Designing quality management systems with minimal management commitment

Øgland, Petter
Department of Informatics, University of Oslo, Norway

Abstract

In literature on quality management design it is "common knowledge" that quality management will not work without management commitment. But, not always is it possible to achieve management commitment, so what to do then? Based on a longitudinal study of quality management design in a software development organisation, a systems thinking approach is presented and discussed. The main idea is that management commitment is observed through the eyes of the people being managed, and may thus be created as an illusion to compensate for a lack of the real thing. The case study shows how this can be done in a way easy to replicate.

Introduction

Beckford (2002: 311) concludes his discussion of quality management from a systems perspective by saying that "management commitment is the single most critical issue in the pursuit of quality. Without it, the programme will fail – as so many do".

However, in a busy software development environment, there may be limits to how much management can dedicate itself to setting quality objectives, making quality plans, reading defect reports and doing quality management systems reviews. The real problem of many software development organisations may thus be related to the problem of *how to design quality management systems with minimal management commitment*.

In this paper, a study of COBOL computer programmers is used as an example in order to illustrate design challenges and design ideas for a software quality management system (QMS) with minimal management commitment. The theory presented in section two refers to programmers psychology, programming standards and QMS design. In section three follows a description of the research methodology, which in this case has been chosen as design research. Results are presented in section four, and in section five follows a discussion of what has been learned through the failures and problems with the current QMS design.

Theory

The theory chapter does not aim to give a complete description of the fields discussed, but try to emphasise aspects of domain knowledge and QMS knowledge that is expected to be to of significant importance for good QMS design in an environment consisting of COBOL programmers.

Psychology of COBOL programmers

Strauss (in: Davis & Hersh, 1986: 179-186) discusses the psychology of computer programmers by starting with the comment that computer programmers are people who would much rather communicate with machines than people. He suggests that the psychology of the computer programmer may be somewhat similar to the psychology of the formula one race car driver, in terms of having a total dependence on the surrounding technology and thus expressing oneself through technology. Furthermore, he suggests that computer programming is basically something that has to be experienced from within, as it from the outside may look both forbidding and dull.

Weinberg (1971) provides a more positive picture of programmers in terms of social skills and interests, but to a certain degree the message is the same. Computer programming is associated with the ability to sustain long attention spans and the ability to concentrate. Although Csikszentmihalyi (1992) does not focus exclusively on computer programmers when researching how flow and happiness is achieved through an "autotelic" personality, his theories on how flow is achieved through relaxed balance between skills and challenges makes computer programmers an interesting example, as they frequently seem to be in states of flow for long periods of time. The activity of writing computer code is an activity that requires both discipline and creativity, and could perhaps be used as a prime example of how to achieve intellectual flow.

Programming standards and participatory design

The COBOL programming language appeared in 1959 and is one of the oldest programming languages still in use. The programming standard was designed by a committee. Is has later been redesigned as ANSI standards as COBOL-68, COBOL-74, COBOL-85 and COBOL 2002 (Ghezzi & Jazayeri, 1997).

COBOL 85 was not compatible with earlier versions. Furthermore, older versions of COBOL lacked local variables, and could not support structured programming. Joseph Brophy, of Travelers Insurance, spearheaded an effort to inform users of COBOL of the heavy reprogramming costs of implementing the new standard. As a result, the ANSI COBOL Committee received more than 3,200 letters from the public, mostly negative, requiring the committee to make changes (Wikipedia, 2007).

Standards may stir up controversy. Not particularly focused on design of internal programming standards, the Scandinavian countries in the 1960s and 1970s developed an approach to design called "participatory design" as an attempt to actively involve end-users in the design process to ensure the product designed met their needs and would be usable (Bjerknes, Ehn and Kyng, 1987).

Using academia as a part of organisational design

The Scandinavian studies of participatory design experiments were facilitated as collaborative research between industry and academia. One way of trying to achieve the effect of combining action and research is through the method of "action research" (Reason and Bradbury, 2006).

One of the more recent applications of action research in the Scandinavian community has been in the area of researching health information systems in the developing world. One of the major theoretical insights gained from these studies has been that one can use the international network of scientists as a

platform for building a network of information systems in developing countries. Information systems researchers function as an "information infrastructure" for practical development of health information systems to build upon this infrastructure, just like computer systems are being built upon the information infrastructure of the Internet (Braa, Monteiro and Sahay, 2004).

Design of quality management systems

The international standard ISO 9001:2000 provides a set of requirements that define the internationally agreed upon concept of a quality management system. The standard is generic, and may be applied to all sorts of organisations or parts of organisations, regardless of size, sector and line of business. The requirements are grouped into a structure (ISO 9001 model) illustrated in figure 1.

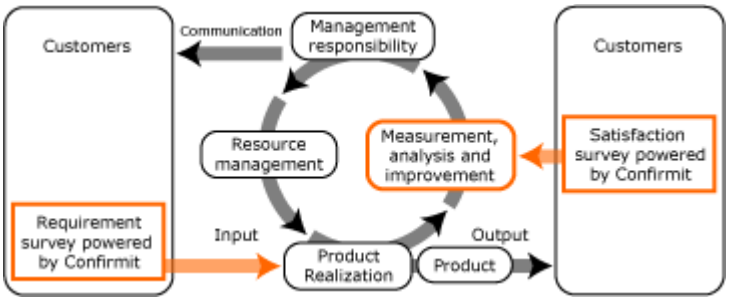

Figure 1 ISO 9001 model (http://www.confirmit.com/solutions/subject/iso_9000/print.asp)

Along with ISO 9001 there are a number of other international guidelines and standards that give requirements and expectations for design in more specific areas, such as the guidelines ISO 90003 for interpreting the ISO 9001 requirements in a software development organisation, or standards ISO 12207 for managing the software development lifecycle.

As the ISO 9001 standard is a list of requirements for producing a certifiable quality management system, it does not give advice on how to design the QMS in order to fit with both the organisation and the requirements of the standard. As pointed out by Seddon (2000), providing a good QMS design that fits with the ISO 9001 requirements is not a trivial task, illustrating his point by giving examples of cases where bad QMS design has resulted in poor quality or causing the situation to become even worse than it was before the QMS was implemented. A design approach that is more likely to succeed, he suggests, is to apply the systems thinking approach that he finds in the writings of Deming (1986; 1992).

Methodology

Design research

Simon (1996) argues that the research approach used for investigating the natural world is not the best approach for investigating the artificial world, and he defines design research as a scientific method that is different both from the positivist methods used in natural science and also different from the phenomenological methods used for understanding purpose and meaning.

The illustration in figure 2 is inspired by the guidelines provided by the Association of Information Systems (AIS, 2007) for using design research to study information systems. The inner loop turning clock-wise illustrates a typical systems developing method consisting of four steps. First the problem is formulated and a design process is used for providing a solution in the shape of a model. The model is then implemented as a system, and the system is run and monitored in order to perform an evaluation. The evaluation is used for identifying opportunities for improvement, and the cycle starts again.

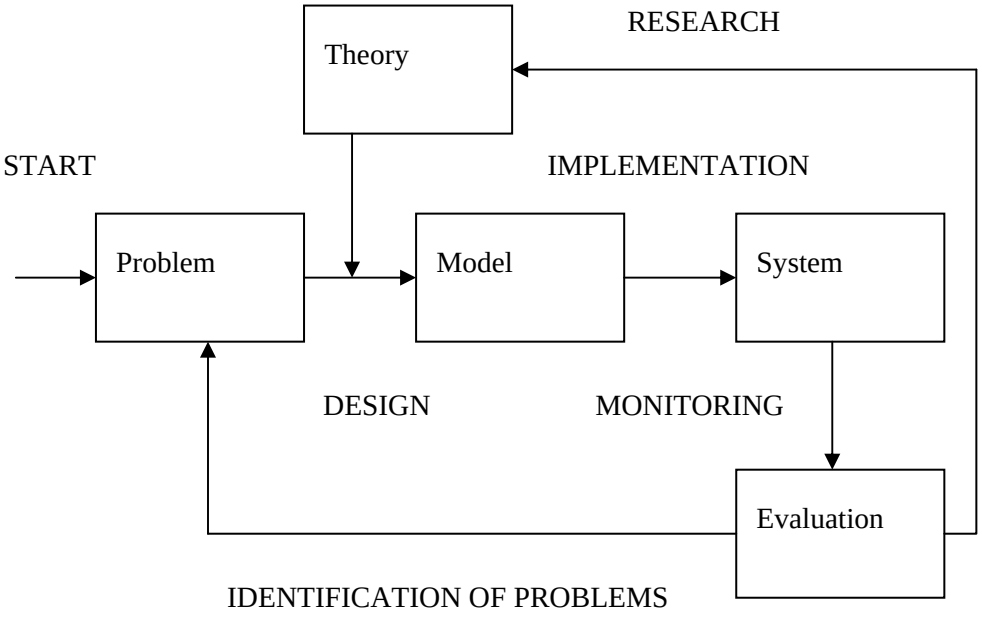

Figure 2 System design and design research

Design research is represented by the process that runs from the evaluation of a system, focusing on the problems and challenges of the system in action, in order to contribute to the theory that was used in the design. This simplified description of design research corresponds with how design and research are sometimes viewed as inverse processes (e.g. Krick, 1965).

The socio-technical quality management system

The ISO 9001:2000 standard contains requirements both with respect to humans and artefacts, meaning that the quality management system cannot only be understood as the quality manual, procedures and other documents, but the QMS must be seen as the total socio-technical system consisting of the humans and non-humans that collaborate in managing the organisation with respect to quality.

In this study, the writer of this paper has been working as a management consultant within the IT function of a large Scandinavian public sector organisation during the period from 2000 to 2007. The aim of this particular case study is to investigate the design of a subsystem of the total QMS that deals specifically with making COBOL programmers following programming standards.

The aim of the QM subsystem is to make the COBOL software more easily maintainable. In 1997 one of the COBOL programmers died, and others had to step in to maintain and improve critical software. As there was no standardised way of programming, the situation got critical, and it became ob-

vious to everybody that there was an intense need to make sure that all COBOL software was standardised and restructured to fit with modern principles of structured programming. In the strategic plan for the IT department, published in 1998, it was stated that all COBOL software should be compliant with internal standards based on the COBOL 85 standard and good programming practice.

The QM subsystem consists of three groups of programmers, comprising 40 COBOL programmers and three managers. These managers report to the software maintenance manager. The software maintenance manager reports to the head of the IT department. During the first five years of this study, the management consultant reported to the innovation manager, and the innovation manager reported to the head of the IT department. For the past two years, the management consultant has reported directly to the head of the IT department. The COBOL software consisted of about 1.4 million lines of code, distributed among 10 software projects.

Results

In order to design the details of the QMS in figure 1, the diagram in figure 3 illustrates relationships and flows in the part of the QMS called "product realization".

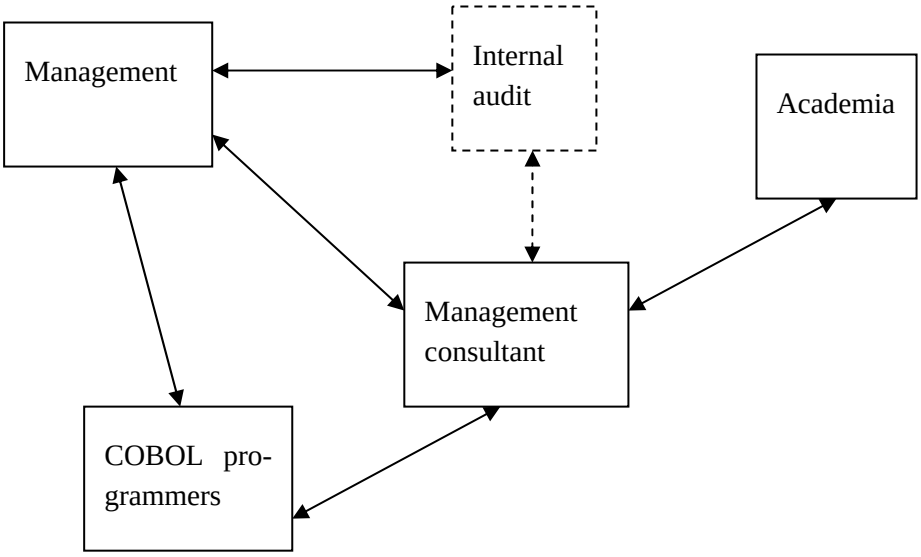

Figure 3 COBOL standardisation management system ("product realisation" in QMS)

In this design the term "management consultant" is used as the term for describing an owner of the QMS, meaning that this is a person who can destroy the system (Checkland, 1981).

Interaction between consultant and programmers

The interface between the consultant and the programmers consists of the consultant producing a note summarising the situation for the particular group of programmers, including statistical trends illustrating how the group is progressing towards the goal of making the software compliant with the standard ("zero defects"), a statistically estimated guess on how long it will take until they have completed the task (if continuing at current speed), and benchmarking results of the group against fellow groups.

The note is presented to a representative of the group of programmers during an improvised interview. The consultant is thus observing the reactions of the programmer when confronted with the statistics, and they are able to discuss the situation.

If the representative of the programming group is not available, the statistics are delivered by post, and the consultant will not be able to observe reactions. In such cases, the consultant sometimes talks to the manager instead, especially if the statistical trends seem to go in the wrong direction.

Interaction between consultant and management

Given the initial problem of lack of management commitment, one of the core design ideas was to minimize the interaction between consultant and management. In order to utilise the principles of "what managers focus on tends to get done" (Schein, 1992), it seemed worth a try just to send a copy of the evaluation results to management so that the programmers understood that managers where being informed.

By the end of each annual cycle, a full report was constructed. As the aim of the report has gradually been to develop theory that would improve the relationship with academia, the report was difficult to understand both for programmers and managers, but according to the "what gets focused gets done" hypothesis, the important issue was to make visible that quality control and academic research was getting done rather than what was the details of the academic work. The practical aspects, i.e. whether the trends showed improvement or not, were explained through executive summaries.

Interaction between consultant and internal audit

As the purpose of the internal audit is to make sure that the internal control is working properly, the annual reports from the standardisation project were also distributed to the internal audit. Just like in the case of management, the design idea behind this interface had more to do with getting the programmers to see that the internal audit were being informed than with what the internal audit were actually doing with the information they were being given. Actually, as the internal audit was a small group of people with a large internal control system to monitor, the consultant never got any feedback on the COBOL project, and never bothered to discuss this particular project with the internal audit.

Although not illustrated on figure 3, there was a continuous effort to enlarge the system by including various actors such as the National Audit, the National Quality Management Organisation and a "best practice" network of similar organisations from neighbouring countries. The idea remained the same; to give the programmers an illusion of management commitment by becoming the centre of a system that was growing all the time, over seven years. However, just like the case of the internal audit, none of these new components of the system showed any interest whatsoever in what was going on, and had no interest in performing mutual benchmarks, exchanging "best practices" or doing any sort of interaction.

Interaction between consultant and academia

Due to challenges in the organisation due to programmers not being to happy with being monitored and management not being too happy with being given statistics that indicate the processes are not running as smoothly as one would like, there is a looming chance that management will terminate the system and the consultant being unable to perform process improvement.

The reason for including academia in the QMS design is thus to use the "cultural capital" of academia as a source of power make making the QMS survive despite producing statistics that may challenge the organisation. The general idea is to use the QMS for improving the organisation rather than having the organisation terminate the QMS.

So far, three academic conference papers have been produced. One paper got accepted at a Scandinavian workshop conference on information systems research. The other paper got rejected at a European information systems conference, but was modified and got accepted for a Scandinavian workshop conference. The current paper was accepted for a systems society conference. The research questions and research approach has been slightly changed based on the feedback received through each conference feedback.

Discussion

As the hypothesis for the paper is that the current QMS design is a successful design, the purpose of this discussion is to try to explain why the design appears successful and to challenge the hypothesis by focusing on some of the weaker aspects of the design.

Overview of significant findings

Designing an environment for the COBOL programmers

When there are weak ties with management, the design idea suggests this to be compensated by stronger ties with the programmers, or at least a method for providing the programmers with an experience that would be somewhat similar to having strong management commitment.

The idea informing this particular QMS design was Csikszentmihalyi's studies of "flow" and happiness (Csikszentmihalyi, 1992). By presenting the programmers with trend curves and Pareto diagrams illustrating how they were performing against the standard and against each other, this was supposed to help the programmers find the right balance between challenge and mastery, and thus use these statistics as a way of achieving flow. Consistent with Csikszentmihalyi's theories, there were no incentives associated with the numerical feedback beyond the fact that the programmers knew that the information was also presented to management, internal audit and, more recently, also used as input for academic research. According to flow theory, the state of relaxed happiness is most easily achieved by the autotelic personality, i.e. the state of doing work for the pleasure of doing it.

From a systemic perspective, the focus of the design was to create an environment that would make the programmers work as an adaptive and evolving system, and there was consequently no need to investigate too deeply what the programmers where thinking. Preferably, in order to achieve flow, they should not be thinking too self-consciously about the standards and measurements, but just raise their personal standards to gradually match with the challenges provided by the QMS.

Applying participatory design for developing programming standards

In order to prevent unnecessary resistance and complaints from the programmers about being alienated by the internal programming standards, having the standards forced upon them, or making questions concerning the competence of the standards, one of the QMS design ideas was to have the programmers themselves design the internal programming standard, but nevertheless make sure that it was formulated in terms of explicit demands ("shalls"), so that it would be easy to produce measure-

ment software that could be used for checking COBOL software for non-compliance with the standard. In collaboration with the programmers, various parts of literature on programming practice was used, such as Dijkstra (1968), the introductory chapter in Aho, Hopcroft and Ullman (1983) and various pieces of advice found on the internet.

Both the standard and the measurement system derived from the standard were discussed and developed among the programmers, and was signed by management, thus showing a minimal management commitment.

Whether the involvement of the programmers through a participatory design of having them design their own standards was necessary or not for achieving a good QMS design is difficult to say. Despite having defined the standards themselves, they were not particularly happy with being forced to follow a standard, although each and every one seem to be quite happy with their fellow programmers being forced to follow it.

Differences as compared to traditional QMS design

Although the Toyota quality management system seems to have been based on a similar systems thinking type of design (Fujimoto, 1999), during the literature research done as a preparation for this study, no traditional QMS design literature was found to support the idea of designing quality management systems with minimal management commitment. On the contrary, the advise found in all the literature studied, including Tricker (2005), Schlickman (2002), Deming (1986), Crosby (1979), Juran (2006), and the ISO 9000 guidelines, were supportive of the comments made by Beckford in the introduction of this paper. However, looking at the discussion forum of the American Society for Quality (ASQ, 2007) there is a thread related to complexity theory in quality management. It is possible that there may be some similarities between the design approach presented in this paper and how complex adaptive systems (CAS) may be used as a theory for QMS design (e.g. Dooley et al, 1995).

The importance of aligning with academia

As illustrated in the story of the internal audit and other units, people tend to be interested in collaboration only when they are being forced to collaborate or there are clear indications of "what's in it for me". The internal audit and the national audit were interested in auditing according to their own audit plans, and had little use or interest in being fed with additional information. People working with quality management systems in other organisations were focused on their own problems, and had little interest in cooperation or learning by benchmarking. Also, from the perspective of management, such informal international quality management networks were of little use as they could not be used for any political purpose unless the relationships were made tighter through formalisations, and if it should turn out that the benchmarks turned out less favourable for the organisation, this would only produce yet another threat for the destruction of the QMS.

Using academia as a component in the QMS design, however, proved to be a much more useful idea, provided one could find the right academic niche, such as scientists researching general systems theory, management systems theory or information systems theory. Establishing a contact between the management consultant working with the day to day practical problems of QMS design and information systems scientists developing theories on management information systems design, and testing such theories through a network of international information systems studies, proved to be a success.

Unless the previous relationships, that were established in order to create an illusion of management commitment, the relationship with academia was a different type of relationship as it was a two-way relationship for creating a research-action process that would be of mutual interest. Besides, estab-

lishing contacts with academia was something that gave prestige to management, and thus reducing the risk of the QMS being destroyed.

Consideration of the findings in light of exiting research studies

It is "common knowledge" in quality management literature in general and software process improvement management literature in particular that quality management systems are difficult to design, difficult to implement, and seldom have a life span longer than two years Furthermore, in order to achieve success, there are several factor that need to be present, such as management commitment, measurements connected to economic parameters and other issues that seem important but may be difficult to achieve.

In the study documented through this paper, none of the usual "must have" conditions are present, and yet the QMS has lasted for over seven years and is still viable. What seems to be radically different in this particular study, as compared to others studies, is the emphasis on treating the organisation as a system. Rather than starting by a detailed clock-work design of each component of the system and making sure that everything is perfectly optimal, the driving idea has been to view the QMS as an organic entity and try to produce and environment for the system that may work as a learning environment, appealing to the sportsmanship, pride, pleasure of doing good work, etc of the programmers by providing them with numerical feedback.

Implications for current theory

Looking at the issue of designing quality management systems from a soft systems perspective rather than a hard systems perspective, makes it possible to challenge the assumption of management commitment. What this particular case study illustrates is that lack of management commitment can be compensated by trying to establish and illusion of management commitment. What matters is not what the situation looks like for somebody observing the system from the outside, but what the situation looks like from the perspective of the COBOL programmers. If it is possible to convince the COBOL programmers that there is a strong management commitment, meaning that following programming standards and producing high quality maintainable software is something that is considered to be of great importance, then it doesn't really matter what management may think.

Although such ideas are not specifically discussed by Seddon (2000), the ideas seem to fit well with his soft-systems interpretation of Deming's later writing (Deming, 1992).

Careful examination of findings that fail to support the hypothesis

Checkland (1981: 318) uses the term "system owner" for the person or persons who might be capable of destroying the system. In the case of the quality management system discussed here, the main owners seem to be the head of the IT department, the management consultant and the COBOL programmer who is delivering statistics to the management consultant. As this particular programmer is about to retire, something have to be done if the QMS is not to be destroyed, and it clearly reveals that depending too much on key personnel for running is a bad design feature in the long run, although it was perhaps an essential feature of the early design, in order to build on the few key people who had the knowledge and motivation for making the system work.

Whether the management consultant is capable of destroying the system or not is not of great concern for the management consultant per se, but it ties him up with this particular quality management project, demanding resources and making him less flexible in order to investigate similar projects. Of course, from the perspective of the organisation, it may not be all that useful to have projects depending too much on singular people.

The greatest threat to the system, however, is top management. The design of the QMS is based on the assumption that "no new is good news" whenever management is concerned. If the statistical trends do not show improvement, it may be easier for management to shut down the system and play ignorant than actually do something about it. Thus the problem of the management consultant is to convince management that everything is fine and working smoothly while, on the other hand, convincing that programmers that they need to shape up and focus on keeping with the programming standards and generally improving whatever they are doing. On the other hand, if the programmers are stressed too much, they may complain to management, thus making management shutting down the quality management system anyway.

Limitations of the study that might affect the validity or generalisation of the results

It is difficult to generalise from a single case study. There are various factors that might be questioned, such as the study being carried out among software programmers in a Scandinavian public sector organisation. Software programmers may respond differently to standards, measurements and quality management systems than other professions. Scandinavian culture may be different to other cultures. Public sector may be different to private sector. Nevertheless, the study provides a method and a repeatable example of what has been successful in one organisation. As the results appears to be contrary to common knowledge among academics and practitioners in management consulting, the study should at least be a possible reference point for doing more studies of a similar kind.

Recommendations for further research

Although the ISO 9001 requirement standard is used as a part of the theory in this study, the QMS design in section four was not fully compliant with ISO 9001:2000. In developing a design theory for quality management systems with minimal management commitment, it would be interesting to investigate to which extent such a theory could be generalised for designing quality management systems compliant with the international minimum requirements for quality management systems, i.e. ISO 9001. In this context, it is particularly interesting that there are clauses in ISO 9001:2000 setting requirements to management commitment, something that would make the "management without managers" approach (Koch and Godden, 1996) interesting to study.

Implications for professional practice or applied settings

As the this paper documents the personal experience of an internal quality management consultant in a public sector IT organisation, the story of the QMS design has been a story of personal transformation from a traditional analytical quality management approach to a systems thinking quality management approach. The experience has not been an experience of finding anything wrong with the ISO 9000 standards or experts in industrial engineering, but rather an experience in terms of ideas from typical "hard systems thinkers" proving even more insightful when combined with a "soft systems approach".

Although the ideas in this study is influenced by how various systems models may be used for augmenting the quality management strategies, as suggested by Beckford (2002), the results from this study is more optimistic than Beckford, suggesting that even with minimal management support, the quality manager (management consultant) need not despair. Insights from systems thinking may stimulate good results nevertheless.

References

Aho, A.V., Hopcroft, J.E. and Ullman, J. (1983) Data structures and algorithms, Addison-Wesley, USA.

AIS (2007) Association for Information Systems, Design Research in Information Systems, http://www.isworld.org/Researchdesign/drisISworld.htm

ASQ (2007) American Society for Quality, Discussion Boards, Complexity Theory in Quality, http://www.asq.org/discussionBoards/forum.jspa?forumID=50

Beckford, J. (2002) Quality: A Critical Introduction. Second edition, Routledge, UK.

Bjerknes, G., Ehn, P., & Kyng, M. (Eds.), Computers and Democracy - A Scandinavian Challenge, Aldershot, UK.

Braa, J., Monteiro, E. and Sahay, S. (2004) 'Networks of Action' MIS Quarterly, Vol 28 pp 1-26.

Checkland, P. (1981) Systems Thinking, Systems Practice, Wiley, UK.

Crosby, P.B. (1979) Quality is Free, Signet, USA.

Csikszentmihalyi, M. (1992) Flow: The Psychology of Happiness, Rider, USA.

Deming, E.W. (1986) Out of the Crisis, MIT Press, USA.

Deming, E.W. (1992) The New Economics, MIT Press, USA.

Dijkstra, E.W. (1968) 'Go To Statement Considered Harmful' Communications of the ACM, Vol 11 No 3 pp 147-48.

Davis, J.D. and Hersh, R. (1986). Descartes' Dream: The World According to Mathematics, HBJ, New York.

Dooley, K., Johnson, T. and Bush, D. (1995) 'TQM, chaos, and complexity' Human Systems Management, Vol 14 No 4 pp 1-16.

Flood, R.L. (1993) Beyond TQM, John Wiley and Sons, UK.

Fujimoto, T. (1999) The Evolution of a Manufactoring System at Toyota, Oxford University Press, USA.

Ghezzi, C. and Jazayeri, M. (1997) Programming Language Concepts, Wiley, USA.

Juran, J.M. (2006) Juran on Quality by Design: The New Steps for Planning Quality into Goods and Services, Free Press, USA.

Koch, R. and Godden, I. (1996) Managing without Management: A Post-Management Manifesto for Business Simplicity, Nicholas Brealy Publishing, UK.

Krick, E.V. (1965) An Introduction to Engineering and Engineering Design, Wiley, USA.

Reason, P. and Bradbury, H. (2006) Handbook of Action Research: Concise Paperback Edition, Sage, UK.

Schein, E. (1992) Organisational Culture and Leadership, Pfeiffer Wiley, USA.

Schlickman, J. (2003) ISO 9001:2000 Quality Management System Design, Artech House Publishers, USA.

Seddon, J. (2000) The Case Against ISO 9000: How to Create Real Quality in your Organisation, Oak Tree Press, UK.

Simon, H. (1996) The Sciences of the Artificial, MIT Press, USA.

Tricker, R. (2005) ISO 9001:2000 for Small Businesses, Butterworth-Heinemann, UK.

Weinberg, G. (1971) The Psychology of Computer Programming, Dorset House, USA.

Wikipedia (2007) COBOL, http://en.wikipedia.org/wiki/Cobol