Achieving Business Excellence in Software Quality Management

Michael Elliott[1], Ray Dawson[2], Janet Edwards[2]

[1]Software Quality Manager, AWE plc., Aldermaston, Reading, Berkshire. UK

[2]Department of Computer Science, Loughborough University, Loughborough, UK

Abstract

Many companies have had difficulties in achieving success with software process improvement initiatives or have had adverse experiences in implementing quality systems. With a plethora of standards available and the numerous frameworks to apply best practice, none appears to act as a panacea to guarantee fulfilment or realise a true Return-on-Investment. This paper proposes a holistic approach to software process improvement, describing a range of supporting tools and methods highlighting a true understanding of the customer base and associated cultures.

The research aim was to develop and evaluate a demonstrably effective and efficient software quality management methodology suitable for a technical company. To be effective the methodology must deliver real process improvement conformance to the best practice quality standards. To be efficient the methodology must deliver a real Return-on-Investment.

A range of case studies are described including audits, self-assessment, training, system design, marketing, and the people skills associated with a consultation process are all examined in detail. Each case study provided a further opportunity to measure and analyse the success or otherwise of that method for further refinement.

The research methodology has demonstrated its success as the data collected during these case studies show that steady improvement in implementing the software quality system has occurred year on year. This success has been validated by third party ISO 9001 assessments and has led to an enhancement in reputation. The approach has overcome cultural resistance and changed working practices. With a philosophy of customer care, consultation, and active engagement, practitioners adopt best-practice quality management principles. The cost effectiveness of this methodology means its adoption could be considered by any organisation whether large or small.

1. Introduction

This paper documents a successful methodology for improving the level of implementation of the software quality management system (SQMS) within a large and diverse technical organisation. The claim for achieving excellence can be justified by the significant improvements in the working practices used by software and IT practitioners, feedback from third party companies assessing the implementation of the SQMS, a noticeable change in culture and attitude from practitioners towards quality systems in general, and the savings as cost benefits gained from improved working practices.

The improvements in working practices are demonstrated by direct measurement of the level in implementation of the SQMS over a nine year period. The methodology improved the level from a baseline value of 34% to that of 81%. This is considered an outstanding achievement by all stakeholders and validated by reports from third party auditors on ISO 9001 assessment. These figures are the average values for each year from a number of assessments conducted on software systems throughout the Company, see Table 1.

The requirements of the SQMS had to take into account both the ISO 9001 standard, utilising the supporting guidance standard ISO 9000-3 and the Defence Standard 05-95, Quality System Requirements for the Design, Development, Supply and Maintenance of Software and the subsequent revisions

The research presented in this paper and research thesis [1] was facilitated by the first author’s role as a Software Quality Manager in the organisation central quality assurance group, with the responsibility for the SQMS definition, inline with the contractual standards. He had at his disposal the internal audit team to check, or perhaps more aptly, enforce compliance.

Of particular interest was the significant difference after the application of the methodology described in this paper that occurred in behaviour from software practitioners, the customers of the SQMS. At the initiation of the improvement project, some components of the culture could be described as “thoughtless tick-in-box compliance”. Some of the outputs from various processes mandated or documents produced in response to the SQMS requirements could lack quality. A detailed review of many of these documents would conclude that some were produced; “just to get them out of the way”, indeed this was admitted by many of the document or software system owners. From earlier research on the implementation of the SQMS [2] and the related auditing process [3], it was evident that there was a level of resentment in having to implement what were not fully understood controls. The success of the improvement programme was to provide software and IT practitioners with a good understanding of the SQMS requirements, and to encourage active engagement with these customers to deal with their concerns. As a result, more quality documents were produced that contained the output of good decision making that improved both the effectiveness and efficiencies of local working practices. It could therefore be demonstrated that implementing the requirements of the SQMS truly added value and this could be further emphasised by well maintained systems.

2. Literature Review

One of the major factors to investigate within this research is why auditing is not a popular process [4]. Culturally, they could be seen as a necessary evil. According to Wealleans [5], they are often viewed as a means to finding and recording of trivial issues, and this has blighted the perception of the auditing process [5]. In the "audit world" this is very clearly recognised, many publications focus on this issue. Yet despite this understanding, the negative perception of the process has not significantly improved, suggesting the new focus may not have been effective. It is an assertion within this research investigation that a tangible cost benefit has to be attributed to the corrected action, otherwise this activity will not necessarily actually add value.

Many standards exist to provide a framework for a software quality management system defining the organisational arrangements and the working processes. The ISO 9001 standard [6] and the Software Engineering Institute's (SEI) Capability Maturity Model Integration (CMMI) (Carnegie Mellon, 2001) are often contractually specified. The guidance (ISO 9004, 2000) to the revision of the ISO 9001 standard from 1994 to 2000 highlights eight principles for a successful management system: customer focus, leadership, involvement of people, process approach, system approach to management, continual improvement, factual approach to decision making and mutually beneficial supplier relationships. As the SEI CMMI conveys five levels of maturity for a company to be world class, the initial key process areas to obtain the first level of maturity would be the most important. These consist of: configuration management, quality assurance, sub-contract management, project tracking and oversight, project planning, and requirements management. There is a correlation between these two standards, various management processes and requirements management, but there are also some differences, suggesting disagreement on what would be a magic formula for a software quality management system. Further correlation of the importance of management processes and requirement management comes from the Standish Research Group's Chaos Report (Standish, 1994), which stated that the top eight reasons why projects are cancelled or severely fail to deliver on time, cost or performance, were: incomplete requirements, user involvement, lack of resources, unrealistic expectations, lack of senior management support, changing requirements, inadequate planning, system no longer needed. Again, five relate to requirements management, and three to what would be considered general management processes. The follow-up Standish Report (Standish, 2001), highlighted its top ten factors for project success as: executive support, user involvement, an experienced (project) manager, clear business objectives, minimized scope, standard infrastructure, firm basis requirements, formal methodology, reliable estimates, others, including: small milestones, proper planning, and competent staff and ownership. Again there is some correlation, but there are also differences.

Organisational culture does not feature highly in these standards or reports, although, as Spencer et al. [11] state competency normally encompasses motivation along with knowledge and experience. Culture is also conservative about change. Wagner [12] also suggests that motivation can be influenced by cultural factors. Indeed many excellent technical books on software quality management do not focus on culture. Horch [13] warns of the pitfalls, starting a software quality programme is doomed to failure with inadequate preparation, misused terms, lack of planning and failure to recognise the individual role of members of the organisation. Galin [14] suggests that having colleagues involved in the development and, at least, the review of company procedures, will help convince other colleagues to abide by them. The little attention to culture is perhaps surprising, indicating that the effect of culture on the adoption and implementation of a software quality management system has been under-estimated [15].

According to Sandi et al. [16] training programmes are an essential feature of organisational life. Training initiatives are widely acknowledged to be a salient feature of the competitive organisation's corporate strategy and, in times of great change, learning is the key skill [17] Employees, managers and organisations rely on training as a solution to enable issues to be resolved, yet Hale [18] reports that only 35% of UK companies have measured the effectiveness of their training and development programmes. To value and reward learning, an organisation must have a method of determining performance in learning and gaining knowledge. Morey and Frangioso [19] state that a method to assess the value of learning is to measure the performance of employees in creating items of business value from the learning. They suggest that for training to be effective it must have specific objectives and outcomes which directly lead to business benefit. It is surprising that despite heavy investment in training, organisations fail to evaluate adequately the value or success of their training programmes and many that do, do so inadequately [18]

The ultimate aim of the software engineering training was to improve the application of the software quality management system. Training and education are a key component in any improvement initiative. To further evaluate the success of this training in terms of value, the financial benefits should be considered as outlined by Philips [20].

Lee and Quazi [21] report that the recent increase in the use of self-assessment as a process improvement tool has been, in part, due to organisations setting the goal of attaining a recognised quality award. These awards include The Malcolm Balridge National Quality Award (MBNQA), and The European Quality Award (EQA). An integral part of obtaining such an award is to conduct self-assessment of organisational performance against the award criteria or framework. A key element of the self-assessment process is the question-set. Typical question-set methods are pro forma or matrix as described by Samuelson and Nilson [22].

Samuelson and Nilson point out that responses to questions need to be carefully managed as issues such as bias and inconsistency can distort analysis, though inconsistency can be minimised with a suitably large number of samples to "average-out" false variations. Waina [23] documents characteristics of the use of self-assessment. In its simplest form it can be administered in one or two hours. In this situation the disruption is low.

With the significant number and high profile of software project failures, as reported by Standish [9], a considerable amount of research has been undertaken on risk concepts to try and avoid these failures [24]. Jones [24] describes the "Common Colds" of software management:

- Excessive schedule pressure – cultural issues that requires therapy to deal with the sociological issues.
- Poor quality – technology problem with cultural aspect. Requires new methods in measurement, defect prevention and removal, plus cultural awareness of the true value of quality to the business.
- Inaccurate estimation – essentially a technological problem, but has a cultural aspect. Requires good estimating and planning tools. However measurement is needed for accuracy and this is where the cultural aspect comes in.

Jones' [24] philosophy is that the value of identifying the risks for each software application is to proactively plan for the appropriate risk mitigation action. Significantly these can be the quality assurance activities required from the adoption of quality standards that could be captured in a quality plan.

In considering the cost-benefits and value of software quality or engineering practices the effectiveness of reviews, particularly software inspections [25] are often cited. For example, Freeman and Weinberg [25] report that they cut testing costs by 50% - 80% or that they remove 80% - 95% of faults at each development stage and Gilb & Graham [26] state that they can reduce schedules by up to 25% and produce a 28 times reduction in maintenance costs and can cost only 5% - 15% of development cost. These are supported by Yourdon [27] who states that testing reviews or inspection can produce a 10 times reduction in faults reaching the testing phase. If ever a case could be made for the value of a quality assurance system, these few statements make it. The important of design information in a maintenance regime is given by Bennett et al [28] in that with a complex system, 50% to 95% of the cost needed to make a change can be taken to understand the program. The need for independent testing is highlighted by Freeman and Weinberg [25] in their research that testers find only 30% to 40% of their own faults.

As this research presents a cost-effective methodology, it is of interest to review the themes of Cost-Benefit and Return-On-Investment (ROI) in the work of Rico [29]. ROI is the quantification of the benefits received in financial terms of any investment scheme such as the introduction of a SQMS or software process improvement project. Waina [23] states that the drive to invest in these systems is to make the business more successful by producing better products, more quickly and cheaply whilst improving customer satisfaction. Significant investment is then required in process definition and documentation, training and education, as well as tools, technologies and techniques.

A powerful strategy to aid adoption of any management system is to demonstrate its cost-benefits and ROI. As the start-up costs and effort are apparent in any software process improvement or quality system initiative, a framework or method to convey benefits will surely provide an incentive to accelerate its adoption. Rico [29] analyses a number of techniques and standards in terms of training costs, project costs, life cycle benefits, benefit-to-cost ratio and ROI. The adoption of standards has additional preparation and assessment costs. The systems assessed for ROI are Software Inspections [30] Personal Software Process^{sm} [31] Team Software Process^{sm} [31], Software Capability Maturity Model (SW-CMM) [32] ISO 9001 [33] and Capability Maturity Model Integration (CMMI) [7] The conclusion from Rico's analysis is that due to the high start-up cost for the adoption of the standards SW-CMM, ISO 9001, and CMMI, they provide the lowest ROI. According to Rico [29], the highest ROI was achieved by PSP, then the software inspections.

3. Methodology

A project to improve the adoption of the SQMS was initiated at the completion of a comprehensive assessment on the level of implementation [1]. This comprehensive assessment, in fact, became the first step of a seven-step methodology for effective implementation. These seven steps are described in the remainder of this methodology section:

Step 1 : Establishing a Baseline
A sample of fifty-five software systems from a diverse range of applications, were assessed against all the requirements of the SQMS. These requirements were documented in a comprehensive audit question-set that were grouped into software quality topics. These included general management, requirements management, development, testing, use, configuration management, and disaster recovery.

A range of 0 to 4 implementation rating system was applied to each topic. This rating system could be described as an anti-dote to the culture of "tick-in-the-box" compliance as level 4 was a test that the local area valued their implementation of the SQMS. Increased levels on the rating for a topic could be achieved with an increase in the number of requirements for that topic that were complied with. This has similarities to the Software Process Improvement Capability Determination (SPICE) model of ISO Standard 15504 [34]. However, level 4 could not be achieved unless the initial actions to implement the SQMS were updated when changes occurred, so that the system retained its integrity and was maintained. This would act as a demonstration that the local area felt what had been implemented was of sufficient value to be maintained. This provided an indication that SQMS requirements were an established way of working and had become institutionalised, and not completed and then forgotten. Each system assessment attracted a pro-rata percentage of implementation that was collated to calculate a company average for the year [1]. The result came to 34% and this was a concern for senior management. Of particular concern was that neither the ISO 9001 assessments for certification nor the internal audit process had raised any major issues. It was directed that corrective action for improvement had to be conducted in a low profile manner until significant improvement could be demonstrated. A programme of assessments continued each year as can be seen in Table 1. Unfortunately, due to a temporary reassignment on to other tasks for most of the fourth year, only five assessments took place. The assessments were conducted on business critical software systems that were all well managed and demonstrated a high degree of compliance. Unfortunately due to the low sample the data is not representative.

Step 2 : Gathering Feedback from Users

In parallel with Step 1, during the assessments comments, problems and other barriers to adopting the SQMS were gathered. These issues were documented as "causes" (e.g. direct, contributory, root) for non-compliance on the assessment pro-forma. There were clear themes presented in these comments. There was certainly a significant cultural resistance to the procedures which, in many cases, stemmed back to the original ISO 9001 certification process, when many practitioners felt that their concerns on the system definition were not taken into consideration. As a result, a culture of annoyance and an assumption of lack of relevance was evident concerning the SQMS. This would present a significant challenge to improvement. From a technical viewpoint it was clear that some terminology was not understood, and termed as "quality speak". There was criticism that all the procedures had to be read thoroughly before you knew what rules to follow. This was often described as an inability to visualise the various processes. There was a lack of understanding of many software quality and engineering concepts. Best-practice requirements management, life-cycle selection, configuration management and effective reviews were not fully understood.

Step 3 : Review and Revision of Quality Procedures

To address the lack of clarity of the procedures and deal with some of the other comments or criticism of the SQMS, all the procedures were reviewed and revised. Based on the feedback from Step 2, key considerations for the revision were to deal with the lack of coherence of the current system, visualisation of processes and the provision a simple framework or introduction to the main requirements. The solution to the issue of applicability was to develop a system to grade or tailor the number of requirements based on risk and complexity. The re-drafted procedures formed the basis of a significant consultation process. The drafts were sent out for comment, and were facilitated by a number of presentations and workshops. The aims of this consultation were marketing the benefits of implementing procedures and dealing with any concerns so as to gain the buy-in from the practitioners, the users of the system. It was hoped that this would help overcome much of the resistance that had built-up over the years. These workshops and presentations certainly raised the awareness of the procedures, but it became evident that education on software engineering, quality principles and terminology conversion was required. This whole process took four months, however, the improvement in understanding and reduction in resistance meant this time was well spent and ultimately very effective.

Step 4 : Facilitated Self-Assessment
The next phase of the improvement programme was to carry-out a series of facilitated self-assessments. This would continue the non-threatening and low profile approach established from the consultation process. Also, when advice and guidance was requested from the central quality group by software developers and IT practitioners, they were encouraged to conduct a self-assessment. However, it was not mandated to engage in a process improvement project. In addition, practitioners were encouraged to attend a training course on software engineering principles and quality. Self assessment was particularly useful in support of the training as not only did it baseline the current level of implementation, but it helped gain an insight as to the amount of understanding of software engineering or quality processes. This helped tailor the training to specific individual training needs and present examples that were more relevant to attendees.

Step 5 : Software Engineering Training
Step 5 was carried out in parallel, and working closely with Step 4. The first training course was started in late in the second year. The philosophy for training was to provide a good introduction to the principles of software engineering and link this into how to implement the company software procedures [35]. A partnership was established with a training provider that not only had a ready made software engineering course, but also had experience of implementing the methods and techniques from a diverse range of software systems. The first author contributed to the training by providing the links to the Company Software Procedures. A feature of this part of the training was to not only provide examples of good practice but also to identify some situations where people had misinterpreted the requirements in a manner that was slightly humorous. This helped give the course a lighter feel as software engineering and quality can otherwise be a dry subjects.

From the onset it was felt important to ensure attendees felt the course was of value. This was not only achieved through the monitoring of course evaluations but also by comments requested at the end of each course tutorial. The evaluations consisted of a comprehensive set of questions on course objectives, value, applicability, joining instructions, food, etc., to be answered on a six point scale. After a number of courses it became clear that the company specific elements was considered the most valuable, so the course altered to become a completely bespoke course, tailor-made to the company software system. Nearly all tutorials were geared to further implementing the SQMS requirements. Attendees would document there own processes, collate inventories and assign categories and software product baselines. They also learned to understand system measures and metrics and apply their own. The courses were not made compulsory but the skills provided by the course were documented in a competency framework within the company software procedures. Over a six year period, 130 people completed the training course, and a total 69 full assessments were conducted as part of the training.

Step 6 : Providing Support and Guidance

Step 6 was also undertaken in parallel with Steps 4 and 5. Another issue that needed solving that became apparent during the consultation process was the considerable time was spent explaining many aspects of the software quality system on the telephone. Frequent questions were on some quality terminology and how to implement various requirements in a range of differing situations. There were many requests for examples. These questions continued after the procedures were published. The role of central quality assurance could be compared to that of a helpdesk. Unfortunately many of these calls would last typically 20 minutes with a significant number around 45 minutes. The response to this required a significant resource and presented a clear need to support or underpin the main procedures with guidance information. Certainly the ability to refer, in the first instance, to a document would alleviate the amount of time spent of the phone. In response to this need, a programme was established to produce guidance documents. A total of twenty-two guidance documents were issued to provide the detailed explanation, examples and templates to streamline software engineering document production. The first document produced was a template on how to produce a software control plan. As this would provide the biggest return-on-investment and immediately improve both the quality of the processes and significantly raise the level of compliance. Thereafter, the guidance was produced on a priority basis that reflected the weakest implemented processes, such as software configuration management and guidance for conducting reviews, and the most frequently asked questions, such as "what is the category of my software?" and "what is the difference between verification and validation?"

Step 7 : Monitoring and Evaluation

The final step is closely associated with Step 1, Establishing a Baseline. The exercise to assess the level of implementation of the SQMS was continually monitored with each new quality assessment, with the level of compliance with the SQMS being recorded. At the end of each year the results were compiled into Tables 1 and 2, to gain a measure of what improvement in implementation was being achieved, which then gave a measure of the success of the overall improvement methodology.

In practice, all the above steps were subject to many iterations. Each self-assessment helped identify areas of improvement in the training, in the support and guidance and even in the quality procedures themselves. The questions trainees asked in training sessions also led to improvements in support documentation and the greater understanding the training imparted led to improved self-assessments. The questions received in the support process identified areas of training need and the training courses and support contacts helped recruit further candidates to carry out self assessments. This continual improvement in all of the steps in the methodology was an essential part of the approach, allowing the improvement process itself to grow in its effectiveness as it was being carried out and developed.

During the final two years of the research, some of the recommendations from the earlier work were tested. This again formed quite an integrated approach as the recommendations on auditing were combined on a new audit programme mandating the self-assessment spreadsheet as a check-list [36]. Further, the cost models described were also included in the self-assessment spreadsheet[36] to test the response of financial factors being applied to inefficiencies related to inadequate practice, or in auditing terms, non-conformance.

The audits and assessment conducted for years 8 and 9 were from management areas that they had not engaged on improvement initiatives over the last few years or had not undertaken the self-assessment as part of the training programme. The selected areas were placed on the company audit programme with the self-assessment spreadsheet mandated as part of the audit process with guidance and support available when needed.

As audit deficiencies were raised as formal non conformances they were presented to senior management to ensure support and drive for improvement actions. So the formality of the audit process was used to maximum effect.

A further research objective was to test the Cost-Benefit Audit Methodology (C-BAM) model described in an improved process for internal auditing [37] and depicted in Figure 1. The concept was to include example inefficiencies for non-compliance and have the cost models integrated in the self-assessment spreadsheet. The summary page on the spreadsheet contained the percentage implementation and also an inefficiency rating cost based on not fully complying with the SQMS. The idea was this would stimulate debate on the cost benefits of applying best practice methods and to further the model to the costs to the software system and management arrangements under investigation. The cost benefit of improvements and the resultant savings could then be collated to demonstrate the cost benefit that improvement actions would provide. The response was quite mixed, from genuine interest to develop the cost models to fierce criticism and challenge to the concept of even trying to include it in an audit process.

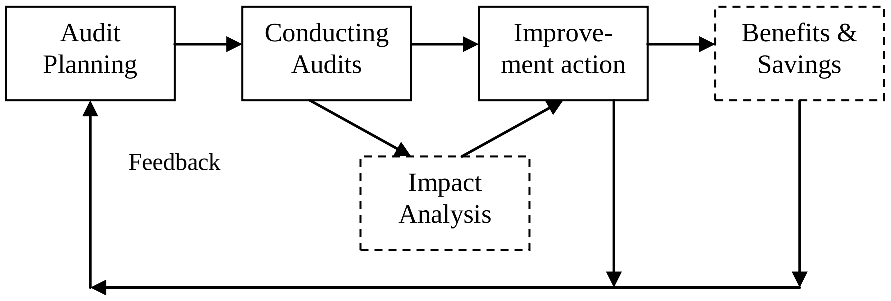

Figure 1. Improved top level process map for internal auditing

4. Results

Table 1. depicts the assessment results of software systems checked against a question set based on practices within a software quality system. Up to one hundred questions could be applicable depending on the type of software system. The assessments were evidence based as documentary evidence was required to prove relevant practices had been adhered too. The results show the average value of all systems assessed that year.

Table 1. Software System Assessment Results

Year	Number of Systems Assessed	Average % implementation
1	55	34%
2	18	46%
3	19	54%
4	5	n/a
5	22	60%
6	21	66%
7	16	74%
8	29	73%
9	51	81%

The assessment question set was compiled under seven main headings. The figures presented in Table 2 are the average values of all system that year for that particular subject.

Table 2. Software Quality Topic Percentage Improvements by Year

Topic	Y1	Y2	Y3	Y4	Y5	Y6	Y7	Y8	Y9
General Management	36%	51%	57%	n/a	61%	65%	72%	78%	81%
Requirements Management	26%	32%	42%	n/a	57%	71%	76%	82%	83%
Development Processes	26%	44%	48%	n/a	59%	60%	69%	71%	80%
Testing Arrangements	40%	53%	60%	n/a	68%	71%	76%	79%	81%
System Use	n/a	59%	70%	n/a	74%	75%	94%	78%	84%
Configuration Management	20%	26%	33%	n/a	50%	52%	67%	45%	73%
Disaster Recovery	36%	47%	61%	n/a	58%	57%	74%	77%	85%

5 Analysis of Results

The results in Table 1 show a steady improvement in the overall level of implementation. As well as the increase in the yearly average, the number of systems below 40% reduced from 27 in year 1 to just the one in year 6. By year 9 the lowest assessment result was 48%. Although there was a programme of assessments a significant number of systems assessed each year came from either people requesting advice or from attending the training course. In this respect the sampling of assessments contained an element of randomness adding to the validity of the improvement.

Another indication of the general upward trend is that a number of systems were assessed more than once and on each occasion an improvement was achieved. The owners were therefore implementing further best practices to implement the SQMS through their own initiative. This is further indication that once understood the SQMS requirements are seen as adding value.

The topic improvements in Table 2 provide a good indication that the targeted training, guidance and support emphasised on both configuration management and requirements management have had significant impact as they are the topics with highest improvement. The most significant improvement in the development section was in the knowledge provided for design decisions based on reliability and maintenance factors and the effective selection of appropriate life cycles. The significant increase in the "system use" section revolves around the fact that people can see an immediate payback to supporting the use of the software system with guidance and work instructions. Customer and user satisfaction provides its own drivers. .

It was reassuring that management areas visited in year 8 and 9 that had attended the training course but had not undertaken self-assessment had reasonable scores, despite concerns to the contrary. The company average score of 73% for year 8 is a good reflection on the progress for the process improvement actions identified from both the audit and self-assessment. It was noticeable that the drive to complete audit actions which are tracked on a company data base provided more impetus to improvement action than normally witnessed through facilitated self-assessment without audit. Again the drive given to progress audit action accelerates the audit and self-assessment process improvement activity. The final improvement to 81% is seen as an outstanding achievement.

This pilot scheme to include financial models for cost benefit analysis was successful as one or two areas engaged in the discussion on how accurate the models were to identifying the cost of non conformance and the associated cost benefits and savings attributed to improvement actions. The challenge to the approach described in the method section must be noted and the conclusion here is that the request to consider costs to non-conformance was too much of a surprise and seen as a threat to local management.

6. Conclusion

It is clear from the data that steady improvement in implementing the software quality system has occurred year on year and that the overall approach in facilitating these improvements has been successful and effective.

An independent validation of the improvement is given by the ISO 9001 Certification process where the number of problems found in software issues had been greatly reduced. Significantly, only one, minor nonconformity had been raised in the last three years of the research. The third party audit reports frequently praised the holistic approach being taken to improve compliance to the SQMS. The training methodology was found to an exemplar when compared against best practice evaluation frameworks and industry in general.

Perhaps the most pleasing aspect of the improvement is that it reflects a difficult to achieve success for software process improvement projects that this approach had truly won the hearts and minds of practitioners to bring about a change in culture. With a philosophy of customer care, consultation, and active engagement, practitioners now produce documentation of good quality that, in turn, facilitates good decisions that have proved to be effective and efficient.

The research has demonstrated a practical solution to achieving process improvement for software quality management. The cost effectiveness of the approach is attractive to any organisation, as value and a return-on-investment are keys to success. Any system that has a beneficial impact on “the bottom line” will win. Another pleasing aspect of the research is that elements of the methodology have also proved successful individually and, as such, each could be applied in their own right.

As this research has surfaced methods that have not been addressed elsewhere by current literature and the practical aspect of the level of achievement have been self validating, the author believes the findings of this research make a significant contribution to the world of process improvement. With the number of process improvement failures still making headlines, it is suggested that companies adopt the suggestions that have emerged from this research to add another dimension to the effectiveness of their process improvement programmes.

7. References

1. Elliott, M., (2008), *Achieving Business Excellence in Software Quality Management*, PhD thesis, Loughborough University, UK https://dspace.lboro.ac.uk/2134/8121
2. Elliott, M., Dawson, R.J. and Edwards, J., (2007a), *An Analysis of Software Quality Management at AWE plc*. Software Quality Journal, Volume 15, Number 4, pp 347-363, Springer Netherlands, ISSN 0963-9314.

3. Elliott, M., Dawson, R.J. and Edwards, J., (2006), *Towards real process improvement from internal auditing—A case study*, Software Quality Journal, Volume 14, Number 1, pp 347-363, Springer Netherlands, ISSN 0963-9314.
4. Cangemi, M. P. and Singleton, T. (2003), *Managing the Audit Function*, John Wiley & sons, Hoboken, New Jersey, USA, ISBN 0-47128-119-0.
5. Wealleans D. (2000), *The Quality Audit for ISO 9001:2000, A Practical Guide*, Gower Publishing Company, ISBN 0 566 08245 4.
6. International Standards Organisation (ISO), (2000), ISO 9001 : 2000 Quality Management Systems – Requirements
7. Carnegie Mellon University, (2001), *Capability Maturity Model Integration*, Software Engineering Institute, Pittsburgh USA.
8. ISO 9004:2000, (2000), ISO 9001 : 2000 *Quality Management System – Guidelines for performance improvements,* International Standards Organisation, Geneva, Switzerland
9. Standish Group International, (1994), *The Chaos Report*, Standish Group web site, www.standishgroup.com, (visited November 2005)
10. Standish Group International, (2001), *Extreme Chaos*, Standish Group web site, www.standishgroup.com, (visited November 2005)
11. Spencer L.M., McClelland D.C., Spencer S.M., (1992), *Competency Assessment Methods, History and State of the Art,* Paper presented at the American Psychological Association, Annual Conference, Boston, USA.
12. Wagner H., (1999), *The Psychobiology of Human Motivation,* Routledge, Taylor & Francis (UK), Abingdon, Oxon., England, ISBN 0-415192-95-7
13. Horch, J.W. (1996), *Practical Guide to Software Quality Management*, Artech House Publisher, Boston, USA, ISBN 0-89006-865-8
14. Galin D., (2004), *Software quality assurance, from theory to implementation*, Pearson Education Limited, Essex England, ISBN 0-201-70945-7
15. Siaksa S, and Georgiadou (2002), *Empirical Measurement of the Effects of Cultural Diversity on Software Quality Management*, Software Quality Journal, Issue 10 No 2, September 2002, pp 169-180, ISSN: 0963-9314
16. Sandi, M. and Robertson, I.T., (1996), *What should training evaluations evaluate?* Journal of European Industrial Training Vol. 20/9 pp 14-20 MCB University Press, ISSN 0309-0590
17. Tennant, C, Boonkrong, M. and Roberts, P.A.B., (2002), *The design of a training programme measurement model*, Journal of European Industrial Training Vol. 26/5 pp 230-240 MCB University Press, ISSN 0309-0590
18. Hale, R., (2003), *How training can add real value to business: Part 2,* Industrial and Commercial Training, Vol. 35 No 2 pp 49-52, MCP UP Limited, ISSN 0019-7858
19. Morey, D. and Frangioso, T. (1998), *Aligning an organization for learning: The six principles of effective learning,* Journal of Knowledge Management, Vol. 1 No 4, pp 308-314, Emerald Publishing, ISSN 1367-3270
20. Phillips, J.J. (2002), *Return on investment in Training and Performance Improvement Programs*, 2nd ed., Butterworth-Heinemann, Woburn, MA.
21. Lee, P. and Quazi H. A. (2001), *A Methodology for Developing a Self-assessment Tool to Measure Quality Performance in Organisations*, International Journal of Quality and Reliability Management, Vol 18 No 2 pp 118-141, MCB University Press, 0256-671X

22. Samuelson, P. and Nilson L (2002), *Self-Assessment Practices in Large Organisations, Experiences from using the EFQM Excellence Model*, International Journal of Quality and Reliability Management, Vol 19, No 1, pp 10-23, MCP UP Limited, 0265-671X
23. Waina, R.B. (2001), *Five Critical Questions in Process Improvement,* web site http//www.chips.navy.mil/ archives/ 01_summer/five_critical_questions_in_proce..htm. visited April 2006.
24. Jones Capers (1994), *Assessment and Control of Software Risks,* Yourdon Press, Prentice Hall Building, Englewood Cliffs, New Jersey, ISBN 0-13-741406-4
25. Freeman & Weinberg, (1991), *Handbook of walkthroughs, inspection and technical reviews*, Dorset House Publishing Co Inc.,U.S, ISBN 0932633196
26 Gilb, T. and Graham, D., (1993), *Software Inspections*, Addison-Wesley, Massachusetts, USA, ISBN 0-201631814
27. Yourdon, E., (1989), *Structured Walkthroughs*, Prentice Hall, New Jersey, USA. ISBN 0138552893
28. Bennett K H, Munro M, Knight C & Xu J (2000). *Informatics Centres of Excellence; Research Institute for Software Evolution.* IEE Computing and Control Engineering Journal 11(4): 179-186.
29. Rico, D.F. (2002), *Software process improvement: Modelling return-on-investment (ROI)*, Software Engineering Institute (SEI) Software Engineering Process Group Conference (SEPG 2002) Phoenix, Arizona.
30. Fagan, M.E. (1976), Design and code inspections to reduce errors in program development, IBM Systems Journal, 12(7), pp 744-741.
31. ^{SM}Personal Software Process, PSP, Team Software Process, TSP are service marks of Carnegie Mellon University.
32. Humphrey W.S. (1989), *Managing the Software Process,* Addison-Wesley, Massachusetts, USA, ISBN 0-201180-95-2
33. ISO 9004:2000, (2000), ISO 9001 : 2000 *Quality Management System – Guidelines for performance improvements,* International Standards Organisation, Geneva, Switzerland
34. ISO 15504, (1998), *Software Process Assessment,* International Standards Organisation, Geneva, Switzerland
35. Elliott, M., Dawson, R.J. and Edwards, J., (2009), *Providing demonstrable return-on-investment for organisational learning and training,* Journal of European Industrial Training, Vol. 33 Iss: 7, pp.657 – 670, Emerald Group Publishing Limited, ISSN: 0309-0590
36. Elliott, M., Dawson, R.J. and Edwards, J., (2009), *An evolutionary cultural-change approach to successful software process improvement,* Software Quality Journal, Volume 17, Issue 2, pp 189-202, Springer Netherlands, ISSN 0963-9314
37. Elliott, M., Dawson, R.J. and Edwards, J., (2007b), *An Improved Process Model for Internal Auditing*. Managerial Auditing Journal, Volume 22, Numbers 6 and 7, Emerald Group Publishing Limited, ISSN 0268-6902.