Transactions on Information and Communications Technologies vol 8, © 1994 WIT Press, www.witpress.com, ISSN 1743-3517

# The software process - what it is, and how to improve it

S. Zahran
*Digital Consulting, Digital Equipment Corporation, The Crescent, Jays Close, Basingstoke, RG22 4BS, UK*

ABSTRACT

The work of W. Edwards Deming has convinced industry that it must first measure quality and then emphasise process to improve quality. In response to Deming's arguments and in light of the perception that the software industry is unable to produce quality products on schedule, and within budget, more software development organisations are now emphasising process measurement, monitoring, and assessment. This paper describes what is meant by the software process and discusses an approach for its measurement and improvement.

## INTRODUCTION

Unreliable software makes big news, from emergency services disasters to social security payment blunders. Improved software quality is essential to ensure reliable products and services, and gain customer satisfaction. While software development has existed for more than four decades, we failed so far to make it as an industry and as an engineering discipline rather than a craft.

Developing reliable and usable software that is delivered on time and within budget still represents a difficult endeavour for many organisations. As the role of software becomes increasingly critical for businesses as well as for human lives, the problems caused by software products that are late, over budget, or that do not work, become magnified. If lives are lost or people inconvenienced due to incapable computer software, the news media is there to make big stories. Organisations are realising that their fundamental problems is the immaturity of their software process.

Robert Lai (1993) proposes that the process improvement is the second maturity wave of the software industry. He states that *"the first wave of software was developed using the waterfall model in the 1970's. Today we are*

*in the midst of a second wave - a maturity movement - as we attempt to formally define the development process and the best ways to continuously improve it"* (Lai 1993).

Taking the lead in this area has been Watts Humphrey (Humphrey 1989, 1990, 1991), and the Software Engineering Institute (SEI) at Carnegie Mellon University. The SEI started in 1986 to develop a process maturity framework to help organisations appraise the maturity of their software process, and to provide a guidance for organisations to improve their software process capability. A brief description of the framework was released in September 1987, including a maturity questionnaire. The SEI evolved the model and questionnaire into the 5 -Level Capability Maturity Model (CMM) in 1991. In February 1993, it released the CMM version 1.1 (Paulk et al 1993).

This paper discusses the software process and describes the capability maturity model (CMM). It also shows how the CMM can be used as a basis to measure maturity of the software process of an organisation and plan its improvement.

## SOFTWARE QUALITY AND PROCESS IMPROVEMENT

In his book "Quality is Free" Phil Crosby states that: *"Quality is free. It is not a gift, it is free. What cost money are the unquality of things - all the actions that involve not doing the jobs right the first time"* (Crosby 1980). If such statement is true for many disciplines, it is particularly true for software. The evidence is abundant in the number of software products which exceeded their budget, were produced late, failed to satisfy the user requirements, and are full of bugs. The demand for improved software quality is increasing to ensure reliable products and services.

The benefits of improved quality comes in the form of reduction in failure costs. For software projects failure costs include (DTI 1992) :

- costs of correcting defects, both before and after delivery
- overruns against time and budget
- unnecessary high maintenance costs
- indirect costs which users incur due to poor quality software

The link between process maturity and software quality is expressed in the premise that *"The quality of the software system is governed by the quality of the process used to develop and maintain it"*. One stumbling block to improving software quality seems to be that not enough attention is paid to the overall development process itself. While software professionals typically devote their time to developing, testing or documenting software products, no one has prime responsibility for improving the software process. Experience has shown that if

no one is working on the software process, orderly improvement is unlikely. The process certainly won't improve itself, rather, most likely, it will deteriorate over time. Continuous improvement can occur only if a process infrastructure is in place.

Watts Humphrey argues in an early article published in Datamation, April 1989 that: *"Without work on the process, there will be little or no progress in improving software".*

## THE SOFTWARE PROCESS

Adopting a process view of software development represents a revolutionary change in perspective. A process orientation to software development involves elements of structure, focus, measurement, ownership, skills, and supporting technology.

In this section we investigate what is meant by the software process.

### What Is The Software Process

According to Webster's dictionary, a process is *"a system of operations in producing something .. a series of actions, changes, or functions that achieve an end result".* Chambers Concise Dictionary defines a process as *"a series of actions or events .. a sequence of operations or changes undergone".* The IEEE defines a process as *"a sequence of steps performed for a given purpose".* In the general business context, a process is defined as *"a structured, measured set of activities designed to produce a specified output for a particular customer or market"* (Davenport 1993).

These definitions put a strong emphasis on "HOW" work is done, in contrast to a product focus's emphasis on "WHAT". Accordingly, a process can be considered as *"a specific ordering of work activities across time and place, with a beginning, an end, and clearly identified inputs and outputs: a structure for action".*

In this paper we will adopt the following definitions quoted in (Humphrey 1990, Paulk et al 1993) are intended to encompass software throughout its life, which covers new development, enhancement, and repair.

The software process is *"The set of activities, methods, and practices used in the production and evolution of software".*

The software engineering process is *"The total set of software engineering activities needed to transform a user's requirements into software".*

## People and the Software Process

Software development is still a people-intensive activity. Talented people are the most important element in any software organisation. Even if you get the best people available, if they do not follow a common process, if everyone wrote in different programming languages, used different conventions, or didn't co-ordinate their design and code changes with their peers, the results will be chaos. Successful software organisations have learned that even the best professionals need a structured and disciplined environment in which to do cooperative work. Software organisations that do not establish such disciplines condemn their people to endless hours of repetitively solving technically trivial problems. The obvious fact is that attracting the best people is vital, but it is also essential to support them with an effectively managed software process.

## Technology and the Software Process

Another myth is the widespread belief that some technologically advanced tool or method will provide a magic answer to the software crisis. This is not only wrong, but it is dangerous. Organisations which jumped on the bandwagon of CASE tools, and ended up in failure and wasted time and effort have learned their lesson the hard way. Just ask yourself before introducing technology: What do I want to automate? In the absence of a defined, practised, and managed process, introducing automation can lead to increased chaos. There are several factors which limit the effective use of software technology: an ill-defined process, inconsistent process implementation, and poor process management. Software technology cannot be fully effective until these problems have been properly addressed.

## The Need for a Defined Process

If no effort is made to define and enhance the software development process across the whole organisation, each software development project latches on to its own tools, methods, and practices with little guidance available on how to use them. This ad hoc approach will not be sufficient to tackle the task of developing complex software systems. The goal of software process management is to enable organisations to produce software that meets cost, schedules, and quality objectives. The principles are the same that underpins statistical process control-principles that have been successfully used in controlling scientific experiments and in high -volume manufacturing operations. Statistical concepts have been found to be just as applicable to the software development as they are to the production of manufactured goods such as motor cars. Applying such concepts will only be possible if there is a defined and managed process.

## Software Process Models

A software process model is defined as *"One specific embodiment of a software process architecture"*. Most organisations have at least some policies, procedures, and standards. They generally follow some intuitive process models both prescriptively and descriptively. To be fully effective, these process models should be explicit and should relate to each other. These factors define the level of process maturity of the organisation.

A complete process model needs to contain functional, behavioural, structural, and conceptual views (Kellner et al 1988). However, for process management purposes, a simpler process model is appropriate as long as it does not artificially constrain process execution. Most of the process-improvement approaches available today are organised either as models or as questionnaires based on models. Well known approaches to process modelling include:

i) The Software Engineering Institute's Capability Maturity Model (CMM), defined in (Paulk et al 1993) and discussed in this paper.
ii) Capers-Jones software measurement model, described in (Jones 1991).
iii) Model-based Process Assessment described in (McGowan et al 1993).
vi) IEEE standard for software life cycle processes, described in (IEEE 1988).

These models reflect the fact that process improvement is cyclic and continuous. They are suitable for use to support decision making for training and technology insertion, and to identify the current state for a software development organisation. The CMM is the most popular model to date.

Variations of the software process

Software engineering is not a routine activity that can be structured and regimented like a repetitive manufacturing or clerical procedure. Rather it is an intellectual activity that must dynamically adjust to the creative needs of the professionals and their tasks. A trade-off is thus required between the individual need for flexibility and the organisational need for standards and consistency. Some factors to be considered are:

i) The specific nature of the project Software projects have differences, so their software engineering processes must have differences as well.
ii) Different organisations have different needs Organisations and projects must define processes that meet their own unique needs.
iii) Project-specific characteristics The process used for a given project must consider the experience level of the members, current product status and the available tools and facilities.

## SOFTWARE PROCESS MATURITY

Setting sensible goals for process improvement requires an understanding of the difference between immature and mature software organisation. Let us contrast the characteristics and symptoms of an immature versus mature software organisations

### Immature Organisations

A genius description of an immature organisation is provided by Deming in his book Out of the Crisis (Deming 1986) as follows:

*"One gets a good rating for fighting a fire. The result is visible; can be quantified. If you do it right the first time, you are invisible. You satisfied the requirements. That is your job. Mess it up, and correct it later, you become a hero."*

This sums up the culture and behaviour in an immature organisation. A closer look at an immature software development organisation will reveal the following behaviours and symptoms:

- Software processes are generally improvised by practitioners and their management during the course of the project. Even if a software process has been specified it is not rigorously followed or enforced.
- Managers are usually focused on solving immediate crisis (fire fighting).
- Product functionality and quality are often compromised to meet schedules.
- There is no objective basis for judging product quality, therefore product quality is difficult to predict.
- Activities intended to enhance quality such as reviews and testing are often curtailed or eliminated when projects fall behind schedule.

### Mature software organisations

On the other hand mature organisations display most of the following characteristics:

- An organization-wide ability for managing software development and maintenance processes.
- Work activities are carried out according to the defined and planned process.
- Roles and responsibilities within the defined process are clear throughout the project and across the organisation, and the software process is accurately communicated to both existing staff and new employees.

- Managers monitor the quality of products and the processes that produced them, there is an objective, quantitative basis of judging product quality and analysing problems with the product and the process.
- Schedules and budgets are based on historical performance and are realistic; the expected results for cost, schedule, functionality, and quality of the procedures are actually achieved.

In general, a disciplined process is consistently followed because all of the participants understand the value of doing so, and the necessary support infrastructure exists to support the process.

Software process maturity framework

Capitalising on the observations about immature and mature software organisations requires a construction of a software process maturity framework. This framework describes an evolutionary path from ad hoc, immature processes to mature disciplined software processes. Such framework is essential for process improvement programs to become effective. A maturity framework provides the necessary foundation for supporting successive improvements. The software process maturity framework presented in the SEI Capability maturity model (CMM) emerges from integrating the concepts of software process, software process capability, software process performance, and software process maturity. A definition of the software process maturity and some of related concepts is given in (Paulk et al 1993a) as follows:

Software process maturity *"is the extent to which a specific process is explicitly defined, managed, measured and effective"*. Maturity implies the potential for growth in capability and indicates both the richness and consistency with which software processes are applied across the organisation. As an organisation matures, the software process becomes better defined and more consistently implemented throughout the organisation.

Software process capability *"describes the range of expected results that can be achieved by following a software process"*. The software process capability of an organisation provides one means of predicting the most likely outcomes to be expected from the next software project the organisation undertakes.

Software process performance *"represents the actual results achieved by following a software process"*. Thus, software process performance focuses on the results achieved, while software process capability focuses on results expected.

The Capability Maturity Model (CMM) developed by the SEI provides an evolutionary path which defines the process improvement stages necessary for

an organisation to follow in order to increase the maturity of its software process. The CMM framework is also useful for assessing the degree of repeatability and measurability that an organisation has put into its development process. But it makes no attempt to assess the quality of the resulting product, or the appropriateness of any specific process model for the task.

## Process Maturity Levels

The staged structure of the CMM is based on principles of product quality that have existed for the last sixty years including the work by W. Edwards Deming (Deming 1986) and Juran (Juran 1984). The maturity framework into which these quality principles have been adapted was first inspired by Phil Crosby in his book Quality is Free (Crosby 1984). Crosby's quality management maturity grid describes five evolutionary stages in adopting quality practices. This maturity framework was adapted to the software process and brought to the Software Engineering Institute (SEI) by Watts Humphrey in 1986.

Humphrey added the concept of maturity levels, and developed the foundation for its current use throughout the software industry. The outcome took the form of the capability maturity model (CMM). The CMM provides a conceptual structure for improving the management and development of software products in a disciplined and consistent way. It also provides a framework for organising these evolutionary steps into five maturity levels that 'build' successive foundations for continuous process improvement. The basis for choosing the CMM's five maturity models have been chosen because they:

- Reasonably represent the actual historical phases of evolutionary improvement of real software organisations.
- Represent a measure of improvement that is reasonable to achieve from the prior level.
- Make obvious a set of immediate improvement priorities, once an organisation's status in this framework is known.

This section describes the CMM and shows how it delineates the characteristics of a mature, capable software process. The progression from an immature, unrepeatable software process to a mature, well-managed software process is also described in terms of the maturity levels in the model.

## Software Capability Maturity Model (CMM)

The capability maturity model (CMM) is an:

- Application of process management and quality improvement concepts to software development and maintenance.
- Guide for evolving toward a culture of engineering excellence.

- Underlying structure for reliable and consistent assessments.
- Framework for continuous process improvement.

The CMM represents a common-sense engineering approach to software process improvement. The CMM concepts and structure have been extensively discussed and reviewed within the software community. While the CMM is not perfect, it does represent a broad consensus of the software community and a useful tool for guiding software process improvement efforts.

The CMM organises the process improvement steps into five maturity levels as shown in figure 1. These five maturity levels define an ordinal scale for measuring the maturity of an organisation's software process and for evaluating its software capability. The levels also help an organisation prioritise its improvement efforts.

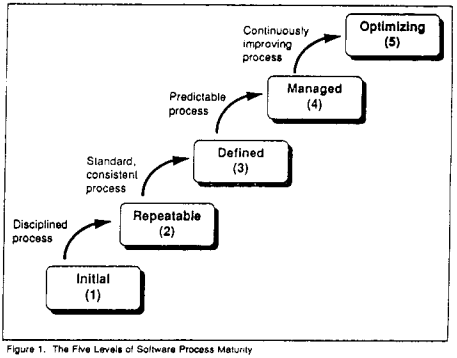


Figure 1. The Five Levels of Software Process Maturity

A maturity level is a well-defined evolutionary plateau toward achieving a mature software process. Each level comprises a set of process goals that, when satisfied, stabilise an important component of the software process. This in turn results in an increase in the process capability of the organisation. Organising the CMM into the five levels shown, prioritises improvement actions for increasing software process maturity. The 5 levels are defined as follows:

*Level-1* Initial : At the Initial Level, the software process is characterised as ad hoc, and occasionally even chaotic. Few processes are defined, and success depends on individual effort. Until the process is under statistical control, orderly progress in process improvement is not possible. While there are many degrees of statistical control, the first step is to achieve rudimentary predictability of schedules.

*Level-2* Repeatable : At the Repeatable Level, basic project management processes are established to track cost, schedule, and functionality. The necessary process discipline is in place to repeat earlier successes on projects with similar applications. The organisation has achieved a stable process with a

repeatable level of statistical control by initiating rigorous project management of commitments, cost, schedules, and changes.

*Level-3* Defined : At the Defined Level, the software process for both management and engineering activities is documented, standardised, and integrated into a standard software process for the organisation. All projects use an approved, tailored version of the organisation's software process for developing and maintaining software. The organisation has defined the process as a basis for consistent implementation and better understanding. At this point advanced technology can be usefully introduced.

*Level-4* Managed : At the Managed Level, detailed measures of the software process and product quality are collected. Both the software process and products are quantitatively understood and controlled. The organisation has initiated comprehensive process measurements and analysis. This is when the most significant quality improvements begin.

*Level-5* Optimising : At the Optimising Level, continuous improvement is enabled by quantitative feedback from the process and from piloting innovative ideas and technologies. The organisation now has a foundation for continuing improvement and optimisation of the process.

Internal structure of the CMM

There is more to the CMM than the maturity levels. Figure 2 illustrates the CMM internal structure. As illustrated each maturity level decomposes into its constituent parts, which range from abstract summaries of each level down to the operational definition where each maturity level is composed of a number of key process areas (KPA's).

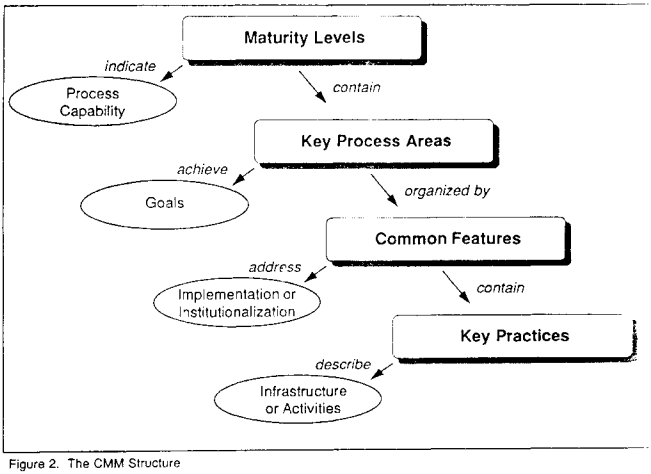


Figure 2. The CMM Structure

A key process area is a cluster of related activities that, when performed collectively, achieve a set of goals considered important for enhancing process capability. Each key process area is organised into five sections called the common features. The common features specify the key practices that, when collectively addressed, accomplish the goals of the key process area. The key process areas residing at the specific maturity levels are shown in figure 3. To achieve a maturity level, the key process areas for that level must be satisfied. To satisfy a key process area, each of the goals for the key process area must be satisfied.

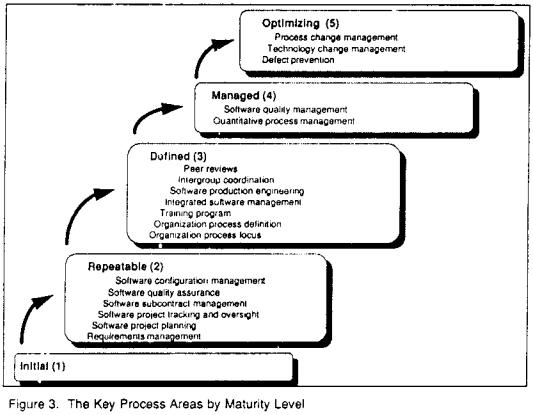


Figure 3. The Key Process Areas by Maturity Level

Key process areas of the CMM represent one way of describing how organisations mature. These key process areas were defined based on many years of experience in software engineering and management over the SEI's five years of experience with software process assessments and software capability evaluations. Every key process area has goals associated with it.

The goals summarise the key practices of a key process area. They can be used to determine whether an organisation or project has effectively implemented the key process area. The goals signify the scope, boundaries, and intent of each key process area. All the goals of a key process area must be achieved for the organisation to satisfy that key rocess area. When the goals of key process area are accomplished on a continuing basis across projects, the organisation can be said to have institutionalised the process capability characterised by the key process area.

## Uses of the CMM

There are at least four possible uses of the CMM :

- Assessment teams use of the CMM to identify the strengths and weaknesses of the organisation.

- Evaluation teams use of the CMM to identify the risks of selecting among different contractors for awarding business and to monitor contracts

- Managers and technical staff use of the CMM to understand the activities necessary to plan and implement a software process improvement program for their organisation

- Process improvement groups use of the CMM as a guide to help them define and improve the software process in their organisation.

The CMM on its own does not guarantee that software products will be successfully built, or that all problems in software engineering will be adequately resolved. However, current reports from CMM-based process improvement programmes indicate that it can improve the likelihood with which a software organisation can achieve the cost, quality and productivity goals. Examples of the results of such improvements can be found in (Dion 1992), (Humphrey et al 1991), (Lipke et al 1992), and (Kitson et al 1992).

## SOFTWARE PROCESS ASSESSMENT (SPA)

The CMM and its process maturity structure is intended for use with an assessment methodology along the lines described below.

### Assessment objectives

Process assessment helps software organisations improve themselves by identifying their critical problems and establishing improvement priorities. The basic assessment objectives are:

- To learn how the organisation works,
- To identify its major problems,
- To enrol its opinion leaders in the change process.

Assessment helps organisations to focus first on the problems before jumping to solutions. Without preliminary problem analysis, the "solutions" are seldom effective. It helps identify an organisation's specific maturity status, and the management actions for implementing the priority improvement once its position in this maturity structure is defined. The organisation can concentrate on those items that will help its advance to the next level. A definition of the software process assessment is:

*"Software process assessment is an appraisal by a trained team of software professionals to determine the state of an organisation's current software process, to determine the highest priority software process-related issues*

*facing an organisation, and to obtain the organisational support for software process improvement"* (Paulk et at 1993).

Assessment phases

Assessments are typically conducted in three phases: preparation, assessment, and recommendations. These three phases are:

*Phase 1*: *Preparation* The objective of the preparation phase is to gain senior management commitment to the process, by agreeing to participate personally, and commit to take action on the resulting recommendations. This phase concludes with a brief two to three day training program for the assessment team.

*Phase 2: On-site Assessment* This activity typically takes several days, although it can take two to three weeks, depending on the size of the organisation and the assessment technique used. This phase concludes with a preliminary report of the findings to local management.

*Phase 3: Recommendations* In this phase, findings and action recommendations are presented to the local managers. A local action team is then assembled to plan and implement the recommendations.

The SEI has developed an assessment method along these lines and developed an assessment questionnaire derived from the CMM model. Figure 4 illustrates a typical SEI Assessment schedule.

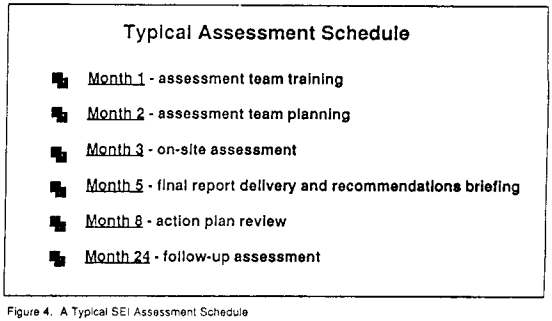


Figure 4. A Typical SEI Assessment Schedule

This schedule is for SEI-authorized assessment. For self assessments and non-SEI authorised assessments, this schedule can be modified to match the specific management objectives and the organisation requirements.

Assessment Principles

The basic requirements for a good assessment are a competent team, sound leadership, and a cooperative organisation. However, because the software

process is human-intensive, the following special considerations, when observed should contribute to the success of the assessment:

- Start with a Process Model as a Basis for the Assessment
- Observe Strict Confidentiality
- Involve Senior Management
- Keep an Open mind and Respect People's Views
- Focus on Action

Finally, it should be noted that software process assessment is different from software capability evaluation. The latter is defined as follows:

Software capability evaluation is *"an appraisal by a trained team of professionals to identify contractors who are qualified to perform the software work or to monitor the state of the software process used on an existing effort"* (Paulk et at 1993).

## SOFTWARE PROCESS IMPROVEMENT (SPI)

As a software organisation improves its maturity, it institutionalises its software process via policies, standards, and organisational structures. Institutionalisation entails building an infrastructure and a corporate culture that supports the methods, and procedures of the business so that they endure after those who originally defined them have gone.

### Process Improvement Steps

Continuous process improvement is usually based on many small, evolutionary steps rather than revolutionary innovations. The CMM provides a framework for organising these evolutionary steps. To successfully improve software capabilities, organisations must have a precise picture of their ultimate goal and some way to gauge progress along the way. The following six steps are suggested as landmarks on the road to improvement:

1. Understand the current status of the software development process. This can be achieved through software process assessment (SPA).
2. Develop a vision of the desired process.
3. Establish a prioritised list of process improvement actions.
4. Produce a plan to accomplish these process improvement actions, through a process improvement programme (PIP).
5. Commit the resources, execute the plan, and track progress against the plan.

6. Assess the results, update the goal and start all over again.

These steps form iterative procedure where the process is continuously improving. Every iteration should result in an increasing improvement building on the previous iteration.

Return on Investment (ROI) in Process Improvement

It is not easy to prove to senior management in terms of hard numbers the ROI for process improvement effort. This is due to the following evident reasons:

- There is a need for sometime to pass before measurements are available for general publication. Since the process maturity movement is relatively recent, such historical data are not available.
- Many benefits that result from process improvement are qualitative rather than quantitative. It is easy to challenge such improvements.

There are two well publicised cases in the literature discussing real life experiences of practical process improvement initiatives at Hughes Aircraft (Humphrey et al 1991), and Raytheon (Dion 1992, 1993). The industry still needs a convincing business case for process improvement based on practical results of real life cases. This should help us convince management of the value of investing in process improvement. Developing such business cases will require champions and pioneers who believe in process maturity and who are dedicated enough to measure the improvements over a period of time and publish the results to the whole industry. This should help in making the process maturity movement take off, and the second wave of the software industry becoming a reality.

REFERENCES

(Aguayo 1991) Rafael Aguayo,"Dr Deming",Fireside-Simon & Schuster,1991.

(Boehm 1988) Barry Boehm,"A Spiral Model for Software Development and Enhancement", IEEE Computer, 1988.

(Crosby 1980) Phil Crosby,"Quality Is Free",McGraw Hill,USA,1980.

(Crosby 1984) Phil Crosby, "Quality Without Fears" ,McGraw Hill,N.Y.,1984.

(Davenport 1993) Thomas Davenport, "Process Innovation", Harvard Business School Press, Boston, Mass.,1993.

(Deming 1986) W.E. Deming,"Out of the Crisis", Cambridge University Press,1986.

(Dion 1992) R. Dion,"Elements of a Process Improvement Programme",IEEE Software, July 1992, pp.83-85.

(Dion 1993) R. Dion,"Process Improvement and the Corporate Balance Sheet",IEEE Software, July 1993,pp.29-35.

(DTI 1992) Department of Trade & Industry, UK, "TickIt, A guide to Software QMS Construction using ISO 9001/EN 29001/BS 5750), Feb.1992.

(Gardner 1965) John Gardner,"Renewal of Organisations",20th Annual Meeting of the Board of Trustees, Midwest Research Institute,Kansas City,MO,May 3,1965.

(Humphrey 1989) Watts Humphrey,"Improving the Software Process",DATAMATION,April 1,1989.

(Humphrey 1990) Watts Humphrey,"Managing the Software Process", Addison-Wesley,1990.

(Humphrey et al 1991) W.S. Humphrey, T.R. Snyder & R.R. Willis,"Software Process Improvement at Hughes Aircraft",IEEE Software,July 1991, pp. 11-23.

(IEEE 1988) Institute of Electrical and Electronics Engineers, Inc., "Standard for Software Life Cycle Processes", P1074/D2.1, Dec. 1988.

(Jones 1991) T.Capers Jones,"Applied Software Measurement", McGraw Hill,1991.

(Juran 1984) J.M. Juran,"Juran on Planning for Quality",The Free Press, N.Y.,1984.

(Kellner et al 1988) M. I. Kellner and G. A. Hansen,"Software Process Modelling",Technical Report CMU/SEI-88-TR-9,Software Engineering Institute, Carnegie Mellon University,May 1988.

(Kitson et al 1992) D.H. Kitson and S. Masters,"An Analysis of SEI Software Process Assessment Results:1987-1991",Tech. Report CMU/SEI-92-TR-24, Software Eng. Institute,Pittsburgh,1992.

(Lai 1993) Robert Lai,"The Move to Mature Process",In IEEE Software,July1993.

(Lipke et al 1992) W.H. Lipke and K.L. Butler,"Software Process Improvement: A Success Story",Crosstalk,Nov. 1992,pp. 29-31.

(McGowan et al 1993) C.L. McGowan and S.A. Bohner,"Model Based Process Assessments", Proc. Int'l Conf. Software Eng.,IEEE CS Press, Los Alamitos, Calif.,1993,pp.202-211.

(Osterweil 1987) Leon Osterweil,"Software Processes are Software Too",Proc. Int'l Conf. Software Engineering,IEE CS Press,1987,pp. 2-13.

(Paulk et al 1993a) M.C.Paulk, et al,"Capability Maturity Model for Software,Version 1.1",Tech. Report CMU/SEI-93-TR-24,Software Engineering Institute,Pittsburgh,1993.

(Paulk et al 1993b) M.. Paulk, Bill Curtis, M.B. Chrissis & C. Webber,"Capability Maturity Model, Version 1.1",IEEE Software,July 1993,pp. 18-27.

(Royce 1970) W. Royce,"Managing the Development of Large Software Systems", Proc. Wescon,IEEE Press,New York,1970.

(Schultz 1988) H.P. Schultz,"Software Management Metrics", Tech. Report, Mitre Corp.,Mass.,1988.

(Toffler 1981) A. Toffler,"The Third Wave",Bantam Books,New York,1981.