STRATEGIC QUALITY -

A SOFTWARE ENGINEERING APPROACH

by

ROELOF JOHANNES VAN STADEN

DISSERTATION

submitted in partial fulfillment
of the requirements for the degree

MAGISTER INGENIERIAE

in

ENGINEERING MANAGEMENT

in the

FACULTY OF ENGINEERING

at the

RAND AFRIKAANS UNIVERSITY

SUPERVISOR: PROF L PRETORIUS
CO-SUPERVISOR: PROF JHC PRETORIUS

DECEMBER 2004

Abstract

Software engineering organizations face a struggle for daily survival in an extremely volatile climate. Numerous times it has been shown that the quality of a service or product could make the difference between an organization existing or closing down. The way in which quality is approached in any organization is part of a strategy; unbeknown to the managers and employees in many instances. Even though there are numerous books, articles, internet sites and other sources devoted to the subject of quality, total quality management, and strategic planning, not many of these information sources link quality and the strategy of the organization in such a way as to consider the quality of the organization's products and services to be a major factor of strategic survival – or even the very existence of the organization.

Quality is known under various names and terms in organizations, these terms and definitions will be investigated to grasp the true meaning of software quality and strategy as it concerns modern software engineering organizations. The tools and techniques required to improve and measure strategy and quality will also be scrutinized. One dominant factor about quality and strategy should be borne in mind, and that is that these programs depend on teamwork and management support as the major underlying framework.

There are many tools and techniques that leaders and members of software teams can employ, but one of the most important factors is to gain a picture of the total process of continuous improvement and measurement. For this reason the author has included a large section on Jack Welch, who managed to use continuous improvement techniques to create one of the best, and biggest international organizations in recent years. It is vital that all people realize that they need an improvement and measurement model, and they need guidance in using such a model.

This dissertation investigates the reasoning behind implementing strategic quality processes in software engineering organizations. Following the investigation into the necessity for a quality strategy, the methods, processes, tools and techniques that are required for a strategic quality framework (improvement and measurement model) for software engineering organizations will be researched to provide a basic framework and guidance in implementing such a model.

Opsomming

Sagteware ontwikkelingsorganisasies word daagliks in die gesig gestaar deur ‘n geveg om oorlewing in ‘n baie onstuimige ekonomiese klimaat. Menige male is dit al bewys dat die kwaliteit van ‘n diens of produk die verskil kan maak tussen ‘n organisasie wat vandag bestaan, of môre hulle deure sluit. Die manier waarop kwaliteit benader word in enige organisasie is deel van ‘n strategie; of die bestuur en werknemers hiervan bewus is of nie. Selfs al is daar baie boeke, artikels, webwerwe and ander bronne inligting beskikbaar oor kwaliteit, kwaliteitsbestuur, strategiese beplanning en bestuur, is daar relatief min gedoen om die kwaliteit van die organisasie se dienste en produkte te beskou as ‘n kritiese faktor van strategiese oorlewing – nog minder oorlewing van die organisasie.

Kwaliteit is bekend onder verskeie vaandels, en hierdie terme en definisies is ondersoek om die ware betekenis van sagteware kwaliteit en strategie te verstaan, soos dit betrekking het op moderne sagteware ontwikkelings organisasies. Die gereedskap en tegnieke om strategie en kwaliteit te verbeter en meet het ook onder die vergrootglas gekom. Een van die dominante faktore omtrent kwaliteit en strategie wat in gedagte gehou moet word, is dat al hierdie programme aandring op spanwerk en ondersteuning van die bestuur as die onderliggende raamwerk.

Daar is vele tegnieke en metodes wat leiers en spanlede van sagteware spanne kan inspan om die kwaliteit van sagteware te verbeter, maar een van die mees belangrikste faktore is om ‘n geheelbeeld van die totale proses van kontinue verbetering en meting te verkry. Om hierdie rede het die skrywer ‘n groot gedeelte gewy aan die korporatiewe strateeg Jack Welch, wat kontinue verbeteringstegnieke gebruik het om een van die beste en grootste internasionale organisasies in onlangse jare te skep. Dit is belangrik dat mense besef dat hulle ‘n verbeterings- en metingsmodel nodig het, en dat hulle leiding kry om so ‘n model toe te pas.

Hierdie verhandeling ondersoek die denkwyse agter strategiese kwaliteitsprosesse in sagteware ontwikkelingsorganisasies. Die metodes, tegnieke, en prosesse wat nodig is vir ‘n strategiese kwaliteitsraamwerk word ook ondersoek vir sagteware ontwikkelingsorganisasies om ‘n basiese raamwerk en aanwysings te verskaf om so ‘n model toe te pas.

Dedication

To all engineers working in the software engineering arena, trying to make things better for all.

"Our circuits are so overloaded by the technology that keeps us wired – the cell phones and modems and fax machines – that we've lost the capacity to daydream." – James Atlas

Table of Contents

Appendices.....149

List of Figures

List of Tables

About the Author

Roelof van Staden has been involved in the information technology industry since 1998. He holds a Bachelors degree (honours level) in Electrical and Electronical Engineering from the Rand Afrikaans University, in Johannesburg, South Africa. He has dealt with numerous applications of computing, including military applications, banking systems, asset management systems, warehouse and logistics systems, barcode and wireless software applications, enterprise resource planning applications, and high-throughput data feeds and database management systems.

At the time this dissertation was in progress, the author was in the employ of one of the major corporate and investment banking groups as a quality assurance manager for their global front office technology department. His daily tasks involved developing test plans, creating software configuration management models, implementing test plans, software reviews, and management meetings amongst other things. The author realized that the need for high quality software has become more and more evident as the business needs changed – poor quality software will only result in poor business results. No longer just another support function, the software engineering process became a strategic objective.

Preface

Having worked as a software developer, software engineer and a quality assurance manager, all in the past couple of the years, the author has come to realize that the issues of quality management, quality software, and above all strategy, are sorely lacking in the software industry in Southern Africa. The passion for a more coordinated approach to the issues of quality in the software engineering profession has led to the drive behind this research dissertation.

The topics that are covered have been selected in such a way as to answer questions people ask frequently about organizational strategy, software and quality. This dissertation aims to put solid knowledge and information in the place of many myths that abound in the software engineering industry.

Some of the chapters cover practical issues and provide step-by-step suggestions for improving the quality and processes in your own environment. There are even answers to some of the more common questions that people ask, like why does software crash? What is software reliability? Software quality? How do we improve software quality? Is quality my responsibility? Is quality a strategic organizational objective?

At the end of this dissertation there is a large, annotated glossary that is more than just a handy reference of definitions; it includes some of the key concepts related to strategy and quality topics that may assist in the field of quality assurance.

The information in this dissertation is not a complete reference work on all matters related to software, quality, or strategy, but may equip the reader to make more sense of the claims and counterclaims in strategy, quality, software and some general management issues. One should be able to get more involved and proactive in quality management and general improvements in the software engineering industry, and even to make some informed decisions about your career if quality is your passion.

For more detailed information about a specific topic, the sources that are cited in the list of references (bibliography) may be consulted.

List of Abbreviations and Symbols

ABC	Activity Based Costing
AI	Artificial Intelligence
ANSI	American National Standards Institute
BCG	Boston Consulting Group
BPR	Business Process Re-Engineering
BSI	British Standards Institute
CAD	Computer-Aided Design
CAE	Computer-Aided Engineering
CASE	Computer-Aided Software Engineering
CC	Change Control
c.d.f.	Cumulative Distribution Function
CM	Configuration Management
CMM	Capability Maturity Model
EFQM	European Foundation of Quality Management
EVA	Economic Value Added
FMECA	Failure Mode Effects and Criticality Analysis
FTR	Formal Technical Review
GE	General Electric
IEEE	Institute for Electrical and Electronical Engineering
ISO	International Standards Organization

IT	Information Technology
KPA	Key Process Area
KPI	Key Performance Indicator
MBO	Management by Objectives
MTTF	Mean Time to Failure
p.d.f.	Probability Density Function
POFOD	Probability of Failure on Demand
QA	Quality Assurance
QC	Quality Control
QFD	Quality Function Deployment
QMS	Quality Management System
R&D	Research and Development
RFP	Request for Proposal
RH	Relative Humidity
ROCE	Return on Capital Employed
ROCOF	Rate of Occurrences of Faults
ROI	Return on Investment
RONA	Return on Net Assets
RS	Requirements Specification
RUP	Rational Unified Process
SBU	Small Business Unit

SD	Standard Deviation
SDLC	System Development Lifecycle
SEI	Software Engineering Institute
SLA	Service Level Agreement
SPC	Statistical Process Control
SQA	Software Quality Assurance
SRS	System Requirements Specification
STP	Straight Through Processing
SVA	Shareholder Value Analysis
SWOT	Strengths, Weaknesses, Opportunities, Threats
TQC	Total Quality Control
TQM	Total Quality Management
UML	Unified Modeling Language
WACC	Weighted Average Cost of Capital
WBS	Work Breakdown Structure
ZD	Zero Defects

INTRODUCTION

PROBLEM DEFINITION AND RESEARCH APPROACH

Chapter 1

Introduction and Research Approach

"The formulation of a problem is far more often essential than its solution, which may be merely a matter of mathematical or experimental skill. To raise new questions, new possibilities, to regard old problems from a new angle requires creative imagination and marks real advancement in science." – Albert Einstein

1.1 EXECUTIVE SUMMARY

There are numerous books, articles, and Internet sites, that are devoted to the subject of quality, total quality management, strategic planning and strategic management. Not many of these information sources link quality and the strategy of any organization in such a way as to consider the quality of the organization's products and services to be a major factor of strategic survival; even the very existence of the organization [83].

In the modern economy, organizations face a struggle for daily survival in an extremely volatile economic climate, yet it has been shown on numerous occasions that the quality of a service or product could make the difference between an organization existing today, or closing its doors this afternoon - permanently. Thus it only makes sense to investigate the influence that quality processes, procedures, methods, and in some instances just the word "quality", has on many organizations, and in particular software engineering organizations, and their strategies for economic survival.

Software engineering organizations have never faced a tougher challenge in the markets than the last couple of years. The huge losses that were experienced on the world stock markets during and after the year 2000 made investors and the general public very wary of the way in which they viewed the information technology industry [33]. The tech-stock losses led to major re-organization and re-engineering of processes. The next major hurdle for software engineering organizations will be to

align their core processes with the business processes, in order to be market leaders in order to achieve survival [15].

Quality is and probably always will be a hot topic in organizations. Quality is also known under various different names and terms, such as improvement and competitiveness. Most software engineering organizations have launched quality management programs under titles such as the quality advantage, quality improvement, quality systems, quality action, quality leadership and the like [40].

Quality programs such as total quality management (TQM) have produced significant benefits and have one very obvious and dominant characteristic. *These programs depend on teamwork as the major strategy and underlying framework.* Although the team-centered characteristic is the most dominant, there are other factors to be aware of [41]:

- Customer Satisfaction.
- Continuous Improvement.
- Assessment and Measurement.
- Supplier Performance.
- Environmental Support.
- Systems Support.
- Training.
- Innovation.
- Financial Implications.
- Ethics.
- Services.
- Products.

In order to produce a world-class quality strategy to enhance, strengthen, and improve the organization's competitive advantage, Kinlaw [44] states that:

> *"…there are many tools that people need in order to undertake improvement projects in a careful and systematic way. They need problem-solving tools, such as flow-charts and cause-and-effect diagrams. They need to know to create run charts, histograms and Pareto charts. They need statistical tools to control their systems. They need to know to plan and collect data and how to use check sheets."*

But beyond all these particular tools, the leaders and members of teams need a picture or overview of the *total* process of continuous improvement and measurement. They need a way to visualize the interconnectedness of all the improvements that they might make. In short, they need an improvement and measurement model and they need guidance in using such a model [41].

This dissertation aims to investigate the reasoning behind implementing strategic quality processes in software engineering organizations. Following the investigation into the necessity for a quality strategy, the methods, processes, tools and techniques that are required for a strategic quality framework methodology, (improvement and measurement model), for software engineering organizations will be researched to provide a basic framework and guidance in implementing such a methodology.

1.2 RESEARCH OBJECTIVES

The purpose of this research into strategic quality planning is to discover what the benefits are to be derived from implementing quality processes and methods as part of a software engineering organization's strategic objectives. Strategic planning is a management function aimed at the improvement of the organizational functions and products [46], [68], [83].

Specifically, the author intends to investigate the tools and techniques for implementing a quality strategy. Methods to determine whether the gains are determinable, actionable, measurable and specific to any type of software engineering organization will be discussed. After describing the quality strategy tools and techniques, the author will illustrate how one can go about implementing a quality strategy for a software engineering organization.

1.3 THE RESEARCH PROCESS

The author approached the research process in this dissertation by following the sequential process that Cooper and Schindler outline in Business Research Methods [11]. The book points out that:

> *"Writers usually treat the research task as a sequential process involving several clearly defined steps. No one claims that research requires completion of each step before going to the next. Recycling, circumventing, and skipping occur."*

Accordingly some steps in the research process were given more attention than other steps, returning to some stages in the research process to clarify issues.

The research process was approached in this dissertation by using a simplistic version of the question hierarchy as proposed by Cooper and Schindler. The research approach was initiated by the discovery of a significant management dilemma that led to a management question. The management question was then used to state the management dilemma in terms of a question, to facilitate the solution of the management dilemma.

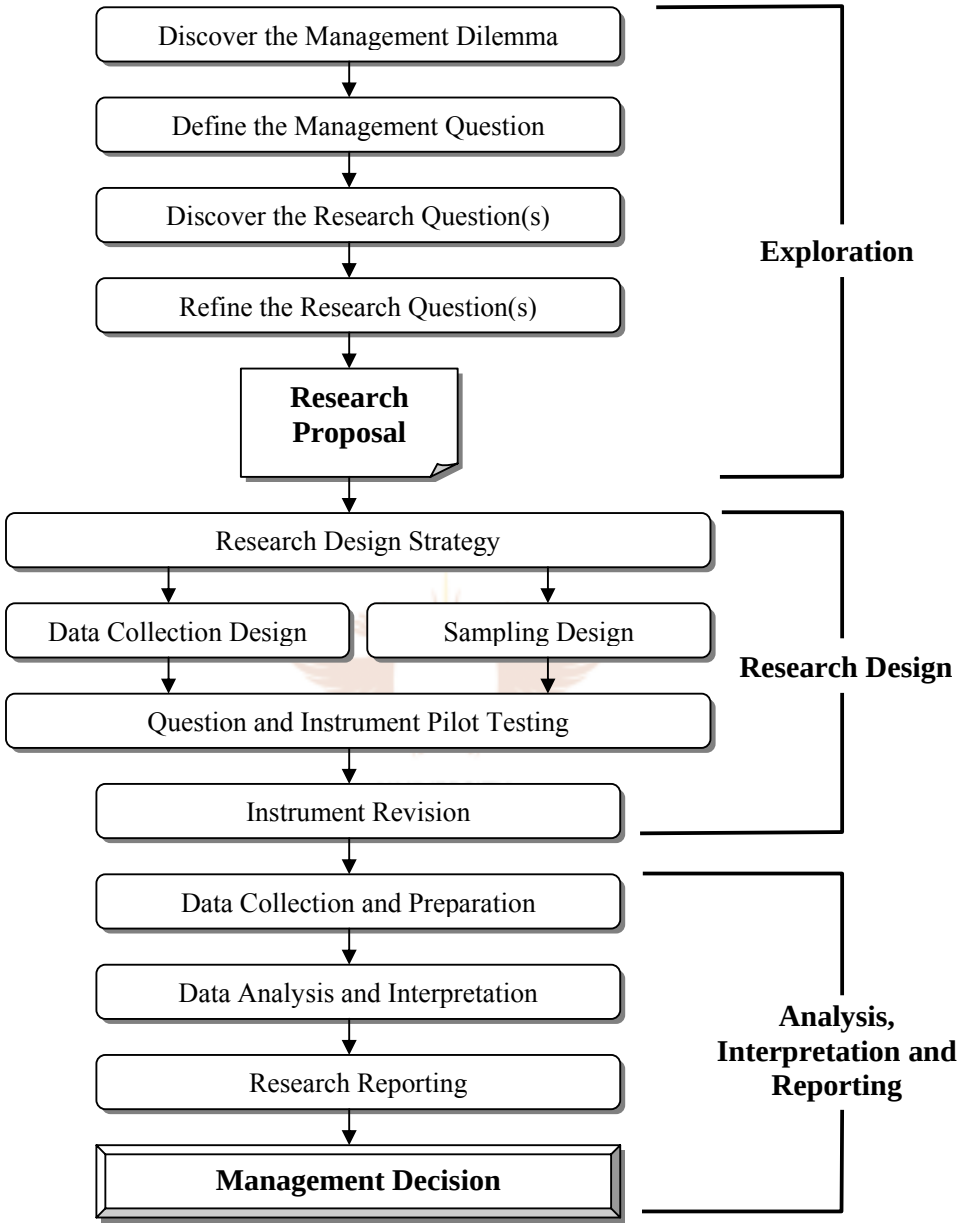
The management question led to the research questions, which needed to be refined to such a level as to be clear and concise enough to provide the guidance for the management research. Only once the research questions were formulated, and the relevant parties to the management dilemma agreed that the formulated research questions are factually correct, could the research proposal be generated. This approach is illustrated in Figure 1.1.

The management question is of paramount concern when approaching any research or investigation; as Cooper and Schindler [11] points out:

> *"In our view of the research process, the **management question** – its origin, selection, statement, exploration and refinement – is the critical activity in the sequence."*

This statement is extremely important to remember, as the title of the dissertation is a direct reflection of the management question, and all the research and effort that has

gone into preparing this dissertation focuses exclusively on the highlighted management question – strategic quality.

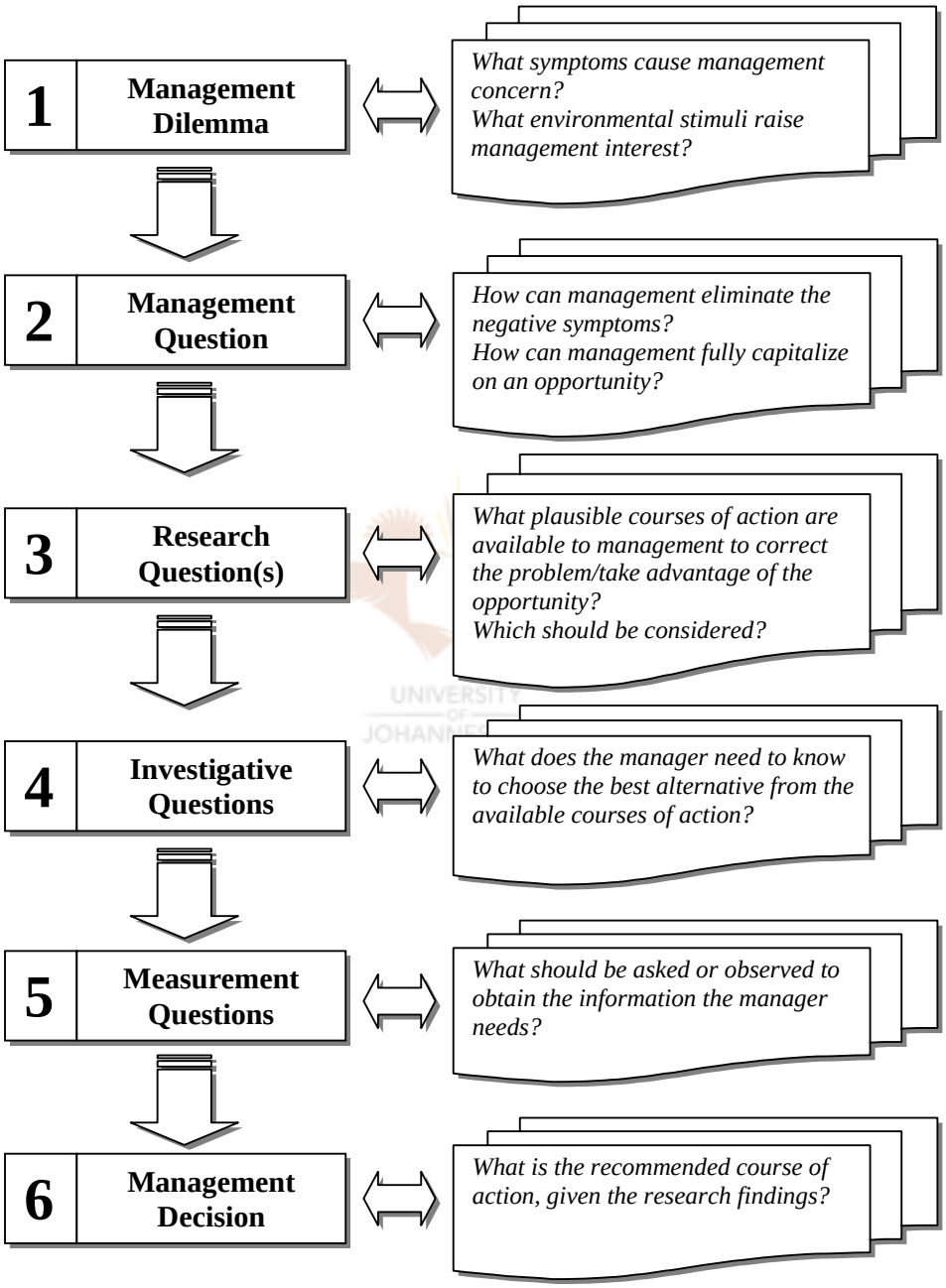


Source: Adapted by the author from Cooper and Schindler [11], Exhibit 3-1

Figure 1.1: The Research Process

The management question was used to restate the management dilemma in question form, in order to enable one to proceed with the research process. Management questions can be broadly divided into three categories:

- Choice of purpose or objectives.
- Generation and evaluation of solutions.
- Troubleshooting or control situation.



Source: Adapted by the author from Cooper and Schindler [11], Exhibit 3-2

Figure 1.2: The Management-Research Question Hierarchy

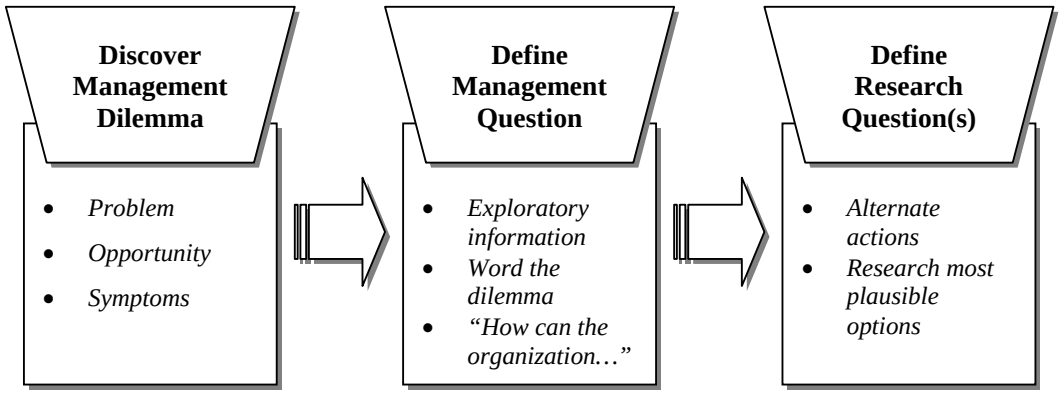
Figure 1.2 illustrates the management-question hierarchy that the author followed to produce answers to the management question that was posed.

The author approached the management question very cautiously, as there is a tremendous amount of literature focused on software engineering, software quality and reliability and even strategic management. The difficult part was in establishing the link between management strategy and software quality. Therefore most of the focus in the research process and even research design sections were to ensure that the management question was understood fully and to ensure that a good understanding of the topic, as well as the potential myths and pitfalls were outlined.

According to Cooper and Schindler [11]:

> *"A **research question** is the hypothesis of choice that best states the objective of the research study. It is a more specific management question that must be answered. It may be more than one question, or just one. A research process that answers this more specific question provides the manager with the information necessary to make the decision he or she is facing."*

The author utilized the first three steps in the management-research question hierarchy as depicted in Figure 1.2 to arrive at the research question, as can be shown in Figure 1.3:



Source: Adapted by the author from Cooper and Schindler [11], Exhibit 3-3

Figure 1.3: Formulating the Research Question

Following the basic research question methodology that Cooper and Schindler [11] depicted in Figure 1.3, the author defined the management dilemma, management question and research question as follows:

Management Dilemma

- Lots of talk about quality without any real commitment to quality programmes.
- Lack of understanding about what quality really means.
- Quality processes do not form part of the normal product development cycle.
- Quality is not seen as a "real" business function.
- Also, quality not seen as an income generating function.
- Quality is more of a sales tactic, than a real strategic tool.
- Speculation that improved quality will not necessarily lead to improved sales and profits.

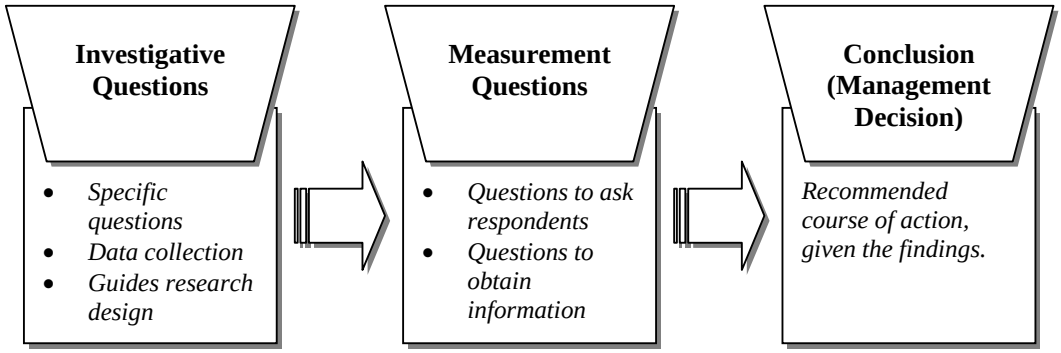
Management Question

Should quality, and quality programmes become part of the organizational strategy, and if so, how should the implementation of quality programmes in software engineering organizations be approached?

Research Question(s)

- Can quality become a method for strategically enhancing the organization to a level above that of its competitors?
- Should quality become part of the strategic plans of an organization?
- How can one strategically plan for quality, if quality becomes part of the organizational strategy?
- How does one implement a quality programme in a software engineering environment?

Having formulated the research questions, the answers to these problems needed to be obtained by following a combination of investigative as well as descriptive research in line with the methods outlined by Cooper and Schindler [11].



Source: Adapted by the author from Cooper and Schindler [11], Exhibit 3-2

Figure 1.4: Formulating the Analysis and Interpretation of Results

The author utilized the last three steps in the management-research question hierarchy as depicted in Figure 1.4 to arrive at the investigative and measurement questions. According to Cooper and Schindler [11]:

> *"**Investigative questions** are questions the researcher must answer to satisfactorily arrive at a conclusion about the research question. To formulate them, the researcher takes a general research question and breaks it into more specific questions about which to gather data."*

The main reason why the author considered investigative questions so important, is because Cooper and Schindler [11] stated that:

> *"They are the foundation for creating the research data collection instrument."*

Through the investigative questions, one should be able to arrive at a conclusion, or a management decision, which should provide one with a reasonable course of action, given the research findings. Following the basic research question methodology that Cooper and Schindler [11] depicted in Figure 1.4, the author defined the investigative questions as follows:

Investigative Questions

- What is strategic management?
- What is software engineering?
- What is quality?
- What is software quality?
- How is software quality measured?
- How do we define an organizational strategy?
- How do we include quality as part of the organizational strategy?
- How can we implement a quality program as part of the organizational strategy?

The measurement and conclusion will be discussed in the conclusion at the end of this chapter.

1.4 THE RESEARCH DESIGN AND METHODOLOGY

The author makes use of mostly descriptive research [11] to highlight the definitions of key terms and concepts such as "quality" and "strategy" to aid in the attainment of the stated objectives of the research. The bigger part of the dissertation relies on descriptive research and literature studies as the mechanisms via which the underlying fundamental concepts inherent in the dissertation are brought to the reader's attention. Four topics receive such treatment, being the disciplines of strategic management, quality management, software engineering and software quality. The descriptive research is then followed by the investigative research, which is the practical part of the dissertation.

Throughout the dissertation numerous case studies and incidents in the real world will be highlighted as examples of the need for an organized strategy in tackling quality issues. For a more detailed description on the outline and structure of the dissertation consult Section 1.5.

The term "strategy" or "strategic management" is used extensively throughout the dissertation. The term "strategy" is used to describe *the planning and tactics required in order for an organization to be more competitive.* In a similar manner, for the

purposes of this dissertation, the term "strategic management" is used to *describe the management processes and tasks that are required to define the organization goals for value creation, and the ways and means in which the organization will attempt to achieve these goals*. An example of a strategy could be the use of benchmarking tools to determine the current state of the organizations' technology. Following the use of such tools, activities can then be improved, and monitored against a best practice standard. This is similar to process monitoring and improvement processes as described by total quality management processes [22], [39], [40].

The term "quality" is also used extensively throughout the dissertation. The author finds that the ISO definition of quality is the most apt for this dissertation. The International Standards Organization (ISO) [22] states that quality is:

> *"The totality of features and characteristics of a product or service that bear on its ability to satisfy specified or implied needs."*

This definition will be assumed to be sufficient for the purposes of most (if not all) discussions related to software quality and strategy within this dissertation. The implementation of a strategic quality plan for software engineering organizations uses this definition as the main focus of quality in software.

1.5 AN OVERVIEW OF THE DISSERTATION STRUCTURE

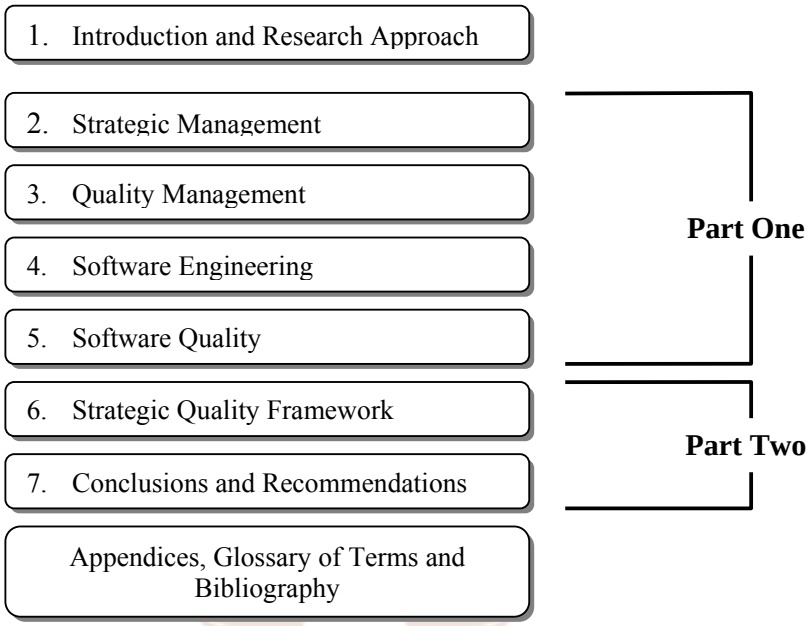
The dissertation was structured in such a way as to ensure that the topics flow logically from one point to the next in order to maximize reader comprehension of the topics presented. The topics are not naturally related to one another, so it is a good idea to follow the structure of the dissertation in order to understand the applications of the concepts.

The function of **Part One** of the dissertation, which includes Chapters Two to Five, is to provide the reader with all the necessary background information regarding strategic management, quality management, software engineering and software quality concepts.

Part Two of the dissertation provides a practical method of improving the software quality in a software engineering organization, using the concepts and techniques

described in Part One of the dissertation. Part Two of the dissertation terminates with Chapter 7: Conclusions and Recommendations.

A schematic outline of the dissertation is illustrated in Figure 1.5:



Source: Created by Author

Figure 1.5: Dissertation Structure

There are a number of additional features included in the dissertation to aid in the readability and understanding. The features include:

- A **Glossary of Terms** to explain the meaning of technical and management concepts used in the dissertation.
- A list of **Appendices** to include extra information regarding quality awards, as well as numerous practical checklists to aid in improving the software quality in any software engineering organization.

1.6 CONCLUSION

The research aims to provide a meaningful answer to the problem of the relationship between quality and strategy, and to provide a framework (guidance) for implementing a strategic quality plan for a software engineering organization.

The results of this research will provide a management strategy for planning quality activities within a software engineering environment to reduce development cycles, effort and costs. Let us start with the meaning of strategy, and how management view strategy as a means for organizational improvement.

PART ONE

LITERATURE SURVEY

Chapter 2

Strategic Management

"A person who, seeing farther and probing deeper than other people, …has energy enough to give effect to this extra vision." – George Bernard Shaw

2.1 INTRODUCTION

Organizations create value by offering products to markets and distribute value earned in these markets to the providers of resources that sustain the organization's value-creation processes. All the employees in an organization need to have the same goal in mind, together with an understanding of how their decisions can affect the profitability and value for all the stakeholders [87].

In order to get everyone in the organization to change their behavior, to learn and to enjoy working, and to find ways to improve the way the organization functions, it is necessary to *develop a strategy, and to implement the strategy* [46], [78]. It needs to be understood that *a strategy's main aim is to improve the accomplishments and profitability of the organization in the short- and long-term on a sustainable basis*. One of the main ways to do this is to encourage employees to expand their knowledge, and to improve their abilities to engage in business activities [87].

Approximately 70% of all strategies fail because the strategy was not executed correctly [83], [87]. The importance of people supporting and understanding the strategy cannot be overstated. One of the biggest benefits of a good organizational strategy is that the employee's skill is almost immediately improved. [87]

Employees often do not understand how the organization earns its money. As soon as employees gain an insight into this, they can make better decisions to support the organizational objective. The employees also know what their individual roles are, and what effect their decisions can have on the organization [87].

A map is just like a strategic framework, model or image, that helps to focus minds, and helps people to take a particular course [15]. It should thus be apparent that to make a real difference in the organizational strategy all the relevant people should be well-informed as to the purpose of the organizational strategy, and that steps should be taken to implement immediate change in order for the strategy to become effective.

But strategy is more than just words or action. The meanings and definitions behind certain terms need to be understood before any action can be taken.

2.2 DEFINITIONS

For the purposes of clarity, most of the definitions were taken directly from the quoted sources.

2.2.1 Strategy

According to the English Usage Dictionary [2], a strategy is the skilful planning or tactics for winning a battle, contest, or competition. In engineering terminology, one could define a strategy as a plan or tactics for improving current processes and methods. In business terms, a strategy aims to improve the accomplishments and profitability of the organization in the short- and long-term on a sustainable basis [87].

2.2.2 Strategic Culture

A strategic culture is one that is sensitive to the customer's requirements, but also recognizes the need for the organization to remain profitable while satisfying these requirements [74].

2.2.3 Management

Management is the term that is used to denote the processes through which an organization tries to maintain and improve its ability to create and distribute value by coordinating the interactions of participants in the activities of the organization as a system. The activities that managers try to coordinate may include both the internal interactions among the organization's own participants and the external interactions of an organization's participants with other people and organizations [78].

2.2.4 Strategic Management

Strategic management refers to management processes that are concerned with two major tasks:

- Defining the organization's goals for value creation and distribution.
- Designing the way the organization will be composed, structured, and coordinated in pursuing its goals for value creation and distribution. [16], [46], [78], [83]

2.2.5 Organizations

Organizations are collections of people who interact with each other in specific, usually repeated ways over some period of time. To attract people to its activities and thereby to assure its continued existence, an organization must bring some form of benefit – whether psychological, social, cultural, professional, or economic – to the people who participate in its activities [78], [83].

2.2.6 Organizational Competence

Organizational competence is the ability of an organization to sustain coordinated deployments of resources in ways that help the organization to achieve its goals [78].

2.2.7 Competence Building

Competence building is any process by which an organization creates or accesses qualitatively new kinds of resources (including new assets or new capabilities) or develops new abilities to coordinate and deploy new or existing resources in ways that help the organization achieve its goals for value creation and distribution [78].

2.2.8 Competence Leveraging

Competence leveraging is the use of an organization's existing competencies to create and distribute value in ways that do not require qualitative changes in the resources the organization uses or in the ways it coordinates its resources. It is in effect, the exercise of one or more of an organization's current strategic options created by prior competence building activities [78].

2.2.9 Resources

Resources refer to any tangible or intangible assets and any human skills and capabilities that are useful and available to an organization in pursuing its goals for value creation and distribution [78].

2.2.10 Stakeholders

Stakeholders are individuals or other organizations that provide essential resources to an organization and that therefore have an interest or stake in receiving some form of value from the organization in return for the resources they provide [46], [53], [78].

2.2.11 Value Creation and Distribution

Value creation and distribution is the creation of benefits for participants in organizational activities. Value distribution could also be used to describe the ways in which the value that an organized activity creates is allocated among the participants in the activity [78], [83].

2.3 TOOLS AND TECHNIQUES FOR STRATEGIC MANAGEMENT

This section addresses some of the tools and techniques that are useful in the implementation of organizational strategies. These tools offer basic planning, measurement and control insights into organizational strategies.

2.3.1 Activity Based Costing

A strategy must first be preceded by clear knowledge of how profits (or losses) are currently being made by customer and by product or service offering. To understand profit by customer and profit by product, revenue is relatively easy to determine in most instances. Activity based costing (ABC) is the process of assigning costs - traditional accounting systems often do not provide such data. Determine the activities, group and allocate overhead costs by activity, assign costs to products and customers based on usage of activity. Specific jobs can then be assigned to the overhead costs based on the amount of activities that they consume. [43], [46], [83]

This is a powerful process as activity based costing [12], [43], [83]:

- Clearly indicates products and customers requiring added attention (watch for premium pricing which may not be sustainable or may attract new niche market entrants).

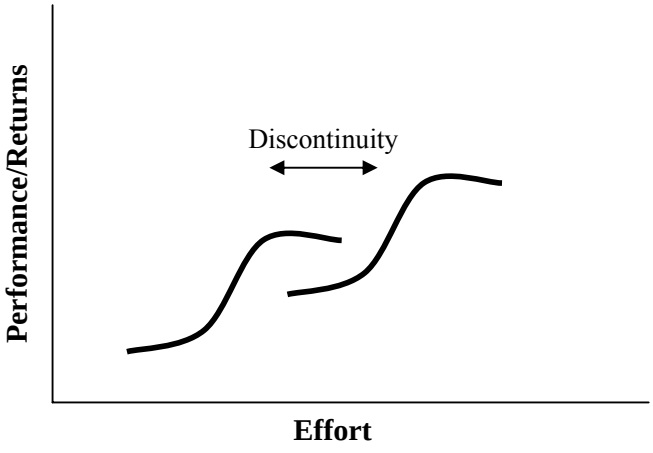
- Indicates value destroying products and customers, often requiring repricing.

- Indicates products and customers, which should not be made or serviced (clearly understanding the interplay of fixed and variable costs and the role of marginal costing).

2.3.2 Adoption Curves

Adoption curves provide a framework to determine the timing and amount of investment in the next generation of technologies. It was founded in the concept of SDLC (system development lifecycle) and adapted to indicate the discontinuities between technologies or business models [19].

As funds are invested in new technologies, initially the returns are low. Once a critical threshold or base is reached, then incremental efforts lead to relatively large returns. Finally the technology reaches a stage where increased competition and reduced profitability make the incremental returns on additional efforts non-viable.



Source: Foster [19]

Figure 2.1: System Development Lifecycle Returns

Adoption curves such as shown in Figure 2.1, ensure an understanding of the fundamental performance drivers of the technology. The organization needs to track improvements or value added versus the effort expended in each driver. The limits of

improvement of the technology needs to be determined and plotted on the S-curve to manage the investment cycle and the allocation of resources accordingly (i.e. minimal resources are allocated to mature technologies and the focus on investment in new technologies is funded by returns from mature technologies) [19].

2.3.3 Balanced Scorecard

Research with twelve organizations at the leading edge of performance management resulted in the formulation of a "balanced scorecard" – a set of measures to give top managers a fast but comprehensive view of the business. The balanced scorecard includes financial measures that tell the results of actions already taken. It complements the financial measures with operational measures on customer satisfaction, internal process and the organization's innovation and improvement activities, all operational measures that are the drivers of future financial performance [41]. [79].

The balanced scorecard will be dealt with in a separate section in this dissertation.

2.3.4 Benchmarking

Benchmarking is a performance improvement process aimed at identifying best practices, and rating organizations against these identified criteria. The aim is to identify superior performance in units within a group, competitors (local or global) or industries where parallels can be drawn. Such practices should then be incorporated into the whole organization.

The activities of benchmarking are shown in Table 2.1:

Activities of Benchmarking
Identify the key activity to improve.
Identify Key Performance Indicators (KPI's).
Choose and entity against which to benchmark (i.e. superior products).
Collect data on KPI's and the supporting activities or processes.
Implement and monitor against best practice standards.

Source: Created by the Author from Boxwell [8]

Table 2.1: Benchmarking Activities

The activities shown in Table 2.1 can be used to improve performance, speed the learning curve through imitation, overcome relative cost and performance

disadvantages, and to become globally competitive [8], [49], [52]. The dissertation of W.P. Lindeman [49] also indicates the importance of benchmarking, as he states:

> *"Benchmarking is the cornerstone of business performance management, because no one can manage what is not measurable."*

Benchmarking also leads to the creation of balanced scorecards based on performance measurements [49]. Thus benchmarking is a tool that should be used in conjunction with balanced scorecards.

2.3.5 Business Process Re-Engineering

Business Process Re-Engineering is the re-design of key business processes to bring about quantum improvements in productivity, cost, or value delivered to customers. The processes often develop over time, largely unplanned or piecemeal, and the value created by taking an end-to-end optimization view. This was popularized in the 80's/90's rollout of enterprise wide technology systems.

A simplified view of business process re-engineering is to:

- Identify value propositions that customers truly value (e.g. delivery time).
- Identify those processes that are involved.
- Use cross-functional teams to redesign end-to-end these processes.

The target is often a combination of improved quality, improved cycle times or reduced cost. The focus is on eliminating hand-overs, or unproductive, or insufficient value adding tasks [25], [28].

2.3.6 Client Retention

It is always more cost effective to keep clients than lose existing clients and go find new clients. Small changes in client retention rates often significantly impact the financial results of an organization.

In order to perform a client retention analysis, one should explicitly link the impact of customer loss (and the cost of replacement) to the financial impact. Thereafter one can

create a client retention strategy by identifying the drivers of "defection" largely based on why clients have defected to date. These reasons are often insidious and relatively easy to fix once identified, e.g. a poor complaints department and high staff turnover rates [49].

A process of client retention is shown in Table 2.2:

Step	Description
1	Identify target client retention rates.
2	Identify and eradicate the drivers of defections.
3	Build retention targets into incentives and budgets.
4	Create the "magic space" where the organization interacts with customers to create a relationship building organization.

Source: Created by the Author from Liswood [49]

Table 2.2: Client Retention Process

Once the organization realizes that its client base has decreased, the organization might embark on a strategy of increasing sales to existing clients (i.e. trying to sell more to the clients that are left). This is not always a good strategy as the existing client base might not be interested in new deals through the increased sales effort, and sales might start dwindling soon [49].

2.3.7 Competitor Analysis

Competitor analysis is a formal and systematic process of reviewing and understanding competitors in order to gain competitive advantage as an input into the organizations own strategy (differentiated positioning often allows margin extraction versus head-on-head competitive positioning). This is assessed at a strategic level and derivations of the Boston Consulting Group (BCG) matrix can be used with multiple axes as variables – or at the operational product or process level (with benchmarking as a guiding principle). A process for performing a competitor analysis is shown in Table 2.3:

Step	Description
1	Identify the current or future competitors.
2	Gather public information.
3	Gather information internally in the organization about the knowledge of competitors.
4	Develop a view of their strategies.
5	Perform a SWOT (strengths, weaknesses, opportunities, threats) analysis.

Source: Created by the Author from Keiser [42]

Table 2.3: Competitor Analysis

From this data, create distinctive grids conceptualizing competitive positioning. Consider strategies and possible counter moves that are likely to be adopted. The main goal for the organization is to focus on the weaknesses of their opposition, in order to capitalize on these areas and to inhibit the strengths of their competitors. Once the organization is faced with the realization that they do have some strong competition in the market often motivates the organization to take action [42].

2.3.8 Core Competencies

A core competency is an attribute that a firm possesses that creates differentiated value to the customer. This is often not easy to identify and creates a basis for sustainable competitive advantage. Often these value creation activities are embodied in the collective organization and are often not discreet but rather a bundle of activities that have been linked in a way different to competitors to create customer value [26], [27]. To be a core competency it should adhere to requirements as shown in Table 2.4:

Core Competency Requirements
Does it make a significant difference to the value of the customer, actual or perceived?
Is it difficult to imitate?
Able to leverage (replicate) into other markets or across business units?
Does it identify what the customers value?
Does it isolate the key abilities that deliver the value?
Does the set of activities compare well with those of competitors?
Does it understand which activities the customers truly value?
Does it nurture such activities to build competencies (either existing or those strategically required)?
Is it a recognized technology?
Does it have procurement value?

Source: Created by the Author from Hamel [27]

Table 2.4: Core Competency Requirements

Core competencies are often used to tie business units or functional areas together, to enable the integration of processes and technologies in different units and to encourage group learning and communication [26], [27].

2.3.9 Customer Service Guarantees

Customer service guarantees are commonly known as Service Level Agreements (SLA's). Customer needs should be understood measurably and objectively (e.g. defect ratio, time to repair etc). The customer needs are then documented and contracted to with the customer. The basis of this is to guarantee service delivery to differentiate the offering [30].

The consequences of non-achievement of customer service guarantees are often financial (i.e. penalties) and consistent non-performance often leads to contract cancellation as it is so explicit. The objective benefits are that this could be considered (in a form) the ultimate scorecard as the customer is doing the explicit scoring [30].

The organization benefits from focusing on delivering specific needs articulated by the customers as well as what the benchmarks are. This methodology can be taken back into the organization with explicit SLA's between departments who are dependent upon each other (e.g. production and sales). Further enhancements often include gain-sharing models above targeted benchmarks [30].

2.3.10 Economic Value Added

The Economic Value Added model measures whether a corporation is creating or destroying value by returning money in excess, or below, its required rate of return on capital. This is primarily a shareholder measure. It can be done on past historic returns or on future returns (based on discounting the cash flow streams from scenarios). Previously what was known as SVA (shareholder value analysis) was popularized as EVA by Joel Stern. Under SVA there were various sub-measures and ratios such as ROCE (return on capital employed), RONA (return on net assets), and ROI (return on investment) [82].

2.3.11 Flat Organizations

Flattening organizations (or organizational structures) refers to the process of removing hierarchies (often of non-value adding management layers) to create empowered employees and an empowered and more responsive organization [63].

This process requires increased reliance on teamwork and staff with increased responsibility and skill sets. The process is aided with documented knowledge management methodologies in an organization [63].

A process to follow in flattening an organization is shown in Table 2.5:

Step	Description
1	Prepare the organization's culture for change.
2	Organize teams around key business processes.
3	Transfer ownership to the teams with clear team leadership assigned.
4	Improve the skill levels of team members.
5	Create incentive-based remuneration structures.
6	Ensure that the culture supports information sharing, coaching and training.

Source: Created by the Author from Ostroff [63]

Table 2.5: Flattening an Organization

2.3.12 Innovation

Innovation proponents focus on the perspective that long-term sustainability only comes from out-innovating the competition, be that in products, business processes or business models. The innovation model is anti-incremental and research and development (R&D) supportive. No real model exists other than certain hard targets e.g. R&D% spend of turnover. More modern techniques and cultures allow innovative cultures to thrive [46], [64].

There are several proponents hereof, the best known being Tom Peters. A derivation hereof is the focus on the 'learning organizations' as outlined by Peter Senge in his book "The Fifth Discipline". [64]

2.3.13 Lessons from Jack Welch

Jack Welch was the CEO of General Electric and considered a guru on his aggressive style of management. His message was primarily about *getting ideas into action in the pursuit of shareholder value.*

Some of the main ideas and methods behind Jack Welch's management style were to develop a vision for the business, and then to change the culture to achieve the vision. One of the ways in which he achieved this was to insist that managers share their ideas, information and experiences with their colleagues and to let people manage their delegated business as they see fit [30].

Jack Welch set measurable goals for all-important measures, and treated them as solid commitments that management must keep. A similar style could be adopted by briefing oneself fully on everybody working in the same department and to recognize employees [30]. Be tough, but do not be hard, with everyone with whom one may have dealings. The lessons that could be learned from Jack Welch about corporate strategy will be dealt with in a separate section.

2.3.14 Mass Customization

The mass customization model is based on delivering profiled information into a production or delivery environment to derive customized goods and services. The key is to add the most amount of customer value added through customization with adding the least possible incremental cost to each market segment product [83]. Market knowledge, technology platforms and responsive production and delivery systems (flexible) often bring this about. (The software industry, especially the internet with active server pages, are a prime example of personalized services).

An organization can introduce product sets that are segment-customized with minimum setup or change over costs. This is often designed around components that can be modularized so that the end product is customized. The organization structure is then transformed to allow horizontal information flow (e.g. marketing needs to production) with a focus on reducing time to market. The ultimate goal is within ever-smaller segments, to ultimately cater for the segment of one without significantly increased cost structures. If this is built into the process of interacting with the market and customizing products seamlessly (e.g. web site personalization) then the ability to respond quickly to changes in the market are quick and seamless [46].

2.3.15 Mission and Vision Statements

A rich view of the future and the defined mission statement of the organization is vital. The difference between mission and a vision is described as follows [12], [83]:

- A mission statement defines the methods of doing business and its objectives.
- The vision describes the desirable end state.

The way in which the mission statement and vision of the organization is defined varies from organization to organization. It is not important how the mission and vision are created, but how the mission and vision are used in the organization. Internally it aligns the organization, provides a framework for decision-making, is often motivational and gives a sense of common purpose. It serves to position the organization to outside parties and to serve as a corporate customer relations tool.

By defining a mission statement, the organization has defined a general direction that usually gives rise to organizational goals. Very often the mission statement is rooted in the values of the organizational culture [12], [46].

2.3.16 Pay for Performance

The pay for performance technique is commonly used in management by objectives (MBO) models. The compensation and remuneration models are linked directly to the achievement of corporate, management and personal objectives with the focus to create more directed effort, motivation and alignment between individual efforts and the corporate objectives.

The pay for performance goals must be explicit, agreed, controllable and measurable. The system must be aligned to how shareholders create value and must focus on two aspects:

- A performance measurement system. This should include financial and non-financial targets. It should also create a weighting of team based, and individual based efforts [83].
- A remuneration structure that rewards performance. Depending on the industry and culture of the organization, the rewards should be both financial (with

differing time frames created through the mix of bonuses/gain sharing or stock option type structures) and non-financial rewards (e.g. recognition systems) [83].

As the pay for performance system impacts remuneration, the process must be implemented carefully and with due consideration to organization specific dynamics. If implemented incorrectly it could lead to increased cost levels without increased output; if implemented correctly the true contributors are identified and secured and the laggards identified and remunerated accordingly [40], [83].

2.3.17 Porter 5 Forces Model

The Porter 5 Forces model was developed by Michael Porter and was a landmark work. It assesses the attractiveness, and hence sustainable profitability, of an industry. The 5 forces identified by Porter affecting industry attractiveness that need to be assessed are illustrated in Table 2.6:

Force	Description
Intensity of competitive rivalry	*Review the product likeness, concentration levels and intensity of competitiveness or industry "management".*
Threat of new entrants	*To be assessed against barriers to entry but also review regulatory barriers or competitor reactions to potential entrants.*
Threat of substitutes	*The ability to substitute products. Assess in terms of product functionality, pricing, and likelihood of customer substitution and technology changes.*
Bargaining power of suppliers	*Assess the impact of industry costs, and the leverage that upstream players can exert. This is often assessed by industry concentration, substitutability, or regulatory frameworks.*
Bargaining power of buyers	*This is assessed by buyer concentration, volume buying power, substitutability, regulation or incentives to bargain.*

Source: Created by the Author from Porter [68]

Table 2.6: Porter 5 Forces Model

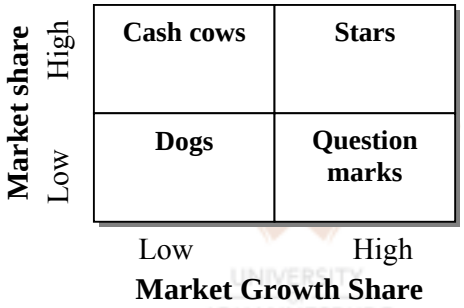
The model is used to assess long-term returns in an industry. Often used for strategic decisions to enter, exit or remain in a business unit or industry segment. This also guides alignment decisions with regard to suppliers or buyers [68], [70].

2.3.18 Portfolio Analysis

The portfolio analysis model attempts to conceptualize and optimize a portfolio of business units or products or activities. The model attempts to classify units into

strong or weak performers or those with high growth potential. The underlying purpose is to identify future resource allocation (e.g. potential star products requiring investment) and accompanying strategies (e.g. cash generation from 'cash cows'). It attempts to balance portfolios, with some products generating cash to fund, future high potential, units requiring cash investment support.

The model can use various axes or variables to create multiple views of activities. The base analysis however originates in the Boston Consulting Group (BCG) grid which ranks the competitive capability (usually indicated by current market share position) versus the market attractiveness (i.e. high growth markets). The reasoning behind the BCG matrix is to have high market shares in low growth industries (usually cash generators) to fund high market shares in high growth markets (stars). An example of a BCG matrix is shown in Figure 2.1:



Source: Kotler [46], Figure 4.2

Figure 2.2: Boston Consulting Group Growth-Share Matrix

The assumption is that organizations want to be in high growth rate markets. It is also assumed that high market shares for product result in sustainability and the ability to extract a profit margin – resulting in the need to build Stars (high share value in the right markets). Often these rapid growing markets require investment phases and this is to be funded by Cash cows (i.e. high market shares in low growth stable markets). The Dogs are often culled (i.e. low market shares in the wrong markets). Question marks are in the correct high growth markets but a decision must be made as to the attractiveness of investment to garner additional market share to move them over time to the Stars quadrant [46].

The BCG matrix is used for investment decisions, resource and capital expenditure allocations as well as determining cash generation requirements [46].

2.3.19 Satisfaction Surveys

Satisfaction surveys (face to face, telephonic, electronic, etc.) and focus-groups are some of the ways to identify, understand and anticipate how to fill customer needs. The surveys are often left within a marketing department for feedback and is seldom brought to a strategic level [79].

Satisfaction surveys are used to ascertain the 'needs' of customers. A feedback loop monitoring system is required for measuring the achievement of certain benchmarks. These surveys should be done irrespective of the subjective nature and sample sizes of the benchmarking exercise. The process of measuring customer satisfaction should be expanded to continually assess the developing needs of the market and how existing products could be modified to service the emerging needs of clients [46], [79].

Software is notoriously user-unfriendly, and hence good service will go a long way to build and enhance the relationship between the organization and their clients.

2.3.20 Scenario Planning

Scenario planning is used to build robust "what-ifs" by taking various alternative views as to what may unfold in the future and then modeling these organizational scenarios. These scenarios can either be static or more dynamic models (i.e. changing various variables simultaneously). A process for scenario planning is shown in Table 2.7:

Step	Description
1	Identify the baseline model.
2	Agree time frames for analysis and key variables that may change.
3	Model the circumstances, including the positioning and financial consequences.

Source: Created by the Author from Wack [90]

Table 2.7: Scenario Planning Process

The process of being forced to think through more than just the one consensus view is of immense value. It forces broad thinking and avoids "group think". The process identifies sensitivities to key variables and leverage points. Alternative strategies often result in contingency planning [90].

Scenario planning creates deeper, richer future views and brings to the surface implicit assumptions and widely held views that are often taken as fact without being

thought through. It creates system or process thinking and often leads to Game Theory implications of strategy and counter-strategy [90].

2.3.21 Self-Directed Teams

A self-directed team attempts to cut across the traditional functional or business unit boundaries to take full responsibility for delivery. This either occurs on a project basis or ongoing in key customer value adding activities. Teams work together, taking full responsibility, often using a project management methodology [87].

The teams must have freedom beyond traditional boundaries in order to redesign business processes and have input into all activities that will impact their focus area. Through the creation of a self-directed team a leadership mentality is instilled in the individuals and the responsibility of team players are usually expanded, with less reliance on traditional management structures [87].

The self-directed team concept developed from the concept of flattening the organizational structure and empowering employees. The focus area is therefore less dependent on management and more on team motivation, the environment together with increased attention to quality and any area requiring improvement [87].

2.3.22 Strategic Alliances

In the wave of globalization many firms are focusing on niche vertical markets rather than horizontal expansion. This makes the need for “partnerships” and alliances more pressing. Strategic alliances occur where firms, often within a value chain of a solution, mutually commit resources and agree to work together to meet common objectives. Kotler [46] describes the global partnership between fifteen airlines – Star Alliance – allowing travelers to make hassle-free and seamless connections to numerous destinations. This is a prime example of various organizations within the same value chain works together to meet the common objective of integrated connections for airline passengers.

The primary drivers for strategic alliances include accessing new markets (extending the reach or the access of client bases), sharing resources and risks, or inhibiting the positioning of competitors, often by being able to create dominant positions rapidly through leveraging on the alliance partner’s reach or skills [46], [48], [83].

Based on the strategy chosen for the strategic alliance, the organization must identify where alliances could supplement the traditional approach of "going it alone". Organizations should evaluate potential partners and the possible synergy while ensuring the strengthened strategic positioning through the alliance will not pose a future threat. Finally, the strategic alliance should include an agreement with the identified counter party with formal systems that measure the rollout against the pre-agreed objectives, being careful to manage the scope of the alliance [48], [83].

2.3.23 Target Marketing

Target marketing is the process of using information and profiling techniques to develop ever-decreasing market segments and the differing needs therein, which are targeted in order to satisfy customer requirements on a differentiated basis. Often these initial segments are moved into tracking mechanisms to refine segmentation knowledge and evolve the marketing strategies. An understanding of the customer needs using existing data for information analysis can also be supplemented by focus groups and point of sale tests [5], [46].

2.3.24 Time Based Competition

Time based competition aims to reduce the time taken to perform key business processes that add value to customers. The focus is on identifying and re-engineering those activities that slow delivery such as decision-making and non-value activities [17]. The main focus is placed on reducing the time to market of new products and services of the organization.

The procedural approach to mapping and shortening key activities often lead to faster delivery times at lower cost. The focus areas are initially often pure operations, then extending to delivery and logistics. The same methodology is then applied to new product development. The so-called 'new economy' led to many processes being automated or STP models (straight through processing) coming to market [17], [83].

2.3.25 Total Quality Management

Total quality management is a systematic approach to the delivery of products and services based on customer needs and then ensuring that a zero defect specification approach is taken in the pursuit thereof. This was a process popularized by the

Japanese in the early 1980's, together with continuous improvement processes and quality assurance [20], [39], [68], [83].

The process for achieving total quality management can be simplified as shown in Table 2.8:

Step	Description
1	Understand both current and future customer needs.
2	Design products and services to match these needs while attempting to exceed expectations.
3	Identify areas in the business that hamper zero defect delivery.
4	Create processes and work with staff to work towards zero defects.
5	An incentive linked process must create feedback loops and QA monitoring processes.

Source: Created by the Author from Taguchi [83]

Table 2.8: Total Quality Management Process

The goals of total quality management are numerous:

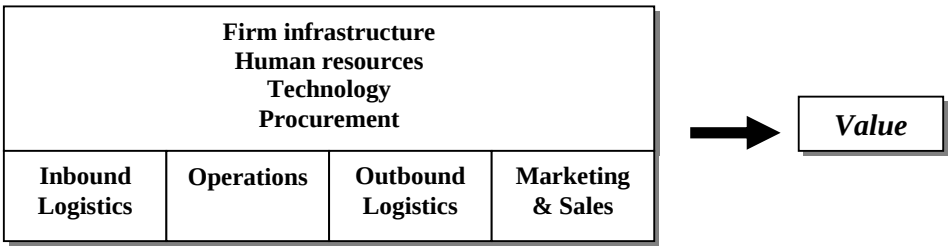
- Customer satisfaction across all needs.
- Increased productivity.
- Lower rework.
- Lower cost – ultimately competitive advantage.

The ultimate objective of total quality management is that if you are going to do it, do it once and do it right [20], [38], [68], [83].

2.3.26 Value Chain Analysis

The value chain analysis model was developed by Michael Porter [46], [68]. The value chain identifies the strategic activities that create value in a specific business area [46]. Cost and performance should then be analyzed in order to try and improve these business functions.

Value chain analysis can be graphically depicted as shown in Figure 2.3:



Source: Adapted by the Author from Kotler [46], Figure 3-3, and Porter [65]

Figure 2.3: Value Chain Analysis

Value chain analysis examines the organization's activities in terms of cost, the value that the activities add for customers. The functions of the model include:

- Identifying the business activities.
- Assign costs to the business activities.
- Understand the linkages between activities.
- Understand the performance of the activities relative to competitors.
- Identify what creates value for the customer.
- Develop a strategy using the various activities that maximize differentiation value to the buyer and minimizes increases in cost.

Value Chain Analysis focuses on the activity level to find sources of competitive advantage to gain cost advantages or alternatively increases levels of differentiation that are important to the buyer while limiting the increase in cost [46], [68].

2.4 LESSONS FROM JACK WELCH

2.4.1 Background

One of the best and probably most prolific corporate icons have been Jack Welch [30]. Some of his principles and practices were based on the principles of total quality management – Jack Welch also believed in continuous improvements, hence the author used him as an example of a corporate strategist.

Jack Welch can best be described as a corporate icon and leader who almost single-handedly transformed General Electric (GE) into one of the world's most valuable organizations. Jack Welch took over General Electric in 1981 and the stock market value was $13,9 billion. When he retired in 1999, the value had increased over 30 times to $410 billion. The organization's stunning success was driven by many of Welch's principles – such as streamlining cost structures, concentrating on core business with leadership positions and forcing responsibility down to subordinates [30].

Jack Welch was born in 1936 and joined GE in the plastics division in 1960, from where he won accolades for turning the 300 staff division around, and steadily moving up the ranks, being seen as a "cage rattler" and a bit of a maverick. He reduced the GE workforce by 100,000 staff (including 400 corporate planners). Jack Welch was a hands on, confrontational, face-to-face manager who believed that "the idea flow from the human spirit is absolutely unlimited" and spent endless hours talking to those around him. This led to the achievement of short-term exceptional profit growth and investments for future growth [30].

In a question as to which is more important, families of former employees or shareholder value Jack Welch's reply was: "In a global economy, you cannot manage a company in a paternalistic way. If you do not sort out things in good time they will eventually explode in your face, then you have to become brutal and cruel. GE won for three straight years America's most admired company. The attributes assessed were admirable performance; innovativeness; employee talent; financial soundness; use of corporate assets; long-term investment value; social responsibility and quality of products and services" [30].

2.4.2 Make the Managers Lead

Jack Welch believed that in order to turn managers into leaders, a 7-point programme for management is required [30], as shown in Table 2.9:

Step	Force	Description
1	Vision	*Develop a vision for the business*
2	Culture	*Change the culture to achieve the vision*
3	Organization Structure	*Flatten the organizational structure*
4	Bureaucracy	*Eliminate unnecessary bureaucracy*
5	Empowerment	*Empower individuals in the organization*
6	Quality	*Raise quality*
7	Boundaries	*Eliminate all boundaries*

Source: Created by Author from Heller [30]

Table 2.9: 7-Point Programme for Management

Jack Welch felt that a leader's job is to define direction and allocate resources, and his 7-point programme was used to drive this change in management thinking. It is important to note that raising quality is seen as a key step in the process of turning managers into better leaders.

Bureaucracy slows decision-making, blurs responsibility, and creates undoable jobs. The layers of bureaucracy need to be removed and the span of control increased. In theory small business units (SBU's) makes the unwieldy manageable, yet the flaws are often fatal: businesses and departments do not cooperate and the focus becomes short term and bureaucracy and control often reign.

Meetings should be used to share information, but not to enforce bureaucracy. Meeting leaders must discuss any relevant issues keep people alert for possible problems. Meetings could also be used to pursue best practice in the organization through the process of highlighting achievements and to point to successes that are held up as best practices.

Leaders in the organization should be able to delegate responsibility through a combination of hands-on and hands-off leadership, and monitor explicit and exact performance targets. Leaders should also win the hearts and minds of those people around them in order to empower middle management. This is key for productivity gains. The steps that Jack Welch recommended for empowering management are shown in Table 2.10:

Step	Description
1	Free managers to manage and rise
2	Defeat bureaucracy and rigidity
3	Generate and use new ideas
4	Empower workers to flourish and grow
5	Accompany rewards by frank, face-to-face evaluations

Source: Created by Author from Heller [30]

Table 2.10: Empowering Management

A man like Jack Welch spent half his time on people issues, getting to know people and how to energize them. A concept of Jack Welch was "E to the fourth power" – Energy, Energize others, competitive Edge, and Execution [30].

Future leaders could be tested by moving them around to see how they react under pressure. Traits to look for in leaders include energy, the ability to excite others, the ability to define a vision, the ability to find change fun and not paralyzing, feeling comfortable in any environment, and the ability to talk to all kinds of people. Jack Welch felt that if the change isn't big enough, bureaucracy could beat you. For this reason he suggested that people:

- Face reality as it is, not as you wish it were.
- Be candid with everyone.
- Don't manage, lead.
- Change before you have to.
- If you don't have a competitive advantage, don't compete.
- Control your own destiny, or someone else will.

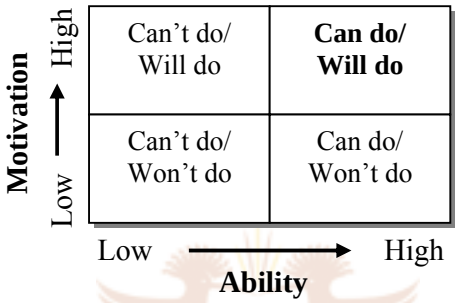
On leading any type of team, the six basics include:

- Pick the right people.
- Have a clear purpose.
- Put it in writing.
- Work to a strict timetable.

- Plan action.
- Act on the plan.

2.4.3 Mobilize the Workforce

Positive behavior can be reinforced (and even demanded) by living the values, being an entrepreneur, hitting high targets, starting decisively, embracing change, doing what you say, concentrating focus, managing on the facts, forgiving honest error and organizing oneself. Jack Welch demanded high standards and used the evaluation shown in Figure 2.4 to guide staff decisions [30]:



Source: Heller [30]

Figure 2.4: Staff Evaluation

If people are in the can do/will do section, they are your top performers. People evaluated to be in the can't do/will do section should be trained to assist them in gaining the ability to perform the required jobs as they are more than willing to perform such tasks. People in the can do/won't do section should be motivated to perform better, and lastly, people in the can't do/won't do section should be re-assigned. It is all about improving employee value and gaining competitive strength [30].

Another evaluation method employed by Jack Welch is to richly reward the top 20% of employees, look after and energize the key 70% of employees and re-assign or even dismiss the bottom 10% of employees [30]. The last step of re-assignment or even dismissal becomes more difficult each year, but could be an effective tool to rid the organization of the worst performing employees.

The process of mobilizing the workforce is started by banishing traditional bosses and managers. The new management team should have three ideals, namely speed,

simplicity and self-confidence. It is important to remember that only changed managers can lead a changed workforce. In the same manner, executives must live the values of their organization. If executives try to achieve any objectives by force, through being a tyrant, bully, or even autocratic, their stay will be short lived despite making organizational targets. Organizational values are more important than meeting goals in modern lean organizations Executives must remember to involve employees in decision making, transferring ideas between different areas and to simplify production and other organizational processes.

Organizational efficiency was driven through three initiatives as shown in Table 2.11:

Force	Description
Workouts	*Using workshops to generate ideas and proposals for radical improvement and driving out non-value activities.*
Best practices	*Identification of both internal and external practices to learn and benchmark against. Create experts, give programmes a name and direction.*
Process mapping	*Motivate the workforce to succeed by mapping processes and procedures to specific areas and people.*

Source: Created by the Author from Heller [30]

Table 2.11: Organizational Efficiency Initiatives

Mobilization of the workforce is a continuous process that is based on empowerment. Mobilization is what happens when people become excited about finding solutions to their problems.

Jack Welch insisted that managers make their decisions clearly and simply. Managers should also start off improvement teams on quick fixes to:

- Eliminate administration issues.
- Adopt total quality methods to save costs, raise productivity, and delight customers.
- Never relent in the insistent pursuit of better personal and organizational performance.
- Use quality programmes as the key means of management development.

The quality of General Electric was to be made so special and valuable to their customers, to be an important factor in their success, that General Electric products became their only real value choice [30].

2.4.4 Gain a Competitive Advantage

Jack Welch believed that an organization must only compete where it can win, i.e. "be number one or number two in your market, or else". An organization must not compare their returns versus a competitor's returns in a specific market segment, but rather compare the required return on capital. Organization must compete in the right sectors and then compete only where it has a competitive advantage – "don't play with businesses that cannot win". This no. 1 or no. 2 rule was a global view that Jack Welch had [30]. "There will only be one standard for corporate success: international market share".

In order to gain a competitive edge, organizations should move into global markets after building a domestic base. Acquisitions and the pooling of assets could be used to strengthen local businesses for expansion [30].

2.4.5 Pursue Shareholder Value

Jack Welch felt that growth in shareholder value is the final deliverable in his management strategy. To achieve growth in shareholder value, management should clearly name and communicate the target profit levels of the organization. Thereafter each business area is allowed to set their own targets. This view of sustained wealth creation by Jack Welch comes from four tenets [46]:

- Lead the market only from a number 1 or number 2 position.
- The returns are to exceed inflation.
- Create a unique selling proposition by giving value to customers that competitors cannot.
- Build on what the organization does best.

The other focus in the creation of shareholder value is a concept of "integrated diversity", meaning that businesses are run independently, yet pool their ideas, people,

experiences, and best practices. This is a long-term emphasis. Jack Welch felt that anybody can manage an organization short term, and anybody can manage long term, but balancing these two things is what management is about [30].

General Electric used the principles (and not in its pure form) of EVA (economic value added) to get managers to think like owners, i.e. the organization must earn more on the capital employed than the cost of capital [30].

2.4.6 Exploit the Forces of Change

Managers should face the competition and understand that they are accountable as they as managers "own" the business. Jack Welch challenged managers and employees into change – he felt that change is key and will always be easier to propose than to achieve. Managers need to distinguish between evolutionary change and revolutionary change in the face of competitive threat [30].

Jack Welch realized that a big risk for any organization is the pursuit of too many initiatives in too short a time [30]. The organization must be changed and educated through the creation of themes and the communication of these themes. Organizations need overarching themes to create change. If it is just somebody pushing a gimmick or a programme, without an overarching theme, it will probably fail.

2.5 THE BALANCED SCORECARD

In the preceding section, the management style and strategies of Jack Welch were discussed. This section aims to provide a technique for measuring and implementing organizational strategies.

An organization's measurement system strongly affects the behavior of managers and employees. Managers should also understand that traditional accounting financial measures like return-on-investment and earnings-per-share can give misleading signals for continuous improvements and innovation – the activities that today's competitive environment demands [46].

Managers and academic researches have tried to remedy the inadequacies of current performance measurement systems, by focusing on making financial measures more relevant. Others have said, "Forget the financial measures. Improve operational

measures like cycle times and defect times; the financial results will follow." However, managers should not have to choose between financial and operational measures. No single measure can necessarily provide a clear performance target, or focus attention on the critical areas of the business. Managers ostensibly need a balanced presentation of both financial and operational measures [41].

> *The balanced scorecard is like the dials in an airplane cockpit: it gives managers complex information at a glance [41].*

The balanced scorecard was devised to provide top managers with a set of measures that could give a fast but comprehensive view of the business – the balanced scorecard [41], [46]. The balanced scorecard includes financial measures that depict the results of actions already taken. It also complements the financial measures with operational measures on customer satisfaction, internal processes, and the innovation and improvement activities of the organization – operational measures that are the drivers of future financial performance.

The balanced scorecard allows managers to look at the business from typically four important perspectives. It provides answers to four basic questions [41]:

- How do customers see us? (A customer perspective).
- What must we excel at? (An internal perspective).
- Can we continue to improve and create value? (An innovation and learning perspective).
- How do we look to shareholders? (A financial perspective).

While giving senior managers information from 4 different perspectives, the balanced scorecard minimizes information overload by limiting the number of measures used. Organizations rarely suffer from having too few measures. More commonly, they keep adding new measures whenever an employee or a consultant makes any worthwhile suggestions. The balanced scorecard forces managers to focus on the handful of measures that are most critical for the organization.

Several organizations have already adopted the balanced scorecard. Hewlett Packard uses a customer-based scorecard to monitor approximately twenty business areas. Sales and profits are monitored to identify areas where improvements in marketing strategies could lead to improved financial results [46]. The early experiences using the scorecard have demonstrated that it meets several managerial needs. The scorecard brings together in a single management report, many of the seemingly disparate elements of an organization's competitive agenda: becoming customer orientated, shortening response time, improving quality, emphasizing teamwork, reducing new product launch times, and managing for the long term.

The scorecard can also be seen as a way to clarify and simplify the vision at the top of the organization for operational purposes. It is always a good idea to start the design of the balanced scorecard with the focus on a short list of critical indicators of current and future performance.

2.5.1 Customer Perspective

The customer perspective is the way in which customers see the organization. The importance can be found in the corporate mission statement of many organizations that focus on the customer. "To be number one in delivering value to customers" is a typical mission statement. How an organization is performing from its customers' perspective has become, therefore, a priority for top management. The balanced scorecard demands that managers translate their general mission statement on customer service into specific measures that reflect the factors that really matter to customers [41], [46].

Customers' concerns tend to fall into four categories: time, *quality*, performance and service, and cost. Lead time measures the time required for the organization to meet its customers' needs. For existing products, lead time can be measured from the time the organization receives an order to the time it actually delivers the product or service to the customer. For new products, lead time represents the time to market, or how long it takes to bring a new product from the product definition stage to the start of shipments [83]. *Quality measures the defect level of incoming products as perceived and measured by the customer. Quality could also measure on-time delivery and the accuracy of the organization's delivery forecasts* [41]. The

combination of performance and service measures how the organization's products or services contribute to creating value for its customers.

> *The balanced scorecard shows how the results are achieved: did the costs of set-ups fail because of shorter set-up times or bigger batch sizes? [41]*

In order to put the balanced scorecard to work organizations should articulate their goals for time, quality, and performance and service. Once these goals have been determined, these goals should be translated into specific measures. For example, the organization should establish the general goals for customer performance and then translate these general goals into specific goals and identify an appropriate measure for each. These general goals could include reducing the time to market or developing innovative products.

To track the specific goal of providing a continuous stream of attractive solutions, the organization can measure the percent of sales from new products and the percent of sales from proprietary products. This kind of information should be available internally. Sometimes the organization will be forced to gather data from outside the organization, for example, if the organization wanted to assess whether it was achieving its goal of providing reliable, responsive supply, the organization will have to turn to its customers.

Benchmark procedures are yet another technique organizations use to compare their performance against competitors' best practice [49], [83]. Many organizations have introduced "best of breed" comparison programs: the organization looks to one industry to find the best distribution system, to another industry for the lowest cost payroll process, and then forms a composite of those best practices to set objectives for its own performance.

In addition to measures of time, quality, performance and service, organizations should remain sensitive to the cost of their products. Customers see price as only one component of the cost they incur when dealing with their suppliers. Other supplier-driven costs range from ordering, scheduling delivery, and paying for the materials; to receiving, inspecting, handling, and storing the materials, the scrap, rework, and obsolescence caused by the materials, and schedule disruptions from incorrect deliveries [41].

2.5.2 Internal Business Perspective

Customer-based measures are important, but they must be translated into measures of what the organization should do internally to meet its customers' expectations (what must the organization excel at?). After all, excellent customer performance derives from the processes, the decisions, and the actions occurring throughout an organization. Managers need to focus on those critical internal operations that enable them to satisfy customer needs. The second part of the balanced scorecard gives managers that internal perspective [41].

The internal measures for the balanced scorecard should stem from the business processes that have the greatest impact on customer satisfaction – factors that affect cycle time, quality, employee skills, and productivity, for example. Organizations should also attempt to identify and measure their organization's core competencies, the critical technologies needed to ensure continued market leadership. Organizations should then decide what processes and competencies they must excel at and specify measures for each [41].

To achieve goals on cycle time, quality, productivity, and cost, managers must devise measures that are influenced by employees' actions. Since much of the action takes place at the department and workstation levels, managers need to decompose overall cycle time, quality, product, and cost measures to local levels. That way, the measures link top management's judgement about key internal processes and competencies to the actions taken by individuals that affect overall corporate objectives. This linkage ensures that employees at lower levels in the organization have clear targets for actions, decisions, and improvement activities that will contribute to the organization's overall mission.

Information systems play an invaluable role in helping managers disseminate the summary measures. When an unexpected signal appears on the balanced scorecard, managers can query their information systems to find the source of the trouble. If the aggregate measure for on-time delivery is poor, for example, managers with a good information system can quickly look behind the aggregate measure until they can identify late deliveries, day by day, by particular plant to an individual customer.

If the information system is unresponsive, however, it can be the Achilles' heel of performance measurement. Some of the issues that managers might face are [41], [46]:

- The absence of an operational information system.
- The scorecard information is not timely.
- Reports are generally a week behind the organization's routine management meetings.
- The measures have yet to be linked to measures for manager and employees at lower levels of the organization.

Hewlett-Packard used a metric called breakeven time (BET) to measure the effectiveness of its product development cycle. BET measures the time required for all the accumulated expenses in the product and process development cycle (including equipment acquisition) to be recovered by the product's contribution margin (the selling price less manufacturing, delivery, and selling expenses) [41], [46].

2.5.3 Innovation and Learning Perspective

The customer-based and internal business process measures on the balanced scorecard identify the parameters that the organization considers most important for competitive success. But the targets for success keep changing. Intense global competition requires that organizations make continual improvements to their existing products and processes and have the ability to introduce entirely new products with expanded capabilities.

An organization's ability to innovate, to improve, and learn ties directly to the organization's value. That is, only through the ability to launch new products, create more value for customers, and improve operating efficiencies continually can an organization penetrate new markets and increase revenues and margins – in short, grow and thereby increase shareholder value [41], [46].

Many organizations use the percent of sales from new products as one of its innovation and improvement measures. If sales from new products are trending

downward, managers can explore whether products have arisen in new product designs or new product introductions.

Business Balanced Scorecard

Financial Perspective	
Goals	**Measures**
Survive	Cash flow
Succeed	Quarterly sales growth and operating income by division
Prosper	Increased market share and ROE

Customer Perspective	
Goals	**Measures**
New products	Percent of sales from new products
Responsive supply	Percent of sales from proprietary products
Preferred supplier	On-time delivery (defined by customer)
Customer partnership	Ranking by key accounts Number of cooperative engineering efforts

Internal Business Perspective	
Goals	**Measures**
Technology capability	Manufacturing geometry vs. competition
Manufacturing excellence	Cycle time, unit cost, yield
Design Productivity	Engineering efficiency
New product introduction	Actual introduction schedule vs. plan

Innovation and Learning Perspective	
Goals	**Measures**
Technology leadership	Time to develop next generation
Manufacturing learning	Process time to maturity
Product focus	Percent of products that equal 80% of sales
Time to market	New product introduction vs. competition

Source: Kaplan [41]

Table 2.12: The Balanced Scorecard

Table 2.12 shows an example of a balanced scorecard that is used to measure the financial perspective, customer perspective, internal business perspective and the innovation and learning perspective in an organization.

One organization, Milliken & Co., required that managers make improvements within a specific time period. Milliken did not want its "associates" (Milliken's word for employees) to rest on their laurels after winning the Malcolm Baldrige Award. (Appendix A describes the Malcolm Baldridge National Quality Award in more detail). Chairman and CEO Roger Milliken asked each plant to implement a "ten-four" improvement program to measure process defects, any missed deliveries, and scrap. These identified areas were to be improved over the next four years as part of its improvement program. These targets emphasize the role for continuous

improvement in customer satisfaction and internal business processes in organizations [41].

2.5.4 Financial Perspective

Financial performance measures indicate whether the organization's strategy, implementation, and execution are contributing to bottom line improvement, and how the organization looks to shareholders. Typical financial goals have to do with profitability, growth, and shareholder value as shown in Table 2.12. Some organizations state their financial goals simply: to survive, to succeed, and to prosper [41]. Survival can be measured by cash flow, success by quarterly sales growth, and operating income by division, and prosperity by increased market share by segment and return on equity [46].

However, given today's business environment, should senior managers even look at the business from a financial perspective? Should they pay attention to short-term financial measures like quarterly sales and operating income? Many have criticized financial measures because of their well-documented inadequacies, their backward-looking focus, and their inability to reflect contemporary value-creating actions [41]. Shareholder value analysis (SVA), which forecasts future cash flows and discounts them back to an estimate of current value, is an attempt to make financial analysis more forward looking. But SVA still is based on cash flow rather than on the activities and processes that drive cash flow [46].

Assertions that financial measures are unnecessary are incorrect for at least two reasons. A well-designed financial control system can actually enhance rather than inhibit an organization's total quality management program [68]. More important, however, the alleged linkage between improved operating performance and financial success is actually quite tenuous and uncertain as shown in the following example.

Over the three-year period between 1987 and 1990, a NYSE electronics company made an order-of-magnitude improvement in quality and on-time delivery performance. Outgoing defect rate dropped from 500 parts per million to 50, on-time delivery improved from 70% to 96%, and yield jumped from 26% to 51%. Did these breakthrough improvements in quality, productivity, and customer service provide substantial benefits to the organization? Unfortunately not. During the same three-

year period, the organization's financial results showed little improvement, and its stock price plummeted to one-third of its July 1987 value [41].

The considerable improvements in manufacturing capabilities had not been translated into increased profitability. Slow releases of new products and a failure to expand marketing to new and perhaps more demanding customers prevented the organization from realizing the benefits of its manufacturing achievements. *The operational achievements were real, but the organization had failed to capitalize on them.* The organization should rather have focused on appropriate measures in the balanced scorecard to try an achieve success.

Measures of customer satisfaction, internal business performance, and innovation and improvement are derived from the organization's particular view of the world and it's perspective on key success factors. However, the organizational view is not necessarily correct. An excellent set of balanced scorecard measures does not guarantee a winning strategy. The balanced scorecard can only translate an organization's strategy into specific measurable objectives. A failure to convert improved operational performance, as measured in the scorecard, into improved financial performance should send executives back to their drawing boards to rethink the organization's strategy or its implementation plans.

As one example, disappointing financial measures sometimes occur because organizations do not always follow up their operational improvements with another round of actions. Quality and cycle-time improvements can create excess capacity [46]. Managers should be prepared to either put the excess capacity to work ore else get rid of it. The excess capacity must be either used by boosting revenues or eliminated by reducing expenses if operational improvements are to be brought down to the bottom line [46].

Ideally, organizations should specify how improvements in quality, cycle time, quoted lead times, delivery, and new product introduction will lead to higher market share, operating margins, and asset turnover or to reduced operating expenses. The challenge is to learn how to make such explicit linkage between operating and finance. Exploring the complex dynamics will likely require simulation and cost modeling.

2.5.5 Measures that Move Organizations Forward

As organizations apply the balanced scorecard, they should understand that the scorecard represents a fundamental change in the underlying assumptions about performance measurement. Managers cannot necessarily implement the balanced scorecard without the involvement of the senior managers who have the more complete picture of the organization's vision and priorities.

The balanced scorecard puts strategy – not control – at the centre.[41]

Traditional measurement systems have sprung from the finance function, hence the systems tend to have a control bias [41]. The traditional performance measurement systems specify the particular actions they want employees to take and then measures the actions taken to see whether the employees have in fact taken those actions, resulting a system that attempts to control behavior.

The balanced scorecard puts strategy and vision at the center, establishing goals, assuming that people will adopt the behaviors necessary to arrive at these goals. This new approach to perform measurement is consistent with the initiatives under way in many organizations: cross-functional integration, customer-supplier partnerships, global scale, continuous improvement, and team accountability rather than individual accountability [41], [46].

The balanced scorecard can help managers comprehend at a high level the interrelationships between the financial perspective, the customer perspective, internal business perspective and innovation A better understanding of these interrelationships can lead to improved organizational decision making. Ultimately, the balanced scorecard attempts to keep organizations looking and moving forward, instead of backward.

2.6 ACHIEVING STRATEGIC OBJECTIVES

2.6.1 The Strategic Logic of Organizations

Sanchez [78] defines strategic logic can be defined as the operative rationale of an organization for achieving its goals through value-creation and distribution. Defining and implementing a strategic logic of an organization is the most fundamental responsibility of the organization's strategic managers.

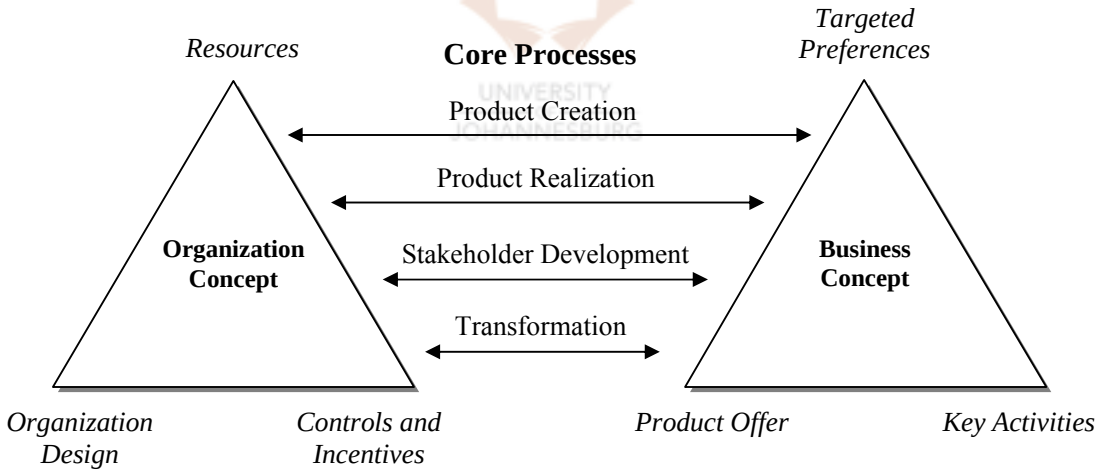
A strategic logic consists of three interrelated components [78]:

- A *business concept* that identifies the intended customers of the organization and the product offers as well as key activities that the organization will use to create value for those customers.

- An *organization concept* that defines the resources the organization will use in its value-creating activities, the organization design for coordinating its activities, the controls it will use to monitor its value-creating activities, and the incentives – or plan for value distribution – that the organization will offer to attract and motivate resource providers in its value-creating process.

- The *core processes* of product creation, product realization, stakeholder development, and organizational transformation through which an organization tries to create and distribute value on a sustainable basis.

The strategic logic of an organization can be graphically depicted as shown in Figure 2.10:



Source: Sanchez [78], Figure 5.1

Figure 2.5: The Three Components of a Strategic Logic

Defining and implementing a viable strategic logic for value creation and distribution in an organization could be an intellectually demanding task as it challenges strategic managers to be insightful in their understanding of the markets for both products and resources. The strategic managers also need to be creative in imagining better ways to

attract both customers and resources to the organization for value-creation and distribution [78].

The business concept can be seen as the "demand" side of the organization's strategic logic. The organization must be able to offer products and services that appeal to some people who then become customers for the organization's products or services. Th basis behind the business concept is that every potential customer will have preferences for certain kinds of goods and services, and that in order to persuade people to buy its products or services, an organization must discover and serve well the specific preferences of people who might become customers [46], [78].

Groups of people who share similar preferences (or wants) for a given kind of product or service are known as market segments [46]. Each organization must choose which preferences in a market it will try to serve, as no organization can be all things to all people. A business concept is therefore founded on a strategic organizational decision to undertake to serve certain identified preferences shared by the people who make up a targeted market segment.

In order to carry out an organization's business concept, managers should devise an effective approach to organize and coordinate the organization's value-creating activities – the organization concept [78]. Identifying the resources needed to carry out the business concept is a task of the management team. Resources are any assets that are available and useful to an organization in its value-creating activities. Resources include tangible assets like land, buildings, machines and equipment; intangible assets like knowledge, capabilities, reputation, brands, relationships and human assets (employees) [46], [53], [78].

Managers should also create an organization design for coordinating the organization's use of its resources in value-creating activities. In designing an organization, managers should determine task allocations (who will do what), the distribution of authority (who will decide what) and the information flows (who will know what). Management feedback is essential, hence managers should establish controls to measure and monitor the organization's value-creating activities [78].

The organization's business concept and organization concept determine the value-creating and distribution activities that the organization will undertake and how it will

attempt to perform them. The types of activities that any organization will be involved with can be grouped into four fundamental categories of activities that are named the core processes of an organization.

Product creation is the first core process. An organization decides, whether through creativity or imitation, to create a product or products that the organization will offer to markets [20], [88]. After the product has been designed, the product needs to be produced and delivered to the organization's customers. The product also needs to be supported in various ways, including maintenance and repair services [6], [73], [88].

Product realization are all the activities that an organization undertakes to make, deliver, service and support its product offers. Like product creation, product realization is a core process that all organizations must sustain in order to survive [78].

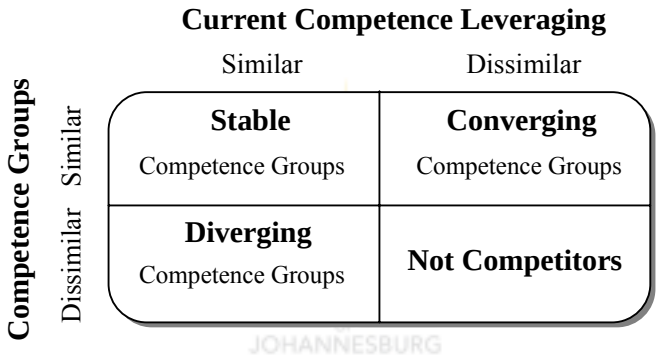
Stakeholder development is the process of attracting and developing resources. The organization must attract and use many different kinds of resources to create and realize product offers. The organization needs to develop relationships with providers of many kinds of resources and should work with these resource providers to develop specialized capabilities suited to the organization's products and targeted market segments [46], [78].

Transformation is the process by which an organization may undergo a change in the way it makes sense of its environment, defines the best opportunities for creating value, organizes its value-creation processes, distributes its value to its resource providers, and undertakes other fundamental aspects of organizational life. Examples of recent changes include movements such as total quality management (TQM), business process re-engineering (BPR), environmental sustainability and social responsibility [20], [68], [78]. All of these movements have changed management thinking in general and the way in which organizations work [68], [78].

Every organization is unique, and hence it can be expected that each organization will have at least some characteristic differences in its goals for and approaches to value creation and distribution. Even though organizations are unique, certain patterns of behavior suggest that the strategic logic of different organizations have important elements in common. Concepts such as cognitive maps, dominant logics, and strategic

groups suggest ways in which different organizations composed of different people may nevertheless begin to share common cognitive elements in their strategic logics. Kotler [46] mentions that organizations such as IKEA, Southwest Airlines and Dell Computers are prime examples of strategic groups carrying out an organizational strategy that will make the most profits. Typically these organizations are seen as providing high value at a low cost in a specific market segment [46].

Competence groups could be used to identify the similarities and the differences in the competence building and leveraging activities of organizations. Figure 2.6 indicates that organizations in stable competence groups share similar competence building and leveraging activities, suggesting that their strategic logic and competitive activities have much in common [78].



Source: Sanchez [78], Figure 5.2

Figure 2.6: Competence Groups

Other organizations that currently engage in similar competence leveraging activities, but exhibit important differences in the new competencies they are building, can be categorized as constituting a diverging competence group, suggesting important differences in the longer-term goals in their strategic logic and in the future competitive activities of those organizations. Organizations currently engaged in different competence leveraging activities but nonetheless building similar competencies for the future are identified as being in converging competence groups [78].

Organizations that do not have similar competence leveraging, nor similar competence building activities are not competitors in product markets and do not form a competence group.

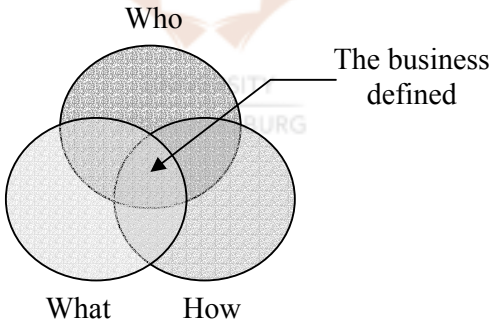
Understanding competence groups should aid managers in making decisions regarding the structure and grouping of individuals. The groups of people can then be aligned according to the strategic objectives of the organization, which will be much easier to do once specific competence groups have been identified.

2.6.2 Defining the Business Concept

In order to define an effective business concept, strategic managers need to determine:

- Who will be served by the organization (customers).
- What will be offered to customers (products/services).
- How will products and services be created and offered to customers.

Figure 2.7 suggests the overlap that defines the business of the organization will not be able to serve all the potential customers equally well. Strategic managers must be aware of this fact, and set boundaries on the groups of customers the organization will try to serve. Extending the boundaries too far leads to unhappy customers [78].



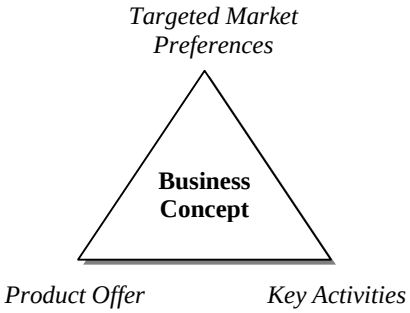
Source: Sanchez [78], Figure 6.1

Figure 2.7: Defining the Business Who, What, and How

Similarly, no organization can provide all the goods and services any group of customers might need. Strategic managers must therefore determine the kinds of products that the organization will create and offer to targeted customers.

Strategic managers should make important choices about how the organization will provide its selected products to its targeted customers, the technologies it will apply, the distribution channels it will use, the way its products will be serviced, its customers supported, etc. Once these questions have been answered and a good "fit"

of customers, products, and means identified a clear sense of the way an organization will try to create value through its activities can be established [78].



Source: Sanchez [78], Figure 6.2

Figure 2.8: The Three Elements of the Business Concept

As suggested in Figure 2.8 the nature of a business is defined by the targeted market preferences, the product offers, and the key activities. Key activities are those activities that an organization should perform well in order to create and deliver product offers to the targeted markets and customers. The key activities in any organization's business concept become top priority concerns of management in designing core processes in an organization's strategic logic [78]. Hence, if quality becomes a critical success factor, then the strategic management of the organization needs to give serious attention to any and all issues of quality that affect the nature of the business.

In designing organizations as value-creating open systems, strategic managers must adhere to certain design principles. Two system design principles are outlined by Sanchez [78]:

- The three elements of a business concept – targeted market preferences, product offer, and key activities – must be internally logically consistent.

- Taken together, the three elements of a business concept – targeted market preferences, product offer, and key activities – must have a clear, credible rationale for superior value creation.

2.6.3 Implementing Strategies

In the case of a small organization, management may develop a strategy and then implement the strategy by communicating it to all employees. In a large organization

this is a more complex process, as a lot of detail has to reach employees at all levels in the organization. If one looks at a bank, everyone, from the tellers who deal with the public, right up to top-level management, needs to be informed about how their behavior affects the welfare of the organization, and how changed behavior following a new strategy can improve this [87].

Once everyone understands that a new strategy and approach is necessary, and by stimulating the necessary decision-making, a strategy can then possibly be implemented successfully. Things that need to be taken into account include organizational growth, profitability, cash flow, even how planning is executed [78], [87], [92].

Implementing a new strategy should be approached with the help of experts, knowledgeable in the field of organization change. The more complex the organizational structure, the more complex the communication process. The general communication procedure is that the managing director discusses the strategy with his/her management team, who will then go and distribute it to employees that they manage [87].

The manner in which employees interpret messages depends on their reference framework. This could possible lead to confusion and misinterpretation if the reference framework of the management team and the employees differ. Therefore even though people are working towards a common goal, they might not understand how it can be done in the best possible manner. It is very important for the goals and steps of a strategy to be communicated in such a way as to be understandable to all levels of employees. This would lead to the influence of the strategy becoming more visible to employees. The information about the strategy should become processed information, that they may feel have not been forced onto them. Employees should be made a part of the strategy according to their skill levels and expertise if it is possible [20], [87].

2.7 CONCLUSION

In their book Competing for the Future, Gary Hamel and C.K. Pralahad [26] offers some insights into business management for the 21st century. A forward-thinking strategy is mentioned where organizations should “seize the future” rather than

maintaining their status quo. They argue that organizations that are focused entirely on the present are merely running in place, and will lose sight of the future, only to be outpaced by their competitors.

The corporate icon Jack Welch also took a similar stance by developing a vision for the business, and then changing the culture to achieve the vision, rather than waiting for the organizational culture to drive the change. He had three methods to put ideas into action; go for the three ideals of: speed, simplicity, and self-confidence. Managers had to make their decisions clearly and quickly, and get improvement teams started on implementing quick fixes.

One of the methods he firmly believed in was the adoption of total quality methods to save costs, raise productivity, and delight customers. Employees should also never relent in the insistent pursuit of better personal and organizational performance. Another key factor of improvements was the use of quality programmes for management development. These programmes helped General Electric to become the market leader in the business areas they competed in, and there is no reason that a similar strategy should not be effective in other environments [30].

Jack Welch believed in having a plan to become the best, and using best practices and measurement control to drive performance on operational indicators.

Hall and Rosenthal [25] suggested that organizations need to go beyond "strategic planning" and embracing "strategic architecture building". Planning must not just focus on the short-term of a percentage of market share here or there, but organizations also should not just focus on forecasts in the attainment of unrealistic goals. A better strategy would be to look at ways to enhance and improve existing opportunities and competencies in the organization. This could lead to increased productivity for the future, enabling the organization to rise to the challenge of new demands from customers.

One of the ways to enhance and improve existing competencies in an organization is to re-look the quality of the products and services that the organization supplies. Quality is a difficult concept to pinpoint – the lack of quality is often the first indication of a need for any type of improvement. In order to improve quality, one firstly needs to understand the full meaning of quality...

Chapter 3

Quality Management

"There is nothing either good or bad, but thinking makes it so." – William Shakespeare

3.1 INTRODUCTION

Numerous definitions for the word quality exist, most notable being the link between quality and user needs. Quality is still hard to define and difficult to measure, but is most easily recognizable in its absence, such as when a piece of software is not working, or when something new does not seem to work.

Some insights to remember about quality according to Gillies [22]:

- Quality is not absolute, and can have many different meanings in different situations.
- Quality is multidimensional with many contributing factors and is thus difficult to summarize in a simple, quantitative way.
- Quality is generally subject to constraints such as cost and resources.
- Quality is about acceptable compromises, such as reliability considered to be more important than functionality.
- Quality criteria are not independent and interact with each other causing conflicts.

The problem with quality definitions are the context-dependence of quality, the ensuing is then an attempt to define quality and quality management in a more general sense.

3.2 DEFINITIONS

The definitions in this section were included to clarify some of the more general quality terms that were used in this chapter of the dissertation. Most of the definitions were sourced from the references cited.

3.2.1 Quality

The English Usage Dictionary [2] states that quality is the degree of goodness or excellence. This definition is insufficient for the purposes of this dissertation as the nature of "excellence" must be considered in more detail to make the definition more effective.

According to Gillies [22] a more formal definition by the International Standards Organization (ISO) states that quality is:

> *"The totality of features and characteristics of a product or service that bear on its ability to satisfy specified or implied needs."*

This definition associates quality with the ability of a product or service to fulfill its function through the features and characteristics of the product. This definition of quality could be applied to products/system, from motor cars to software, and will be the general definition used for quality in this dissertation unless otherwise noted.

3.2.2 Quality Assurance

Quality assurance consists of the auditing and reporting functions of management. The aim of quality assurance is to provide management with the data necessary to be informed about product quality meeting its goals [45]. Freeman-Bell [20] states that quality assurance is also a method of managing all the activities that affect the quality of goods or services in order to prevent faults.

3.2.3 Quality Assurance System

According to Kit [45] a quality assurance system can be defined as the organizational structure, responsibilities, procedures, processes and resources for quality management.

3.2.4 Quality Audit

A quality audit as stated by O'Connor [62] is an independent appraisal of all the operations, processes and management activities that can affect the quality of a product or service. The objective is to ensure that procedures are effective, that the procedures are understood and that they are being followed.

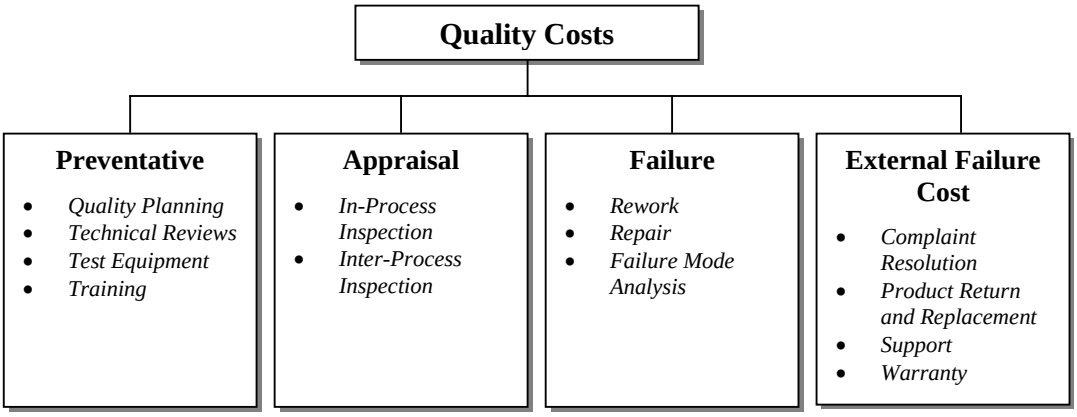
3.2.5 Quality Control

Quality control is the process of variation control [20]. The reasoning behind quality control is that all processes are monitored and controlled to ensure that they are capable of meeting requirements – hence quality control consist out of a series of inspections, reviews, and tests generated throughout the development life cycle to guarantee that each work product meets the requirements placed upon it.

Quality control activities may be fully automated, manual, or a combination of automated tools and human interaction [45]. Quality control brings with it the principles of inspections with the consequent costs of possibly finding faulty products [20].

3.2.6 Quality Costs

Quality costs include all the costs incurred in the pursuit of quality or in performing quality related activities [20]. Cost of quality studies are conducted to know the current cost of quality and to find out the opportunities for reducing the costs of quality and to provide a basis for comparison [20], [45].



Source: Kit [45]

Figure 3.1: Quality Costs

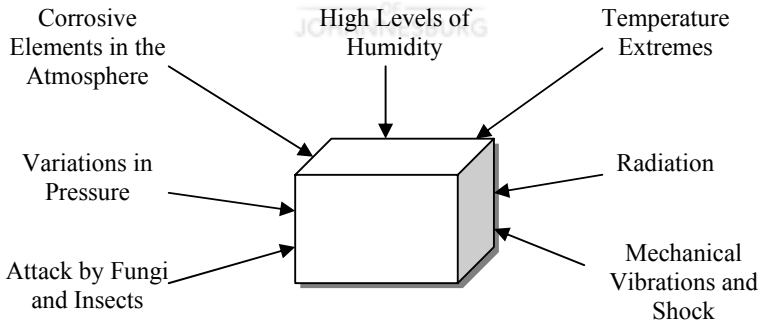
Figure 3.1 depicts the four general groups of quality costs; preventative, appraisal, failure, and external failure costs. These are general quality costs, and are typically used in cost of quality studies.

3.3 QUALITY CONCEPTS AND TERMINOLOGY

This section on quality concepts and terminology describes some of the concepts related to quality such as environmental effects and human reliability. These concepts need to be grasped, as these factors contribute to a lack of perceived quality of software.

3.3.1 Environmental Effects and Interfaces

The prevailing conditions of the environment in which any system is operating have a profound effect on its perceived quality and reliability. The environment affects the system when it is operating (active) or even switched off (static), or stored. Sometimes the effects are even worse when the equipment is switched off (static) as there is no heat to counteract the effects of moisture. This is particularly true for electronic equipment such as computers and their interfaces. This leads to computer equipment always being left switched on [31], [51].



Source: Hignett [31]

Figure 3.2: Environmental Effects

Figure 3.2 illustrates some of the environmental effects that might affect electronic equipment such as computers. To achieve the highest reliability when using computer and electronic equipment, it would be best to operate the system at a fairly constant and medium value of temperature, say 20° +/- 5% Celsius, the sort of conditions in an air-conditioned office.

Humidity, the level of moisture in the air is another environment factor that needs to be taken into account. It may vary from as little as 3% RH for deserts and arid regions, up to nearly 100% Rh in the tropics. Relative humidity (RH) is the ratio of the amount of water vapor present in the air to the amount that would saturate the air at the same temperature [31].

All instruments and computers when transported and handled will experience mechanical vibrations and shock to some degree. Atmospheric pressure variations are important at high altitudes where the pressure falls to a low value, and can cause leaking of seals in components. Usually the conditions around a working piece of equipment are a combination of several environment factors, and the effects of all must be carefully assessed during the design and development stage [31], [51].

Smith [81] describes the electromagnetic interference tests that are performed on computer systems to test for data corruption due to mains and airborne interference. This could lead to system-wide degradation of the system and is common with electronic equipment.

3.3.2 Human Failures/Human Reliability

Hignett [31] discusses the modeling of human reliability and failures of protective functions and risk. The protective function concerns human operations which promote safety, whilst risk deals with maloperations which may lead to hazardous states. O'Connor [62] also states that "human reliability" is used to cover the situations in which people, as operators or maintainers, can affect the correct or safe operation of systems. People are fallible, and can cause system failure in many ways.

It should be emphasized that in modeling human behavior the human element often bridges the boundary between safeguard actions and the initiation of dangerous demands. An alarm system, which relies on a human operator to take corrective action, is significantly dependent on human reliability, which is uncertain over a large range due to variable human reactions and transient extraneous influences. Such a system is clearly not viable in major hazard situations [31].

When designing a system, human reliability and failures should be considered in the design if there is a high possibility that human fallibility might affect reliability and

safety. Factors such as incorrect operation, incorrect maintenance, the ability to detect and respond to failure conditions, ergonomic and any other factors that might influence safe operation should be taken into account during system design. [62].

Reliance on human operator performance could be significantly reduced by the employment of hardware and software systems to monitor and apply corrective actions and hence increase the confidence levels in system reliability performance. However, it should be pointed out that automated systems of increasing complexity require maintenance and modification at higher technical levels of human influence. Elimination of dangerous human operator failures can be partly offset by failures of the human element in design and maintenance.

3.3.3 Inspection and Test

The basic idea behind testing and inspection is that products are checked after production to see if any faulty product can be detected and rejected. This method is probably the oldest method that is in use, and will be in use for many years to come [20], [22], [62].

Inspections pose a number of problems, such as only detecting problems at the end of the production process. This could be due to the entire production line working incorrectly. Inspections can also add a cost factor if products are to be inspected one-by-one, resulting in a cost to the customer [20], [62].

A culture of blame is common with inspection practices, as people will rather hide mistakes than correct mistakes. Inspection also does not lend itself well to service-oriented products, as inspections cannot always be performed on intangible products that cannot be measured physically [20].

3.4 QUALITY MANAGEMENT APPROACHES

In order to guide organizations in their quests for better quality of products and services, a number of initiatives and standards were created. This section describes some of the better known approaches to quality.

3.4.1 ISO 9000

The ISO 9000 quality assurance models treat an enterprise as a network of interconnected processes. For a quality system to be ISO compliant, these processes must address the areas identified in the standard, and must be documented and practiced as described. Documenting a process helps an organization understand, control and improve what is the opportunity to understand, control, and improve the process network that offers the greatest benefit to organizations, that design and implement ISO compliant quality systems [20], [38], [68].

ISO 9000 describes the elements of a quality assurance system in general terms. These elements include the organizational structure, procedures, processes, and resources needed to implement quality planning, quality control, quality assurance, and quality improvement. However, ISO 9000 does not describe how an organization should implement these quality system elements. Consequently, the challenge lies in designing and implementing a quality assurance system that meets the standard and fits the organization's products, services, and culture [20], [38].

3.4.2 ISO 9001

ISO 9001 is the quality assurance standard that applies to software engineering, as well as design and manufacturing. The standard contains twenty requirements that must be present for an effective quality assurance system [45], [95].

The twenty requirements delineated by ISO 9001 [95] address the topics as shown in Table 3.1:

No.	ISO 9001 Requirements
1	Management responsibility.
2	Quality system.
3	Contract review.
4	Design control.
5	Document and data control.
6	Purchasing.
7	Control of customer supplied product.
8	Product identifications and trace ability.
9	Process control.
10	Inspection and testing.
11	Control of inspection, measuring and test equipment.
12	Inspection and test status.
13	Control of non-conforming product.
14	Corrective and preventive action.
15	Handling, storage, packaging, preservation and delivery.
16	Control for quality records.
17	Internal quality audits.
18	Training.
19	Servicing.
20	Statistical techniques.

Source: Freeman-Bell [20], Table 7.2

Table 3.1: Clauses in ISO 9001

Each of the aforementioned requirements gives rise to a need for a documented procedure in the organization, defining the actions that will be taken to ensure that the requirement will be met.

In order for a software organization to become registered to ISO 9001, it must establish policies and procedures to address each of the requirements noted in Table 3.1, and then demonstrate that these policies and procedures are being followed. Software quality is dealt with in a separate chapter.

3.4.3 Total Quality Management

The term total quality management (TQM) is often used to describe a system whereby all the activities that contribute to product quality, not just production quality control, are appraised and controlled by one manager. In this context quality is defined as the totality of features which determines a product's acceptability, and as such includes appearance, performance, reliability, support, etc [20], [39], [62].

Under TQM the QA manager has very wide authority for setting and monitoring quality standards, throughout all functions of the organization. The QA manager then

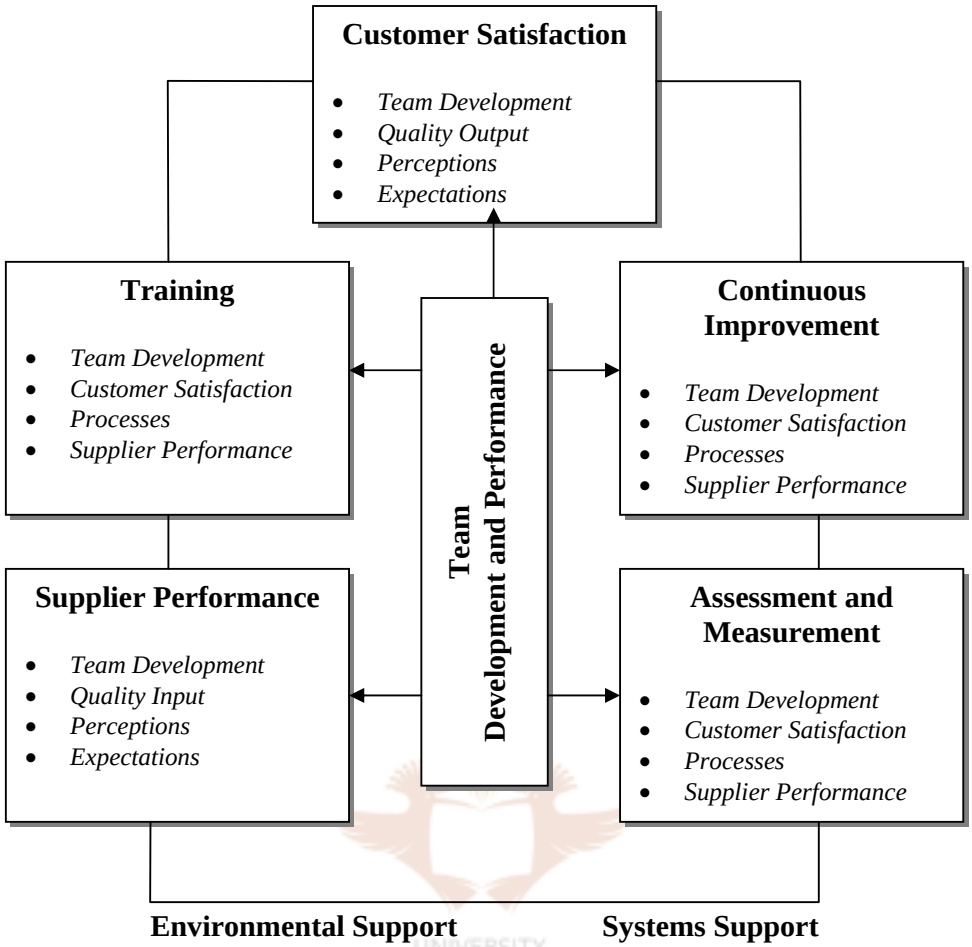
reports directly to the managing director or chief executive. It remains essential for line functions such as design, test, and production to retain responsibilities for their contributions to quality. However, the QA manager is responsible for ensuring that the total approach is coordinated, through the setting of standards, training and performance monitoring [20], [39].

The total QA approach to quality can be very effective, particularly when applied to correct a situation in which quality is perceived as being lower than required, but the reasons cannot be pinned down.

A TQM approach demands exceptional talents of persuasiveness and ability from the QA manager. It is easy for the QA manager and the whole organization to become dissociated from corporate realities, and the authority of the QA manager might be questioned by line departments and project managers [20], [39].

The most obvious solution to this problem is that the managing director or chief executive should take the responsibility. This has an advantage of showing that quality is a top level concern in the organization. All the quality functions can then be integrated with design and production, coordination of standards and training through a chief executive's QA committee [20].

Kinlaw [44] suggests that an integrated approach of the quality functions and the rest of the product lifecycle be used to ensure better quality. The integration of quality with all the organizational functions is also known as a team-centered approach to TQM and is shown in Figure 3.3:



Source: Kinlaw [40], Figure I-1

Figure 3.3: Team-Centered Total Quality Management

The team-centered model for TQM as shown in Figure 3.3 displays the TQM characteristics and depicts the interdependence on team development and performance [44].

3.4.4 Zero Defects

The zero defects (ZD) approach to quality control was developed in the United States in the 1960s. The zero defects approach is based upon the setting of quality control standards, the publication of results, award presentations and poster campaigns. It is an approach that attempts to ensure that all processes are performed without error, by providing training and motivation to all of the people concerned [20], [62], [83].

Many successes were claimed for the ZD approach, but in reality it is hard to sustain the initial enthusiasm for ZD. Also, very few managers exist that are able to set up

and maintain a ZD programme. The difficulties experienced in trying to sustain a ZD programme means that very few organizations are willing to implement ZD programmes [62], [83].

3.4.5 Quality Circles

The quality circles movement started in Japan in the 1950s, and is now used worldwide [62]. The idea is largely based on Drucker's management teaching, developed and taught for quality control application by W.E. Deming and K. Ishikawa. It uses the methods of operator control, consistent with Drucker's teaching, that the most effective management is that nearest to the action. This is then combined with basic statistical process control and problem solving methods to identify and correct problems at their sources. Quality circles are an inherent part of the total quality management approach [20], [62], [83].

In quality circles, the operator is considered to be the person closest to the process he/she operates, and hence will likely understand the problems better. The problem is that the person does not have the authority, knowledge or influence to make the necessary changes, but the quality circle system organized the workers in smaller groups to be able to give them the knowledge and influence. Quality circle teams manage themselves, select their own leaders and members, and the problems to be addressed. If the improvement methods are under their control, they introduce them, otherwise their suggestions get recommended to management who must respond positively.

Quality circles differ from zero defect approaches due to the fact that quality motivation is a normal working practice at the individual and even team level in quality circles. Zero defect approaches need to instill motivation through close management involvement and support [62].

Analytical techniques and problem solution methods are generally taught to quality circles to help identify problems and to generate solutions [20]. The *seven tools of quality* that these quality circles make use of are [62]:

- Brainstorming sessions, to identify and prioritize problems.
- Data collection.

- Data analysis methods, such as trend charts and regression analysis.
- Pareto charts.
- Histograms.
- Cause-and-effect (Ishikawa, or fishbone) diagrams.
- Statistical process control (SPC) charts.

Quality circles should be organized with care, and trained properly. Management support from senior and middle management should be provided to the quality circle. Even so, quality circle recommendations should still be carefully assessed and only actioned whenever they will be effective. If any quality circle recommendations are not actioned, then good reasons should be given for not following those recommendations.

Quality circles have proved to be highly successful in motivating people to produce better quality, and has been part of the Japanese industrial revolution since the Second World War [20], [34], [62], [83]. Introducing quality circles can be an effective measure towards quality improvement in the organization if there does not exist a formal quality control approach, as is often the case with small organizations [34], [62].

3.4.6 Six Sigma

The six sigma approach was developed by the Motorola company in the 1980s [62], [72]. The six sigma approach focuses on two specific areas (also known as a two-pronged approach). The first area of focus is on processes – every process should operate within 6σ limits. The 6σ limit implies that no more than approximately one operation per million should be defective or wrong. This is a misleading statement (and unrealistic) as it implies that every process follows the s-normal distribution [62].

The second focus area is the application of statistical and other measures to identify any opportunity for process improvement [62]. In order to focus on statistical methods, specialists should be trained or expert consultants hired to seek out

opportunities. In order to focus on statistical measures, the support of management is essential [72].

Similar to the zero defects approach, the six sigma approach has also been credited with generating savings in several organizations [62]. The six sigma approach can be costly as trained experts are required to guide employees in improving their tasks [72].

3.4.7 Quality Awards and Benchmarking

One of the first national quality awards to be introduced were the Deming Awards in Japan [62], [68], [83]. The Deming awards are presented every year to individuals, groups, and organizations that achieve notable quality levels or improvements [68], [83].

Later the idea was followed in the United States with the Baldridge Awards, named after the late US Secretary for Trade [56], [57], [58], [59], [60], [62], [68], [89]. Organizations in the United States submit themselves to an assessment process, which is conducted by independent assessors, to determine whether these organizations achieve the highest scores in a range of categories. The Baldridge award winners generally generate improvements as a result of their efforts to impress the assessors.

In Europe a different approach is used that is called the European Foundation of Quality Management (EFQM). The EFQM has produced a self-assessment guide, which organizations can use to conduct their own evaluations [62], [68].

3.4.8 Australian Technology Network

The Australian Technology Network (ATN) of Universities has developed a unique approach to quality and planning that is underpinned by ten key operating principles derived from research on how successful universities manage ongoing change [3].

The ATN quality and planning principles state the following:

- A shared language and understanding of key quality terms and concepts.
- It is clear which processes and systems are set up to prove quality, and which are set up to improve quality.

- There are clear links between quality developments in core activities, and changes in the infrastructure and administrative systems that support them.
- Decision making about quality accreditation, improvement, and innovation is responsive, agile, efficient, and evidence-based, not ad hoc or anecdotal.
- There is an efficient and comprehensive set of tracking mechanisms in place which is used to make informed decision about quality accreditation, improvement and innovation.
- Explicit attention is given to the core values, mission and objectives when priorities for improvement and innovation are determined.
- There are systems in place that ensure that agreed quality improvement and innovation priorities are addressed promptly, wisely and collaboratively.
- Careful attention is given to assuring and developing the quality of staff, especially leaders.
- The support given to staff during the implementation of key quality improvements and innovations acknowledges that change is a learning process for all concerned and not a single event.
- Everyone involved understands that the process of quality assurance, improvement and innovation unfolds in a cyclical and not a linear process.

3.5 QUALITY MANAGEMENT TOOLS AND TECHNIQUES

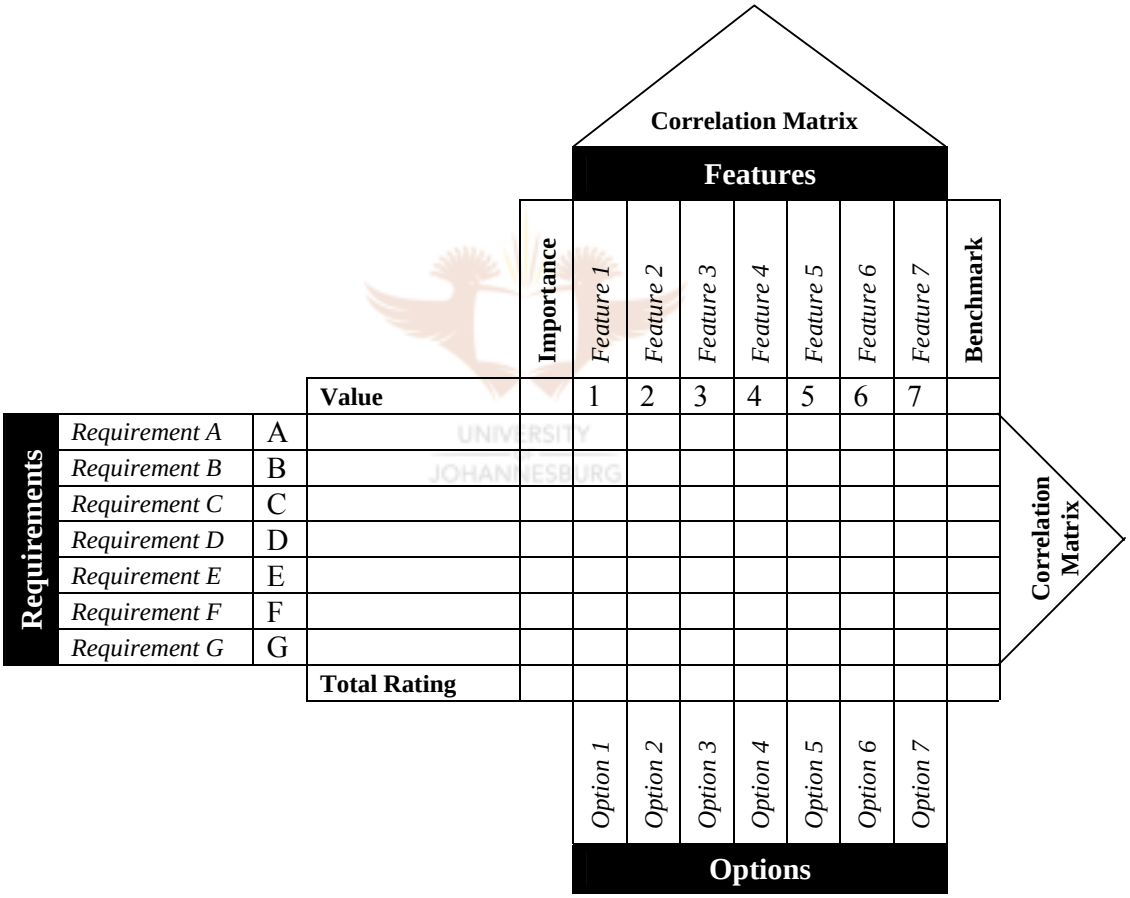
Quality tools and techniques were created to aid organizations and individuals in their pursuit of better quality. Some of the more commonly used quality tools and techniques are described in this section.

3.5.1 Quality Function Deployment

Quality function deployment (QFD) is a technique that incorporates customer requirements into the design of a product. QFD covers aspects such as customer preferences for look and feel, and is a useful and systematic way to highlight the

design, the process, and the control activities that are necessary to ensure reliability, and also quality in the product [46], [62], [83].

QFD is a process that begins with a team consisting of the key marketing, design, production, reliability and quality staff working their way through the project plan or specifications, in an attempt to identify the features that will require control, the control methods and the responsible people. Any constraints and risks should also be identified, as well as the resources needed. No analysis or detailed planning is performed during the QFD process, but the methods to be employed should be identified [62], [83].



Source: O'Connor [62], Figure 7.1

Figure 3.4: Quality Function Deployment Chart

Charts and matrices are commonly used as part of the QFD. These charts and matrices enable the requirements, control methods, responsibilities, constraints, etc. to be listed in a tabular form, as all these requirements relate to design, analysis, test, production, etc. Figure 3.4 is an example of a QFD chart:

Requirements are rated on an importance scale (1-5) on the chart, linked to the design features that can affect them. Each feature is in turn rated against its contribution to each requirement, and a total rating of each feature is derived, by multiplying each rating with the importance value, and then adding these values. This process will highlight the most critical design factors [62], [83].

A "benchmark" column (also called a competitive assessment or evaluation) could be used to rate each requirement, as perceived by potential customers, against those of competitive products. Multiple benchmarks could be used for each different competitor. Adding a benchmarking column is a useful way of showing how the requirements relate to marketing perspectives [62], [83].

The correlation matrices indicate the extent to which requirements and features interact; plus signs (+) indicate positive correlation, and minus signs (-) negative correlation. This indicates if the requirements affect one another in some manner. Any minus signs in the correlation matrix indicate conflicting requirements, hence necessitating further modeling. The options section is used to include the variables that need to be included in such work [62], [83].

The shape of the QFD chart has led to it being called the "House of Quality" due to its house-like appearance [83]. It should be remembered, however, that quality is used here in the widest sense – to include all aspects of the product that will affect its reputation and cost.

Through the use of QFD charts, every aspect of design and production, analysis, test, production process control, final inspection, packaging, maintenance and other functions can be systematically evaluated and planned for, always in relation to the more important product requirements. Requirements or features that are not important will show up as being minor issues for quality purposes, which could lead to cost reductions, and even reliability improvements [62].

The QFD chart is just another quality tool, and should be adjusted to suit the environment and application for which it is being used.

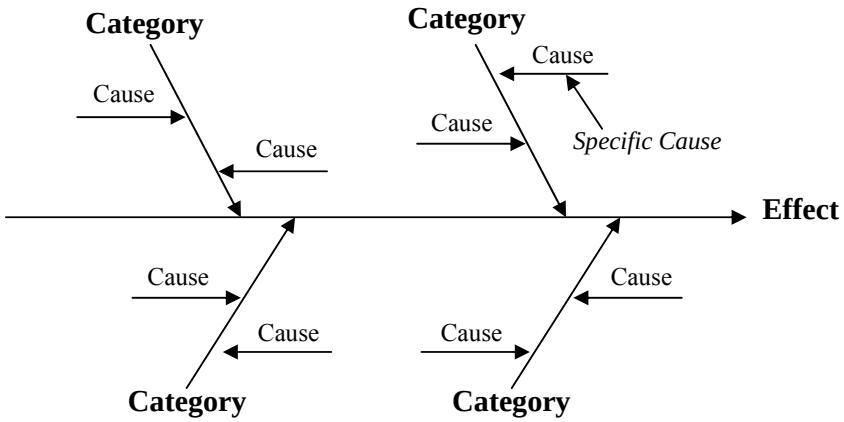
3.5.2 Statistical Process Control

Shewhart developed the methods of statistical process control (SPC) with the emphasis on using the data and charting methods to identify and reduce assignable variation, and to keep processes in control. SPC is the term used for the measurement and control of production variability [62]. Statistical process control essentially evaluates the output of a process to determine its acceptability [83].

SPC rely heavily on the s-normal distribution (see Appendix B for an explanation of the s-normal distribution). Samples from the process under investigation are taken regularly and compared with a predetermined standard. Any variations should be corrected, and follow-up action taken to ensure that all the problems have been corrected [83]. Conventional SPC methods are often simplified and should be approached with caution [62].

3.5.3 Cause-and-Effect Diagrams

The cause-and-effect diagram was invented by K.Ishikawa as an aid to structuring and recording problem-solving and process improvement efforts [20], [62], [83]. The diagram is also called a *fishbone* or *Ishikawa* diagram. The main problem is indicated on a horizontal line, and possible causes are shown as branches, which in turn can have sub-causes, indicated by sub-branches, and so on [20], [62]. An example of a generic fishbone diagram is shown in Figure 3.5:



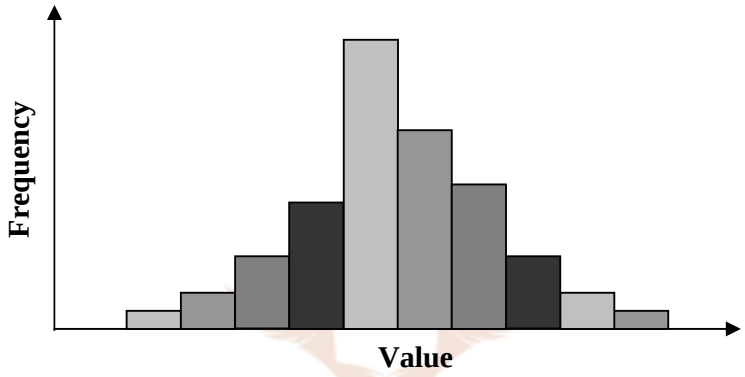
Source: Freeman-Bell [20], Figure 7.9

Figure 3.5: Basic Cause-and-Effect Diagram

Some of the more common categories used in the cause-and-effect diagram are men, methods, materials, machines and environment. The categories on the diagram should generally not exceed six. There is no maximum for the causes.

3.5.4 Histogram

A histogram is a plot (bar chart) of measured data or values that vary about an average [62], [83]. A histogram is a graphical method of keeping count of how often something happens. This will provide one with a representation such as shown in Figure 3.6:



Source: O'Connor [62], Figure 2.3

Figure 3.6: Frequency Histogram of a Random Sample

If more samples are taken, then the width of the vertical bars will be narrower, giving a much better indication of the results of the measurement samples. A histogram has the advantage of giving a visual indication of the relevant sizes of the samples shown on one chart, combining a lot of information into a more compact visualization. As more samples are taken, this can be added to the histogram, without redrawing the complete diagram, allowing data to be gathered over a long period of time [20].

3.5.5 Pareto Charts and Analysis

The Pareto principle states that there is a "significant few and the insignificant many" and is also known as the 80:20 rule, due to a few causes giving rise to a majority of effects [20], [62]. It is often found that a large proportion of failures in a product are due to a small number of causes, i.e. 80% of failures arises from 20% of all the faults.

Therefore, if one analyzes the failure data, one should be able to determine how to solve the largest proportion of the overall failures with the most economical use of

resources [20, [62], [83]. Using the Pareto principles for analysis helps to focus on the main issues and to prioritize work effort [20].

3.5.6 Data Analysis

Data analysis can provide many meaningful insights and forecasts into processes and procedures [53]. It should be borne in mind that these mathematical and statistical data analysis processes have limitations. Mathematical models do not necessarily reflect reality in the way that deterministic, physics-based formulae do [62]. Hence it is necessary to remember that some of the issues to remember when applying statistical methods to engineering are [62]:

- Real variation is seldom s-normal.
- The most important variation is usually that in the tails, where there is inevitably less (or even no) data, the data is more uncertain, and where conventional statistical models can be most misleading.
- Variation can change over time, so that the patterns measured at one time might not represent the true situation at another.
- There may be interaction effects between variables, causing combined effects that are more significant than those of individual variations.

3.5.7 Brainstorming

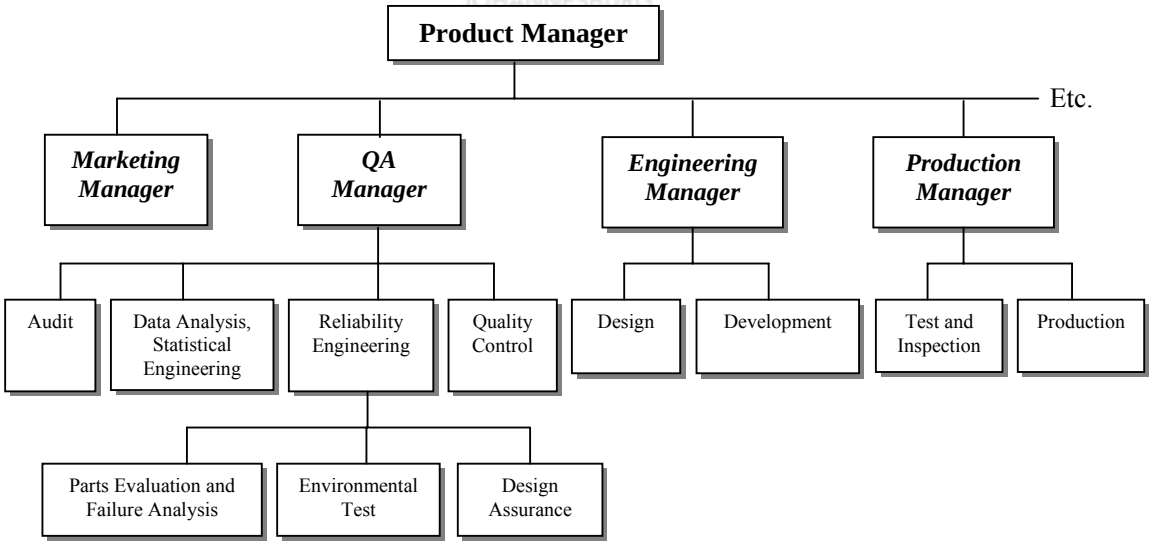
The brainstorming approach is associated with the Taguchi method, and can be used for multiple applications, not just quality purposes [20], [83]. In brainstorming sessions, all the people involved in the design of a product and its production processes meet and suggest which are the likely important control factors, and plan the experimental framework.

The team should consider all sources of variation, deterministic, functional, random, and their likely ranges during the brainstorming sessions, so that the most appropriate and cost-effective approach is determined. The team generally needs a person who is skilled and experienced in the design and analysis of statistical experiments. It is important to create an atmosphere of trust and teamwork, and the whole team must agree with the plan once it has been finalized [20], [62].

3.6 ORGANIZATION STRUCTURE

The author has described the definitions, tools and techniques used by management teams in the pursuit of quality in the previous paragraphs. Another topic that is equally important in grasping how quality management is applied in organizations is the actual structure of the organization. Without the correct people in place, the best policies, procedures and strategies may still fail. Hence the subject of organizational structure needs to be touched upon before highlighting other important aspects of quality management.

Many different activities contribute to the quality of a product, and it is difficult to be categorical about the optimum organizational structure to ensure effective management of quality. Design, development, production, quality control, control of subcontractors and maintenance all play a part in the quality and reliability of products. All the activities need to be coordinated, and resources applied in relation to the requirements of the product. Quality management functions should be integrated with project management functions to ensure that the necessary attention is given to the quality functions. Once the quality management functions are integrated with the project management functions, resources and time should be allocated to achieve the desired quality.



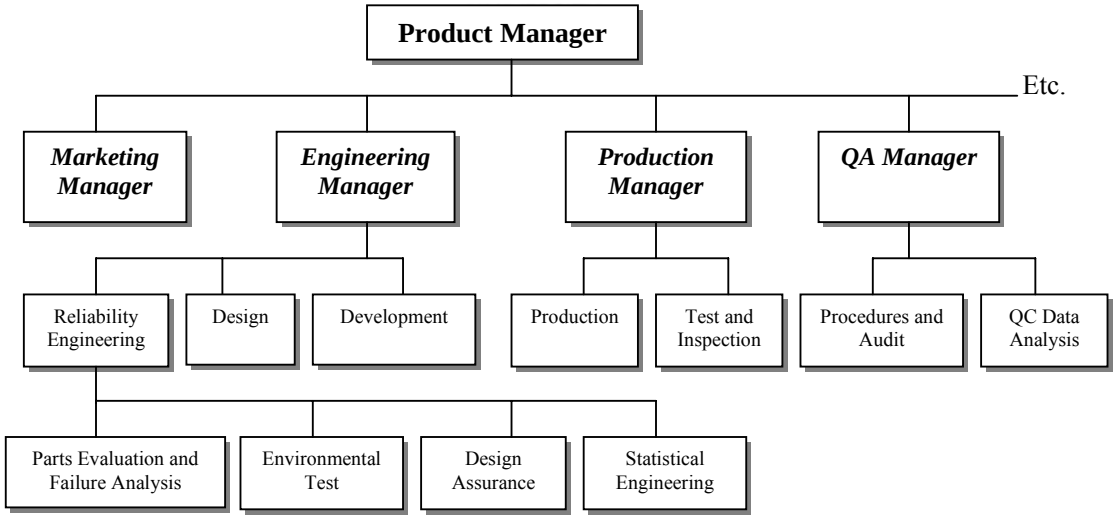
Source: O'Connor [62], Figure 15.5

Figure 3.7: Quality Assurance Based Organization

A quality assurance (QA) based organization as shown in Figure 3.7 places the responsibility for issues such as reliability, quality audits, data analysis, and general quality control with the QA management, which then controls the quality of design, maintenance etc [62]. This organizational form is based upon the definition of *quality as the totality of features which bear upon a product's ability to satisfy the requirement* [62]. There should be a close cooperation with other engineering departments for the purposes of quality control; this leads to a common failure data collection and analysis system to be used.

The QA based organization as shown in Figure 3.7 allows easier integration of some tasks that are common to design, development and production. Failure data systems, statistical methods, test tools and methods are very similar in application and could be re-used for multiple products and projects.

There are many other forms of organization structures, such as the pure product organization, the matrix organization, and a mixed organization structure [48], [53], [62]. The most important approach should be a team-based approach to quality, right from the start of the project, irrespective of the organizational structure [38].



Source: O'Connor [62], Figure 15.6

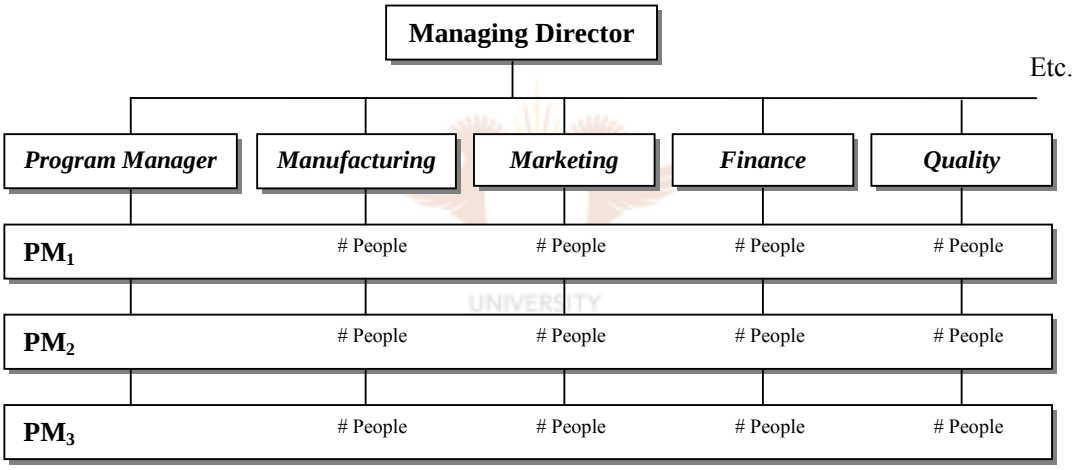
Figure 3.8: Engineering Based Organization

One structure that is also fairly common is the engineering based organization form, shown in Figure 3.8. The main functions of the QA team is the focus on quality

procedures, audits and data analysis, while the issues of quality in design and reliability now falls under the control of the engineering manager.

In the engineering based organization the QA manager is just responsible for controlling production quality and may report directly to the product manager or to the production manager [32], [62]. This is not an ideal situation for software engineering environments as the QA manager and his team need to be part of all the development, design and requirement efforts. If the quality assurance functions are part of the development and production process at the final stages of the product lifecycle, the QA team do not add value, and hence this approach should be avoided.

The most suitable organization structure in the author's opinion that would bring the greatest benefits in terms of quality is the matrix organization as shown in Figure 3.9:



Source: Adapted from Meredith [53], Figure 4.3

Figure 3.9: Matrix Organiztion

The individuals necessary for each different project manager (PM_1, PM_2, etc) to perform marketing, manufacturing and quality functions will come from their respective functional divisions, and are assigned to the project full-time or part-time, depending on the project's needs. It is important to note that the project manager controls when and what these people will do, while the functional managers control who will be assigned to the project and what technology will be used [53].

The type of organization that is finally decided upon depends on where the responsibility for quality activities in the organization mainly lies in relation to the effort that is considered necessary and sufficient during design and developmental

stages. The type of organizational structure is actually less important than the need to ensure that the quality activities are integrated with all activities in the organization.

3.7 RESPONSIBILITIES FOR QUALITY DEPARTMENTS

The responsibilities outlined in this section should not be seen as an exhaustive list of the QA department's responsibilities. The explanations are merely intended to explain some of the more general functions that a QA department would perform as part of their daily duties.

3.7.1 Setting Organizational and Production Quality Standards

The quality manager should decide on the production quality standards to be met in the organization [20], [62]. The quality standards may be decided upon by the customer, as is increasingly the case in both commercial and non-commercial organizations. [62].

The quality standards should apply to the resources, production processes and functions required to create a product. It is the task of the quality manager to decide on the quality methods to be used in inspections and conformance tests [62].

3.7.2 Monitoring Production Quality Performance and Costs

The quality manager should be satisfied that the quality objectives are being met, or that action is being taken to ensure this [20], [62]. These include quality cost objectives [83].

QA staff is responsible for the monitoring of functions such as defect reporting, data analysis, inspections and testing. The QA department should prepare quality reports, and should monitor and assist with problem-solving [62].

3.7.3 Quality Training

Quality control training is typically the responsibility of the quality manager. Training is an important part of the quality process as basic quality concepts should be employed by all the employees of the organization [20], [62].

3.7.4 Specialist Services

The Newport Group [55] states that a critical success factor in achieving continuous quality assurance in dynamic application environments is having the flexibility to draw on the right expertise at the right time, from within the organization as well as from qualified services organizations.

Quality departments provide services such as vendor appraisal, resource assessment, data collection, reporting and analysis. Using an integrated approach to quality engineering yields the greatest benefits during the manufacture, support and maintenance of a product [62].

3.8 TEAM INVOLVEMENT AND COMMITMENT

Quality improvement is more than just meeting the required standards that have been set in the organization. Constant improvement and measurement is also concerned with any factor that could affect the quality of services and products in the organization. In order to become a more quality focused organization, the following list of factors need to be considered [44]:

- The leadership styles of supervisors, managers and executives.
- The activities of entrepreneurs.
- The flexibility of organizations and their capacities to change.
- The aggressive acquisition of new technology.
- The planning processes of organizations.
- The control systems in organizations.
- The visions of executives.
- The cultures of organizations.

All these causes are valid, but they do not capture the day-to-day reality of improvement. Kinlaw [44] states that:

> *"…,improvement, however it is conceived, planned, or initiated, almost invariably comes down to the cooperative actions of teams of people."*

The statement made by Kinlaw [44] relates to the ISO requirement of having management commitment for the quality processes that are implemented in the organization. Management commitment is one of the most important requirements if the quality strategy in an organization is to succeed.

3.9 CONCLUSION

The last half of the twentieth century has been marked by a revolution in the way that quality has affected lives. Many people are unaware of the benefits that an increase in quality has brought to general living standards, and the methods that have brought about this upswing.

Many reasons could be cited for this revolution, such as people just showing what they can achieve, or can do; another driving force has been the awareness that people have started to show in more durable, reliable, high-quality goods, leading to a consumer-driven approach to quality. This drive for higher quality goods and services led to formal definitions of quality, together with quality processes, tools, techniques, procedures and standards such as ISO 9000 described in this chapter.

The Japanese people were among the first in the world to realize that quality should be driven from management levels all the way down to the employees. The Deming quality award originated in Japan, and soon afterwards similar quality initiatives spread around the world. One of the most notable being the Malcolm Baldridge National Quality Award in the United States of America (see Appendix A for more details on the Malcolm Baldridge National Quality Award). The Japanese decision to make quality and reliability national priorities have borne fruit that other countries struggle to mimic [38], [62], [83].

Managers such as Peter Drucker and W.E. Deming showed that continuous improvement could be achieved while costs are reduced and production increased, all at the same time, thereby changing the way people traditionally thought about quality and production processes [62]. To implement a quality assurance system successfully

takes solid planning, reliable tools, the appropriate experience and expertise. Above all, quality management should become an ongoing process.

The ISO 9000 standard includes management responsibility and involvement as a prime requirement. This agrees with authors such as O'Connor [62] who states that the quality approach should be a total approach, requiring a management philosophy that inspires, supports, trusts, teaches and leads, and does not inhibit, frighten or constrain.

In this chapter the author gave a brief overview of general quality principles, tools and techniques. One should realize that without these basic concepts, software quality would not have evolved to its present state. Before one looks at the concepts of software quality, software engineering needs to be examined in order to grasp the subtleties that make the quality of software such a unique topic.

Chapter 4

Software Engineering

Knowledge is power, and the computer is an amplifier of that power…The American computer industry has been innovative, vital successful. It is, in a way, the ideal industry. It creates value by transforming the brainpower of knowledge workers, with little consumption of energy and raw materials. Today, we dominate the world's ideas and markets in this most important of all modern technologies. But what about tomorrow? – Feigenbaum and McCorduck

4.1 INTRODUCTION

Pressman [71] describes a software process as a framework for the tasks that are required to build high-quality software. The issue that arises is that a software process defines the approach that is taken as software is engineered, but software engineering also defines technologies for the process, the technical and non-technical methods, the tools and people.

The IEEE [25] has developed a comprehensive definition that states:

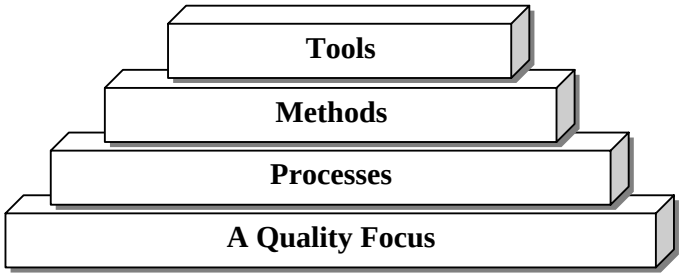
> *Software Engineering: (1) The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, that application of engineering to software. (2) The study of approaches as in (1).*

Another interesting definition of software engineering has been stated by Fritz Bauer [54]:

> *Software engineering is the establishment and use of sound engineering principles in order to obtain economically software that is reliable and works efficiently on real machines.*

These definitions of software engineering does not address *software quality* per se, but it does mention *engineering principles* such as reliability, which can be used as the baseline for starting a discussion about quality in software engineering, Engineering principles covers issues such as customer satisfaction, on-time product delivery, measurement and metrics, and mature processes, hence this will become the starting point for a quality strategy.

Software engineering can be considered a layered technology as shown in Figure 4.1:



Source: Pressman [71], Figure 2.1

Figure 4.1: Software Engineering Layers

Figure 4.1 illustrates that the organizational approach to engineering (including software engineering) should rest on a commitment to quality. Any quality philosophy leads to continuous process improvements, and this ultimately leads to mature approaches to software engineering [71].

The key process areas that lead to effective delivery of software depends on the process layer. The process layer includes the management control of software projects and establishes methods in which software work products are produced, milestones are established and change is managed [71].

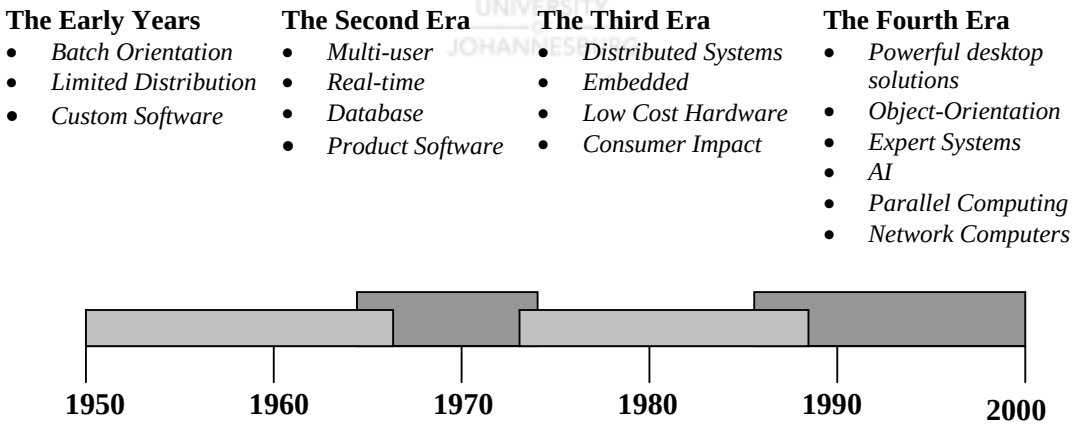
The software engineering methods provide the knowledge (how to) for actually building the software. Basic principles such as requirements analysis, design, program construction, testing and maintenance are all part of the methods. These basic principles govern each area of the software technology and include modeling activities and other descriptive techniques. Software tools such as code generators, compilers and automated testing applications provide automated or semi-automated support for the processes and the methods [71].

CASE tools are commonly used to combine hardware, software, and a software engineering database to create software engineering environments that are analogous to CAD/CAE for hardware [71].

The Software Engineering Institute (SEI) has developed the Capability Maturity Model (CMM) over the past two decades. This initiative has had a tremendous impact on efforts to improve software engineering practices [9], [22], [71], [81]. The CMM has generated a lot of debate, but even so, it provides a very good indicator of the attributes that must exist when solid software engineering is practiced [71], [81].

4.2 THE EARLY YEARS

Software fulfills many roles, both as a product and as a means to deliver products. Software has undergone many changes throughout the latter part of the twentieth century, driven to an extent by hardware developments, changes in computing architectures, increases in memory and storage devices, and a wide variety of input and output devices. Together with these advances in sophistication, problems and complexities started to arise for the people tasked with building complex systems out of a dazzling array of building blocks [71].



Source: Pressman [71], Figure 1.1

Figure 4.2: Evolution of Software

Figure 4.2 illustrates the historical milestones in software development, and shows that in the past 50 years there have been major changes, advances, and technical innovations as far as software engineering is concerned.

During the early years of software evolution, software was used primarily for scientific (number-crunching) applications. Software was not seen as part of the main organizational activities, but more of an afterthought. Software development was considered an art due to the lack of any formal systematic methods and procedures. Software only came to the attention of management once costs escalated due to schedule slippage [71].

The second era of software evolution ranged from the mid-1960s to the late 1970s. Software engineering organizations became more common. New software applications came into being, such as product software, multiprogramming, multi-user systems, not only mainframe applications. Software engineering organizations began selling mass-produced software in order to make a quick profit, leading to issues with software which resulted in the need for software maintenance [71].

The third era began in the mid-1970s and introduced distributed computing, local and global networks, high-bandwidth digital communications and instant data access. Advances in computer hardware occurred rapidly during this period along with increasing capabilities of software. Personal computers was not yet common, hence software was still only developed for business and academic purposes. The one exception to this rule was microprocessors that started to be used in any electrical device, from automobiles to microwaves. At the end of the third era personal computers did become a reality due to the decline in price of computer hardware [71].

The fourth era started moving away from individual computing to powerful networked computing architectures that feature a decentralized client/server environment. Even though rapid advances have taken place in the software engineering industry, numerous issues still plague the software industry [71]:

- Software cannot tap the full potential of the hardware.
- The demand for newer, better programs is still bigger than the ability to supply programs.
- The reliability of software need to increase as one cannot build software that has a high enough reliability and quality to match the requirements of most customers.

These are just a few issues that we face today in the world of software engineering, and by realizing that there are issues that we face, we are in a better position to come up with solutions to these challenges.

4.3 SOFTWARE APPLICATIONS

Software can generally be applied to any situation where a pre-specified set of procedural steps (algorithm) has been defined. Modern software now includes expert systems and artificial network software that are applied to non-procedural situations. The different types of data (information content) together with the order and timing (determinacy) of data determines the nature of most software applications [71].

It is important to remember that as software complexity grows, the clear definition of the type of software disappears, but there are several general types of applications as shown in Table 4.1:

Application	Description
System Software	*Collection of programs written to service other programs. Compilers, editors, file management utilities, etc.*
Real-Time Software	*Programs that monitor/analyze/control real world events as they occur.*
Business Software	*Business information processing. Payroll, accounts payable/receivable, inventory, etc.*
Engineering and Scientific Software	*Number-crunching algorithms. Astronomy, CAD, CAM, stress analysis, biology, manufacturing, etc.*
Embedded Software	*Software residing in read-only memory to control products for the consumer and industrial markets. Performs limited and esoteric functions, or even provide significant function and control capability, depending on application. Microwave ovens, television sets, automobile controls, etc.*
Personal Computer Software	*Word processing, spreadsheets, computer graphics, multimedia, entertainment, database management, personal and business financial applications.*
Artificial Intelligence Software	*Non-numerical algorithms solves complex problems that are not amenable to computation or straightforward analysis.*

Source: Adapted from Pressman [71], Section 1.2.3

Table 4.1: Software Applications

This table is not exhaustive, and should only be used as a guideline, as the field of software engineering is evolving, and this will lead to numerous other types of software applications in the future.

4.4 SOFTWARE MYTHS

Many of the modern-day issues that software practitioners face can be traced back to a mythology that arose during the early years of software development. Software developers spread rumors that had some elements of truth, but were generally misleading and caused problems for managers and technical people alike [71].

One of the most common management myths that anyone in software engineering will encounter, and it is also one of the key reasons for starting this dissertation, is that managers feel that there are a number of standards and procedures for software engineering, and that this will enable the software team to develop high-quality software applications. The existence of a quality standard does not guarantee high quality software, as these standards and procedures might be outdated, and the modern processes and procedures cannot conform to the "old" way of doing things. Another simple problem is that these standards and procedures are rarely complete, and in many cases simply outlines an academic exercise in standardsm [71].

Another common software myth (and probably one of the most widely known) is the "Mongolian Horde Concept" [10]. If a project gets behind schedule, more programmers are added to catch up. This has been shown by Brooks [10] to be a huge mistake. New people that are added to a software project need to be trained by the existing staff complement, thereby reducing productive time that could have been spent on software development effort. In the long run, however, this strategy might pay off, but only if the process has been well planned and coordinated [10], [53].

One only has to read through internet newsgroups, websites, computer magazines, Dilbert cartoons, and talk to other software professionals to realize that the amount of software myths is ever increasing, and to realize that there is a definite need to correct some of the myths.

4.5 SOFTWARE PROBLEMS

Almost all electronic devices nowadays has some sort of "code" or software running on it. Think about washing machines, microwave ovens, engine management systems, etc. The problem is, software is generally full of bugs and faults [81]. Generally software is acknowledged as not being very reliable. Many of these errors can be avoided if only people are willing to take the initiative, and make software more reliable and of a better quality [4], [81]. This can only happen if people are going to be made responsible for the consequences of their poor quality software.

Poor quality software can lead to financial losses, or even worse, to death and injury or people. In 1997 a Korean passenger plane crashed when the height-warning system failed to work properly – 228 people died. In 2000 more than 30 000 trucks and tractors, and more than 6 000 school buses in America were recalled due to poor quality software controlling the brake systems [2].

Another example is the 1999 Mars landing craft that was destroyed when poor quality software shut the engines off 30m above the surface of the planet. The estimated loss was $165 million. Some organizations in the USA estimate that unreliable and poor quality software costs the country $59 000 million per year [2].

According to the Software QA/Test Resource Center the main reason that is being given for this unfortunate set of events is that software is generally not tested very well [33]. Other conditions that are cited include [33], [71], [81]:

- The complexity of modern software packages.
- The pressure to get the software product to market.
- The software industry does not accept any liability for errors.
- Poor work methodologies.
- Miscommunication or even no communication at all.
- Egos, because people like to believe that they are capable of many things.
- Poorly documented code.

- Software development tools that are flawed and contain bugs.

In general people spend about 50% of their time to write a program, and then the other 50% of their time trying to find the errors. In some programs that have been tested, more than 10 faults per 1000 lines are observed, which equates to 10 000 faults for a program that has 1 million lines of code [4].

Five common problems in the software development process are shown in Table 4.2:

Problem	Description
Poor requirements	*Unclear, incomplete, and too general requirements.*
Unrealistic schedule	*Too much work crammed into too little time.*
Inadequate testing	*No one knows if the program is any good until the customer complains or the system crashes.*
Featuritis	*Requests to pile on new features after development is under way.*
Miscommunication	*If the software developers do not know what is needed, or if the customer have erroneous expectations, then problems are guaranteed.*

Source: Created by the Author from Hower [33]

Table 4.2: Common Software Problems

The good thing about software development, is that there are a number of initiatives that one can follow when trying to achieve higher levels of reliability and quality:

- SEI – Software Engineering Institute.
- CMM – Capability Maturity Model.
- ISO 9001.
- IEEE Standards for Software.
- ANSI – American National Standards Institute.

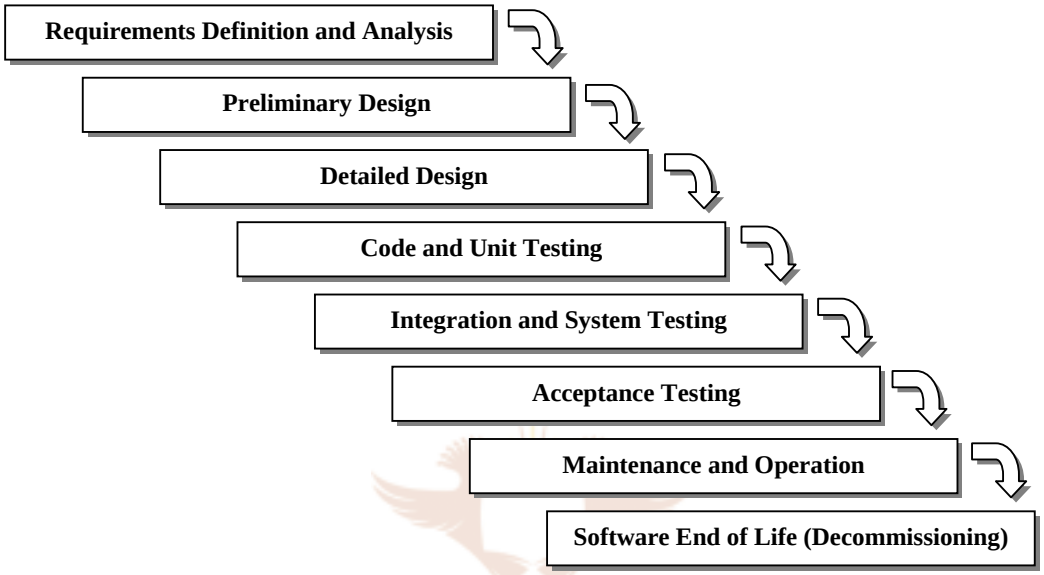
4.6 SOFTWARE DEVELOPMENT METHODOLOGIES

In the software engineering industry, there are several methodologies that are being used to develop software [40], [61], [71], . Some of these are:

- The classic waterfall development cycle.
- Booch and Rumbaugh.

- Extreme Programming (XP).
- The Rational Unified Process (RUP), incorporating the Unified Modeling Language (UML) to develop software specifications.

The most important, and generic, is the classic waterfall software development cycle, as shown in Figure 4.3:



Source: Created by the Author from Pressman [71]

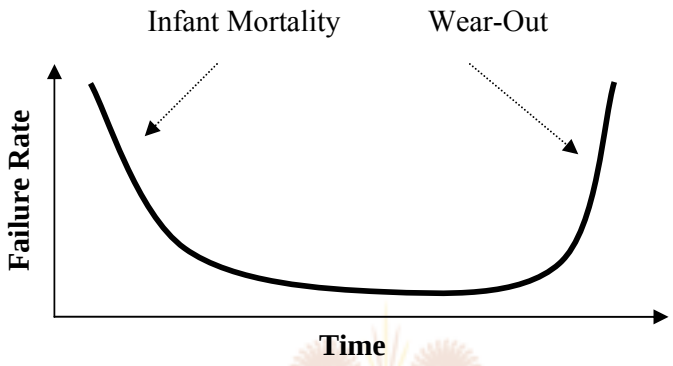
Figure 4.3: Classic Waterfall Software Development Lifecycle

The main reason for only elaborating on the waterfall software development cycle, is that all the development methodologies follow this mindset in some way or another, and just add iterations and loops to the process. Knowing the whole process is important, because one needs to know when to introduce quality requirements, and how to enforce this during the life of the product.

Quality requirement analysis should be done in conjunction with the normal software requirement analysis and definition, as this takes place right at the beginning of the software development phase, and cannot be introduced successfully into the project at a later stage. The sooner the requirements are drawn up, the sooner people will understand the challenges that are facing them, and what they need to do in order to be able to achieve the quality goals that they have set for themselves.

4.7 SOFTWARE RELIABILITY

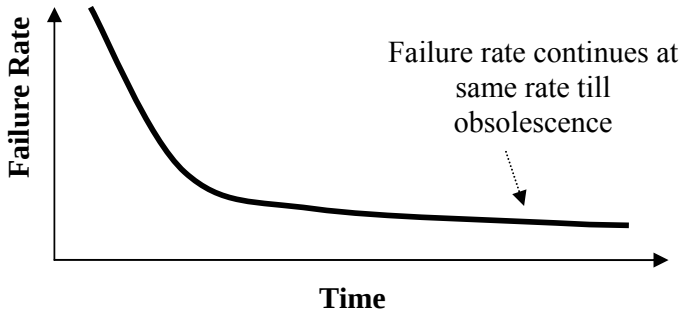
One of the software myths is that "software does not wear out", which is factually true, but practically incorrect. Software does deteriorate with age, and this is one of the reasons why software reliability has become an important topic in the past two decades. Software deterioration is due to the fact that the environment surrounding the software application changes (hardware upgrades, software upgrades, data corruption, etc), causing the software to crash or experience failures.



Source: Pressman [71], Figure 1.2

Figure 4.4: Hardware Failure Curve

The "bathtub" curve, shown in Figure 4.4 depicts hardware failures, indicating that hardware exhibits high failure rates early in its life (design or manufacturing defects) [62], [71], [73]. As the defects are corrected, the failure rate drops to a steady-state level for some time. As time passes the hardware will start to experience rising failure rates. This is simply the graphical depiction of hardware that is starting to wear out [62], [73].

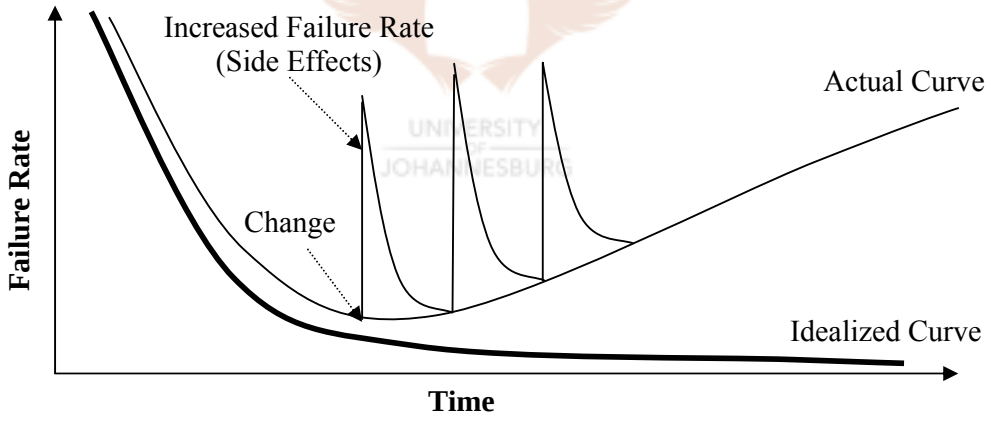


Source: Pressman [71], Figure 1.4

Figure 4.5: Idealized Software Failure Curve

Software, on the other hand, is not susceptible to the environmental factors that affect hardware. Hence, in theory the failure curve for software should look something as shown in Figure 4.5. Defects such as incorrect logic that are not discovered will lead to high failure rates early in the life of software. However, in the ideal world these will be discovered and fixed, which leads to the flattening of the failure curve for software [71].

In the real world, software will undergo maintenance (change) which will lead to new defects being introduced, causing the failure rate to increase (spike), not decrease, as shown in Figure 4.6. In order for the curve to return to the original steady-state idealized curve, more changes are requested, causing the curve to increase and spike again. This leads to an increase in the minimum failure rate, which depicts the deterioration of the software due to change [71]. This is a highly simplified model of software failures, but it does illustrate one important point – software does not wear out…but the software can deteriorate.



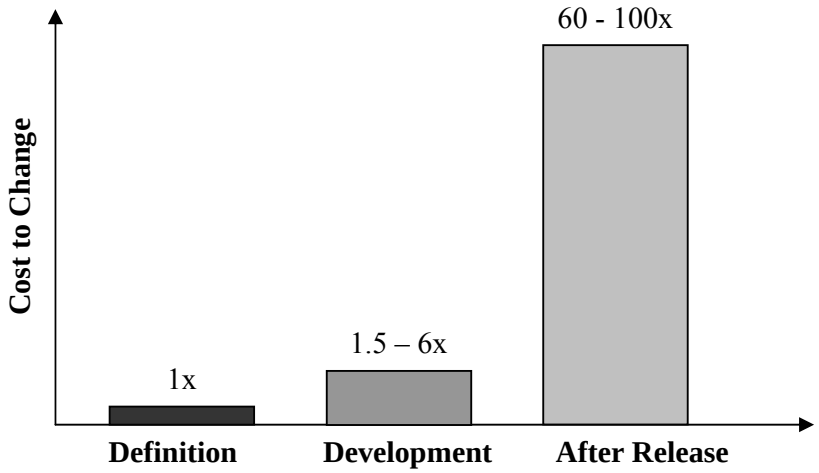
Source: Pressman [71], Figure 1.3

Figure 4.6: Actual Software Failure Curve

An important aspect of wear illustrates one of the major differences between software and hardware. If and when a hardware component wears out, it is replaced by a spare part. Software does not have "spare parts". Any error in software indicates an error in the design, code or the process through which the design was implemented in machine code. Hence software maintenance involves more complex processes than the simple replacement of a hardware item.

4.8 IMPACT OF CHANGES TO SOFTWARE

Software requirements often change, and the impact of the changes to the software is not always felt till late in the software development lifecycle.



Source: Pressman [71], Figure 1.5

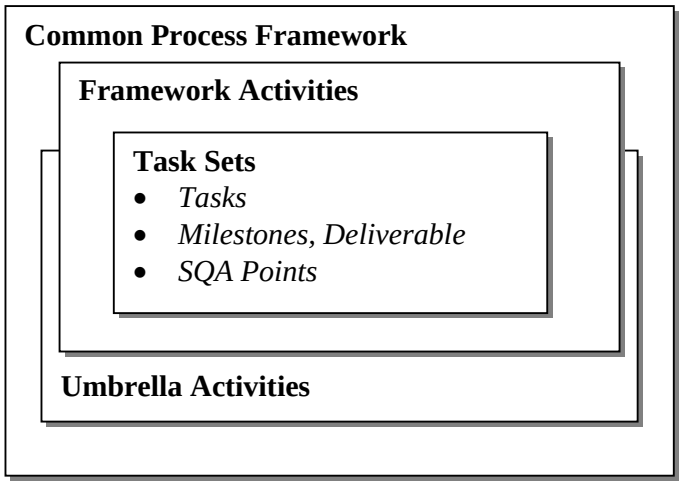
Figure 4.7: The Impact of Change on Software

If serious attention is given early in the development cycle to ensure that the project definition is clear, changes can be accommodated relatively easily. The customer will be able to review the requirements and will be able to recommend modifications with relatively little cost impact. If the changes are requested during the software design phase, then the costs escalate rapidly. The reasons are that resources are now committed and have already started working on the project, and a design framework is in place that will have to be changed. Once the project has reached the implementation (coding, testing) stage, any change will lead to severe impacts on cost [71].

Figure 4.7 also indicates the costs associated with changes requested once the software application has been placed into production. The costs are estimated at an optimistic factor of between 60 and 100 times the cost of changes requested at the time of project definition. This can lead to an *exponential* cost increase [71].

4.9 SEI CAPABILITY MATURITY MODEL

According to Pressman [71], the majority of software processes can be graphically depicted as shown in Figure 4.8:



Source: Pressman [71], Figure 2.2

Figure 4.8: The Software Process

The *common process framework* is defined through the framework activities that are applicable to all software projects, regardless of their size or complexity. *Task sets* are collections of software engineering work tasks, project milestones, software work products and deliverables, and quality assurance points. This enables the framework to be adapted to the unique characteristics of the software project and the requirements of the project team. Umbrella activities overlay the process model and are independent of any framework activity and typically occur throughout the process. The activities include software quality assurance, software configuration management and measurement [71].

The Software Engineering Institute has developed a maturity model that describes the process maturity of software engineering organizations [40]. This is a comprehensive model that is based on a set of software engineering capabilities that should be present as organizations reach different levels of process maturity. An assessment questionnaire and a five-point grading scheme is used to determine an organization's current state of process maturity. The reason for using the grading scheme is to determine compliance with a *capability maturity* model (such as the SEI CMM model) that defines key activities required at different levels of process maturity [71].

Level	Description	Definition
1	Initial	*The software process is characterized as ad hoc, and occasionally even chaotic. Few processes are defined, and success depends on individual effort.*
2	Repeatable	*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.*
3	Defined	*The software process for both management and engineering activities is documented, standardized, and integrated into an organization-wide software process. All projects use a documented and approved version of the organization's process for developing and maintaining software. This level includes all characteristics defined for level 2.*
4	Managed	*Detailed measures of the software process and product quality are collected. Both the software process and products are quantitatively understood and controlled using detailed measures. This level includes all characteristics defined for level 3.*
5	Optimizing	*Continuous process improvement is enabled by quantitative feedback from the process and from testing innovative ideas and technologies. This level includes all characteristics defined for level 4.*

Source: Created by the Author from Pressman [71]

Table 4.3: SEI Process Maturity Levels

Table 4.3 defines the SEI process maturity levels. These levels provide a measure of the global effectiveness of an organization's software engineering practices. The five levels are the results of the SEI assessment questionnaire that is based on the CMM.

The SEI has also associated key process areas (KPA's) with each of the maturity levels. The KPA's describe the software engineering functions that must be present to satisfy good practice at each level. Table 4.4 identifies the KPA characteristics.

Characteristic	Description
Goals	The overall objectives that the KPA must achieve.
Commitments	Requirements that must be met to achieve the goals, and that provide proof of intent to comply with the goals.
Abilities	The things that must be in place to enable the organization to meet the commitments.
Activities	The specific tasks that are required to achieve the KPA function.
Methods for Monitoring Implementation	The manner in which the activities are monitored as they are put into place.
Methods for Verifying Implementation	The manner in which proper practice for the KPA can be verified.

Source: Created by the Author from Pressman [71]

Table 4.4: Key Process Area Characteristics

The KPA's are mapped into different levels of the process maturity. The KPA's that should be achieved at different levels of each process maturity are defined as shown in Table 4.5:

Level	Definition	Key Performance Area
2	Repeatable	*Software Configuration Management* *Software Quality Assurance* *Software Subcontract Management* *Software Project Tracking and Oversight* *Software Project Planning* *Requirements Management*
3	Defined	*Peer Reviews* *Intergroup Coordination* *Software Product Engineering* *Integrated Software Management* *Training Program* *Organization Process Definition* *Organization Process Focus*
4	Managed	*Software Quality Management* *Quantitative Process Management*
5	Optimizing	*Process Change Management* *Technology Change Management* *Defect Prevention*

Source: Created by the Author from Pressman [71]

Table 4.5: Process Maturity Key Performance Areas

Each of the KPA's is defined by a set of key practices that contribute to satisfying its goals. The key practices are policies, procedures and activities that must occur before a key process area has been fully instituted. The SEI defines key indicators as "those

key practices or components of key practices that offer the greatest insight into whether the goals of a key process area have been achieved" [71].

4.10 CONCLUSION

Software is the key element in the evolution of computer-based systems and products. Software has developed from specialized problem-solving and information analysis tools to an industry in itself. However this legacy has also created a number of problems that are still in existence, even today. Software can be a limiting factor in the evolution of computer-based systems if the software engineering process is not approached with a quality frame of mind [71].

The major intent of software engineering is to provide a framework for building software of a higher quality, yet it seems that it has not always made good on this promise. The next step to evaluate in the search for software excellence is how quality relates to software, and how quality of software can be improved…

Chapter 5

Software Quality

The problem of quality management is not what people don't know about it. The problem is what they think they do know… – Philip Crosby

5.1 INTRODUCTION

Any software engineering approach should have one goal in mind, and that should be to produce *high-quality software*.

Software quality is an extremely problematic area when compared to areas such as manufacturing [40], [81]. IT professionals have been asked why software quality is so different from other types of quality, and it was suggested that each type of product has its own set of quality demands, but that computer software was notably problematic for the following reasons [40], [71], [81]:

- Software does not have a physical existence.
- There is seemingly a lack of client needs from the start of the project.
- The client needs change many times over the duration of the project.
- There are rapid changes in the hardware and software during the lifetime of the project.
- Customers have high expectations with respect to adaptability.
- It is considered as an intellectual object.

The expectations of the customers with regard to adaptability and flexibility adds to the issues experienced with software quality.

According to Pressman [71] and other authors such as Kit [45] the term software quality assurance (SQA) refers to the umbrella activity that is applied throughout the software process. SQA encompasses (1) a quality management approach, (2) effective software engineering technology (methods and tools), (3) formal technical reviews that are applied throughout the software process, (4) a multi-tiered testing strategy, (5) control of software documentation and the changes made to it, (6) a procedure to assure compliance with software development standards (when applicable), and (7) measurement and reporting mechanisms.

In software development, quality of design encompasses requirements, specifications, and the design of the system. Quality of conformance is an issue focused primarily on implementation. If the implementation follows the design, and the system meet its requirements and performance goals, it leads to high quality [71], [81].

In order to create high quality software systems and applications, one needs to apply methods that are effective in helping one achieve quality. Methods that are misleading, ineffective, or just plain counterproductive should be avoided. Management methods, such as software quality assurance, reliability engineering, etc. will be either strategic or tactical. The strategic principles and methods prepare the organization for the work that must be done in the future [78], [83].

The most important strategic methods are shown in Table 5.1:

Important Strategic Methods	✗/✓
Top or upper management commitment to the quality management strategy through all functions and phases.	✓
An effective organization and people.	✓
Effective quality methods, capabilities and procedures.	✓
Supplier, subcontractor, and vendor development.	✓
Training, including management, technology and the tactical methods must be consistent with the organization and procedures.	✓
Research.	✓

Source: Created by the Author

Table 5.1: Important Software Quality Strategic Methods

Some important tactical methods are shown in Table 5.2:

Important Tactical Methods	✗/✓
Reliability design analysis, and the effective use of CASE tools.	✓
Failure reporting and criticality analysis systems (FRACAS).	✓
Quality control and continuous improvement.	✓
Maintenance planning and methods.	✓

Source: Created by the Author

Table 5.2: Important Software Quality Tactical Methods

The software quality methods should be applied over the long term, as the methods do not generate benefits right away. The time scales for effective payback can be anything from a few months to a couple of years, depending on the size of the system, and the benefits can continue to grow as time increases. It is imperative that the quality methods be managed appropriately, and be protected from short-term cost-cutting, as is seemingly the way of the software industry.

Specific quality outputs such as software quality assurance plans, software configuration management plans, test plans and release methodology will not be discussed in this dissertation. The reader should consult the numerous volumes of work on these subject matters if more information is required [7], [9], [22], [24], [33], [36], [37], [40], [45], [53], [61], [62], [68], [71], [81], [83], [91]. Appendix C outlines a sample software quality assurance plan which may provide readers with a basic outline of some of the concepts of a software quality assurance plan.

It should also be remembered that the main focus *of increased quality is an increase in customer satisfaction*, and hence quality awards should be approached with caution, as these may not lead to real benefits for the organization's customers.

5.2 SOFTWARE QUALITY DEFINITIONS

This section on software quality was included in order to clarify some of the more general software quality terms and definitions used in this chapter of the dissertation. Most of the definitions were sourced from the references cited.

5.2.1 Software Quality

The ISO definition for quality that states that software quality is:

> *"The totality of features and characteristics of a product or service that bear on its ability to satisfy specified or implied needs."*

This definition can be used for software, but there are a number of definitions that have been developed that refer specifically to software. The Department of Defense in the United States defines software quality as "the degree to which the attributes of the software enable it to perform it's intended end use"" and combines the need to provide a good solution with the requirement to answer the right question. [22]

The software engineering view is taken from IEEE Standard P1601 [71], which states that:

> *"Software quality is the degree to which software possesses a desired combination of attributes."*

This standard states that defining software quality is equivalent to defining a list of software quality attributes required for that system. The standard also states that:

> *"The final software quality of a system is a function of all the preceding phases of the software development cycle."*

This supports the concept that quality must be built-in during design. General agreement has it that most of the quality is determined by earlier phases of development [37].

5.2.2 Software Reliability

Reliability is one of the quality attributes required of software, due to the various applications of software. Software reliability can be defined as the probability of failure-free operation for a specified time, in a specified environment, for a given purpose or the ability of a program to perform a required function under stated conditions for a stated period of time [62].

5.2.3 Software Reliability Metrics

One needs to decide on the metrics in assessing the reliability of software systems in order to quantitatively define the reliability of such systems. Hardware metrics based upon the design being correct can typically not be used for software as hardware

metrics assume that the design is 100% correct. Hardware metrics are also generally based upon component failures, and the need to repair or replace a component once it has failed.

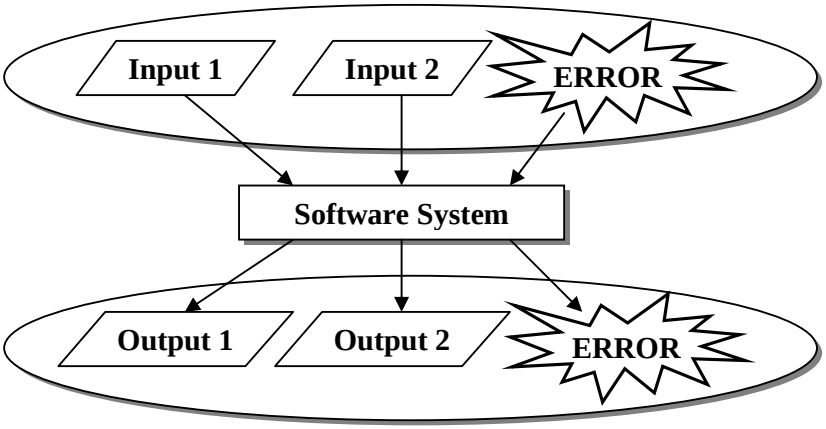
Although a failure may have occurred in software, the system could still be available. This is another major differentiating factor from mechanical systems in general, as a mechanical failure would lead to the mechanical system being unavailable for use.

The reliability metrics that is used most commonly for software reliability metrics include:

- Probability of Failure on Demand (POFOD). This is commonly used for safety-critical or non-stop systems.
- Rate of Occurrences of Faults (ROCOF). This is commonly used for operating and transactional systems.
- Mean Time to Failure (MTTF). This is commonly used for systems with long (time-based) transactional operations.
- Availability. This is commonly used to measure continuously running systems such as telephone exchanges.

5.2.4 Software Failures

Software failures are defined as an unexpected runtime behavior observed by a user of the software system [71]. Figure 5.1 shows the input/output mapping of a typical software system, and what happens when erroneous input is entered.



Source: Created by the Author

Figure 5.1: Software System Inputs/Outputs

Software faults are characteristics inherent in the software that causes failures to occur. Faults do not necessarily cause failures. Faults only cause failures if the faulty part of the software is used. If the faulty part of the software is never used, then the fault will not be noticed, and hence no failure will be observed, as shown in Figure 5.1. Faulty software can still be very reliable.

Software failures can be classified in various classes of failures as described in Table 5.3:

Failure Class	Description
Transient	*Occurs only with certain inputs*
Permanent	*Occurs with all inputs*
Recoverable	*System can recover without operator intervention*
Unrecoverable	*Operator intervention is required to recover from the failure*
Non-Corrupting	*Failure does not corrupt system data or state*
Corrupting	*Failure corrupts system state or data*

Source: Created by the Author from Pressman [71]

Table 5.3: Software Failure Classification

5.2.5 Software Defects

Software defects can be classified into different types as shown in Table 5.4. Defects can be uncovered as software is being developed or after the software has been released to its end user [45].

Software Defects
Incomplete specifications.
Misinterpretation of customer communication.
Intentional deviation from specifications.
Violation of programming standards.
Error in data representation.
Inconsistent module interface.
Error in design logic.
Incomplete testing.
Inaccurate or incomplete documentation.
Error in programming language translation or design.
Ambiguous or inconsistent human-computer interface.

Source: Create by the Author

Table 5.4: Software Defects

5.2.6 Software Quality Factors

Software quality factors, are factors that have been identified in the United States military standard DOD-STD-2167 as shown in Table 5.5:

Quality Factor	Description
Correctness	*The measure in which software is free from defects and from coding defects.*
Efficiency	*The measure to which software performs its intended functions with a minimum consumption of computing resources.*
Flexibility	*The ease with which software can accept enhancements or be modified.*
Integrity	*The extent to which software prevents or controls unauthorized access and modifications to data or code.*
Interoperability	*The ability of two or more products to exchange information.*
Maintainability	*The ease with which software can be maintained.*
Portability	*The ease with which software can be transferred from one computer system or environment to another.*
Reusability	*The extent to which a module can be used in multiple applications.*
Testability	*The ease with which software can be tested.*
Usability	*The extent to which software incorporates human engineering capabilities and features.*

Source: Created by the Authort from DOD-STD-2167

Table 5.5: Software Quality Factors

5.3 THE NEED FOR QUALITY IN SOFTWARE

Software quality is easier to distinguish by it's absence, than by the existence of quality. Statements such as "I'm sorry, but the computer's down, so we cannot take your booking" is heard quite often, as people are became more dependant on computers and software to carry out everyday work and tasks. One of the easiest ways to define poor software quality is through the unavailability of the system. On the other hand, even if we have a system that runs 24 hours a day, 7 days a week, it could be considered a poor quality system if the usability of the system is poor.

The lack of quality has led to increased costs. A Price Waterhouse study [37] found that problem-solving accounted for more than 50% of the total effort in the software industry. The problem solving included problem resolution during testing, changing functionality to meet misunderstood requirements and the maintenance on production systems.

Not only does the lack of quality lead to increased costs, but also to the loss of time. The initial cost of considering quality from the start of a project would be far less if quality is only considered when the software system is delivered to the customer.

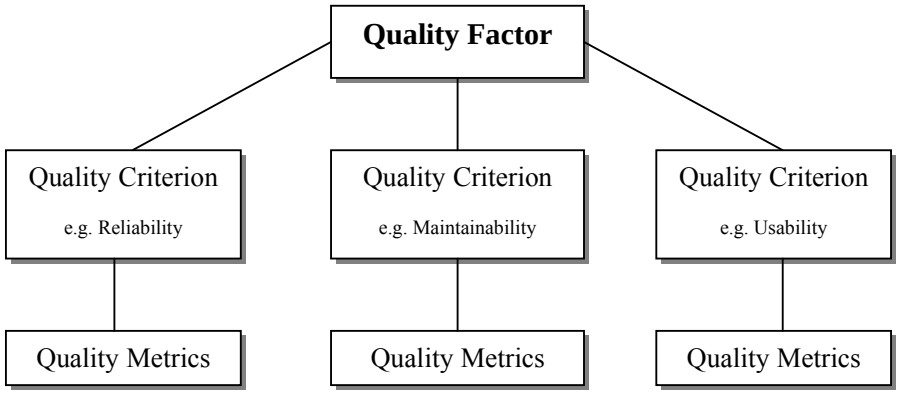
5.4 MODELS OF QUALITY

This section on models of quality describes two software quality models, namely McCall's and Boehm's models of quality. These models are useful as they depict some of the quality factors that users may require of software. These models could also be used to conduct an audit of the quality requirements defined for a software system.

5.4.1 Hierarchical Models

To be able to compare quality in different situations, models of quality need to be established. The idea of a hierarchical model of software quality dates back to 1970. The IBM PC was not yet born, and computing was still centralized within distributed mainframe computing centers. The Boehm and McCall models are typically of this era [22].

A hierarchical model of software criteria is based upon a set of quality criteria, each of which has a set of measures or metrics associated with it. This type of model is shown in Figure 5.5:



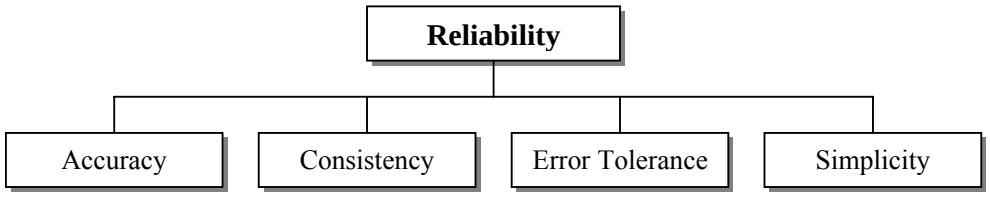
Source: Gillies [22], Figure 2.3

Figure 5.2: Hierarchical Model of Quality

A few examples of quality criteria that is typically employed, include reliability, maintainability, and usability. There are a number of issues relating to the criteria of quality such as:

- The criteria of quality to be employed.
- The inter-relation between the criteria of quality.
- The association of the quality metrics into a meaningful overall measure of quality.

An example of the metrics linked to each characteristic is shown in Figure 5.3:

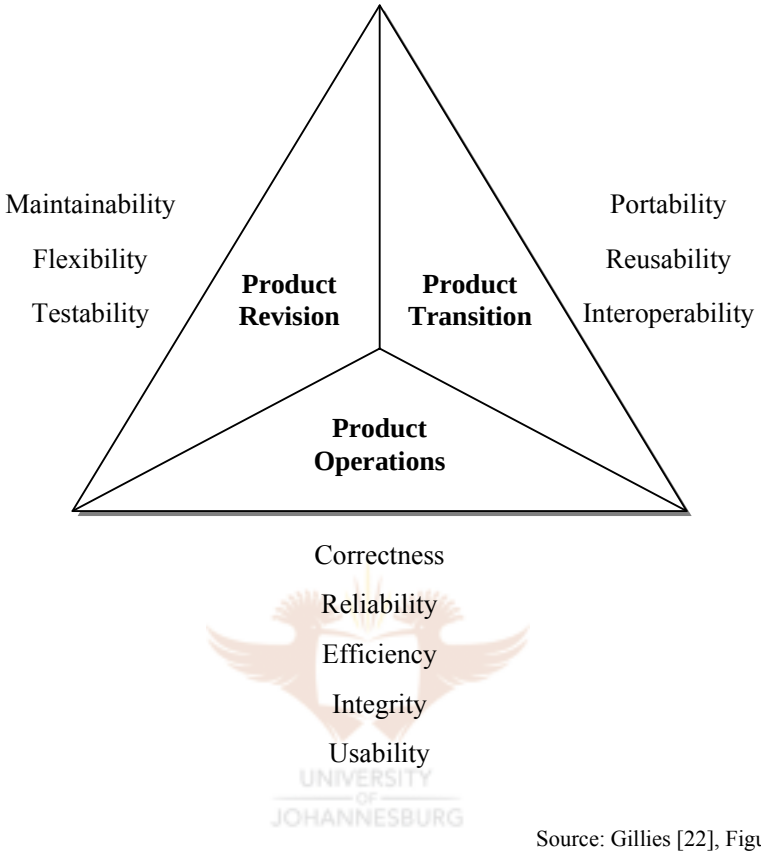


Source: Gillies[22], Figure 2.4

Figure 5.3: Metrics Associated with Reliability

5.4.2 McCall's Model

The model was first proposed by McCall in 1977. McCall's model is shown in Figure 5.4:



Source: Gillies [22], Figure 2.5

Figure 5.4: McCall's Model of Software Quality

This model is aimed at system developers, to be used during the development process. However, in an early attempt to bridge the gap between users and developers, the criteria were chosen in an attempt to reflect the users' views as well as the developers' priorities. The criteria appear to be technically oriented, but they are described by a series of questions which define them in terms acceptable to non-specialist managers.

McCall's model identifies three areas of software work: production operation, product revision and product transition, as shown in Figure 5.4.

- *Product operation* requires that the software product is user friendly, operates efficiently, and that the results produced are correct.

- *Product revision* is concerned with error correction, maintenance and flexibility of the system. This is important because it is generally considered to be the most costly part of software development [22].

- *Product transition* may not be important in all applications of the software system. The move towards distributed processing and the rapid rate of change in hardware are likely to increase its importance.

McCall's quality criteria are described in Table 5.6.

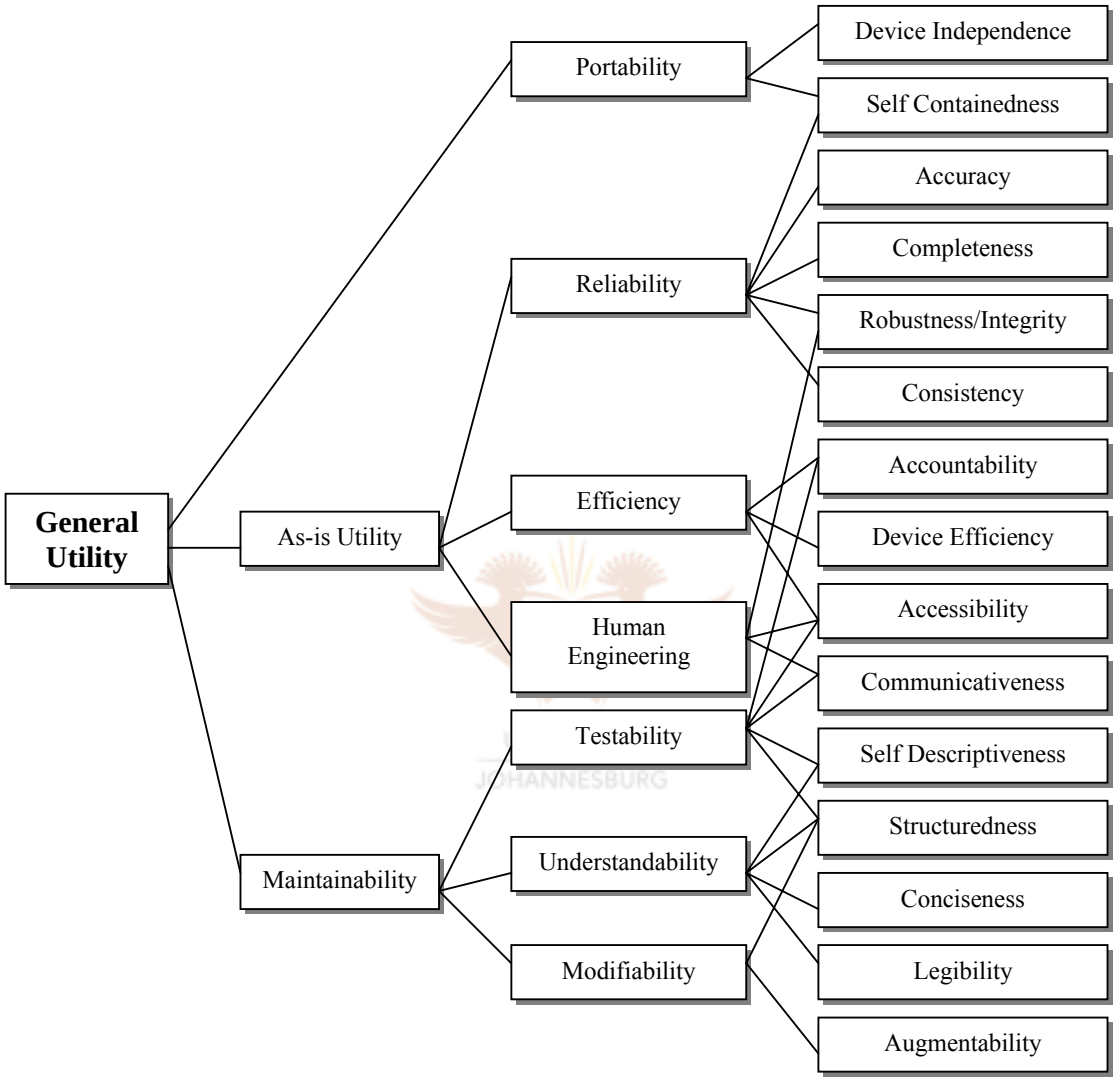
Quality Criteria	Description
Usability	Ease of use of the software.
Integrity	Protection of the program from unauthorized use.
Efficiency	Use of resources, e.g. execution and storage efficiency.
Correctness	Extent to which a program fulfils it's specifications.
Reliability	Ability not to fail.
Maintainability	The effort required to locate and fix a fault in the program within it's operating environment.
Flexibility	Ease of making changes to due to changes in the operating environment.
Testability	Ease of testing the program, to ensure that it is error-free and meets it's specification.
Portability	The effort required to transfer a program from one environment to another.
Reusability	The ease of reusing software in a different context.
Interoperability	The effort required to couple the system to another system.

Source: Adapted from Gillies [22], Table 2.3

Table 5.6: McCall's Criteria of Quality

5.4.3 Boehm's Model

Boehm's model was defined to provide a set of "well-defined, well-differentiated characteristics of software quality" [22].



Source: Gillies [22], Figure 2.7

Figure 5.5: Boehm's Model of Quality

The model is hierarchical in nature, and the hierarchy is extended, so that the quality criteria are subdivided as shown in Figure 5.5. The first division is made according to the uses made of the system. These are classed as "general" or "as is" utility, where the "as is" utilities are a subtype of the general utilities, roughly equating to the product operation criteria of McCall's model. There are two levels of actual quality

criteria, the intermediate level being further split into primitive characteristics which are amenable to measurement [22].

Boehm's model is based upon a much larger set of criteria than McCall's shown in Figure 5.5. The set of criteria that Boehm's model refers to is shown in Table 5.7:

Boehm's Quality Criteria
Usability
Clarity
Efficiency
Reliability
Modifiability
Reusability
Modularity
Documentation
Resilience
Correctness
Maintainability
Portability
Interoperability
Understandability
Integrity
Validity
Flexibility
Generality
Economy

Source: Gillies [22], Table 2.4

Table 5.7: Boehm's Criteria of Quality

5.4.4 Common Characteristics

McCall and Boehm's models share a number of common characteristics arising from the hierarchical nature:

- The quality models focus on the parts that designers of software systems can more readily analyze. Most of the quality criteria are connected with the technical aspects of software quality.
- These hierarchical models cannot be tested or validated until the software system is developed.
- The measurement of overall quality is achieved by a weighted summation of the characteristics.

The nature of the software application should determine the relative importance of these quality criteria. In the quality models the characteristics are closely related. The solution to seek is an optimum balance of quality factors, rather than an ideal solution [22].

5.5 TQM FOR SOFTWARE DEVELOPMENT

Total quality management focus on four focus areas: customer satisfaction, process improvement, a quality culture (human focus) and measurement and analysis. In order to apply TQM successfully in software engineering organizations, the differences that exist between the manufacturing and software development sectors need to be taken into account.

One of the main differences between manufacturing and software development organizations Clients play a big role in the development of software, more so than in the manufacturing sector. The problem arises due to the adaptability of computers and software systems. The needs of the user typically change rapidly and the software is expected to respond to these changes, without affecting the quality of the software in any way.

Applying the techniques of TQM depends on addressing the people and the processes that are involved in the software engineering organizations. Typically, people and processes are linked in software development.

Development processes that are part of a TQM system should be systematic and well understood by all the parties involved. Most of the software development methodologies place a strong emphasis on the gathering of user requirements and verifying that the design process accurately reflects those requirements. Following this type of software development methodology does not always produce the results desired by users [22]. There seems to be a need for an element of prototyping in a development cycle, as users are not able to visualize the software products in the way that they can visualize a manufactured artifact.

One solution to this problem is the use of CASE tools and methodologies encompassed in an iterative loop such as the spiral development method described by Kan [40]. The requirements analysis phase for such methodologies is replaced by a

broader iterative cycle, which aims to promote user involvement in the actual process of software development. Initial requirements are set in consultation with the users.

The system specifications are not intended to be comprehensive or cast in stone. The basis (groundwork) for an initial system (alpha version) should evolve through the evaluation by users. Producing a faithful representation of the user requirements in the implemented code represents a good "match to specification" quality. This means that high quality should be achieved if the specification evolves, rather than remaining static. Strict change control procedures should accompany this iterative methodology to protect the integrity of the software system [22].

Using an iterative approach may improve cost-benefit and on-time delivery through the early identification of problems in the resulting software system. Another benefit to be derived from such an approach is early delivery of the system. This allows the user to start using the system and train users before the formal deadline.

TQM is also concerned with changing people and organizations rather than just the production procedures in the organizations. This means that the implementation of a quality system such as TQM in a software engineering organization depends on change occurring in both the organization, as well as the people within the organization [22].

One of the principal barriers to quality in software development is the relationship between the client and the developer. At one end of the scale, the relationship is adversarial in nature, and the client is regarded as a necessary evil [22]. This typically results in limited user input. Given the vital importance of user requirements in software development, this is often disastrous for the long term user satisfaction with the product, and the result is that it becomes another technical success that fails to be perceived as a quality product by the user.

Changing the attitudes of development teams and making system requirements more dynamic and flexible are key to the success of a TQM approach in the software engineering industry.

System engineers and developers need to recognize that non-technical and business people can make a real and valid input to the system development process. Ideas,

knowledge and expertise could be gained. The involvement of users should be seen as a fruitful partnership in search of a better product. A better mindset would be a recognition that all the groups involved in the development of software are critically dependent on one another [22].

5.6 THE REQUIREMENTS OF ISO 9001

Once an organization has decided to improve the quality of a product or service, it needs to implement a quality system to manage the quality functions. Various types of quality systems are found within organizations, such as the six sigma and zero defects approaches, all varying in their levels of effectiveness. Standards for quality systems have been in existence for almost twenty years or more (ISO 9001 dates back to 1987). Many of these standards and approaches have been incorporated in the ISO standards, as the ISO standards addresses the organization concepts along with quality concepts [37].

A summary of the main requirements of ISO 9001 as they relate to software development is discussed in this section [71], [95]. These items are largely common sense, and very few people would dispute that they should occur. The quality system is the method of ensuring that they do.

5.6.1 Management Responsibility

The section on management responsibility states that management should have an effective quality policy. The responsibility and authority of the employees whose work affects quality should be defined. A management representative, independent of the development process, should be responsible for the quality system.

The effectiveness of the quality system should be reviewed by audits. General quality aims and objectives should be listed in a prominent organizational document, such as the annual report. More detailed quality aims and objectives should be stated in the quality manual. A short guide to the quality manual should exist. Training courses for staff should be held to introduce them to the quality system being used in the organization.

Accurate (and up-to-date) job descriptions must be maintained by the organization detailing broad staff responsibilities. A staffing section should be included in the

project plan describing project-specific responsibilities for all the staff involved in the project – both development and quality assurance staff.

Documentation of any software tools for implementing quality controls should be provided to the staff together with training for the software tools used for implementing quality controls. Training in quality control activities, such as system testing or technical reviewing should be provided.

Procedures for hiring staff required to carry out project-specific or once-off quality controls is part of the management responsibility. A senior manager should be made responsible for overseeing the operation of the organization's quality system and it's adherence to ISO 9001.

Regular reviews of the effectiveness of the quality system, based on documents such as project debriefing reports and summary project audit reports, reviews of training courses and quality tools should be held as far as possible. Quality assurance should be performed independently of the software projects if the organization has enough resources.

5.6.2 Quality System

The section on a quality system states that documented standards, procedures and guidelines should exist that have to be adapted and adopted for all the software projects. A standard and procedure should exist that describes how auditing is used to check that project standards and procedures are being adhered to by individual projects. A sample software quality assurance plan is shown in Appendix C to aid in the development of quality procedures and guidelines for software engineering organizations.

The production of quality records which detail that a particular quality control has been carried out includes the collection and retention of quality records, typically describing defects on a phase-by-phase basis during the project. The quality system should also produce a quality plan for every project describing the quality attributes for the software system that each project produces.

5.6.3 Contract Review

The section on contract review states that an organization must review the contract to ensure that it is understood, and that the organization is able to carry out its obligations before it enters into a contractual agreement. A technical review of the customer statement of requirements should be prepared as part of the requirements analysis.

Documentation of the interaction between the customer and the developer should be maintained. Standards and procedures for carrying out a feasibility study should exist. Requirements reviews should be used to determine infeasible requirements. Project plan reviews should be used to determine whether a project plan provides an adequate framework for the production of a software system with the desired quality attributes. (See Appendix C for an example of a software quality assurance plan).

Requirements specifications should be produced containing all the customer requirements. Traceability should be built into project documents so that the customer requirements can be traced to program code via the system design and detailed design.

5.6.4 Design Control

The section on design control states that the software design process should be properly controlled. Designs and design input should be verified as sufficient and correct by performing test activities (Appendix D shows an example of a software test plan to be used for verification and validation activities). Design output must be of the required quality. Any design changes should be controlled. Validation and verification activities such as code reviews applied to designs should be described in the quality manual. (Appendix E shows an example of a software code review checklist).

Traceability between the requirements specification and the design should be built into project documentation. The interface between the design team and other agencies should be documented in the project plan. Procedures should be available to govern the progress of querying the requirements which are fed into the design process. (Appendix G shows an example of a requirements specification review checklist that can be used for validating requirements).

There must be explicit documentation produced which reflects the fact that the input to the design process – the requirements specification – has been satisfied by the design. The design must come under configuration control, i.e. all changes to the design must be rigorously reviewed, applied, checked and documented.

The outputs from a development phase such as a subsystem design should be documented and reviewed. If any developmental item such as a subsystem design is found not to conform to requirements, it should be tagged and not used until it does conform to requirements.

5.6.5 Document Control

The section on document control states that proper procedures for document approval, issue and removal should exist. Any changes to documentation should be controlled and placed under formal change control. The documentation standards and procedures should be contained in the quality system to enable staff concerned with evaluating change to decide whether a change should be approved or disapproved (See the software configuration management plan as outlined in Appendix Q for an example of a document control procedure.).

The configuration management facilities offered by the quality system should be capable of keeping version data and change history data for all documentation.

5.6.6 Purchasing

The section on purchasing states that any purchased material, including brought-in software, should be checked as conforming to requirements. The same level of requirements specification used on a project should be employed to communicate with software subcontractors. A proper software acceptance procedure should be adopted for all software purchased.

There should be explicit evidence that subcontractors are capable of producing software of the same quality as the developed software. The same level of change control should be adopted for purchased software that is employed for developed software. Acceptance testing should be applied to any external software purchased.

5.6.7 Purchaser Supplied Product

The section on purchaser supplied product states that any material supplied by a purchaser, for example, client-provided software, must be properly managed (configuration management) and tested. Any software supplied by the system purchaser should also be checked as rigorously as the software that is being developed.

Software supplied by the system purchaser should be placed under configuration control for product identification purposes. Purchaser supplied software should be accompanied by a requirements specification and a design if possible.

5.6.8 Product Identification

The section on product identification states that the product must be identifiable at all stages of the software engineering process, i.e. configuration management. Each element of the system, be it documentation or code, should be uniquely identified with as much traceability as possible built in between the elements of a system. Version control should be applied to all the elements of a software system. (Appendix H is an example of a software configuration management plan that can be applied to any software engineering organization).

5.6.9 Process Control

The section on process control states that the entire software engineering process should be properly managed. Quality requirements for the software project should be identified in a quality plan, including any tasks which affect quality control. All the tasks should be specified and associated with planning information such as duration and sequence in time.

Documentation such as project plans and quality standards describing the developmental tasks to be carried out should be provided for the staff responsible for such tasks. Facilities for monitoring the execution and completion of a task via the project plan should be provided. Facilities should be provided to enable quality staff to check that a quality control has been carried out correctly; for example, there should be standards for auditing in place in the quality manual.

Standards should be provided for all the software engineering tasks which affect software quality. The project should specify an adequate development methodology and good practices for task such as design. These methodologies and practices should be publicized in the quality manual. If any special processes are identified during planning, these processes should be marked as such in any contractual documents.

5.6.10 Inspection and Testing

The section on inspection and testing states that the quality system should provide checks that no system or part of a system is released to the customer unless it has been checked as conforming to requirements. This requires effective testing, such as module, system and acceptance testing. Test records should be maintained for all executed tests. The quality system should ensure that the developer documents those parts of a system which have not been tested adequately. (See Appendix D for more information on a software test plan).

Adequate testing, such as unit testing, should be carried out in advance of acceptance testing. Documentation should be provided for testers which not only details the test(s) to be carried out, but what the outcome of the tests should be. Final acceptance tests should be derived from the quality attributes described in the quality plan. Documentation should be generated which confirms that a system or part of a system has passed all the tests applied to it.

5.6.11 Inspection, Measurement and Test Equipment

The inspection, measurement and test equipment section states that if any such equipment is used during the software engineering process, then such equipment should be properly maintained and calibrated. For tools that are developed internally, the same quality process that is used for the project on which the tools are to be used should be employed during their development.

5.6.12 Inspection and Test Status

The inspection and test status section states that the status of all software items should be identified. In software engineering terms, this implies configuration management and release control. Software that is undergoing tests should be documented with the test status.

5.6.13 Control of Non-Conforming Product

The section on control of non-conforming product states that untested or faulty software should be kept out of released products. Adequate tests should be implemented, to ensure that a maintenance change, or a change to a baselined item, has not affected any other parts of the item which should not be affected by the change.

5.6.14 Corrective Action

The section on corrective action states that if any defects are found in a software project, then the cause of the defect should be investigated and documented. Corrective action should be performed and should not only affect the part of the system in which a defect was found, but should lead to the modification of all connected documents (such as designs). (See Appendix I for an example of a weekly report that can be used to document any defects that are found in a software system).

Developers should monitor the cause of defects in order to improve developmental practices and the quality system. Standards and procedures should exist to govern the rework that may occur when a defect is discovered.

5.6.15 Handling, Storage, Packaging and Delivery

The section on handling, storage, packaging and delivery should describe the standards and procedures that are in place to ensure that the software contains the correct versions of the components of the software system to be delivered. Other issues such as storage media being damaged (tapes unreeling in the post, compact discs getting scratched or bent) does not occur during handling, storage, packaging and delivery.

5.6.16 Quality Records

The section on quality records states that quality records should be maintained for all projects. The quality records should describe whether a particular quality control has checked that a specific quality attribute has been built into the software system. The steps taken to control the quality of the process should also be recorded for confirmation of quality checks.

5.6.17 Quality Audits

The section on quality audits states that audits of the quality system should be carried out to ensure the quality system is effective. The quality system should contain standards and procedures to determine the conduct, frequency, and relevant documentation of quality audits, together with details on how the results of an audit are to be used.

5.6.18 Training

The section on training states that quality training needs should be identified, and then met. The organization should have a training plan covering not only the training needs of individuals, but also those of the entire organization. The training plan should be reviewed periodically and training records should be available to any staff that carries out project planning activities.

5.6.19 Servicing

The section on servicing states that the developer should have standards and procedures to ensure that any services specified in a contract are being delivered to the customer if a software product is being serviced.

5.6.20 Statistical Techniques

The section on statistical techniques states that appropriate statistical techniques should be used for prediction. The statistical techniques that are used should be mature.

5.7 A STRATEGIC VIEW OF SOFTWARE QUALITY

To understand how software quality is viewed as a strategic objective, the history of software development sheds some light on this. In the 1970s software quality was concerned with how well the software met the technical requirements contained in the specification, emphasizing conformance to specification view of quality. This gave rise to the development models of McCall and Boehm [22].

In the 1980s, the idea of fitness for purpose had already become important in software quality research. Software people talked about “business correctness” and needed to justify its existence in terms of a return on the resources employed. In this sense, the

software engineering environment (IT) is not much different to other industries as is commonly thought. Juran mentions the issue of fitness for purpose in manufacturing in the 1940s [39].

The way that software is used in organizations has changed from mainframe computing, to distributed architectures and powerful desktop computers. A modern view of the software environment is a more strategic view, one that supports the overall business strategy of the organization. Software has become a strategic objective, changing the scope of software quality to be more strategic in nature. Even if one defines software quality in general terms as fitness for purpose [39], a strategic view of quality might be defined as the degree to which the software engineering function fulfills its objective of enabling the business strategy to be achieved [22].

The purpose of the software engineering approach is defined within the information technology (IT) strategy of the organization. Fitness for purpose is therefore the match between the IT and business strategies. (This is also sometimes described as IT effectiveness). The IT strategy should include the totality of the software engineering effort within an organization, thus representing how effectively an organization is making use of software technology, and how this is reflected in the achievement of the overall business objectives.

Resolving the contributions from IT changes and management changes are tricky, IT changes not accompanied by management changes are less likely to be effective. IT often has to play an enabling role in management changes. Sometimes this encouragement of management change is a valuable part of IT practice [22].

The consideration of IT effectiveness rather than simply software quality reflects a certain maturity in the technology. Traditionally investment has been justified in terms of increased efficiency, but a more sophisticated role for IT in today's commercial environments means that the justification for investment rests upon more sophisticated arguments and forms part of an organization's strategic decision making process.

Many of the claimed benefits for IT do not appear explicitly on a balance sheet, e.g. better decision making, better response time to customers. There is a need to distinguish effectiveness from efficiency. IT efficiency is the ability of the IT function

to enable the organization to reap short-term explicit financial benefits which should appear on the balance sheet. Examples include savings in staff costs, increased production for the same staff cost, etc.

IT effectiveness is sometimes concerned with concepts such as competitive edge, organizational image, and quality of decision making. Two of the general IT concepts to consider for IT effectiveness are productivity and quality. Productivity is visible and quantifiable, whilst quality is much less so. Another distinction that may be made is that quality and effectiveness are long term, while productivity and efficiency show immediate benefits [22].

5.8 CONCLUSION

Software quality is an interesting and important field due to the conflicting views and methods that people take in order to try and improve the quality of modern software. An interesting view is that taken by Gillies [22], by saying that quality is determined by people, because:

- People and organizations have problems that are tackled by computer software.
- People define the problems and specify the solutions.
- People implement the designs and produce code (most of the time).
- People test the code.
- People use the final systems and make judgements about the overall quality of the solution.

The tools, processes and quality management systems are all aids to enhancing quality, provided that the people are capable and motivated towards their effective use. The largest barrier to the effective introduction of CASE tools with their promise of higher quality systems remains the impact upon human working practices. Whilst part of the development process, e.g. the implementation of designs to produce code, may be improved by automation, most tasks remain in the human domain, although the tools and systems may provide effective aids [22].

Thus, the context of quality should be seen in the context of assisting people and enabling people to carrying out their functions effectively. This is where the existence of an organizational quality strategy should pay positive dividends.

The concepts of software quality was examined, together with software quality models such as McCall and Boehm. Failures and quality factors were described to assist one in identifying some of the more common reasons for a lack of software quality, together quality factors which can increase the quality of a software system if considered right from project inception.

One of the key motivating factors behind quality initiatives is to save the organization money and time. The one major message that can be communicated out of all this is, that the initial cost of considering quality, right from the start, is far less than that of discovering down the road that quality was treated as an optional extra. If quality is overlooked, ignored, shelved, then the organization will find itself mired in a morass of hassle that may cost it a great deal more to clean up.

Software quality aims to ensure the software system fulfills the requirements of the customer. In order to ensure that the software quality lives up to the claims a coordinated effort is required. This leads us to the next step in the search for higher quality software, a strategic quality framework.

The software engineering functions in an organization aims to enable the business strategy to be achieved.

PART TWO

STRATEGIC QUALITY FRAMEWORK

Chapter 6

Strategic Quality Framework

"The ancestor of every action is a thought." – Ralph Waldo Emerson

"The great end of life is not knowledge but action." – Thomas Henry Huxley

6.1 INTRODUCTION

The two quotes cited above summarize the goal of this chapter – action. What good is knowledge without being able to put it to good use. This section on a strategic quality framework aims to combine the ideas of strategy and quality into a strategic quality framework for software engineering organizations. The tools, strategies and methodologies for implementing a quality framework for quality will be addressed.

The author did not delve into the intricacies of testing, configuration management, promotion models, etc, because it was felt that there is more than ample information on such topics available in both the literature and on the world wide web [7], [9], [21], [22], [24], [25], [33], [36], [37], [40], [45], [55], [61], [71], [81], [93]. Instead, the focus falls on the implementation, or in some cases, the non-implementation of a framework to allow software engineering organizations to start measuring and improving the quality of their products and services, to ultimately become the quality leaders in their respective fields.

It has been mentioned on numerous occasions in the dissertation that management commitment to a quality strategy and process is vital if any organization is to implement such a process successfully.

6.2 ADVANTAGES AND DISADVANTAGES OF A QUALITY STRATEGY

Even though the advantages outweigh the disadvantages of implementing a quality strategy for software engineering organizations, it is still worth noting that there are some definite short-term *disadvantages* in implementing a quality strategy.

- Time is spent on improving quality and drawing up quality management strategies that could be spent on other projects.
- Money is spent on the quality management efforts that does not improve quality in the near future.
- People are now taken off other projects to work on the quality management strategy, leading to a decrease in development efforts.

These are some of the perceived disadvantages of implementing a quality management strategy, and are a real risk to the benefits and improvements that could be made to any software engineering project. Time and money are the main factors behind the reasoning of the disadvantages.

Management should adopt the quality strategy and explain the tangible benefits that could be given to customers if the strategy is to be implemented in the organization. Only once the strategy has been implemented, and a period of time has passed, may the benefits start to become clearer.

Some of the advantages of developing a quality management strategy are that it shows that management and the people working on the relevant software systems are committed to improving the quality, reliability and availability of the software systems. This could all lead to:

- Increased stability of the system.
- Improved availability and uptime of the software system.
- Fewer visible faults. This does not mean that the software will necessarily be error-free; it merely means there are no visible faults in the sections of the software that is mostly used.

- Less frustration on the part of the users of the software.
- Improved relations between the end-users of the software system, and the people that develop and maintain the software.

When a pro-active approach is taken to managing quality, quality can be expected to improve.

6.3 SELLING THE STRATEGY TO MANAGEMENT

In order to be able to perform the quality management strategy effectively, one needs to obtain the required funding. Management should approve this funding to provide one with the money necessary to perform tasks effectively and timeously.

This can be a daunting and formidable task as one needs to:

- Gain managerial approval to fund new positions for the people to be involved in the quality strategy.
- Train the staff to be involved.
- Build databases of crucial support information.
- Gain commitment from all other supporting departments.

Investments in quality yield significant returns and can add value, but one needs to sell these contributions as capacity assurance, quality, reliability and customer satisfaction to management. Often one will need to re-educate, rather than sell this strategy to management.

Generating a quality strategy will not be enough to create changes in any organization. It does not even matter how good the strategy is, as the strategy will not necessarily sell itself. A person must be able to convince other people of the merits of a quality strategy otherwise it will not benefit the organization.

6.4 THE STRATEGIC SOFTWARE QUALITY FRAMEWORK

6.4.1 Determine the Software Quality Strategy

One of the management responsibilities is to define the responsibilities, the quality policy, and the quality strategy for maintaining or improving the quality of the organization's products or services [83]. The software quality policy should relate to their commitment and positive belief of the quality philosophies, principles and practices. The quality policy is the first substantial visible evidence that management is serious about what they want to achieve with quality. This policy is a guideline of what to be done, not how, and should have the power to assist. It is also extremely important that such a quality policy be applied organization wide.

The strategy that needs to be undertaken when dealing with software quality depends on the state of the software engineering processes in the organization.

Level	Description	Definition
1	Initial	*The software process is characterized as ad hoc, and occasionally even chaotic. Few processes are defined, and success depends on individual effort.*
2	Repeatable	*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.*
3	Defined	*The software process for both management and engineering activities is documented, standardized, and integrated into an organization-wide software process. All projects use a documented and approved version of the organization's process for developing and maintaining software. This level includes all characteristics defined for level 2.*
4	Managed	*Detailed measures of the software process and product quality are collected. Both the software process and products are quantitatively understood and controlled using detailed measures. This level includes all characteristics defined for level 3.*
5	Optimizing	*Continuous process improvement is enabled by quantitative feedback from the process and from testing innovative ideas and technologies. This level includes all characteristics defined for level 4.*

Source: Created by the Author from Pressman [71]

Table 6.1: SEI Process Maturity Levels

The SEI process maturity levels are repeated here from Chapter 5 to assist the reader in determining the level of maturity of the organization. The SEI process maturity levels are shown in Table 6.1.

Very few organizations are at level 5, and even if the organization is at a very high level or maturity, this indicates that continuous improvements are constantly taking place – exactly the kind of mentality needed to keep on improving the quality levels [40].

Once the level of maturity of the organization has been determined using Table 6.1, the quality department of the organization needs to decide on measures to reach a higher level of quality in the CMM model.

Table 6.2 shows the key performance areas that are required for the organization to attain a specific level on the SEI CMM grading system. The key performance areas can be used as a guideline to assist the organization in reaching a higher level on the CMM process maturity scale. Table 6.2 can be used as a guideline for basic quality activities in the organization, as these functions and activities build on one another [40].

Level	Definition	Key Performance Area
2	Repeatable	*Software configuration management* *Software quality assurance* *Software subcontract management* *Software project tracking and oversight* *Software project planning* *Requirements management*
3	Defined	*Peer reviews* *Intergroup coordination* *Software product engineering* *Integrated software management* *Training program* *Organization process definition* *Organization process focus*
4	Managed	*Software quality management* *Quantitative process management*
5	Optimizing	*Process change management* *Technology change management* *Defect prevention*

Source: Created by the Author from Pressman [71]

Table 6.2: Process Maturity Key Performance Areas

The key practices are policies, procedures and activities that must occur before a key process area has been fully instituted. The quality department is responsible for the definition of the software quality policies, and the planning and implementation of the software quality procedures and activities.

Only proceed with the next steps of the quality framework once the quality plan has full management commitment, and the need for quality is realized. Management commitment is essential for the successful implementation of the quality plan.

The quality policy serves as the integrating factor that quantifies the guiding principles of the mission statement into quality objectives. The policy may be scrutinized by both internal and external parties. This means that the organization will have to deliver what it promises or risk affecting its survival as mentioned in Chapter 5. The quality policy should be developed in conjunction with all employees within the organization. The policy illustrates the serious intention of top management, and as such should be written down and distributed to all employees.

The following factors need to be considered in developing a quality policy [38]:

- Who, what and where are the customers?
- What products/services do they require, and when?
- What are the competitors' intentions and what does their quality policy indicate?
- What is the focus of the quality mission?
- Who should be involved in developing the quality policy and who is going to lead its formation?
- Should the supplier(s) be involved?

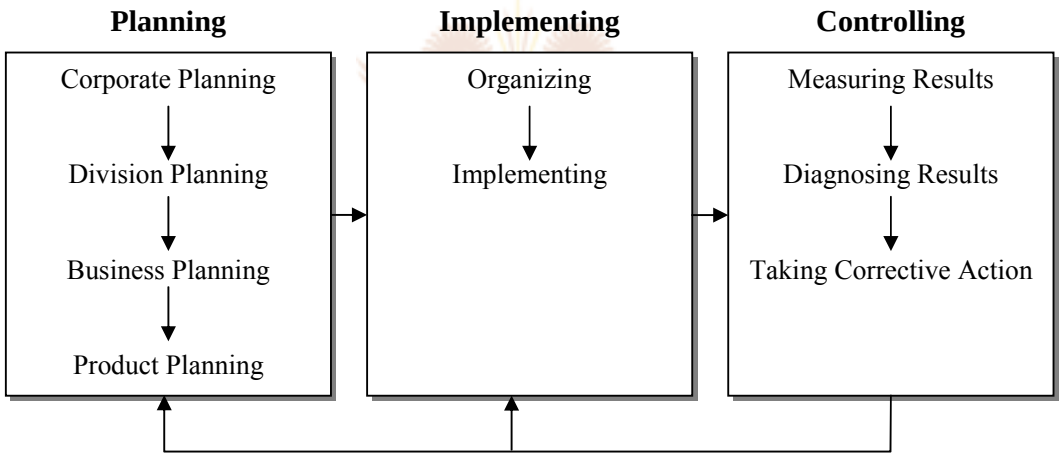
The overall approach is to develop a policy that can be accepted organization wide. The development of the organizational quality policy should not preclude the formulation of localized quality policies at departmental level. The sub-policies at departmental level should be ancillary to the quality policy. The formulation of sub-

policies should be encouraged as it leads to further involvement of staff at operations level.

6.4.2 Planning for the Software Quality Strategy

Kotler [46] outlines methods for achieving the goals of a new strategy. Although his methods are based on marketing principles, the basic ideas can be applied to another situation for which a strategy is devised. The basic process is based on a feedback loop, with the planning phase being the first major milestone in the process. Once the process has been planned, it needs to be implemented. Once the process has been implemented, the control and measurement section of the process is entered. At each of the planning and implementation stages, feedback is given, resulting in an improved process [46].

The basic process is shown in Figure 6.1:



Source: Kotler [46], Figure 4.1

Figure 6.1: Strategic Planning Process

This methodology should also be applied when implementing a quality strategy for a software engineering organization.

Quality planning may result in implementation over a longer period of time. It is the author's view that a total quality management (TQM) type of approach as described in Chapter 5 may also be appropriate for continuous quality improvements in software engineering organizations. To this end, Kinlaw [44] states that:

"Improving work processes is the only way to achieve substantive and long-term effects in the quality of services and products."

With this in mind, the following steps shown in Table 6.3 can be used as the outline for a quality improvement process.

Step	Detail	TQM Tools
1	Understand the opportunity or problem	*Models* *Brainstorming* *Nominal group technique* *Surveys* *Flow charts* *Cause- and effect diagrams* *Pareto charts* *Histograms* *Run charts* *Scatter diagrams* *Control charts*
2	Define the specific improvement target	*Flow charts* *Cause- and effect Diagrams* *Pareto harts* *Histograms* *Scatter diagrams*
3	Define strategies to reach the target	*Models* *Brainstorming* *Nominal group technique* *Cause- and effect diagrams*
4	Design the data links	*Surveys* *Observations* *Devices (mechanical, electronic, pneumatic, optical)* *Interviews*
5	Design the response process to use data from the data links	*Pareto charts* *Histograms* *Control charts*
6	Determine how the project will be managed	*Project management sheets* *Milestone charts*

Source: Kinlaw [44], Figure 3-1

Table 6.3: Project Steps and TQM Tools

Some of the tools that can be used for the attainment of each goal is also shown in Table 6.3. This is not meant to be the definitive guide to quality improvement, but should be used as a guideline. Use the SEI key performance areas as tools to drive the changes in quality processes in a software engineering environment. The goals are the targets to be achieved, and should be [78]:

- Determinable.
- Actionable.
- Measurable.
- Specific.

Use these measures to define the quality goals for the organization. Use the action plans to implement the quality framework. These plans should relate to the implementation issues and outcomes, and should account for details of each implementation. The project plan review checklist that is shown in Appendix F can be used to determine if the goals are indeed measurable, specific, and actionable.

6.4.3 Implementing the Software Quality Strategy

It is important that every person in an organization understands the strategy of the organization, so that they can plan for it. Ueckerman [87] cites mr. Shukri Cornelius, managing director of BTS Africa as saying:

> *"Employees often do not understand how an organization earns its income. As soon as they gain an insight therein, they can start to make well-informed choices."*

A major part of the planning process should be a campaign to increase the awareness of staff to issues such as quality and the effect that better quality can have on the organization. This should be made into a learning process, with the people actually performing quality tasks hands-on, as this increases the effectiveness of the learning process [87].

By participating in the quality initiatives of the organization, employees gain valuable experience of the profit drivers, and the effects of the quality strategy on the organization. Employees should start to see the short- and long-term benefits of the strategy [87].

Implementation, which is the actual doing part, is different from the final stage in the quality framework – controlling (measuring, diagnosing, taking corrective action). If

sufficient planning has not taken place before the implementation phase, then the work carried out could be wasted.

6.4.4 Controlling the Software Quality Strategy

Quality plans require continuous monitoring in order to determine their effectiveness. This means that for all levels of the quality plan there should be monitoring systems. It is a bottom-up process, where the development of the quality plan is a top-down process. One of the crucial elements of the quality plan is the generation and recording of quality data. The quality data can be generated using quality tools, to provide statistical as well as real measures of quality performance [40].

Monitoring and evaluating quality data should also be used to perform strategic and operational quality plan evaluations. These measures can check the viability and effectiveness of the designed quality plan against the measured outcomes. Using such monitoring systems should ensure that organizational issues of the quality plan can be addressed. Quality audits also need to occur at this stage, meaning independent and formal reviews of quality related performance issues of the quality plan.

All these processes will be subject to continuous improvement. This process will evaluate the content of the quality plan and the outcomes to date.

6.4.5 Strategic Quality Process Framework

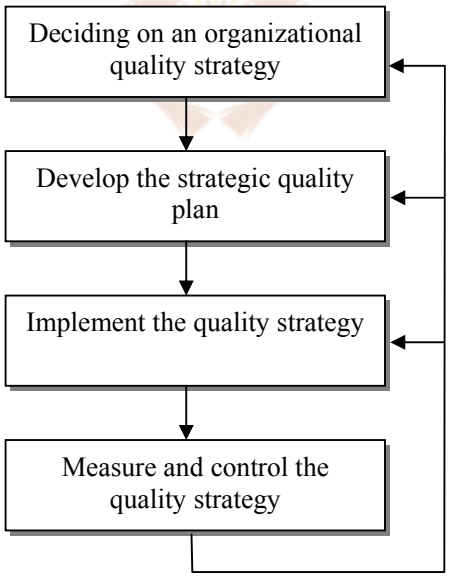
The preceding sections describe the individual steps that culminate in the creation of the author's strategic quality process framework. The basic outline of the process framework is shown in Figure 6.2. The first step in the strategic quality process framework is the establishment of an organizational policy regarding the quality of the software products and services of the software engineering organization. Deciding on a quality strategy is the most important step in the process, as all the preceding steps rely on the guidance provided by the strategy.

The second step in the strategic quality process framework is to plan for the quality strategy. The planning activities should be characterized by project management activities such as brainstorming, the use of Pareto charts, Ishikawa diagrams, models, PERT charts, etc.

The third step following the planning activities is the implementation stage of the strategic quality process framework. Activities such as requirements analysis, software design and development should take place during the third stage conforming to the policies and procedures that were decided upon during the planning stage.

The fourth and final step in the process framework measures and controls the activities that take place during the entire process. Statistical methods and software testing activities should aim to ensure that the software quality lives up to the promise of the quality strategy.

The measurement and control activities of the process framework should provide ongoing feedback to all other activities as shown in Figure 6.2. The main aim of the quality process framework is to create an organization strategy for the software engineering organization that will lead to lasting gains in the quality of the software engineering process, activities, services and products.

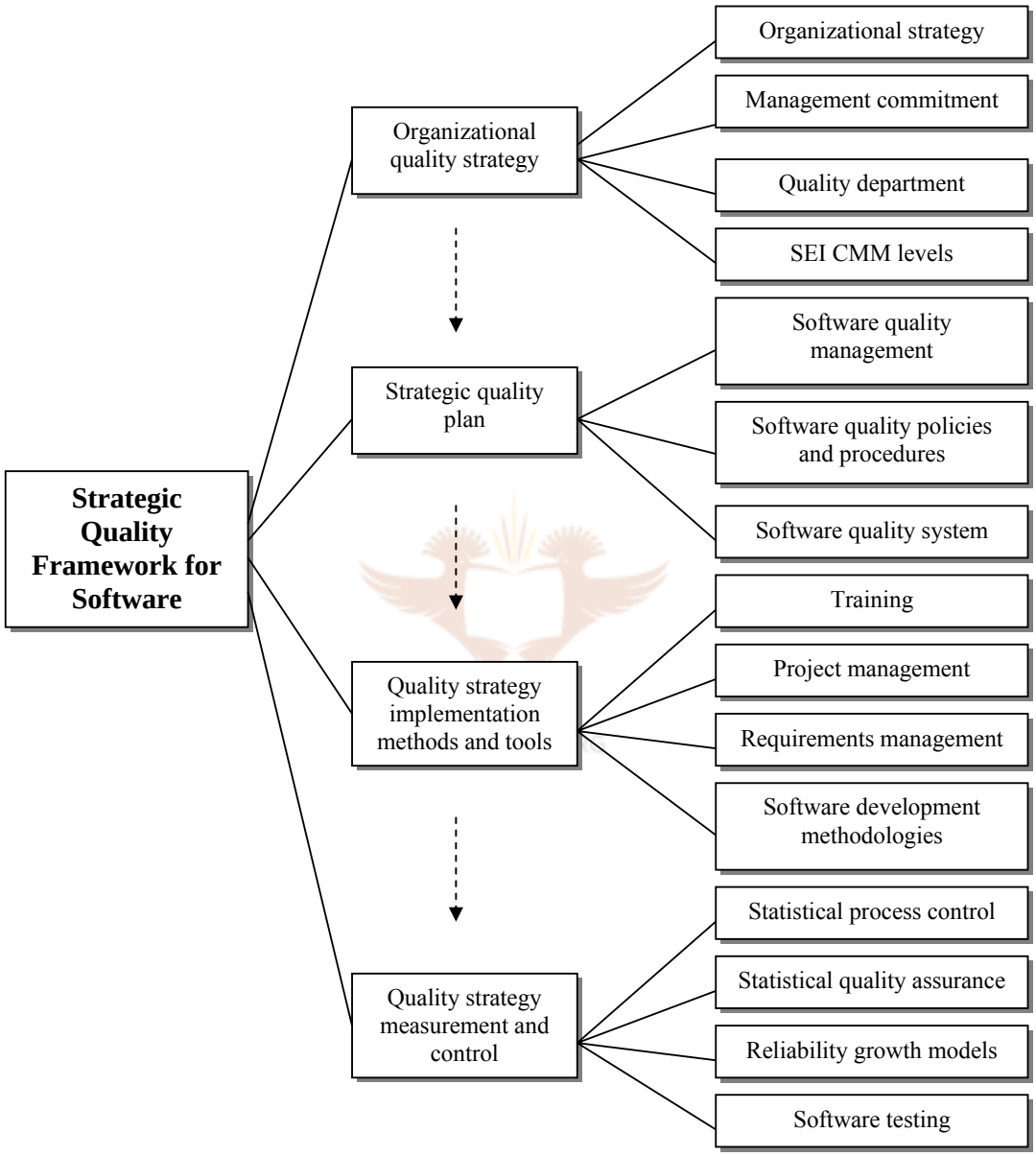


Source: Created by the Author

Figure 6.2: Strategic Quality Process Framework

A more detailed representation of the strategic quality framework is shown in Figure 6.3. The figure shown in Figure 6.3 is not in a process format, but illustrates some of the functions, processes and methods that would typically be associated with the intermediate steps of the quality process, i.e. measurement and control would make

use of statistical methods to provide measurement data to management. The framework should be seen as a dynamic framework, adapting to the needs of the organization implementing a quality strategy.



Source: Created by the Author

Figure 6.3: Basic Strategic Quality Framework

6.5 STEPS TO BETTER QUALITY

This section highlights some of the tools and techniques that the author has found to be helpful in the elimination of software problems and issues. These are only a few of the numerous tools and techniques available.

6.5.1 Software Requirements Definition

According to O'Connor [62] the way to draw up meaningful requirements for any type of system is to focus on applying the following rules to the requirement specification:

- The requirements must be clear and concise.
- The requirements must be realistic.
- The requirements and requirement metrics must be verifiable.

Smith states that [81] most failures occur due to inadequate or incorrect requirements specifications. This situation can be realized if proper tools are used by people trained to draft requirements specifications.

6.5.2 Design Guidelines

Hoschette [32] states that the design process varies from organization to organization and from product to product. Hence only some general guidelines are given to be able to design a better product. He has given a number of useful design guidelines that should be useful in most situations, and can also be used to eliminate mistakes and identify problem areas.

- Before starting a design, write down all the requirements placed on the design or product. Summarize the requirements in a table or matrix for easy review. The list should include requirements such as performance, interfaces, inputs/outputs, safety, operating conditions, special conditions, reliability, test requirements, customer use, rework and maintainability amongst others.
- Get agreement on the product requirements from your system engineer and/or supervisor.

- Generate a list or table of different approaches that you might use for the design.
- Show the various design approaches to people in the organization to find out what they think.
- Analyze the design.
- Create a mock product of the design if possible.
- Identify contingency plans in case something goes wrong.
- Keep track of product costs during design stages.
- Before committing to any design, develop a build and test plan. (See Appendix D for an outline of a software test plan.)
- Write down and document designs.

6.5.3 Software Quality Assurance Plan

The software quality assurance (SQA) plan should be developed during project planning and reviewed by all interested parties. Quality assurance activities performed by the software engineering team and the SQA group are governed by the plan [36]. An example of a software quality assurance plan is included in Appendix C.

The plan should establish inter alia:

- Evaluations to be performed.
- Audits and reviews to be performed.
- Standards that are applicable to the project.
- Procedures for error reporting and tracking.
- Documents to be produced by the SQA group.
- Feedback provided to the software project team.

6.5.4 Software Reviews

Software reviews are a filter for the software engineering process. Reviews are intended to detect errors that software engineers and developers overlook during the creation of software. Reviews are intended to supplement software testing activities [45].

The main goals of a software review are to:

- Point out needed improvement in the product of a single person or group.
- Fix those parts of the code where improvement is needed.

A formal technical review is a software quality assurance activity (software review) that is performed by the software engineers. A formal technical review is actually a group of activities like walkthroughs, inspection, and small group technical assessment of software which is conducted as formal meetings [45].

The aim of the formal technical review is to:

- Unearth errors in functions and logic of the software.
- Check that the software under review meets its requirements.
- Make sure that the software is developed according to predefined standards.
- Achieve software development in a uniform manner.

Formal technical reviews should generally conform to the following guidelines:

- The review meeting should not include more then 3 or 4 people.
- The duration of the review meeting should be agreed upon.
- All the people that are invited to attend the review meeting should prepare for the technical review before attending.
- A proper agenda should be set for the review meeting, and followed.

After the review meeting has been conducted, the attendees should decide to:

- Accept the product that is being developed without modifications.
- Reject the product that is being developed.
- Accept the product, with modifications.

The technical review report should be able to:

- Identify problem areas within the product.
- Serve as an action item checklist.

Some review guidelines to remember are:

- Review the product, not the producer.
- Set an agenda and follow it.
- Limit any debates.
- Take written notes.
- Limit the number of participants, and insist on advance preparation.
- Conduct meaningful training for all reviewers.
- Compare the new reviews with old ones.
- Find out problem areas but do not try to solve them. (It is to be done after the review meetings.)

6.5.5 Statistical Quality Assurance

Statistical quality assurance can be implemented by taking the following steps [20], [40], [62]:

- Collect and group defect information in software.
- Trace each defect to its underlying cause, such as violation of standards, improper communication, etc.

- Once the important causes have been identified, corrective measures should be taken.

6.6 CONCLUSION

Quality assurance is an essential activity for any business that produces products to be used by others. Software developers and consultants will agree that high-quality software is an important goal [45]. In order to produce software systems of a high quality, a strategic quality framework is essential to coordinate all the activities perceived to be necessary in the development of the software system. The basic strategic quality process framework consists out of four sequential steps as shown in Figure 6.2. The four steps include:

- The development of an organizational software quality strategy.
- Planning the process of improving the software quality in conjunction with the organizational quality strategy.
- Implementation of the software quality strategy in the software engineering organization.
- The measurement and control of the software quality strategy.

The first three steps (strategizing, planning and implementation) require constant feedback from the measurement and control step to ensure that the software quality strategy is kept aligned with the organizational objectives. Regular measurement will enable the organization to respond in a quick and efficient manner to any deficiencies in the first three steps of the software quality process framework.

The software quality process framework is a guideline that organizations should adapt to their specific requirements. Thus, the software quality framework lends itself well to adaptation and improvement efforts.

Chapter 7

Conclusion and Recommendations

"Don't find fault, find a remedy." – Henry Ford

7.1 INTRODUCTION

The quotation by Henry Ford stating that one should try and find solutions (a remedy) and not faults, defines the essence of research – the discovery of solutions to problems. Software engineering organizations face a challenge in the delivery of high quality software to customers, even though solutions do exist for the problem of software lacking quality attributes.

Activities such as software testing, software configuration management, release management and quality initiatives such as the SEI capability maturity model all address specific issues of software quality management [9], [22], [24], [25], [33], [36], [37], [40], [45], [55], [61], [62], [71], [81].

A software quality strategy is essential for obtaining management approval on the direction the organization is to take in their software quality endeavours. Quality needs to be a part of the organization's strategic objectives.

7.2 CONCLUDING SUMMARY

Whether a software engineering organization tries to improve its market share, a product or service through strategic planning, there are only a limited number of keys to success. The organization will need business-oriented people who acts in accordance with the organizational strategy to focus on clients and markets, who can work together as a team, who has the driving force and ability to bring plans to fruition, and constantly strives for profitability and excellence. Only a proper strategy can make this happen [87].

Introducing a quality strategy in the organization indicates commitment from the management team for the improvement of the organization's products and services. The quality strategy should be part of the bigger organizational strategy. Tools and techniques such as the balanced scorecards, satisfaction surveys, benchmarking, etc. can be used to define the quality strategy of the organization.

General quality concepts such as ISO 9000 and total quality management (TQM) were investigated in show that these do apply software engineering organizations and software engineering processes and methodologies. The software engineering process should contain a quality layer (as shown in Figure 4.1) in order to ensure customer satisfaction and on-time product delivery.

Software quality definitions were examined, leading to the conclusion that the quality of software is determined by the level of conformance to requirements, or expectations of the customer. Software quality models illustrate some views that authors such as McCall and Boehm used to compare quality. These models led the author to create his strategic quality framework for software engineering activities.

The model aims to align the organization strategy with the software engineering strategy as it relates to quality of the software systems. The strategic quality framework is based upon a feedback process to ensure that corrective measures can be taken to keep on improving all aspects of the software engineering process to provide software that is of a better quality.

Quality assurance is an essential activity for software engineering organizations that produce software products and systems. Software developers and consultants generally agree that high-quality software is an important goal [45]. In order to produce software systems of a high quality, a strategic quality framework as created by the author is essential to coordinate all the activities necessary in the development of the software system.

7.3 RECOMMENDATIONS

The strategic software quality framework is a guideline for the improvement of the quality of software engineering products and services. The framework should be adapted to suit the organization where it is to be applied, adding or removing quality

functions and processes that will not add value to the overall objectives of the organization. The framework features a measurement and control feedback loop, and the process of examining quality data should be performed on a regular basis to ensure that the framework is indeed improving the quality of products and services.

The topics of quality and strategy and the links between the two offer a wealth of research opportunities. The effect of organizational strategies on quality efforts should be investigated further, in an attempt to measure and define the benefits that are gained from adopting such practices. The software engineering industry is a fast paced environment, and moving towards an approach of improved quality is exciting and challenging.

The dissertation focused on the theoretical aspects of implementing a strategic quality plan for a software engineering organization. A practical implementation of a quality strategy for a software engineering organization should offer a wealth of opportunities for further research. Students of organizational change, strategic management, quality control, and software engineering processes could all participate in strategic quality studies. This may just lead to a change in the way organizations operate in the future.

APPENDICES

Appendix A: The Malcolm Baldridge National Quality Award

The United States Congress of 1987 established the Malcolm Baldridge National Quality Award to promote quality awareness, recognize the quality achievements of United States organizations, and to publicize successful quality strategies. Throughout the years the criteria for the award have evolved to represent a general performance and business excellence model. The award had the effect of shaping how people and organizations operate and work [40], [57], [58], [59], [60], [83].

The Malcolm Baldridge National Quality Award is in memory of the late Malcolm Baldridge, who was nominated by president Ronald Reagan of the USA to be Secretary of Commerce on December 1980. Baldridge took the initiative in creating a strategy for creating managerial excellence, which would lead to a long-term improvement of the economy, efficiency and effectiveness of government [56].

Around the world, countries followed suit, and have adopted their own versions of the Malcolm Baldridge National Quality Award. Japan's quality award is known as the Deming Prize, and also launched the Japan Quality Award [40], [57], [58], [59], [60], [83].

The benefits for an organization entering for a quality award is that it will guide the organization in its quality initiative. Organizations adopt the criteria of the award to win customers, to grow, and to remain competitive in a world market place.

The seven criteria for the Malcolm Baldridge National Quality Award are as follows [57], [58], [59]:

- Leadership.
- Strategic Planning.
- Customer and Market Focus.
- Information and Analysis.
- Human Resource Development and Management.

- Process Management.
- Business Results.

The award has helped organizations improve their performance by focusing on two major goals: delivering ever improving value to their customers; and improving the organization's overall performance. More then 2 million copies of the criteria have been distributed since 1988 [57], [58], [59], [60].

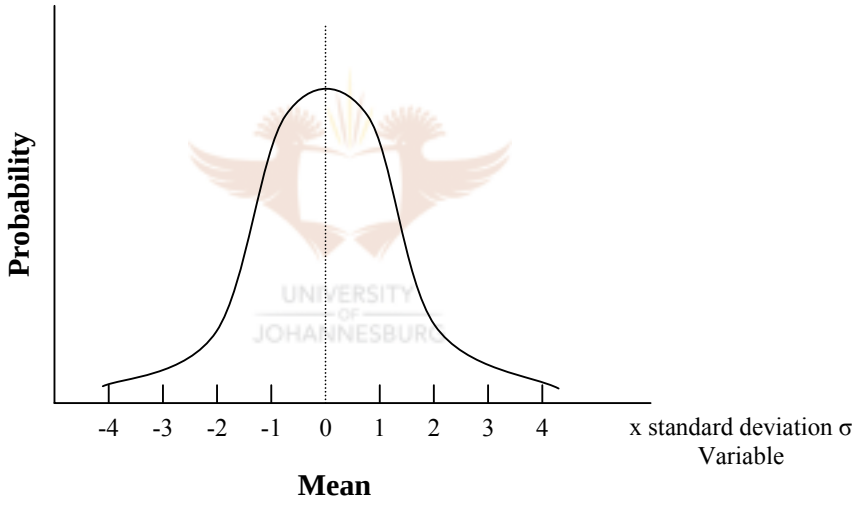
For organizations to be successful in today's environment, they not only need to read and understand the Malcolm Baldridge National Quality Award, but they also need to follow the guidelines.

Appendix B: Normal Distribution

The most widely used model of the nature of variation is the mathematical function known as the *normal or Gaussian distribution*. The s-normal partial differential function is given by [62], [73]:

$$f(x) = \frac{1}{\sigma (2\Pi)^{1/2}} \exp \left[-\frac{1}{2} \left(\frac{x - \mu}{\sigma} \right)^2 \right] \quad \ldots \text{B.1}$$

Where μ is the location parameter, equal to the mean. The mode and the median are coincident with the mean, as the p.d.f. is symmetrical. σ is the scale parameter, equal to the standard deviation (SD).



Source: O'Connor [62], Figure 2.7

Figure B.1: The s-normal (Gaussian) Distribution

A population that conforms to the s-normal distribution has variations which are symmetrically disposed about the mean (i.e. the skewness is zero). Since the tails of the s-normal distribution are symmetrical, a given spread includes equal values in the left-hand, and right-hand tails [62], [73].

For s-normally distributed variables, about 68% of the population fall between +/-1 SD. About 95% fall between +/-2 SD, and about 99.7% between +/-3 SD.

An important reason for the wide applicability of the s-normal distribution is the fact that, when a value is subject to many additive sources of variation, irrespective of how these variations are distributed, the resulting composite distribution can be shown to approach the s-normal distribution. This is known as the *central limit theorem*. It justifies the use of the s-normal distribution in many applications, including engineering, particularly in quality control. The s-normal distribution is a close fit to most quality control and some reliability observations, such as the sizes of machines parts and the lives of items subject to wearout failures, as well as to natural phenomena such as the heights of adults and strengths of materials [62], [73].

$\Phi(z)$ is the *standardized normal* c.d.f., i.e. $\mu = 0$ and $\sigma = 1$. z represents the number of SDs displacement from the mean. Any normal distribution can be evaluated from the standardized normal distribution by calculating the standardized normal variate z, where

$$z = \frac{x - \mu}{\sigma} \qquad \ldots \text{B.2}$$

And finding the appropriate value of $\Phi(z)$ [62], [73].

Appendix C: Sample Software Quality Assurance Plan

Appendix C is an outline for a Software Quality Assurance (SQA) Plan, adapted from the IEEE Standard for Software Quality Assurance Plans (ANSI/IEEE Standard 730-1984 and 983-1986).

This sample Software Quality Assurance Plan serves as a template for SQA activities for new software projects. Tailor the plan to your needs, removing explanatory comments as you go along. Where you decide to omit a section, you might keep the header, but insert a comment explaining why you omit the data.

Items that are intended to stay part of your document are in **bold** and normal; explanatory comments are in *italic* text and should be removed. Text in parenthesis is used where you should replace the text with your own wording.

C.1 PURPOSE OF PLAN

This section provides a simplified overview of the software quality assurance activities so that those approving, performing, or interacting with the Software Quality Assurance Plan can obtain a clear understanding of the SQA plan.

This section explains the purpose and content of the Software Quality Assurance Plan (i.e. the methods and procedures for improving the quality of the software). The Software Quality Assurance Plan documents the methods to be used for identifying software items, controlling and measuring quality, and recording the quality status of the software product. The plan should describe methods to be used for:

- *Project documents.*
- *Models and diagrams.*
- *Technical documents, such as specifications, test plans, etc.*
- *User documents, e.g. help files.*
- *Standards, conventions, practices, e.g. document standards, coding standards.*
- *Reviews and audits.*

C.2 REFERENCES

Section 2 will provide a complete list of documents, standards and manuals referenced elsewhere in the text of the Software Quality Assurance Plan. By definition, some of these documents may originate outside this plan.

C.3 MANAGEMENT

This section provides information describing the allocations of responsibilities and authorities for software quality assurance activities to organizations and individuals.

C.3.1 Organization

This section depicts the organizational context, both technical and managerial, within which the prescribed software quality assurance activities are to be implemented.

C.3.2 Tasks

This section describes the software quality assurance tasks that need to be executed by the organizational units and individuals.

C.3.3 Responsibilities

This section describes the allocation of software quality assurance activities to organizational units.

C.4 DOCUMENTATION

Section 3 will provide a complete list of documents that forms part of the Software Quality Assurance Plan.

C.4.1 Purpose

This section explains the purpose and content of the documents section.

C.4.2 Required Software Engineering Documents

This section lists all the required software engineering documents mentioned within the context of the Software Quality Assurance Plan.

C.4.3 Other Documents

This section lists all the required documents mentioned within the context of the Software Quality Assurance Plan that does not form part of any software engineering documents.

C.5 STANDARDS, PRACTICES AND CONVENTIONS

This section will provide a complete list of all the standards, practices and conventions that forms part of the Software Quality Assurance Plan.

C.5.1 Purpose

This section explains the purpose and content of the standards, practices and conventions section.

C.5.2 Conventions

This section lists all the conventions that are used within the Software Quality Assurance Plan.

C.6 REVIEWS AND AUDITS

This section will provide an overview of the reviews and audits that forms part of the Software Quality Assurance Plan.

C.6.1 Purpose

This section explains the purpose and content of the reviews and audits section.

C.6.2 Review Requirements

Identifies all the items needed for a review.

C.6.2.1 Software Requirements Review

This section lists all the functions and tasks required to perform a software requirements review.

C.6.2.2 Design Reviews

This section lists all the functions and tasks required to perform a design review.

C.6.2.3 Software Validation and Verification Reviews

This section lists all the functions and tasks required to perform software validation and verification reviews.

C.6.2.4 Function Audit

This section lists all the functions and tasks required to perform a function audit.

C.6.2.5 Physical Audit

This section lists all the functions and tasks required to perform a physical audit.

C.6.2.6 In-Process Audit

This section lists all the functions and tasks required to perform an in-process audit.

C.6.2.7 Management Reviews

This section lists all the functions and tasks required to perform management reviews.

C.7 TEST

Identifies all functions and tasks required to test the software system as specified in the scope of the SQA plan.

C.8 PROBLEM REPORTING AND CORRECTIVE ACTION

Identify, name and describe the problems experienced with the software system, and provide possible solutions and corrective actions for identified problems.

C.9 TOOLS, TECHNIQUES AND METHODOLOGIES

Identifies all the tools, techniques and methodologies required by the Software Quality Assurance Plan.

C.10 CODE CONTROL

Code control activities include requesting, evaluation, approval and disapproval of changes to the software code. Changes could include both error correction and enhancements.

C.11 MEDIA CONTROL

This section identifies the methods to used for controlling all changes to any media.

C.12 SUPPLIER CONTROL

For acquired and supplied software, the Software Quality Assurance Plan shall must define the activities to be performed to ensure that the suppliers provide products of a good enough quality.

C.13 RECORDS COLLECTION, MAINTENANCE AND RETENTION

This section must describe how quality records will be maintained and stored for the purposes as described in the Software Quality Assurance Plan.

C.14 TRAINING

This section must list all the training needs on both an organization and individual level.

C.15 RISK MANAGEMENT

This section must list any risks and assumptions made in the Software Quality Assurance Plan. Contingency plans should be specified for each risk identified.

Appendix D: Sample Software Test Plan

Appendix D is an outline for a Software Test Plan, adapted from the IEEE Standard for Software Test Plans (ANSI/IEEE Standard 829-1983). This standard describes a test plan as a document describing the scope, approach, resources, and schedule of intended testing activities. It identifies test items, the features to be tested, the testing tasks, who will do each task, and any risks requiring contingency planning.

This sample Software Test Plan serves as a template for testing activities for new software projects. Tailor the plan to your needs, removing explanatory comments as you go along. Where you decide to omit a section, you might keep the header, but insert a comment explaining why you omit the data.

Items that are intended to stay part of your document are in **bold** and normal; explanatory comments are in *italic* text and should be removed. Text in parenthesis is used where you should replace the text with your own wording.

D.1 TEST PLAN IDENTIFIER

A unique test plan identifier.

D.2 INTRODUCTION

Introduction information provides a simplified overview of the Software Test Plan activities, so that those people approving, performing, or interacting with the Software Test Plan can obtain a clear understanding of the test plan.

The introduction explains the purpose and content of the Software Test Plan (i.e. the methodical testing and recording of all test results). The Software Test Plan documents methods to be used for testing and recording test results and the test status. The plan should describe methods to be used for:

- *Summary of the items and features to be tested.*
- *Need for and history of each item (optional).*
- *References to related documents such as project authorization, project plan, QA plan, configuration management plan, relevant policies, relevant standards.*

- *References to lower level test plans.*

D.3 TEST ITEMS

Lists all test items and their versions, characteristics of their transmittal media, references to related documents such as requirements specification, design specification, user's guide, operations guide, installation guide. List all the references to bug reports related to test items. List all items that are specifically not going to be tested (optional).

D.4 FEATURES TO BE TESTED

List all software features and combinations of features to be tested, include references to test-design specifications associated with each feature and combination of features.

D.5 FEATURES NOT TO BE TESTED

List all features and significant combinations of features which will not be tested, include the reasons why those features will not be tested.

D.6 APPROACH

This section describes the overall approach to testing. For each major group of features, or combinations of features, specify the approach, major activities, techniques, and tools, which are to be used to test the groups. Specify a minimum degree of comprehensiveness required, and identify which techniques will be used to judge comprehensiveness. Specify any additional completion criteria. Specify techniques, which are to be used to trace requirements. Identify significant constraints on testing, such as test-item availability, testing-resource availability, and deadline.

D.7 ITEM PASS/FAIL CRITERIA

Specify the criteria to be used to suspend the testing activity.

D.8 SUSPENSION CRITERIA AND RESUMPTION REQUIREMENTS

Specify criteria to be used to suspend the testing activity, and specify testing activities, which must be redone when testing is resumed.

D.9 TEST DELIVERABLES

Identify the deliverable documents: test plan, test design specifications, test case specifications, test procedure specifications, test item transmittal reports, test logs, test incident reports, test summary reports. Identify test input and output data, as well as the test tools (optional).

D.10 TESTING TASKS

Identify tasks necessary to prepare for and perform testing, including all the task interdependencies and any special skills required.

D.11 ENVIRONMENTAL NEEDS

Specify necessary and desired properties of the test environment: physical characteristics of the facilities including hardware, communications and system software, the mode of usage (i.e. stand-alone), and any other software or supplies needed. Specify the level of security, special test tools, and any other testing needs. Identify the source for all needs which are not currently available.

D.12 RESPONSIBILITIES

Identify groups responsible for managing, designing, preparing, executing, witnessing, checking and resolving, providing the test items identified in the Test Items section, and groups responsible for providing the environmental needs identified in the Environmental Needs section.

D.13 STAFFING AND TRAINING NEEDS

Specify staffing needs by skill level, and then identify training options for providing the necessary skills.

D.14 SCHEDULE

Specify test milestones, item transmittal events. Estimate the time required to perform each testing task. Schedule all the testing tasks and set the milestones, then for each testing resource, specify its period of use.

D.15 RISKS AND CONTINGENCIES

Identify the high-risk assumptions of the test plan, and specify contingency plans for each.

D.16 APPROVALS

Specify the names and titles of all the people who must approve the plan. Provide space for signatures and dates.

Appendix E: Sample Code Review Checklist

Appendix E is an outline for a Code Review Checklist taken from Wiegers [93].

This sample Code Review Checklist serves as a template for code review activities for new and existing software projects. Tailor the checklist to your needs, removing explanatory comments as you go along. Where you decide to omit a section, you might keep the header, but insert a comment explaining why you omit the data.

Items that are intended to stay part of your document are in **bold** and normal; explanatory comments are in *italic* text and should be removed. Text in parenthesis is used where you should replace the text with your own wording.

Code Review		
Reviewer: *[Reviewer name]*		
Meeting Date: ***[dd/MMM/yyyy]***		
Notes: *[Any additional notes]*		
No.	**Structure**	✗/✓
S.1	Does the code completely and correctly implement the design?	
S.2	Does the code conform to any pertinent coding standard?	
S.3	Is the code well-structured, consistent in style, and consistently formatted?	
S.4	Are there any uncalled or unneeded procedures or any unreachable code?	
S.5	Are there any leftover stubs or test routines in the code?	
S.6	Can any code be replaced by calls to external reusable components or library functions?	
S.7	Are there any blocks of code that could be condensed into a single procedure?	
S.8	Is storage use efficient?	
S.9	Are symbolics used rather than "magic number" constants or string constants?	
S.10	Are any modules excessively complex and should be restructured or split into multiple routines?	
No.	**Documentation**	✗/✓
D.1	Is the code clearly and adequately documented with an easy-to-maintain commenting style?	
D.2	Are all comments consistent with the code?	
No.	**Variables**	✗/✓
V.1	Are all variables properly defined with meaningful, consistent, clear names?	
V.2	Do all assigned variables have proper type consistency or casting?	
V.3	Are there any redundant or unused variables?	
No.	**Arithmetic Operations**	✗/✓
A.1	Does the code avoid comparing floating-point numbers for equality?	
A.2	Does the code systematically prevent rounding errors?	
A.3	Does the code avoid additions and subtractions on numbers with greatly different magnitudes?	
A.4	Are divisors tested for zero or noise?	
No.	**Loops and Branches**	✗/✓
L.1	Are all the loops, branches, and logic constructs complete, correct, and properly	

	nested?	
L.2	Are the most common cases tested first in the IF - - ELSEIF chains?	
L.3	Are all cases covered in an IF - - ELSEIF or CASE block, including ELSE or DEFAULT clauses?	
L.4	Does every CASE statement have a default?	
L.5	Are loop termination conditions obvious and invariably achievable?	
L.6	Are indexes or subscripts properly initialized, just prior to the loop?	
L.7	Can any statements that are enclosed within loops be placed outside the loops?	
L.8	Does the code in the loop avoid manipulating the index variable or using it upon exit from the loop?	
No.	**Defensive Programming**	**✗/✓**
DP.1	Are indexes, pointers, and subscripts tested against array, record, or file bounds?	
DP.2	Are imported data and input arguments tested for validity and completeness?	
DP.3	Are all output variables assigned?	
DP.4	Are the correct data operated on in each statement?	
DP.5	Is every memory allocation de-allocated?	
DP.6	Are timeouts or error traps used for external device access?	
DP.7	Are files checked for existence before attempting to access them?	
DP.8	Are all files and devices left in the correct state upon program termination?	

Source: Wiegers [93]

Table E.1: Sample Code Review

Appendix F: Sample Project Plan Review Checklist

Appendix F is an outline for a Project Plan Review Checklist taken from Wiegers [93].

This sample Project Plan Review Checklist serves as a template for project plan review activities for new and existing software projects. Tailor the checklist to your needs, removing explanatory comments as you go along. Where you decide to omit a section, you might keep the header, but insert a comment explaining why you omit the data.

Items that are intended to stay part of your document are in **bold** and normal; explanatory comments are in *italic* text and should be removed. Text in parenthesis is used where you should replace the text with your own wording.

Project Plan Review		
Reviewer: *[Reviewer name]*		
Meeting Date: *[dd/MMM/yyyy]*		
Notes: *[Any additional notes]*		
No.	**Clarity**	✗/✓
CL.1	Is the project plan appropriately decomposed, being neither too detailed, nor too general?	
CL.2	Is the meaning of each plan component clear and unambiguous?	
CL.3	Is the relationship between plan components clear?	
CL.4	Is the document organized in a logical and clear way and consistent with and applicable template or standard?	
No.	**Completeness**	✗/✓
CM.1	Are all estimates documented, along with the method used for deriving them?	
CM.2	Does the project plan include all the items required by the applicable standard, procedure or template?	
CM.3	Are relevant details missing from any plan components?	
CM.4	Are irrelevant details present in any plan components?	
CM.5	Are some plan components unnecessary?	
CM.6	Are all related or reference documents listed?	
No.	**Consistency**	✗/✓
CN.1	Are related components within the plan consistent with each other?	
CN.2	Is the content of the plan consistent with the contents of all related documents?	
CN.3	Is the content consistent with the project's scope and objectives?	
No.	**Correctness**	✗/✓
CR.1	Are any tasks missing from the work breakdown structure?	
CR.2	Are all estimates realistically achievable?	
CR.3	Are all assumptions that went into project planning stated and accurate?	

Source: Wiegers [93]

Table F.1: Project Plan Review

Appendix G: Sample Requirements Specification Review Checklist

Appendix G is an outline for a Requirements Specification Review Checklist taken from Wiegers [93].

This sample Requirements Specification Review Checklist serves as a template for requirements specification review activities for new and existing software projects. Tailor the checklist to your needs, removing explanatory comments as you go along. Where you decide to omit a section, you might keep the header, but insert a comment explaining why you omit the data.

Items that are intended to stay part of your document are in **bold** and normal; explanatory comments are in *italic* text and should be removed. Text in parenthesis is used where you should replace the text with your own wording.

Requirements Specification Review		
Reviewer: *[Reviewer name]*		
Meeting Date: *[dd/MMM/yyyy]*		
Notes: *[Any additional notes]*		
No.	**Organization and Completeness**	✗/✓
OC.1	Are all internal cross-references to other requirements correct?	
OC.2	Are all requirements written at a consistent and appropriate level of detail?	
OC.3	Do the requirements provide an adequate basis for design?	
OC.4	Is the implementation priority of each requirement included?	
OC.5	Are all external hardware, software, and communication interfaces defined?	
OC.6	Have algorithms intrinsic to the functional requirements been defined?	
OC.7	Does the specification include all known customer, or system needs?	
OC.8	Is the expected behavior documented for all anticipated error conditions?	
No.	**Correctness**	✗/✓
C.1	Do any requirements conflict with, or duplicate other requirements?	
C.2	Is each requirement written in clear, concise, unambiguous language?	
C.3	Is each requirement verifiable by testing, demonstration, review, or analysis?	
C.4	Is each requirement in scope for the project?	
C.5	Is each requirement free from content and grammatical errors?	
C.6	Is any necessary information missing a requirement? If so, is it identified as TBD?	
C.7	Can all of the requirements be implemented without known constraints?	
C.8	Are any specified error messages unique and meaningful?	
No.	**Quality Attributes**	✗/✓
QA.1	Are all performance objectives properly specified?	
QA.2	Are all security and safety considerations properly specified?	
QA.3	Are other pertinent quality attribute goals explicitly documented and quantified, with the acceptable tradeoffs specified?	

No.	Traceability	✗/✓
T.1	Is each requirement uniquely and correctly identified?	
T.2	Is each software functional requirement traceable to a higher-level requirement (e.g. system requirement, use case, etc)?	
No.	**Special Issues**	**✗/✓**
SI.1	Are all requirements actually requirements, not design or implementation solutions?	
SI.2	Are all time-critical functions identified, and timing criteria specified for them?	
SI.3	Have internationalization issues been adequately addressed?	

Source: Wiegers [93]

Table G.1: Requirements Specification Review

Appendix H: Sample Software Configuration Management Plan

Appendix H is an outline for a Software Configuration Management Plan (SCMP). The document is an outline for a Software Configuration Management Plan adapted from the IEEE Standard for Software Configuration Management Plans (IEEE Std 828-1990) and the IEEE Guide to Software Configuration Management (IEEE Std 1042-1987).

This sample Software Configuration Management Plan serves as a template for change control activities for new and existing software projects. Tailor the checklist to your needs, removing explanatory comments as you go along. Where you decide to omit a section, you might keep the header, but insert a comment explaining why you omit the data.

Items that are intended to stay part of your document are in **bold** and normal; explanatory comments are in *italic* text and should be removed. Text in parenthesis is used where you should replace the text with your own wording.

H.1 INTRODUCTION

IEEE Standard 828 (Standard for Software Configuration Management Plans) establishes the minimum required content of the Software Configuration Management (SCM) Plan. These standards are supplemented by IEEE Standard 1042 (Guide to Software Configuration Management) that provides approaches to good software configuration management planning.

The Software Configuration Management Plan (SCMP) documents what software configuration management activities are to be done, how they are to be done, who is responsible for doing specific activities, when they are to happen, and what resources are required.

Introduction information provides a simplified overview of the configuration management activities so that those approving, performing, or interacting with the Software Configuration Management Plan can obtain a clear understanding of the SCM Plan.

The introduction explains the purpose and content of the Configuration Management Plan (i.e. the methodical storage and recording of all software components and deliverables during development). The Software Configuration Management Plan documents methods to be used for the identification of software items, control and implementation of change, and recording and reporting change implementation status. The plan should describe methods to be used for:

- *Identification of software configuration items.*
- *Control and implementation of change.*
- *Recording and reporting change and problem report implementation status.*
- *Conducting configuration audits.*
- *Review and approval cycles, as well as approval authority.*
- *Identification of personnel responsible for configuration management.*

H.2 REFERENCE DOCUMENTS, DEFINITIONS AND ACRONYMS

Section 2 will provide a complete list of documents referenced elsewhere in the text of the SCM Plan. By definition, these documents originate outside the project. Also included in this section is a glossary of project specific terms and their definitions and a list of project-specific abbreviations and acronyms and their meaning.

H.2.1 Reference Documents

Lists all project-related documents referred to in the text of the SCM Plan.

H.2.2 Glossary of Terms

Lists and defines all terms that establish meaning within the context of the SCM Plan.

H.2.3 Abbreviations and Acronyms

Lists all alphabetical contractions and their definitions that appear within the text of the SCM Plan.

H.3 MANAGEMENT

This section provides information describing the allocations of responsibilities and authorities for software configuration management activities to organizations and individuals within the project structure.

H.3.1 Organization

This section depicts the organizational context, both technical and managerial, within which the prescribed software configuration management activities are to be implemented.

H.3.2 Responsibilities

Describes the allocation of software configuration management activities to organizational units.

H.3.3 Policies, Directives and Procedures

Any external constraints, or requirements, placed on the SCM Plan by other policies, directives, or procedures must be identified here. A detailed impact analysis should accompany the identification of external constraints.

H.4 ACTIVITIES

Identifies all functions and tasks required to manage the configuration of the software system as specified in the scope of the SCM Plan. Both technical and managerial activities must be identified.

H.4.1 Configuration Identification

Identify, name, and describe the documented physical and functional characteristics of the code, specification, design, and data elements to be controlled for the project. The Plan must identify the items to be maintained in configuration management control.

H.4.1.1 Identifying Configuration Items

Record the items to be controlled and their definitions as they evolve or are selected.

H.4.1.2 Naming Configuration Items

Specify the identification system for assigning unique identifiers to each item to be controlled.

H.4.1.3 Acquiring Configuration Items

Identify the controlled software libraries and describe how the code, documentation, and data of the identified baselines are to be physically placed under control in the appropriate library.

H.4.2 Configuration Control

Configuration control activities request, evaluate, approve or disapprove, and implement changes to the software configuration items. Changes include both error correction and enhancement. This section shall identify the records to be used for tracking and documenting the sequence of steps for each change.

H.4.2.1 Requesting Changes

This section provides information describing the process for requesting changes to software.

H.4.2.2 Evaluating Changes

This section provides information describing the process for evaluating changes to software.

H.4.2.3 Approving or Disapproving Changes

This section provides information describing the process for approving or disapproving changes to software.

H.4.2.4 Implementing Changes

This section provides information describing the process to be followed for implementing changes to software.

H.4.3 Configuration Status Accounting

Record and report the status of configuration items. The following minimum data elements should be tracked and reported for each configuration management item:

- *Approved version.*
- *Status of requested changes.*
- *Implementation status of approved changes.*

H.4.4 Configuration Audits and Review

Configuration audits determine the extent to which the actual configuration management items reflect the required physical and functional characteristics. Configuration reviews may also be designed as a management tool to ensure that a software configuration management baseline is established.

H.4.5 Interface Control

Coordinate changes to the project's configuration management items with changes to interfacing items outside the scope of the SCM Plan.

H.4.6 Subcontractor/Vendor Control

For acquired software, the Software Configuration Management Plan shall describe how the vendor software will be received, tested and placed under software configuration management control.

For both subcontracted and acquired software, the SCM Plan must define the activities to be performed to incorporate the externally developed items into project configuration management and to coordinate changes to these items.

H.5 RESOURCES

Establishes the sequence and coordination for all the software configuration management activities and all the events affecting the Plan's implementation.

H.5.1 Schedules

Schedule information shall be expressed as absolute dates, as dates relative to other project activities, as project milestones, or as a simple sequence of events. Graphic representation can be particularly appropriate for conveying this information.

H.5.2 Resources

Identifies the software tools, techniques, equipment, personnel, and training necessary for the implementation of software configuration management activities.

H.6 PLAN MAINTENANCE

Identifies and describes the activities and responsibilities necessary to ensure continued software configuration management planning during the life cycle of the project. This section of the SCM Plan should state the following:

- *Who is responsible for monitoring the SCM Plan.*
- *How frequently updates are to be applied.*
- *How changes to the SCM Plan are to be evaluated and approved.*
- *How changes to the SCM Plan are to be made and communicated.*

H.7 PLAN APPROVALS

Specify all the names and titles of the people who must approve the plan. Provide space for signatures and dates.

Appendix I: Sample Weekly Report

Appendix I is an outline for a Weekly Report taken from Wiegers [93].

This sample Weekly Report serves as a template for general quality activities for new and existing software projects. Tailor the checklist to your needs, removing explanatory comments as you go along. Where you decide to omit a section, you might keep the header, but insert a comment explaining why you omit the data.

Items that are intended to stay part of your document are in **bold** and normal; explanatory comments are in *italic* text and should be removed. Text in parenthesis is used where you should replace the text with your own wording.

Weekly Report			
Prepared by: *[Team name]*			
Group: *[Group name]*			
Meeting Date: *[dd/MMM/yyyy]*			
Notes: *[Any additional notes]*			
Issues			
System	**Issue**	**Status**	**Possible Solution**
[System name]	*[Issues]*	*[Status]*	*[Possible solution]*
Accomplishments			
System	**Accomplishments**		
[System name]	*[Bulleted list of things accomplished for the week]*		
Next Week Priorities			
System	**Task**		
[System name]	*[Bulleted list of tasks]*		

Source: Wiegers [93]

Table I.1: Weekly Report

BIBLIOGRAPHY

Bibliography

[1] Alred, Gerald J., Charles T. Brusaw, and Walter E. Oliu, *Handbook of Technical Writing*, 7th ed., St. Martin's Press, 2003.

[2] Alswang, J. and A van Rensburg, *English Usage Dictionary*, 3rd ed., EDUCUM, 1988.

[3] Australian Technology Network of Universities, *Planning and Quality*, Australian Technology Network of Universities, <http://www.atn.edu.au/wgroups/quality.htm>. (4 August 2004)

[4] Beeld, *Druk lei tot Brouwerk in Programme*, Beeld, 3 May 2004.

[5] Besser, J., *Riding the Marketing Information Wave*, Harvard Business Review, September/October 1993, pp. 150 – 160.

[6] Blanchard, Benjamin S., *Logistics Engineering and Management*, 5th ed., Prentice Hall, 1998.

[7] Bologna, Sandro and Giacomo Bianco, *Achieving Quality in Software: Proceedings of the 3rd International Conference on Achieving Quality in Software, 1996*, Chapman & Hall, 1996.

[8] Boxwell, R., *Benchmarking for Competitive Advantage*, McGraw-Hill, 1991.

[9] Brinkworth, John W.O., *Software Quality Management: A Pro-Active Approach*, Prentice Hall, 1992.

[10] Brooks, F., *The Mythical Man-Month*, Addison-Wesley, 1975.

[11] Cooper, Donald R. and Pamela S. Schindler, *Business Research Methods*, 8th ed., McGraw-Hill, 2003.

[12] Cooper, R. and R. Kaplan, *Measure Costs Right: Make the Right Decisions*, Harvard Business Review, September/October 1988.

[13] Covey, Steven R., *The 7 Habits of Highly Successful People*, Pan Books, 1996.

[14] Crosby, Philip, *Quality is Free*, McGraw-Hill, 1979.

[15] Cummings, Stephen and David Wilson, *Images of Strategy*, Blackwell Publishing, 2003.

[16] Cummings, Stephen, *ReCreating Strategy*, SAGE Publications, 2002.

[17] Eisenhardt, K., *Speed and Strategic Choices*, Planning Review, September/October 1992.

[18] Feigenbaum, E.A. and P. McCorduck, *The Fifth Generation*, Addison-Wesley, 1983.

[19] Foster, R., *Innovation: The Attackers Advantage*, Simon & Schuster, 1982.

[20] Freeman-Bell, Gail and James Balkwill, *Management in Engineering*, 2nd ed., Pearson Education, 1996.

[21] Getudis, A., *A Quality Assurance Manual (MIL-STD-45208A)*, ALCO Machine Corporation, 1998, <http://www.alcomachine.com/images/qc-manual.pdf>. (23 May 2003)

[22] Gillies, Alan C., *Software Quality: Theory and Management*, Chapman & Hall, 1992.

[23] Gravett, Steve, *The Right Way to Write Reports*, Right Way Publishers, 1993.

[24] Greenhill, G., *Five Steps for Developing a Software Deployment and Management Strategy*, Technical Support, September 2001.

[25] Hall, G., J. Rosenthal and J. Wade, *How to make Reengineering Really Work*, Harvard Business Review, November/December 1993.

[26] Hamel, Gary and C.K. Pralahad, *Competing for the Future*, Harvard Business School Press, 1996.

[27] Hamel, Gary and C.K. Pralahad, *The Core Competence of the Corporation*, Harvard Business Review, May 1990.

[28] Hammer, M. and J. Champy, *Re-engineering the Corporation: A Manifesto for Business Revolution*, HarperCollins, 2003.

[29] Hart, C., *The Power of Unconditional Service Guarantees*, Harvard Business Review 66(4) pp. 54-62, 1988.

[30] Heller, Robert, *Jack Welch*, Dorling Kindersley, 2001.

[31] Hignett, K.C., *Practical Safety and Reliability Assessment*, 1st ed., Chapman & Hall, 1996.

[32] Hoschette, John A., *Career Advancement and Survival for Engineers*, Wiley & Sons, 1994.

[33] Hower, R., *Software QA and Testing Resource Center*, QA/Test Resource Center, 2003, <http://www.softwareqatest.com/qatfaq1.html>. (23 May 2003)

[34] Hutchins, D.C., *Quality Circles Handbook*, Nichols, 1985.

[35] IEEE Standards Collection: Software Engineering, *IEEE Standard 610.12-1990*, IEEE, 1993.

[36] Ince, Darrel, *Software Quality Assurance*, McGraw-Hill, 1995.

[37] Ince, Darrel, *Software Quality and Reliability: Tools and Methods*, UNICOM, 1991.

[38] James, Paul, *Total Quality Management: An Introductory Text*, Prentice-Hall, 1996.

[39] Juran, Joseph M., *Juran on Planning for Quality*, The Free Press, 1988.

[40] Kan, Stephen H., *Metrics and Models in Software Quality Engineering*, Addison-Wesley, 1995.

[41] Kaplan, Robert S. and David P. Norton, *The Balanced Scorecard Measures that Drive Performance*, Harvard Business Review 70(1) pp. 71-79, 1992.

[42] Keiser, B., *Practical Competitor Intelligence*, Planning Review 8(1987) pp. 14-18, 1987.

[43] Kenny, D. and J. Quelch, *Extend Profits not Product Lines*, Harvard Business Review 72(5) pp.153-160, September/October 1994.

[44] Kinlaw, Dennis C., *Continuous Improvement and Measurement for Total Quality: A Team-Based Approach*, Pfeiffer and Company, 1992.

[45] Kit, Edward, *Software Testing in the Real World*, McGraw-Hill, 1998.

[46] Kotler, Philip, *Marketing Management*, 11th ed., Prentice Hall, 2003.

[47] Ladeira, Andre, *Cost Estimation Methods for Software Engineering*, M.Eng dissertation, Rand Afrikaans University, January 2002.

[48] Limerick, C., *Managing the New Organization: Blueprints for Networks and Strategic Alliances*, Jossey-Bass, 1993.

[49] Lindeman, Werner Philip, *A Method for Benchmarking in an Engineering Business Environment*, M.Eng dissertation, Rand Afrikaans University, 1997.

[50] Liswood, L., *Serving them right: Innovative and Powerful Customer Retention Strategies*, Harper Business, 1990.

[51] Loveday, G., *Electronic Testing and Fault Diagnosis*, 2nd ed., Longman Scientific, 1989.

[52] Main, J., *How to Steal the Best Ideas Around*, Fortune, 1992.

[53] Meredith, Jack L. and Samuel J. Mantel, *Project Management – A Managerial Approach*, 5th ed., Wiley, 2003.

[54] Naur, P. and B. Randall, *Software Engineering: A Report on a Conference Sponsored by the NATO Science Committee*, NATO, 1969.

[55] Newport Group, *Continuous Quality Assurance in Dynamic Application Environments: Best Practices*, Newport Group, <http://www.newport-group-inc.com>. (3 March 2004)

[56] NIST, *Biography of Malcolm Baldridge*, NIST, <http://www.nist.gov/Biography.htm>. (4 March 2003)

[57] NIST, *Frequently Asked Questions*, NIST, <http://www.nist.gov/public_affairs/factsheet/baldfacts.htm>. (4 March 2003)

[58] NIST, *Malcolm Baldridge National Quality Award*, NIST, <http://www.nist.gov/public_affairs/factsheet/mbnqagoal.htm>. (4 March 2003)

[59] NIST, *Malcolm Baldridge National Quality Award*, NIST, <http://www.nist.gov/public_affairs/factsheet/mbnqa.htm>. (4 March 2003)

[60] NIST, *The Malcolm Baldridge National Quality Improvement Act*, NIST, <http://www.nist.gov/Improvement_Act.htm>. (4 March 2003)

[61] Norris, Mark and Peter Rigby, *Software Engineering Explained*, John Wiley & Sons, 1992.

[62] O'Connor, Patrick D.T., *Practical Reliability Engineering*, 4th ed., John Wiley & Sons, 2002.

[63] Ostroff, F. and D. Smith, *The Horizontal Organization: Redesigning the Corporation*, Oxford University Press, 1992.

[64] Peters, Tom, *The Circle of Innovation*, Alfred A. Knopf Inc, 1997.

[65] Petrov, Daniel B., *Cost-Effectiveness Investment Analysis for Property Development Projects*, M.Eng dissertation, Rand Afrikaans University, December 2000.

[66] Piquito, Nicholas P., *Financial Product Development: A Strategically Competitive System Engineering Approach to Innovative Risk Based Financial Engineering*, D.Eng dissertation, Rand Afrikaans University, October 1999.

[67] Piquito, Nicholas P., *Technologically Driven Economic Development*, M.Eng dissertation, Rand Afrikaans University, October 1997.

[68] Porter, Leslie J. and Steve Tanner, *Assessing Business Excellence: A Guide to Self-Assessment*, Butterworth-Heinemann, 1996.

[69] Porter, M., *Competitive Advantage: Creating and Sustaining Superior Performance*, Free Press, 1998.

[70] Porter, M., *From Competitive Advantage to Corporate Strategy*, Harvard Business Review, March 1987.

[71] Pressman, Roger S., *Software Engineering: A Practitioner's Approach*, 4th ed., McGraw-Hill, 1997.

[72] Pyzdek, T., *The Six Sigma Handbook*, McGraw-Hill, 2003.

[73] Ramakumar, Ramachandra, *Engineering Reliability*, Prentice Hall, 1993.

[74] Ramaswamy, Rohit, *Design and Management of Service Processes: Keeping Customers for Life*, AT&T, 1996.

[75] Raubenheimer, Pieter J., *A Case Study in Industrial Risk Management*, Rand Afrikaans University, July 2000.

[76] Robbins, Anthony, *Unlimited Power*, Pocket Books, 2001.

[77] Rooney, Alexander C., *Reliability Growth Management of Complex Electromechanical Systems*, M.Eng dissertation, Rand Afrikaans University, June 2000.

[78] Sanchez, Ron, and Aimé Heene, *The New Strategic Management: Organization, Competition and Competence*, John Wiley & Sons, 2004.

[79] Schlesinger, L. and J. Heskett, *The Service-Driven Service Company*, Harvard Business Review, 1991.

[80] Senyurek, Edip, *A Case Study of Customer Needs Identification in Computer Technology*, M.Eng dissertation, Rand Afrikaans University, July 2001.

[81] Smith, David J., *Achieving Quality Software*, 3rd ed., Chapman & Hall, 1995.

[82] Stern, J., *EVA Share Options that Maximize Value*, Corporate Finance no. 105 (Aug) 1993b pp. 31-32, 1993.

[83] Stevenson, William J., *Operations Management*, 7th ed., McGraw-Hill, 2002.

[84] Taguchi, G. and D. Clausing, *Robust Quality*, Harvard Business Review, January/February 1990.

[85] Thibault, Sara, *The National Malcolm Baldridge Quality Award*, Emporia University, <http://www.emporia.edu/ibed/jour/jour14om/sarat.htm>. (4 March 2003)

[86] Thuesen, Gerald J., and W.J. Fabrycky, *Engineering Economy*, 9th ed., Prentice Hall, 2001.

[87] Ueckermann, Helen, *Bemark Strategie aan elke Werker*, Rapport, 29 August 2004.

[88] Ulrich, Karl T., and Steven D. Eppinger, *Product Design and Development*, 2nd ed., McGraw-Hill, 2000.

[89] United States Government, *Malcolm Baldridge National Quality Act of 1987*, One Hundredth Congress of the United States Government, <http://www.nist.gov/Act.htm>. (4 March 2003)

[90] Wack, P., *Scenario's: The Uncharted Waters Ahead*, Harvard Business Review 63(5) pg. 87, September/October 1985.

[91] Wessels, Arie, *Reliability Engineering Course Notes*, M.Eng (Engineering Management), RAU, 2003.

[92] Wheelen, Thomas L. and J. David Hunger, *Cases in Strategic Management and Business Policy*, Addison-Wesley, 1987.

[93] Wiegers, Karl, E., *Process Impact: Inspection Checklists*, Process Impact Website, <http://www.processimpact.com/goodies.shtml>. (4 May 2004)

[94] Young, Allan I., *Complete Plant Operations Handbook: A Guide to Cost Reduction, Quality Control and On-Time Delivery*, Prentice-Hall, 1990.

[95] ISO (1991), *International Standard ISO 9000-3 Quality management and quality assurance standards part 3*, Reference number ISO9000-3:1991(e).

GLOSSARY OF TERMS

Glossary of Terms

The Glossary of Terms contains terms that are used frequently in the software quality and strategic management fields. The terms were cited from their sources for purposes of clarity.

Alpha Test: Also known as acceptance tests or system tests. Used to determine if a system is ready to be deployed [9].

Balanced Scorecard: The balanced scorecard tracks the key elements of an organization's strategy – from continuous improvement and partnerships to teamwork and global scale [41].

Beta Test: Tests of a system in the same configuration as it will subsequently run live [9].

Business Concept: The definition of the intended customers of an organization, of the product offers it will create for those customers, and of the key activities through which the organization's product offers will be brought to its intended customers [78].

C++: A programming language that is based on the idea of object-orientation, which involves writing programs that are easily reusable in other systems [9].

Competence Groups: Groupings of firms by their current competence leveraging and building activities to identify which firms are competitors now and which will be competitors in the future [78].

Computer-Aided Software Engineering: Integrated tools that are used so that information created by one, tool can be used by another; a system for the support of software development [71].

Core Processes: The processes of product creation, product realization, and stakeholder development through which an organization creates and distributes value, plus periodic transforming processes that fundamentally change the way an organization works [78].

Cumulative Distribution Function: The probability that a measured value will fall between $-\infty$ and x [61].

Deliverables: What you propose to give to the user [9].

Financial Costs: The perceived total monetary costs of a customer's purchase, use, maintenance, and retirement of a product [78].

Functionality: What the software system does [9].

Gantt Chart: Standard project planning and control tool that shows tasks and when they are to be done [9], [53].

Key Activities: The activities that are critical in successfully delivering the net delivered customer value for customers in a targeted market segment [78].

Market Preferences: The preferred ways of satisfying basic needs that are usually determined by consumer's lifestyles and economic situations [78].

Market Segments: Groupings of potential customers with similar preferences [78].

Methodology: A proprietary approach to the development of a system [9].

Organization Concept: The definition of the resources, organization design, and controls and incentives through which an organization intends to carry out its business concept [78].

Outsourcing: Delegating a task to someone in a separate company or organization [9].

Procedure: A set of instructions on a computer which describe how a particular software task should be carried out [37].

Product Benefits: The perceived value of the functions, features, and performance levels of a product that enable a user of the product to do things they could not otherwise do, or could not do as well [78].

Quality Control: An activity that ensures that a particular quality factor is present in a software system and its associated documentation [36].

Quality Factor: An aspect of a software product that is important to the customer and/or the developer [36].

Quality Function Deployment: A technique to identify all the factors which might affect the ability of a design or product to satisfy the customer, and the methods and responsibilities necessary to ensure control [62].

Quality Manual: A document that contains standards, procedures and guidelines which can be adopted by a project manager for a particular software project [36].

Quality Plan: A plan that embodies the standards, procedures, and quality controls that are to be used on a project [36].

Quality System: A system to ensure that quality factors identified at the beginning of a project are present in a completed software system [36].

Regression Tests: Tests which are rerun of previous tests. Repeated because a system has changed, to confirm that the changes have not damaged existing functionality [9].

Software Engineering: The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software. Also the establishment and use of sound engineering principles in order to obtain economically software that is reliable and works efficiently on real machines [71].

Software Engineering Database: A repository containing important information about analysis, design, program construction, and testing [71].

Software Quality Assurance: An umbrella activity that is applied throughout the software process. SQA encompasses a quality management approach, effective software engineering technology (methods and tools), formal technical reviews that are applied throughout the software process, a multi-tiered testing strategy, control of software documentation and the changes made to it, a procedure to assure compliance with software development standards (when applicable), and measurement and reporting mechanisms [71].

Standard: A description of the way in which a project document is structured and how information in the document is presented [36].

Strategic Logic: An organization's operative rationale for achieving its goals through its process of value creation and distribution [78].

Time Costs: The perceived costs of the time a customer might spend to acquire, use, maintain, and retire a product [78].

Total Quality Management: A system whereby all the activities that contribute to product quality, not just production quality control, are appraised and controlled by one manager. In this context quality is defined as the totality of features which determines a product's acceptability, and as such includes appearance, performance, reliability, support, etc [38], [62].

Validation: The process of evaluating software at the end of the software development process to ensure compliance with software requirements [37].

Verification: The process of determining whether or not the products of a given phase of the software development cycle fulfil the requirements established during the previous phase [37].