or the lack of it, whether it wants to or not. Either resources are consumed in rectifying non-conformances in the product, or they are spent on QMS and quality improvement techniques in an attempt to avoid non-conformances. Hence a usable definition of the Cost of Quality (COQ) might be that it is the total of prevention costs, appraisal costs, failure costs (rework, retest, planning, time delays) and loss of opportunity costs (custom, goodwill, capacity required for maintenance) involved in the development of a product. Note that these costs involve money, rework effort and loss of future income.

It is always difficult to get an accurate figure for the costs involved in correcting software deficiencies since a company that lacks adequate quality management is least likely to be able to quantify the cost of non-conformance. Similarly, customers are often unwilling to admit to the consequences of their procurement shortcomings. However, several studies [1,2,3] have suggested that, so long as a QMS can significantly reduce the number of defects and omissions in a delivered product by discovering problems at an early stage or improving maintainability, it should prove to be cost effective even on these narrow grounds. This is without considering the effects of user satisfaction on future sales, and contractual wrangles due to faulty products, on an organisation's financial viability.

The cost of initiating a QMS depends upon whether any documented control system already exists within an organisation. The worst case scenario occurs when all software development takes place on a totally ad hoc basis. Also, the extent of the documentation will vary between different organisations, partly dependent on their size and culture and partly on reliability and safety requirements that the products have to meet. However it is possible to suggest minimum levels that must always be attained if a QMS is to serve any useful purpose.

Discussions with a number of software consultants suggest that a small-sized company will need to invest up to 1 person year of effort spread over 1 - 2 years duration to build a QMS from scratch. This will involve both quality assurance (where this function exists) and personnel from other disciplines, since the people who are to use the various procedures should always be involved in their production. Additionally, a new employee may be required to produce, run and maintain the system and provide QA support if this does not currently exist. This does mean that capital and manpower must be committed by senior management at the start of the exercise in the belief that longer term cost savings and improved products will accrue.

A small firm is unlikely to need or afford a big bureaucratic quality system. Company size is such that communication between job functions should be straightforward and no single development project should involve large teams of programmers. Hence a QMS should be crafted to set down how activities are carried out and to provide the minimum of procedures and controls.

If a system can be established without the need to employ more staff to run it then overall setting-up costs may be around the region of £30,000 - £50,000. This still seems a large price to pay for future gains but can often be 'lost' into overall operating overheads if the QMS is developed on a part-time basis by employees who do not have to give up their usual day-to-day functions. This situation is hardly ideal but may prove good enough to get the ball rolling. Note that QMS running costs have not been mentioned as it is tacitly assumed that quality improvements in the company will cover the operating and maintenance costs of the quality system.

One aspect that has not yet been covered is the likely lack of in-house knowledge on how to set up a QMS. If this is the case, then a few days help from an experienced consultant may be of good value; even at a rate of around £700 per day. The most valuable times for a consultancy visit are at the very beginning of the exercise and then later to assess the standard of QMS documentation generated. A number of government schemes operate to help small businesses, and a company should always check first to see whether financial or consultancy support is available at the time. A good starting point is by contacting the local Training and Enterprise Council (TEC).

Once a QMS is believed to be operating satisfactorily, the company may choose to apply for ISO 9001 certification of its quality system (via the TickIT route when the primary end product is software). Again costs will vary but for an organisation of less than 50 people, situated at a single site, with an operational QMS that requires very little change to be fully compliant with ISO 9001, the total outlay should be circa £6,000 with 10 - 20 person days of internal effort. Six monthly reviews would cost about £800 and 3 person days each.

132 Software Quality Management

SETTING UP A QMS IN A SMALL COMPANY

Introducing a quality management system into any organisation is a difficult and time-consuming exercise requiring significant cooperation and support from all quarters. Whether the end product is software or not is largely irrelevant to the actual mechanics of setting up such a system. A more detailed discussion of all aspects can be found in a recent DTI report [4], but the following comments bear repeating.

Prerequisites

It is always necessary to determine whether the right conditions exist within an organisation before proceeding. The main prerequisites are as follows:

- Management commitment at all levels
- The promise of funding
- Agreement to QA independence.

Unless all three are satisfied no further action should be taken.

Initial Activities

Before embarking on implementing a QMS, a thorough review of the current status of the organisation should be carried out. If nobody internally considers themselves competent to perform such a task then a consultant experienced in this field should be employed. The review should aim to establish

- the organisation's characteristics and ways of working
- the global level of quality to be adopted
- what existing standards/working practices there are
- whether any form of QA is already in place.

The results of this review should be presented to management in a report, together with the proposed changes and a costed implementation plan indicating costs versus benefits, timescales and pay-back period. It would also be desirable to establish at this point whether certification of the QMS is a long term goal.

Full backing must be sought from the highest levels of management and a commitment for adequate funding and resources obtained before proceeding to implementation. Sometimes a QMS may be only part of an overall quality initiative such as TQM.

Implementation

A QMS should be developed in-house so that the users can feel that they have some ownership and control over relevant procedures. Trying to buy in a ready made quality system is not sensible. However, initial help and advice from external consultants can be useful.

A hierarchical approach to QMS documentation is usually the most logical for ease of comprehension and maintainability. This implies a top level manual or report briefly describing the major software lifecycle processes and associated support activities, together with references to lower level implementation documents and standards. In this way document interaction can be minimised and maintenance simplified. Documentation should be scaled to reflect the size of the company and the nature of the work. Too large a system is just as bad as no system at all.

The original implementation plan should have covered all major activities, durations, costs and resources. It should also have discussed the need to document specific standards, produce training material and develop schemes for the introduction of new QA functions. On this basis it is then possible to move to the implementation phase of the QMS.

The high level view of events might then be something along the following lines:

- Select the overall management standard (such as ISO 9001) to which the QMS will be tailored.
- Set up a small team with experience of different parts of the organisation to coordinate the QMS's introduction.
- Produce a draft top-level quality document (the Quality Manual) covering all major aspects of product development and the necessary ancillary functions ie. the 'essential processes'. This document should recognise all the phases and processes in a development programme as well as covering the support activities such as purchasing, training, QA, finance and contracts.
- Get this draft document reviewed and agreed by all interested parties.



134 Software Quality Management

- If the QMS is to make use of software tools to ease day-to-day activities, make sure that they are up and running.
- Set up a pilot scheme to try out the QMS in one specific area. Note that other process standards and methods will probably need to be defined at this stage if they do not exist or are found to be deficient. Based on the results of this trial, the QMS should be modified to remove the most obvious shortcomings.
- Gradually phase in the QMS to cover the whole organisation over a period of time.
- Regularly review the operation of the QMS via audits, metrics and user and customer feedback and also ensure that any changes in the organisation and its environment are properly reflected.
- Go for independent certification of the QMS if desired. A company whose primary product is software should seek ISO 9001 QMS certification via the TickIT route.
- Continue to develop and improve the QMS indefinitely in the light of new methodologies, tools and ways of working.

OTHER CONSIDERATIONS

Product Liability

Product liability legislation has come into force in many countries over the last decade or so, spearheaded by laws passed in the USA. It is dangerous to generalise in this field but it is possible to state that these laws normally take the form of 'consumer protection' acts and differentiate between the protection afforded to an individual customer and to a similar corporate purchaser. Strict product liability is commonplace in the USA (the 'Lemon Laws' etc.) and this appears to be the route that the EC will increasingly take. The implication is that a producer of goods cannot hide behind exclusion clauses if a faulty product causes harm to a consumer, and that it is the responsibility of the manufacturer to show that the design or product was not dangerous or that the problem could not have been foreseen.

In the UK the Consumer Protection Act, which came into force in 1988, could have significant consequences for the software industry. Under its provisions, software producers might be held liable for any injury, death or damage to personal property resulting from faulty software used to develop or built into a product. Note that liability extends to culpable individuals within an

organisation as well as the corporate body. Use of a certified QMS in the software development process might provide evidence for a defence of 'good practice'.

Safety

Software that is produced for corporate customers and whose operation could have an impact on the safety of individuals or processes can have implications for the developer in just the same way as product liability. Here it is worth distinguishing between a program that has a primary safety role and one with a secondary impact.

Primary safety critical software is embedded in a system used to monitor or control some other process. An error in the program's operation could result directly in human or environmental harm. Secondary safety critical software is represented by instances where program malfunction could indirectly result in injury or damage. An example of primary safety critical software is the flight control program in a modern aircraft and secondary safety critical software is typified by a computer aided design package whose malfunction might result in a potentially hazardous design flaw in the final product.

The range of measures employed in the development of software with significant safety implications is outside the scope of this paper. The degree of hazard analysis and use of specialized design techniques obviously depends on the level of perceived danger due to malfunctions. It is, however, often difficult to decide where the boundaries of secondary safety lie. For instance, is a hospital out-patient scheduling program that might once fail to set up a consultation for a patient in the secondary safety category?

It is always arguable that complete reliance should never be placed on one piece of hardware or software. The bridge designer should always check the results of his computer aided stress calculations to verify adequate margins. Hence the suppliers of Finite Element packages and the like place exclusion clauses in the user licence agreement to limit the use of a package to non-safety related activities. However we all know that an FE program will be used at some point for just such an activity and it is difficult to be sure that the exclusion clauses will be found to be legally operative. No test cases have yet been fought in a court of law and there is currently much discussion as to the validity of catch-all licence exclusion clauses.

136 Software Quality Management

Tools for Software Support

A 1988 DTI report [2] indicates that a quality system can be more economic to implement and run if good software development tool support is employed within an organisation. There are many development tools on the market covering all phases of the software lifecycle, from specification through to maintenance. Such tools have a bearing on the quality of the delivered product. The more automated checking a support tool can provide, the better the consistency and reliability of the final code should be. For instance, a good high level language development environment should certainly permit rapid, thorough debugging, and it might also collect certain useful metrics such as code length and complexity.

Project support functions should certainly make use of tools where possible. The activities of project planning and estimating, text/documentation preparation, configuration management and quality assurance can all benefit from some level of computer support.

Total Quality Management and Crosby Quality Process

A good QMS will encompass personnel training needs and quality improvement feedback schemes, and should be linked to 'employee empowerment' management concepts such as Total Quality Management [5] or the Crosby quality education system [6] where these exist. In this way a quality culture can be built up in an organisation and be seen as a way of life rather than a set of procedures to which people only pay lip-service.

A QMS resides within the TQM philosophy but is only one part of the overall concept which aims to instill a complete quality culture in an organisation and to get people to care about what they do and to think of ways of improving these activities.

This is a noble aim but opinion is divided on just how much TQM/Crosby actually improves an organisation's status and performance. Any concept that works by changing people's attitudes is fraught with difficulty and success does appear to be heavily dependent on the commitment of management to seeing the process through. A lot of money, training and effort have to be put in right at the beginning of such an exercise before significant gains can be seen. As overall timescales are indefinite, it is all too easy for employees to become disillusioned if they believe that nothing is in it for them. This is especially true if management appear to pay only lip-service to TQM whenever conflicts arise with other business aims.

Since the United Kingdom is renowned for promoting short-term expediency over long-term gain, it is not totally surprising that TQM is often hyped but does not always deliver its promise. However, TQM as a concept should be encouraged and it is significant that the DTI envisages the implementation of a QMS to be done very much within a Total Quality environment [4]. Furthermore, the DoE is currently sponsoring an initiative known as 'Investors in People' which is effectively an implementation of the TQM philosophy in another guise [5].

CONCLUSIONS

Customers are becoming better educated to the costs involved in poor quality software, and are demanding that suppliers meet published specifications and deliver reliable software packages that are both user friendly and engender confidence in their operation. This increase in customer sophistication is most likely to continue and will force suppliers to gain proper control over all their development and production processes. A scientific numeric software suite can often be best described as 'user mean', involving arbitrary commands that have to be painfully learned and a graphical user interface (GUI) that appears to be an afterthought. Shortcomings such as these do not aid reliable data entry or subsequent analysis.

To a certain extent quality is dependent upon how much a customer is prepared to pay for the product. Nevertheless market forces will tend to compel suppliers to improve their product quality. Possibly only such forces are capable of this major change in the attitude of scientific numerical software developers. More than most other sectors, the personnel of these companies are likely to rail against the 'imposition' of quality systems because of their background. It is unfortunate if it takes some significant event (such as a costly legal problem) to make them change their minds. However, such an event does tend to concentrate the mind wonderfully!

Once a small core of companies realise that possessing a QMS is necessary then things will probably snowball. Similarly, certification to an international quality standard may become a necessity when exporting products to certain major economic trading blocks. These comments apply whether a certified QMS actually can be shown to improve quality or not!

Next considering possible legislation, it seems quite likely that the European trend towards strict product liability will continue. As discussed earlier, strict product liability implies that all development processes should be well controlled and independently reviewed to minimise the risk of court action against a company for supplying faulty software products. A certified QMS should be the basis of product and process assurance.



138 Software Quality Management

The constant demand is for software to **handle ever larger problems and to produce faster results**. The supplier that can **best meet these needs** is likely to increase market share. If quality considerations can help to improve the efficiency and reliability of development and maintenance activities, then a developer that implements a QMS should have a market advantage over others.

Finally it is worth suggesting that quality standards will themselves be refined and become more accessible to small companies by the application of proper measurement and evaluation techniques and hence quality management systems will be subject to evolutionary improvement accordingly.

REFERENCES

- [1] DeMarco T (1982). Controlling Software Projects. Yourdon.
- [2] Price Waterhouse (1988). Software Quality Standards: The Costs and Benefits. DTI.
- [3] Boehm B W (1981). Software Engineering Economics. Prentice Hall.
- [4] DTI (1992). A Guide to Software QMS Construction and Certification using EN 29001, Issue 2.0. DTI.
- [5] Oakland J S (1989). Total Quality Management. Butterworth-Heinemann.
- [6] Crosby P B (1979). Quality is Free: The Art of Making Quality Certain. McGraw-Hill.