MOTIVATORS AND DE-MOTIVATORS IN SOFTWARE PROCESS IMPROVEMENT: AN EMPIRICAL STUDY

NATHAN BADDOO

A thesis submitted in partial fulfilment of the requirements of the University of Hertfordshire for the degree of Doctor of Philosophy

The programme of research was carried out in the
Department of Computer Science,
Faculty of Engineering and Information Sciences,
University of Hertfordshire

December 2001

Table of contents:

LIST OF FIGURES: **6**

LIST OF TABLES **7**

ACKNOWLEDGEMENTS **8**

ABSTRACT **9**

CHAPTER ONE: INTRODUCTION **10**

1.1 INTRODUCTION AND PROBLEM OVERVIEW 10
1.2 RESEARCH HYPOTHESIS 13
1.3 METHODOLOGY 15
1.4 OVERVIEW OF THESIS 17
1.4.1 Chapter Two 18
1.4.2 Chapter Three 18
1.4.3 Chapter Four 18
1.4.4 Chapter Five 19
1.4.5 Chapter Six 19
1.4.6 Chapter Seven 20
1.4.7 Chapter Eight 20
1.4.8 Chapter Nine 20
1.5 RELATIONSHIP BETWEEN THIS RESEARCH AND PPP PROJECT 21

CHAPTER TWO: SPI, MOTIVATION AND MOTIVATING SOFTWARE PRACTITIONERS **22**

2.1 INTRODUCTION 22
2.2 BACKGROUND TO SPI 23
2.2.1 Definition 23
2.2.2 The benefits of SPI 24
2.2.3 Approaches to SPI 26
2.2.4 Emerging themes in SPI 29
2.2.5 Limitations of SPI 32
2.2.6 Obstacles to SPI 34
2.3 MOTIVATION THEORIES 37
2.3.1 Process theories 38
2.3.2 Content theories 43
2.3.3 Summary of motivation theories 46
2.4 MOTIVATING SOFTWARE ENGINEERS 47
2.4.1 Case studies on motivators of software engineers 47
2.4.2 Motivating software engineers for SPI 56
2.4.3 Gap in the research 58
2.5 STAFF GROUP PERSPECTIVE OF MOTIVATORS AND DE-MOTIVATORS FOR SPI 59
2.6 SOFTWARE PRACTITIONERS' PERCEPTION OF THEIR ROLE IN SPI 59
2.7 SPI MANAGERS' PERCEPTIONS OF SPI 60
2.8 OUTPUTS OF LITERATURE REVIEW 64
2.8.1 Four themes underpinning software practitioners' support for SPI 64
2.8.2 Guidelines for increasing software practitioners' support for SPI 66

CHAPTER THREE: METHODOLOGY **76**

3.1 INTRODUCTION 76
3.2 EMPIRICAL RESEARCH 76
3.2.1 Definition 76
3.2.2 Relevance of empirical methods 77

UNIVERSITY OF HERTFORDSHIRE
LRC
005.1 BAD
Hat Stack (KOS's
6000 660290

3.3. QUALITATIVE AND QUANTITATIVE RESEARCH METHODS 78
3.3.1 Differences between qualitative and quantitative research methods 79
3.3.2 Subjective and objective data 79
3.4 RESEARCH DESIGN 81
3.4.1 Data collections methods 82
3.4.2 Data analysis methods 92
3.4.3 Verifying research results 113
3.4.4 Limitation of research design 118
3.5 OTHER RESEARCH APPROACHES 119
3.5.1 Grounded theory 119
3.5.2 Ethnography 121

CHAPTER FOUR: SPI MANAGERS' PERCEPTION OF MOTIVATORS AND DE-MOTIVATORS FOR SPI 123

4.1 INTRODUCTION 123
4.1.1. Study aims 123
4.1.2 Rationale and background for investigating SPI managers' perceptions of SPI 123
4.2 SAMPLE PROFILE 125
4.2.1 Respondent profile 125
4.2.2 Sub-Sampling 127
4.3 ANALYSING SPI MANAGERS' PERCEPTIONS 127
4.3.1 Analysis of the inputs 127
4.3.2 Analysis of *implementors* 130
4.3.3 Analysis of SPI outputs 133
4.3.4 Process maturity 135
4.4 CONCLUSION 136
4.4.1 Summary of findings 136
4.4.2 Questions raised from findings 138

CHAPTER FIVE: ANALYSIS OF MOTIVATORS 140

5.1 INTRODUCTION 140
5.1.1 Study aim 140
5.1.2 Study rationale 140
5.2 DATA COLLECTION AND ANALYSIS PROCESS 141
5.2.1 Data collection via focus groups 141
5.2.2 Data analysis: frequency analysis and smallest space analysis (SSA) 141
5.3 FINDINGS OF PRACTITIONER MOTIVATORS 142
5.3.1 Developer motivators 143
5.3.2 Project manager motivators 144
5.3.3 Senior managers' motivators 145
5.3.4 Motivators across practitioner groups 146
5.3.5 Relationship between motivators 148
5.3.6 Limitation of SSA findings of practitioners' motivators for SPI 153
5.4 DISCUSSION OF MOTIVATORS 153
5.4.1 Similarities across practitioner groups 153
5.4.2 Differences in motivators across practitioner groups 156
5.4.3 Predicting developer motivations 158
5.5 CONCLUSION 159
5.5.1 Summary of findings 159

CHAPTER SIX: ANALYSIS OF DE-MOTIVATORS 163

6.1 INTRODUCTION 163
6.1.1 Study aim 163
6.1.2 Study rationale 163
6.2 DATA COLLECTION AND ANALYSIS PROCESS 164
6.2.1 Data collection 164

6.2.2 Data analysis: frequency analysis and smallest space analysis (SSA) 164
6.3 FINDINGS OF PRACTITIONER DE-MOTIVATORS 165
6.3.1 Developers' de-motivators 166
6.3.2 Project managers' de-motivators 168
6.3.3 Senior managers' de-motivators 169
6.3.4 De-motivators across practitioner groups 170
6.3.5 Relationship between de-motivators 173
6.3.6 Limitation of SSA findings of practitioners' de-motivators for SPI 180
6.4 DISCUSSION 180
6.4.1 Major de-motivators 180
6.4.2 Spread of de-motivators across practitioner groups 184
6.5 CONCLUSION 185
6.5.1 Summary of findings 185

CHAPTER SEVEN: PRACTITIONERS' PERCEPTION OF THEIR ROLE IN SPI 188

7.1 INTRODUCTION 188
7.1.1 Study aim 188
7.1.2 Rationale for study on practitioner perceptions of SPI 188
7.2 DATA COLLECTION AND ANALYSIS PROCESS 189
7.2.1 Data collection 189
7.2.2 Data analysis 190
7.3 STUDY RESULTS 190
7.3.1 Developers' perceptions 190
7.3.2 Project managers' perceptions 193
7.3.3 Senior managers' perceptions 195
7.4 DISCUSSION: PRACTITIONERS' PERCEPTIONS OF THEIR ROLE IN SPI 197
7.4.1 Understanding the SPI roles of practitioner groups 197
7.4.2 Similarities and differences in perception of SPI roles across practitioner groups 199
7.5 CONCLUSION 202

CHAPTER EIGHT: RESEARCH FINDINGS 205

8.1 INTRODUCTION 205
8.2 RESEARCH FINDINGS 205
8.2.1 SPI managers' perception of the motivators and de-motivators for SPI 206
8.2.2 Motivators for SPI 207
8.2.3 De-motivators for SPI 208
8.2.4 Practitioners' perception of their roles in SPI 209
8.3 VALIDATING LITERATURE GUIDELINES WITH RESEARCH FINDINGS 210
8.3.1 Visible management support for SPI 211
8.3.2 Secure practitioner buy-in 212
8.3.3 Transfer ownership of processes to practitioners 212
8.3.4 Communicate SPI success 213
8.3.5 Provide SPI training to all practitioners 213
8.3.6 Standardise SPI practices 213
8.3.7 Share SPI best practice 214
8.3.8 Visibly re-prioritise SPI 214
8.3.9 Dedicate resources to SPI 214
8.3.10 Provide internal leadership for SPI 215
8.3.11 Manage internal resistance 215
8.3.12 Reward SPI work 216
8.3.13 Encourage SPI forum 216
8.3.14 Initiate SPI from within the company 216
8.3.15 Make SPI objectives relevant to all practitioners 216
8.3.16 Guidelines not supported by study findings 217
8.4 CONCLUSION 218

CHAPTER NINE: CONCLUSION 219

9.1 SUMMARY OF RESEARCH FINDINGS 219
9.2 RESEARCH RECOMMENDATIONS 221
9.3 APPLICATION OF RESEARCH FINDINGS 221
9.4 CRITIQUE OF METHODOLOGY 223
9.5 SUCCESS AND USE OF METHODOLOGY IN FUTURE APPLICATION 226
9.6 FUTURE WORK 228
9.6.1 Predictive model of software practitioners' motivations 228
9.6.2 Extended Multidimensional scaling analysis of motivators and de-motivators for SPI 229
9.6.3 A survey of the findings of this research 230
9.6.4 Investigating SPI skills that improve software practitioners' support for SPI 230

REFERENCES: 232

APPENDICES 243

APPENDIX A: GUIDELINES FOR IMPROVING PRACTITIONERS' SUPPORT FOR SPI EXTRACTED FROM THE LITERATURE 243
APPENDIX B: COMPANIES IN FOCUS GROUP AND RGT STUDIES 244
APPENDIX C: MATRIX OF RESEARCH FINDINGS BY DATA SOURCES, METHODS AND TYPES 245
APPENDIX D: SAMPLE QUESTIONNAIRE - SPI MANAGERS' PERCEPTION OF SPI 246
APPENDIX E: INTER-RATER RELIABILITY TEST - SOFTWARE PRACTITIONERS' MOTIVATORS AND DE-MOTIVATORS FOR SPI 249
APPENDIX F: DEMONSTRATING THE CALCULATION OF CHI SQUARED 250
APPENDIX G: SUB-SAMPLE PERCENTAGES BASED ON REPORTED FAMILIARITY WITH SPI 251
APPENDIX H: DEFINITION OF MOTIVATORS (CONTENT CATEGORY DICTIONARY) 252
APPENDIX I: DATA MATRICES FOR DEVELOPERS, PROJECT MANAGERS AND SENIOR MANAGERS' MOTIVATORS 253
APPENDIX J: DEFINITION OF DE-MOTIVATORS (CONTENT CATEGORY DICTIONARY) 254
APPENDIX K: DATA MATRICES FOR DEVELOPERS, PROJECT MANAGERS AND SENIOR MANAGERS' DE-MOTIVATORS 255
APPENDIX L: A LIST OF BI-POLAR CONSTRUCTS, ELICITED FROM DEVELOPERS, DESCRIBING THE RELATIONSHIP OF THREE STAFF GROUPS TO SPI 256
APPENDIX M: A LIST OF BI-POLAR CONSTRUCTS, ELICITED FROM PROJECT MANAGERS, DESCRIBING THE RELATIONSHIP OF THREE STAFF GROUPS TO SPI 257
APPENDIX N: A LIST OF BI-POLAR CONSTRUCTS, ELICITED FROM SENIOR MANAGERS, DESCRIBING THE RELATIONSHIP OF THREE STAFF GROUPS TO SPI 258
APPENDIX O: THIS AUTHOR'S PUBLICATION REFERENCES 259

List of figures:

Figure 1: The quality continuum: [Hall and Wilson, 1997] 12
Figure 2: Illustration of research methodology 16
Figure 3: CMM levels in: [Humphrey, 1989; Paulk et al., 1994a] 27
Figure 4: JCT model of motivation in [Couger and Zawacki, 1980] 39
Figure 5: Maslow's hierarchy of needs [Maslow, 1954] 44
Figure 6: Example of a transcribed bi-polar grid 91
Figure 7: One-dimensional representation of statistical relationship 105
Figure 8: An illustration of a SSA plot using CMM KPAs 108
Figure 9: Distribution of years served in computing and present company 126
Figure 10: Familiarity with SPI 128
Figure 11: Experiences of SPI 129
Figure 12: The difficulty of implementing SPI 130
Figure 13: Top-down implementation of SPI 131
Figure 14: Bottom-up implementation of SPI 131
Figure 15: Senior management support of SPI 132
Figure 16: Developers' enthusiasm for SPI 133
Figure 17: Software quality and SPI 134
Figure 18: Cost benefits of SPI 135
Figure 19: SSA plot of developers' motivators 149
Figure 20: SSA plot of project managers' motivators. 150
Figure 21: SSA plot of senior managers' motivators 152
Figure 22: SSA of developers' de-motivators for SPI 173
Figure 23: SSA of project managers' de-motivators for SPI 176
Figure 24: SSA of senior managers' de-motivators for SPI 178

List of tables

Table 1: Herzberg's extrinsic and intrinsic factors in:[Couger and Zawacki, 1980] 45
Table 2: Example of a bi-polar construct 90
Table 3: Inter-rater agreement frequencies 102
Table 4: Overview of the kappa statistics and significance results 102
Table 5: Multivariate correlation - 3 variables 105
Table 6: Multivariate correlation - 4 variables 105
Table 7: Example of a data matrix of non-parametric variables 107
Table 8: Example of a raw grid showing constructs against elements 111
Table 9: Eliciting element to element agreement 111
Table 10: Example of an element by element construct 111
Table 11: Illustrating a construct by construct agreement 112
Table 12: Illustrating a construct by construct agreement with a reversed construct 112
Table 13: People with SPI responsibility 125
Table 14: Correlation matrix of process maturity and questionnaire responses 135
Table 15: Developer motivators 143
Table 16: Project manager motivators 144
Table 17: Senior managers' motivators 145
Table 18: Common motivators 146
Table 19: Motivators unique to particular practitioner groups 147
Table 20: What managers perceive motivates developers 147
Table 21: Summary of de-motivators in study across all groups 166
Table 22: De-motivators cited by developers 167
Table 23: De-motivators cited by project managers 168
Table 24: De-motivators cited by senior managers 170
Table 25: Common de-motivators across practitioner groups 171
Table 26: De-motivators cited in only specific groups 172
Table 27: Developers' element by element agreement matrix 190
Table 28: Construct by construct matrix - developers 191
Table 29: Reconstructed grid with similar elements and constructs next to each other (developers) 192
Table 30: Project managers' element by element agreement matrix 193
Table 31: Construct by construct matrix - project managers 194
Table 32: Reconstructed grid with similar elements and constructs next to each other (project managers) 195
Table 33: Senior managers' element by element agreement matrix 195
Table 34: Construct by construct matrix - senior managers 196
Table 35: Reconstructed grid with similar elements and constructs next to each other (senior managers). 197
Table 36: How literature guidelines are supported by research findings 211
Table 37: Research recommendations 221

Acknowledgements

I would like to express my thanks and appreciation to the following people for their guidance and support:

Dr Tracy Hall, my principal supervisor for her invaluable guidance, vision, support and encouragement throughout the process of researching this thesis.

Prof. Martin Loomes, my second supervisor, for his guidance, vision and encouragement throughout the process of researching this thesis.

Members of the CESPR research team at the University of Hertfordshire for their support and encouragement.

The Faculty of Engineering and Information Sciences at the University of Hertfordshire for the academic and administrative support I received as a Ph.D. student.

Finally, all the UK companies who participated in this study, whom for reasons of confidentiality, I am unable to disclose.

I dedicate this work to my mum Agnes, my dad Nathaniel, my best friend Lynn and Aimee. I love you all very much.

Abstract

Software quality problems are a concern for the software engineering community. Software Process Improvement (SPI) is the most recent and most popular approach adopted to address this problem. SPI focuses on the processes that develop software in order to deliver improvements to the product. Despite this popularity of SPI there is insufficient evidence of its successful impact on software quality. Quality problems in software continue. This has led to some concern in the industry about the effectiveness of SPI in tackling the problem of software quality. There is evidence to suggest that SPI does improve software quality. However, there is also evidence to suggest that SPI is not sufficiently supported by software practitioners. This lack of support may be one of the reasons why SPI appears to be failing at tackling the problem of software quality. In this research it is argued that this lack of support for SPI is caused by companies' inability to manage software practitioners' motivation for SPI properly. Companies may not be managing software practitioners' motivation for SPI properly because they may not understand them. There is therefore a need to better understand what software practitioners' motivations for supporting SPI are.

A review of the literature suggests a set of guidelines that can improve software practitioners' support for SPI. The literature also suggests four themes that underpin software practitioners' motivation for SPI. The four themes are SPI managers' perception of the motivators and de-motivators for SPI, software practitioners' motivators, software practitioners' de-motivators and the differences in software practitioners' motivators and de-motivators. The basis of this research is that exploring the four themes that underpin software practitioners' motivation for SPI improves understanding of the factors that influence support for SPI. This knowledge of the factors that influence support for SPI can then be used to validate and provide an empirical basis for the literature-suggested guidelines. Thereby improving confidence in the guidelines.

The four themes underpinning software practitioners' motivation for SPI are examined through empirical studies. Findings from these studies suggest that SPI managers perceive senior managers as not supportive of SPI. They also perceive developers as not enthusiastic about SPI. The findings also suggest that the key motivators of software practitioners for SPI are visible support and commitment from senior management and empowerment of practitioners, whereas the key de-motivators are related to constraints on resources and a failure to secure practitioners' buy-in for SPI. There are also differences in what motivates and de-motivates different practitioner groups for SPI and these differences are related to the jobs that practitioners do. Finally, software practitioners have different perceptions of their role in SPI, which are related to their software development roles. This suggests that the objectives of SPI should be tailored to the software development objectives of practitioners in order to improve their support for SPI.

Overall, findings from these studies confirm most of the guidelines suggested by the literature. The confirmed guidelines are offered as insight to improving support for SPI, which can in turn help to improve the impact of SPI on software quality.

Chapter One: Introduction

1.1 Introduction and problem overview

Deficiencies in software quality have led to some very high profile commercial disasters. For example, in 1987 a much-heralded computerisation of the London Stock Exchange suffered a setback when the system crashed within hours of its launch. More recently the disappointing performance of some high profile software projects in the UK have brought into sharp focus the issue of quality in software development. Examples include the computerisation of the London Ambulance Service, the Passport Office and House Of Commons projects [PAC, 1999]. Away from these high profile examples, quality problems abound in software development generally [Humphrey, 1998]. There have been increasing calls for the software industry to find solutions to the problem of quality [Crosby, 1986; Deming, 1986; Juran, 1988; Humphrey, 1989; Crosby, 1996]. These calls have become stronger in view of how commercially costly software problems have become to the industry [Fox and Frakes, 1997].

Several approaches have been developed to address this problem of software quality, of which Software Process Improvement (SPI) is currently popular. The aim of SPI is to focus on improving the processes used to develop software, in order to improve the quality of the product, that is to focus on the steps required to develop and maintain software [Humphrey, 1995]. Companies have adopted formal and informal SPI approaches to improve their processes. Some of the most popular formal models are the Capability Maturity Model (CMM) [Paulk et al., 1994a] developed in the US, and SPICE [ISO, 1999], an international model. There is empirical evidence to show that these models can improve software quality. Work done by Humphrey and the SEI amongst many provide such evidence [Humphrey, 1989; Paulk et al., 1994a; Humphrey, 1995]. Also, there are some well documented studies on the benefits of SPI [Herbsleb et al., 1994; Paulk et al., 1994b; Goldenson and Herbsleb, 1995; Herbsleb and Goldenson, 1996; Krasner, 1997; Willis et al., 1998; Krasner, 1999; Pitterman, 2000]

Despite the current popularity of the SPI approach, there remain doubts about the effectiveness of SPI to improve software quality [El Emam and Briand, 1997; Gray and Smith, 1998; Glass, 1999; Leung, 1999]. Glass has suggested that this is due to the lack of sufficient quantitative evidence to support the benefits of SPI [Glass, 1999].

Others like McDermid and Bennet [1999] have argued that the human factors to software process improvement have been ignored over the years and this has been detrimental to the effectiveness of SPI programmes [McDermid and Bennett, 1999]. Hall and Wilson [1997] suggest that to improve the impact of SPI on software quality, greater attention needs to be paid to the people management factors of SPI. This view is illustrated by DeMarco and Lister [DeMarco and Lister, 1987]:

"If you find yourself concentrating on the technology rather than the sociology, you are like the vaudeville character who loses his keys on a dark street and looks for them on the adjacent street because, as he explains, 'The light is better there'".

DeMarco and Lister [1987] report that companies which pay more attention to people management factors are more likely than those who do not to be successful at the projects they implement. In the context of SPI, this suggests that SPI programmes that pay better attention to people management issues are more likely to be successful. Indeed, accounts from companies that report success in software process improvement suggest a link between good people management practices and high SPI success [Herbsleb et al., 1994; Krasner, 1997; Ahuja, 1999; Hammock, 1999].

Few studies, however, have examined the impact that people have on the effectiveness of software development processes. Those that have [Hall, 1995; Hall and Fenton, 1997] emphasise the importance of the relationship between the human and technical aspects of software development. These studies suggest a relationship between the human aspects of software quality improvement and the degree of improvement achieved in the product. This relationship is illustrated by Figure 1.

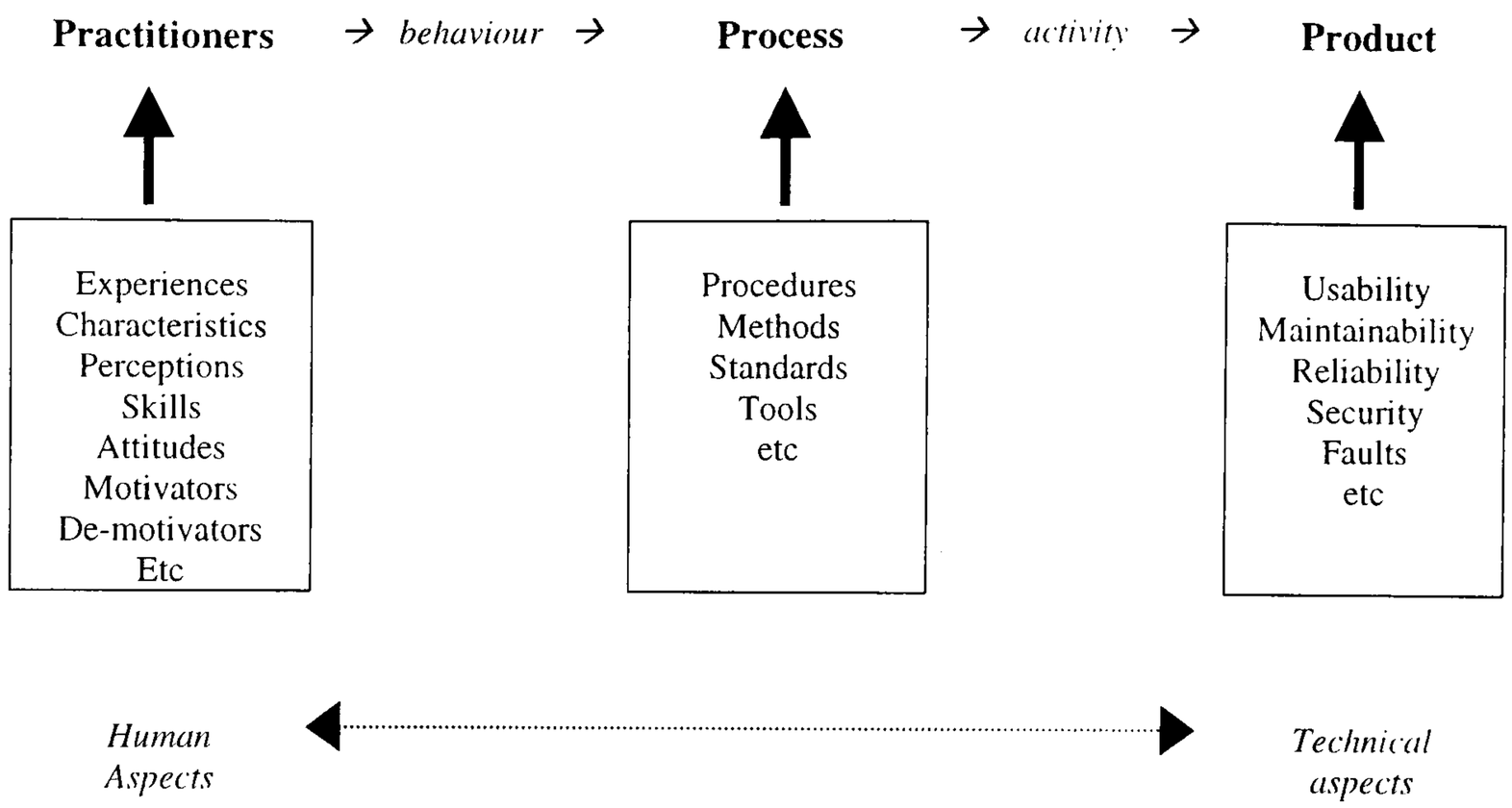


Figure 1: The quality continuum: [Hall and Wilson, 1997]

Hall and Wilson [1997] suggest an important relationship between the human and technical aspects of software development. They suggest that the experiences, characteristics, perceptions *etc* of software practitioners impact indirectly on the quality of the software produced. This implies that such attributes influence how software practitioners behave towards approaches that are adopted to improve software processes. The processes, in turn, impact directly on the attributes of the eventual product. This sequence of events illustrates a continuum between human aspects and technical aspects of software development.

Humphrey [1995] reports that a well-defined process can be improved if the problems of motivating people to properly use it are found and corrected. This suggests that motivators and de-motivators impact directly on practitioner responses to SPI

However, what motivates software practitioners to properly use a process can be elusive. According to Brooks Jr. [1995] much of the progress made in the area of motivating software engineers to work better has concentrated on the reduction of environmental factors that impede their productivity (reducing the de-motivators). Not much effort has

been invested in increasing the factors that improve performance - the motivators. According to Fitz-Enz [1978] the factors that actually motivate software practitioners are directly intrinsic to the jobs that they do. This seems to suggest that if the job-intrinsic factors of SPI are identified, understood and managed then managers may have a better chance of actually motivating software practitioners to support SPI.

In studies carried out in the US in the late 1970's to early 1980's, Couger and Zawacki [1980] established that out of about five hundred different professions, software practitioners were the group with the most personal growth needs. As a result they were the group most likely to be motivated by the factors that were intrinsic to their jobs, for example responsibility and opportunity for advancement. These studies also suggest that understanding and managing job-related factors is instrumental in motivating software practitioners to work better.

Findings from earlier studies in SPI suggest that the way software practitioners respond to SPI differ in relation to their hierarchical staff groups [Hall and Wilson, 1997]. This is because different staff groups have different perceptions of SPI. In their conclusion from this study on SPI involving different staff groups, Hall and Wilson [1997] suggest that these differences in practitioner perceptions of SPI can affect the effectiveness of SPI implementation strategies. SPI strategies that do not reflect the different practitioner perceptions to SPI are unlikely to elicit coherent responses from the different groups of practitioners. For example, project managers may perceive measurement as a useful element in process improvement to monitor project progress, whereas developers may perceive it as monitoring their productivity. Such differences in practitioners' perceptions can influence how they respond to SPI. These different responses to SPI then tend to make the whole SPI initiative less effective.

1.2 Research hypothesis

An empirical study of the opinions of software practitioners in UK companies will confirm the motivators and de-motivators of software practitioners for SPI as suggested by the literature.

The aim of this research is to collect empirical evidence from thirteen UK companies to validate the SPI recommendations reported in the literature.

In this research the term 'opinion' describes the beliefs, attitudes or values expressed by software practitioners as defined by Rokeach in [Rokeach, 1968]. The opinions used in this research are verbally expressed by software practitioners via the research's data collection methods. Opinion data has been shown in many disciplines to be reliable, for example social science research makes extensive use of opinion data. See for example [Axtell et al., 2000].

The term software practitioner is used to reference all people involved in software development (this excludes users). Practitioners are further classified into three hierarchical groups: developers, project managers and senior managers. This follows the approach of other researchers who have classified practitioners similarly [Herbsleb and Goldenson, 1996; Hall and Wilson, 1997; El Emam et al., 2001].

Developers are defined as comprising: engineers, testers, designers, analysts and all grassroots practitioners who are directly involved in the development of the software product. Project managers comprise: team leaders, team managers, project leaders and project managers. Senior managers refer to senior software managers who typically direct the activities of project managers.

The motivators of SPI are defined as the factors, conditions or circumstances around or within the deployment of SPI that encourage software practitioners to support SPI. These factors, conditions or circumstances can either be tangible or intangible, implicit or explicit. This definition stems from the work of Herzberg in [Herzberg, 1987].

Similarly, the de-motivators of SPI are defined as the obstacles to the successful uptake of SPI by software practitioners. These are the factors, conditions or circumstances around or within the deployment of SPI that discourage software practitioners from supporting SPI. Again, these factors, conditions or circumstances can either be tangible or intangible, implicit or explicit. This definition stems from the work of Goldenson and Herbsleb [Goldenson and Herbsleb, 1995].

The thirteen UK companies participating in the study represent companies of varying operational complexity, of different sizes, involved in a range of businesses and producing various types of applications. All thirteen companies are based in the UK and are either UK owned or UK subsidiaries of multinational companies.

The literature refers to publications on software engineering and behavioural sciences. These publications cover a range of topics from software process improvement, quality management to motivation. The types of publication range from books, journals, conference proceedings, published case studies, company reports, technical reports, workshop discussions, magazine articles, published thesis and electronically sourced documents.

1.3 Methodology

1.3.1 Overview

The SPI literature largely consists of single case study reports, company experience reports and high-level software process texts. Very few theoretical or empirical studies are presented in the literature. A review of the literature identified four major themes underpinning practitioner motivation for SPI:

i. SPI change agents.
ii. SPI motivators.
iii. SPI de-motivators.

iv. Different SPI motivators and de-motivators for different groups of staff.

Based on these four themes a variety of specific recommendations and guidelines can be distilled from the literature which claim to identify ways in which practitioner support for SPI can be increased. However the empirical basis of many of these recommendations and guidelines is not clear.

In order to confirm or refute the literature recommendations empirically, four independent studies have been designed. The aim of these studies is to investigate each of the four literature themes empirically in order to establish the validity of the published recommendations. Figure 2 provides a diagrammatic overview of the research methodology.

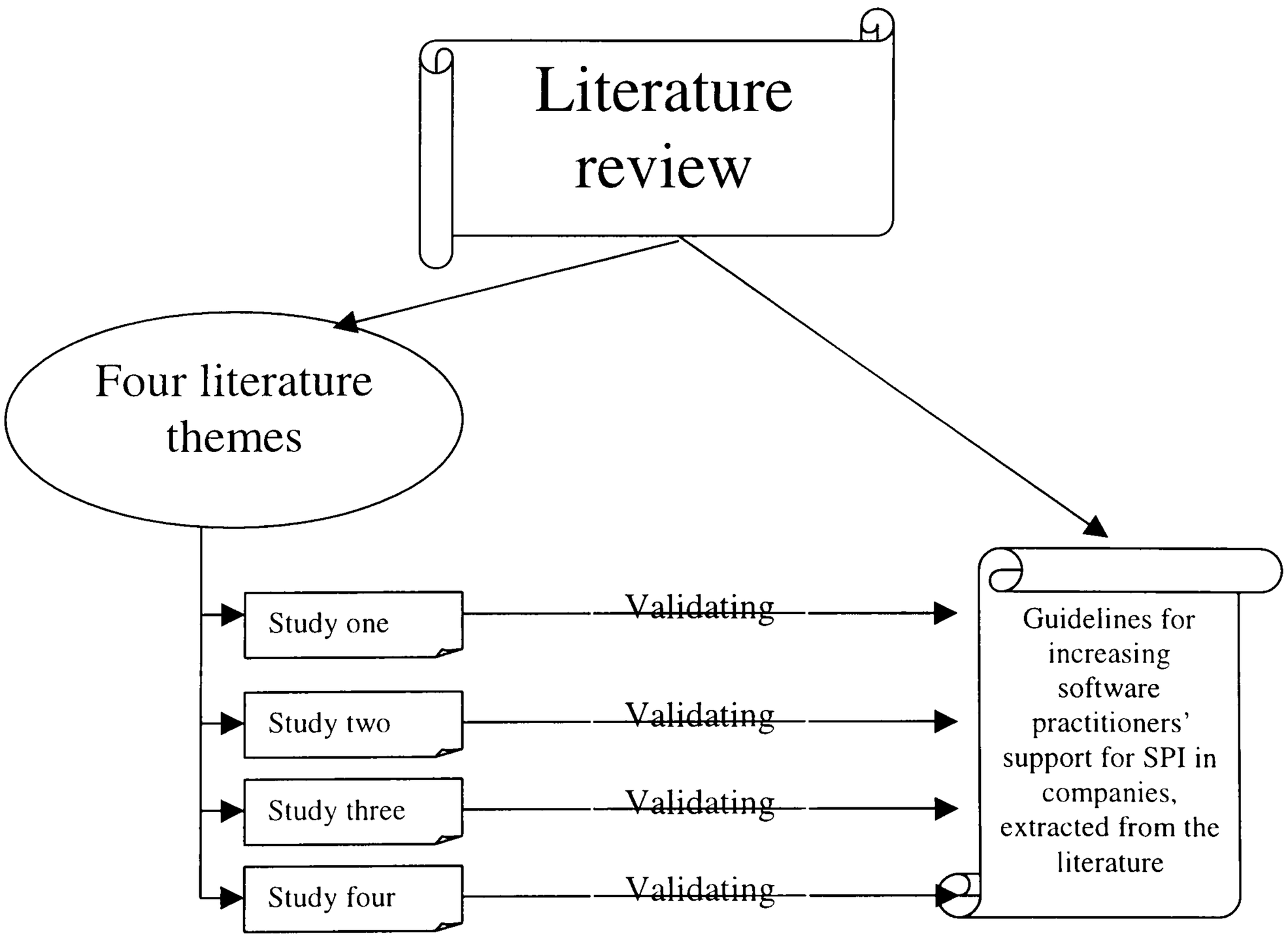


Figure 2: Illustration of research methodology

1.3.2 Four studies

The following describes the four studies conducted to investigate the four themes identified by the literature.

Study One: SPI change agents

In most companies SPI managers act as the change agents for SPI. In this first study a survey of 80 SPI managers is conducted. This survey examines SPI managers' perceptions of the motivators and de-motivators of SPI. This study is reported in Chapter Four and further details of the study can be found in [Baddoo and Hall, in review-b].

Study Two: Motivators for SPI

In the second study 49 focus group sessions are used to elicit software practitioners' motivators for SPI. This study is reported in Chapter Five and further details of the study can be found in [Baddoo and Hall, 2002b].

Study Three: De-motivators for SPI

The third study uses focus group exercises to elicit software practitioners' de-motivators for SPI. This third study examines the factors that inhibit software practitioners from supporting SPI. This study is reported in Chapter Six and further details of the study can be found in [Baddoo and Hall, 2002a].

Study Four: Differences between staff groups

The fourth study is a repertory grid exercise conducted with over 200 practitioners to elicit their opinions on roles in SPI. This study identifies how different staff groups perceive their role in SPI and how those perceptions influence the differences in their motivators and de-motivators for SPI. This study is reported in Chapter Seven and further details of the study can be found in [Baddoo and Hall, in review-a].

1.4 Overview of thesis

This research is organised into nine chapters. Chapter two presents the background to the research areas and shows how the research questions were formulated and how SPI guidelines were extracted. Chapter three describes the research methods used. Chapters

Four, Five, Six and Seven describe studies carried out to validate SPI guidelines derived from the literature. Chapter Eight discusses results from this research. Chapter Nine presents conclusions and introduces future work.

The following is a summary of the chapters.

1.4.1 Chapter Two

Chapter Two sets out to explain the recommendation that the literature makes regarding motivators and de-motivators for SPI. For example, Chapter Two establishes that senior management commitment is a very critical motivator to SPI success. Overall this chapter answers the research question:

What recommendations does the literature make on SPI motivators and de-motivators?

Chapter Two also explains how the four major themes underpinning software practitioners' motivation are identified from the literature.

1.4.2 Chapter Three

Chapter Three presents the design of the research process. It discusses the approach adopted in this research and describes particular research methods and techniques used. It explains the rationale behind choosing the research design and shows how the concept of triangulation was applied in this research.

1.4.3 Chapter Four

Chapter Four sets out to answer the following research question:

What motivators and de-motivators do SPI change agents report?

It describes a study conducted amongst UK SPI managers. This study uses a questionnaire survey to elicit the opinions of SPI managers on SPI motivators and de-motivators. The findings of this study are presented to set the context of the overall research. These findings show the opinions of SPI managers and acts as supplementary evidence to the nature and state of motivation for SPI in UK companies.

1.4.4 Chapter Five

Chapter Five sets out to answer the following research question:

What SPI motivators do software practitioners report?

It presents a study of software practitioners' motivators for SPI. This study uses focus group discussions with 49 groups of software practitioners in thirteen UK companies. The findings of this study characterise software practitioners' motivators according to classic motivation theory. The findings show similarities and differences of motivators across practitioner groups. The findings also show inter-relationship between software practitioners' motivators, indicating the likelihood that motivators will co-occur to each other.

1.4.5 Chapter Six

Chapter Six sets out to answer the following research question:

What SPI de-motivators do software practitioners report?

This chapter presents a study of software practitioners' de-motivators for SPI. This study uses focus group discussions with 49 groups of software practitioners in thirteen UK companies. The findings of this study present the major factors that de-motivate software practitioners from supporting SPI. The findings show similarities and differences in de-motivators for SPI across software practitioner groups. The findings of this study also

show inter-relationship between software practitioners' de-motivators for SPI. This inter-relationship indicates the likelihood of de-motivators co-occurring to each other.

1.4.6 Chapter Seven

This chapter sets out to answer the following research question:

What are the differences in the SPI motivators and de-motivators that senior managers, project managers and developers report?

It presents a study on software practitioners' perception of their role in SPI. This study uses Repertory Grid technique on 46 groups of software practitioners in thirteen UK companies. The findings from this study show how software practitioners see their role in SPI. These findings explain the differences in software practitioners' motivators and de-motivators across the staff groups.

1.4.7 Chapter Eight

Chapter Eight sets out to explain how the guidelines for increasing practitioners' support for SPI that were extracted from the literature are validated by this research. In this Chapter, the overall findings of the four studies are used to refute or confirm the guidelines. The set of guidelines that are confirmed by the study findings are amalgamated and presented as empirically based recommendations of this research.

1.4.8 Chapter Nine

Chapter Nine presents a summary of this research. It reviews the research recommendations and suggests what they contribute to knowledge on SPI. Chapter Nine also reviews the research process and explains how the research hypothesis is proved. Finally, Chapter Nine introduces future work initiated by this research. It explains how research recommendations can be expanded and tested out in future studies.

1.5 Relationship between this research and PPP project

This research is a subset of a bigger research project on SPI: The People Practitioners and Products (PPP) project, which is funded by the Engineering and Physical Sciences Research Council (EPSRC) under grant number GR/L91962. The PPP project is an investigation into the human aspects of SPI implementation in UK companies. It covers a variety of issues from practitioners' understanding of SPI, skills for SPI and the interpersonal politics involved in SPI implementation. This research, however, concentrates specifically on the motivators and de-motivators for SPI.

Some of the data collection processes described in Chapter Three and in the study chapters were undertaken in conjunction with the overall PPP project, whilst others were specifically undertaken for this research. This is explained as follows:

- The data collection process for the study of SPI managers' perception of motivators and de-motivators for SPI, reported in Chapter Four, was undertaken specifically for this research
- The data collection process for the study of software practitioners' motivators and de-motivators for SPI, reported in Chapters Five and Six, were undertaken in conjunction with data collection for the overall PPP project.
- The data collection process for the study of practitioners' perception of their role in SPI, reported in Chapter Seven, was undertaken specifically for this research

All the data analysis reported in this thesis were conducted specifically for this research. The findings reported here are specifically from these analyses.

Chapter Two: SPI, motivation and motivating software practitioners

2.1 Introduction

This chapter presents the literature review for this research. It sets out to explain the recommendations that the literature makes regarding motivators and de-motivators for SPI. It also identifies four themes underpinning software practitioners' motivations for SPI.

The review of the literature in this chapter includes a review of software process improvement as an approach to improving software quality. It also includes a discussion of motivation and an overview of key theories. This chapter then presents a review of work that has been done in the area of motivating software practitioners and presents a summary of the key factors that have been identified to motivate software practitioners. This chapter shows that despite the many studies that have established motivators and de-motivators of software practitioners, generally, there have been few empirical studies on what motivates or de-motivates software practitioners for SPI. Finally, this chapter draws from the literature two outputs:

- A set of recommendations identified as critical to software practitioners' motivators and de-motivators for SPI
- Four themes underpinning software practitioners' motivations.

The aim of this chapter is to provide background material that puts this research into context, sets the scene for the contribution that this research will make to knowledge and above all answers the following research question:

What recommendations does the literature make on SPI motivators and de-motivators?

This rest of this chapter is structured as follows:

Section 2.2 provides background to SPI. Section 2.3 provides an overview of motivation and motivation theories. Section 2.4 presents a review of studies done on motivating software practitioners. In Section 2.5 this research discusses the basis for investigating staff group perspectives of motivators and de-motivators for SPI. Section 2.6 reviews software practitioners' perception of their role in SPI. In Section 2.7, this research reviews the background to SPI managers' perception of SPI. Finally, in Section 2.8 two outputs of the literature review are summarised detailing recommendations for motivators and de-motivators for SPI and the four areas underpinning software practitioners' motivation.

2.2 Background to SPI

In this section an overview of SPI is presented. It includes the aims of SPI, an overview of key SPI models and some emerging approaches adopted in companies to assist SPI implementation. It also reports debate on the limitations of SPI and on the lack of evidence to support the impact of SPI on software quality

2.2.1 Definition

Fox and Frakes [Fox and Frakes, 1997] define SPI as a set of Process oriented Quality Management Systems (PSQM's) that apply a cohesive set of theories, tools, methods and techniques in conjunction with attitudes, values, and model problem solutions. Process oriented quality management in software is a relatively new phenomenon. It has evolved out of work started on software quality in the late 70's to early 80's by Crosby with the introduction of the quality management maturity grid [Crosby, 1979]. This was followed by theories on continuous process improvement with the conception of the Plan Do Check Action (PDCA) cycle by Deming [Deming, 1986] and Juran [Juran, 1988] in the mid to late 80's and parallel work pioneered by Humphrey on software process maturity [Humphrey, 1989].

The basis of SPI is largely agreed to be statistical process control [Paulk et al., 1994a]. The underlying theme of SPI is that by understanding and defining the current state of a company's software development processes, companies can sufficiently determine the areas within the development processes that they can control and manipulate to achieve a particular product effect [Humphrey, 1989]. To understand processes, the people involved in software development first need to collect data about these processes. This data characterises the processes. The measures are in turn used as the basis of assessment to determine how effective the individual areas in the processes are. Companies can then use the assessment to determine which aspects of the processes need improving. In effect, the assessment of processes is a driver for the improvement of processes [Gray and Smith, 1998].

2.2.2 The benefits of SPI

The following are some principal goals of SPI.

- Product improvement

Product improvement is achieved via practices adopted within the development process. These practices are particularly geared towards improving the attributes of the product. For example, practices like inspection, peer reviews, and requirements management are adopted to reduce product faults, improve product maintainability, adaptability and usability and also to satisfy user requirements.

- Process effectiveness

Another aim of SPI is to improve process effectiveness, for example improving time-scales and shortening time-to-market. At the beginning of adopting any improvement programme, companies are looking for control over their software development processes. To achieve such control, the improvement effort should be managed as though

it were a project with individual steps and phases. Some proponents of SPI (e.g. Bill Curtis) have suggested that adopting a project management approach to the improvement effort results in control, clarity and transparency of the development process. Curtis suggests that companies that succeed at SPI are those that have mastered change management [ESEPG, 1999].

- Organisational change management

SPI is also related to organisational change management. For example, Paulk et al argue that the CMM [Paulk et al., 1994a], is also about how to manage change within an organisation. The key elements of change are planning, implementation and communication [Humphrey, 1989]. Planning, implementation and communication are primarily people factors and are as critical to successful SPI as they are in any project.

According to Moitra [1998] if SPI changes are not planned, implemented and communicated properly then there are likely to be major people problems in terms of lack of vision, lack of SPI skills and resistance to change. Resistance to change, for example, could be caused by three main factors [Moitra, 1998]:

- uncertainty about the new processes
- fear of loosing control
- perceived increase on demand for resources

To overcome such resistance it is suggested that changes be made meaningful to practitioners by communicating the benefits that can be made from the new processes [Moitra, 1998]. Otherwise software practitioners will fail to support changes in processes if the benefits are presented as that of senior management only.

Overall, managing SPI by applying the concepts of change management is vital, as intimated by Paulk et al [1994a] and Moitra [1998]. This research suggests that to understand some of the possible factors that de-motivate software practitioners from

supporting SPI it is worth examining the effect of change management factors on software practitioners' motivators. For example, have software practitioners' uncertainties about new processes being addressed? If not, to what extent do these uncertainties contribute towards practitioners' resistance to SPI?

2.2.3 Approaches to SPI

The following provides an overview of various approaches to SPI. It discusses two approaches that are currently popular: CMM and SPICE, and also some other relevant approaches. It also introduces emerging themes and practices in implementing successful SPI. This review of SPI approaches and emerging themes provides an understanding of what is already being done to increase practitioners' support for SPI and sets a basis for building on these current approaches.

- Capability Maturity Model (CMM)

The CMM [Paulk et al., 1994a] was developed by the Software Engineering Institute (SEI) in Pittsburgh, USA. It was inspired by work carried out in the 1980's and 90's by Juran [Juran, 1988] and Deming [Deming, 1986] on continuous improvement and Crosby [Crosby, 1979; Crosby, 1986] on TQM. But the main thrust behind the CMM is the work done on process maturity at the SEI in the late 80's by Watts Humphrey [Humphrey, 1989]. The development of the CMM has been deeply influenced by the requirements of the US Department Of Defence (DOD) [Paulk et al., 1994a]. The DOD wanted to outsource software systems which were mainly safety critical and needed guarantees of organisational stability as well as reliability [Thompson and Mayhew, 1997]. This led software companies that wanted trade from the DOD to adopt the CMM as a measure of their viability. Much of the initial investment into developing the CMM has also been funded by the DOD.

The CMM describes an evolutionary path from ad-hoc immature software processes to optimised, disciplined and mature processes. This is depicted in five stages of maturity [Humphrey, 1989; Paulk et al., 1994a]:

Level	Process description	Process characteristics	Action needed
Level 5	Optimising	Continuously improving. There is also a quantitative basis for continued capital investment in process automation and improvement.	Continuous emphasis on process measurement and process methods to prevent errors
Level 4	Managed	Processes are quantitatively controlled. There is reasonable statistical control over product quality	Quantitative productivity plans and tracking.
Level 3	Defined	Practices are standardised organisation-wide. There are reliable costs and schedule predictions. Quality performance predictions are still improving but unpredictable	Establish process measurements and quantitative quality goals, plans, measurements and tracking
Level 2	Repeatable	Processes are under control. Practices are scheduled and managed, but are not consistent throughout the organisation.	Develop process standards and definitions. Assign resources to processes. Establish methods for stages of software development.
Level 1	Initial	Processes are chaotic and unpredictable. There are unpredictable costs, schedules and quality performances	Planning of size and cost estimates and schedules. Performance tracking. Change control. Commitment from management. Quality assurance.

Figure 3: CMM levels in: [Humphrey, 1989; Paulk et al., 1994a]

Overall, the CMM covers engineering, planning, managing and maintaining software processes. It is accepted as the *de-facto* standard in large companies in North America and is becoming increasingly popular in Europe and in companies in other parts of the world, often where the parent company is North American.

- SPICE

SPICE is a set of international standards on software process assessment [ISO, 1999]. It defines methods for measuring the implementation and institutionalisation of selected processes in the software development cycle. SPICE harmonises existing approaches to process improvement but does not recommend specific paths for improvement. It leaves the determination of a specific improvement path to the practising company.

SPICE is made up of nine parts that range from an introductory part to the vocabulary. However, the key part of this model is Part 2: *A model for process management* [ISO, 1999]. This was formerly know as the Baseline Practices Guide (BPG) [ISO, 1991]. BPG describes the core activities that are critical to good software engineering [ISO, 1999].

- ISO9001

ISO9001 is part of a suite of quality standards. It provides broad guidelines to software developers on how to implement, maintain and improve a quality system capable of ensuring high quality software [Ince, 1994].

- MBNQA

The Malcolm Baldrige National Quality Award is an annual award to US companies that excel in quality achievement and quality management [Lizotte, 1994]. The European quality award is derived from the Baldrige criteria and thirty-two countries have quality awards derived from the same criteria [Wernham, 1992]. The Baldrige award focuses on the organisation as a whole as opposed to software development

- BOOTSTRAP

BOOTSTRAP is an assessment model that concentrates on how software is produced by assessing the company's internal processes and how projects are run within the company [Thompson and Mayhew, 1997]. The BOOTSTRAP model makes use of automation and it is supported by a continuously updated database. BOOTSTRAP as a model is mainly used in Europe.

- Trillium

Trillium is an improvement model used predominantly by software operations in telecommunications. It is heavily based on the SEI's CMM and very much like the CMM

aims to initiate and guide continuous improvement [Thompson and Mayhew, 1997]. It is formed around the concept of a sequence of related activities from which sets of SPI goals are defined.

2.2.4 Emerging themes in SPI

The following are some of the emerging models adopted to support the implementation of SPI in companies.

- Personal Software Process

The Personal Software Process - PSP - [Humphrey, 1999a] addresses how individual practitioners can inject quality practices into their software development processes. PSP examines the Human Activity System (HAS) and concentrates on suggesting methods to improve defect management and accuracy estimation. As a model, PSP focuses on these two areas as key to achieving personal process improvement.

A big advantage of the PSP is that it attempts to provide evidence of the direct benefits of using defined processes to the software practitioner by using data generated by software practitioners themselves. In effect, it aims to get software practitioners, on an individual level, to support SPI by making them generate their own evidence of SPI's benefits. For example, software practitioners come round to accept that code reviews are more efficient at removing faults in the product than testing and debugging when they experience the benefits themselves through the PSP [Humphrey, 1999a].

By addressing individuals, PSP is unique as a process model. However, the drawback about the PSP is convincing software practitioners to follow it in the first place. The seven-step process recommended by the PSP can seem tedious to software practitioners. Humphrey [1999a] himself admits that one of the most difficult aspects of deploying the PSP is getting software practitioners to adopt the new methods.

- Team Software Process

The Team Software Process (TSP) developed by Watts Humphrey [Humphrey, 1999b] is one of the latest aids to implementing software process improvement models. It was actually developed to help engineering teams develop more intensive software and also complements the Capability Maturity Model.

The TSP is used with software teams of two to twenty practitioners. It explores group dynamics theory and takes into account the fact that practitioners do not produce software in isolation. The team software process attempts to address the engineering deficiencies in software development and encourages practitioners to view themselves as teams of engineers. TSP promotes the optimal use of science to build software products [Humphrey, 1999b].

TSP builds upon the improvements that individual software practitioners can make to their development practices by introducing that same rigor within the context of a team. So that in companies that have implemented the team software process, this introduction often begins only when practitioners have been trained in the PSP.

This research suggests that conceptually the TSP, like most of the improvement approaches described can bring improvement to software development. However, there are factors that impede this process. For example Humphrey [1998] suggests that getting software practitioners to change the way they work is an intractable problem because old practices become ingrained with software practitioners. As a result, without stringent supervision, software practitioners will mostly revert to their old practices [Humphrey, 1998].

This research is addressing the same problem of getting software practitioners to adopt new practices as raised by Humphrey [1998]. However, the approach of concentrating on improving the motivators and reducing the de-motivators may appear different from Humprey's suggestion of stringent control. Nevertheless it is possible that for some group

of practitioners stringent control may be a motivator to adopting SPI. This research discusses how this is possible in the latter parts of this chapter and presents some results from studies to highlight this concept in Chapter Five.

- People CMM

The People Capability Maturity Model (P-CMM) [Curtis et al., 1995] offers a guide to improving the company workforce through evolutionary stages. Like the Software-CMM. the P-CMM prescribes maturity levels for the personnel in software development. There are five levels of people maturity in the P-CMM which are described as [Curtis et al., 2001]:

Level 5	Optimising	Continuously improve and align personal, workgroup and organisational capacity
Level 4	Predictable	Empower and integrate workforce competence and manage performance quantitatively
Level 3	Defined	Develop workforce competencies and workgroups and align with business strategy and objectives
Level 2	Managed	Managers take responsibility for managing and developing their people
Level 1	Initial	Workshop practices are applied inconsistently

Each of these levels of maturity consist of process areas and the process areas consist of a set of goals that have to be satisfied to indicate the process area's ability to influence practitioners' capability [Curtis et al., 2001]. In this sense the P-CMM is very similar to the Software-CMM.

Overall, the P-CMM concentrates on workforce activities and offers a way of integrating these with process improvement to establish a culture of excellence.

This research suggests that the P-CMM is the closest human management model currently in use. Its major advantage is that it serves as a benchmark of the processes that should be in place to attain levels of software practitioners' capability. It also serves as a

guide as to how to achieve those processes by indicating the specific goals that should be met.

One major criticism of the P-CMM, however, is that it is tied very much to the Software-CMM thus recommends a hierarchical set of related activities that must be achieved in a particular sequence. This research suggests that the sequential nature of the recommendations can restrict how the model is applied in practice.

A second criticism is that like the Software-CMM, P-CMM can be perceived as too general and not addressing any specific people management concerns of software practitioners.

2.2.5 Limitations of SPI

Despite popularity of SPI approaches and emergence of themes to support SPI, quantitative evidence of the impact of SPI on software companies and their products is not easy to find [Glass, 1998]. This has led to speculation that the anticipated improvements to software quality through SPI have not been fully realised yet [Gray and Smith, 1998].

Also, even though there are advances in technologies and models to improve how software is developed, as depicted by models like the PSP, these advances have not been matched by equal advances in the adoption of these technologies and models in software development [Leung, 1999]. This suggests that the current problem with SPI is not a lack of models showing how to improve software development practices, but rather a lack of an effective strategy to get software practitioners to adopt these models. As Karl Wiegers intimates, there is no need for new models [Weigers, 1998]. What software engineering needs to do is to encourage more companies to use these models.

The following are some of the cited limitations to the uptake of SPI in companies.

- SPI is expensive

Start up costs for SPI are very high, creating a barrier for less commercially viable companies [Glass, 1999], [Leung, 1999] and [Reiblein and Symons, 1997].

Also, SPI can be costly when it fails and the likelihood of such failures is high. According to the Gatner group companies embarking on *"incremental unfocused changes have a better than 50% chance of failure"* resulting in a waste of time effort and money [McGuinness, 1999].

- Portability

Issues of portability also hinder SPI. According to Leung [1999], models like the CMM tend to be more suitable to larger companies. This is because larger companies tend to be in a better position to invest both personnel and financial resources in the recommended practices. Smaller companies tend to find it more difficult to tailor some of the recommendations of SPI models to their particular practices. Leung argues that this issue of portability tends to prevent some companies from embarking upon SPI.

- Lack of immediate successes

More than a decade after SPI became the prominent approach to tackling the problem of software quality and nearly a decade after the publication of some of the most popular SPI models, it is difficult to find companies that have ready accounts of SPI success [Kuilboer and Ashrafi, 2000]. This situation makes it difficult for companies that are not presently practicing SPI to appreciate the value of this approach to software quality. According to Leung [1999] SPI has an unknown value because it is difficult to show tangible benefits for the large-scale expenses spent on SPI programmes.

Such difficulties serve as limitations of SPI to companies that may be considering approaches to tackle their software quality problems.

2.2.6 Obstacles to SPI

Some of the limitations to SPI discussed in 2.2.5 contribute directly to companies' reluctance to embark upon SPI. However for companies that already have a SPI programme in place, there are other factors that may impact negatively on the success of these programmes. According to Zeibe [1996] two out of three software development operations have immature processes. This is despite the fact that many of these software operations have embarked upon either a formal or informal SPI programme of some sort.

In this section, this research presents some of the issues that serve as obstacles to SPI success in companies. For example, the following have been described as the barriers to SPI success in companies [Goldenson and Herbsleb, 1995]:

- Practitioners are discouraged about SPI prospects
- SPI is perceived to get in the way of 'real' work
- 'Turf guarding' (an unwillingness to change due to fear of loosing present position)
- Organisational politics
- When recommendations for SPI are too ambitious
- Need guidance about how to improve
- Need more mentoring and assistance

This research also discusses whether these barriers de-motivate software practitioners from supporting SPI. This is because even though several studies on SPI have reported on the key factors that frustrate the SPI effort in companies, few studies have isolated the issues that de-motivate software practitioners from supporting SPI.

- Resistance, inertia and negative experience

One of the biggest obstacles to introducing any new practice is the unwillingness of practitioners who actually use the practices to take them up. This problem is directly related to how the new practice is change managed. According to change management

theory, such resistance is brought upon when changes are not properly planned. implemented or communicated to practitioners [Moitra, 1998]. The literature suggests that it is important to investigate some of the change management problems associated with SPI to see how they affect software practitioners' motivators for SPI.

For example, software practitioners may resist SPI because they do not perceive any discernible incentives for giving up practices with which they are accustomed and feel comfortable with. Such a reaction may not necessarily be a pro-active response to the new practices being introduced, but rather a need to continue with current and established practices. It reflects the old adage: *"why fix what's not broken"*. Overcoming such inertia is reported as critical to gaining support for new practices [Humphrey, 1998]. According to Humphrey [1998] even 'intelligent' practitioners will not engage in practices that *"logic, experience and even hard evidence suggests that they should"* [Humphrey, 1998]. Humphrey offers several reasons for this including [Humphrey, 1998]:

- Once practitioners learn to develop programs that work they also establish some basic personal practices
- These personal practices become ingrained the more practitioners use them
- Previous bad experience of new tools and techniques, does not make practitioners think that that new practices will improve their output

Humphrey [1998] suggests that practitioners display inertia because they are unwilling to give up learned habits and have previous negative experience of new practices. However, it is possible that practitioners also can actively resist the introduction of new practices due to other factors. For example, practitioners may resist new practices that they perceive as a threat to their autonomy [Moitra, 1998].

- Lack of evidence of benefits

Studies investigating critical success factors of SPI indicate that providing practitioners with evidence of the benefits of SPI is a good motivator for establishing buy-in to SPI [Humphrey, 1998].

On the other hand, software practitioners may display intransigence towards new and better practices which will not necessarily be improved by providing them with evidence [Humphrey, 1998]. Humphrey argues that practitioners will not use new practices even when there is clear evidence that these practices or methods help.

These conflicting views on the influence of evidence on practitioner motivators for SPI warrant further investigation.

- Imposition

There are also implementation implications to the success of SPI in companies. Corporate level SPI initiatives are seen as barriers to successful SPI. According to Hantos and Gisbert [2000] SPI programmes that are initiated from the corporate level often face barriers from software practitioners. This is because software practitioners resist initiatives that they perceive as imposed upon them [Hantos and Gisbert, 2000]. It is suggested that when SPI programmes are initiated from a corporate level, they are often not consultative and have not secured practitioner buy-in.

- Resource constraints

There are resource factors to SPI success. Instrumental to most SPI success is the dedication of time, funds, tools and personnel to the SPI effort. Most studies on critical success factors of SPI acknowledge this [El Emam et al., 1999]. Goldenson and Herbsleb [1995] cite resources as one of the two most important factors to SPI success. According to Pitterman [2000] dedicating resources to SPI was critical to Telcordia's SPI success.

However, little work has been done to examine the impact of the absence of all or a combination of these resource factors on the motivations of software practitioners. As indicated earlier, SPI is expensive and requires the allocation of a lot of resources. There are therefore bound to be companies that embark upon SPI that are unable to dedicate sufficient resources to SPI.

- Commercial pressures

Studies of SPI provide evidence to suggest that commercial pressures serve as barriers to SPI in companies [Zeibe, 1996]. Such pressures are often in the form of meeting project deadlines and getting the product to the market on time and within budget. For example, according to Zeibe [1996] pressures on project schedule mean that often product improvement practices like testing and reviews are not carried out. Such pressures are brought on by the need to gain favourable market positions. SPI can often become a casualty of these commercial pressures.

2.3 Motivation theories

Empirical studies on SPI confirm the importance of motivators to SPI success. Kaltio and Kinnula's [2000] study of deploying defined software processes found that the most important people factors were skills, motivation and time [Kaltio and Kinnula, 2000]. Implying that SPI has a higher chance of success in companies where practitioners experience high motivation for it. Therefore, having explored some of the barriers that can de-motivate software practitioners from supporting SPI, this research also explores the issues that can motivate them to support SPI.

In this section classic motivation theories are reviewed in relation to work that has been done in the area of motivating software engineers. The review identifies gaps in the research on motivating software practitioners for SPI. Such a review can be helpful to understanding how to effectively manage software practitioners' motivators for SPI.

Five motivation theories are reviewed. Even though more recent work has been done in this area, this research presents these five theories in their classic forms. Couger and Zawacki [1980] identify these five theories as the most relevant to understanding motivation.

There are two main types of motivation theories: process theories and content theories.

2.3.1 Process theories

Process theories describe motivation as a sequence of related activities. An overview of three process theories are presented: job characteristics theory [Hackman and Oldman, 1976], stimulus response theory [Skinner, 1976] and equity theory [Couger and Zawacki, 1980]

- Job Characteristics Theories (JCT)

The basic tenet of this theory is that the work itself is the most important motivator. This is because there are certain key characteristics present within a job that makes it motivational to practitioners. Job Characteristics Theory (JCT) refers to these key characteristics as the 'five core dimensions' defined as [Hackman and Oldman, 1976]:

- Skill variety
- Task identity
- Task significance
- Autonomy
- Feedback from the job

The JCT states that the presence of these five dimensions in a job should induce three psychological states in practitioners. These three psychological states are that [Hackman and Oldman, 1976]:

- Practitioners' activities feel meaningful to them
- They feel responsible for their actions
- They receive results from their actions.

As a result of these three psychological states, practitioners will [Hackman and Oldman, 1976]:

- Experience high internal motivation
- Produce high quality work
- Experience high satisfaction with their work
- Reduce their absenteeism.

This relation between job characteristics and personal work outcomes is illustrated in Figure 4.

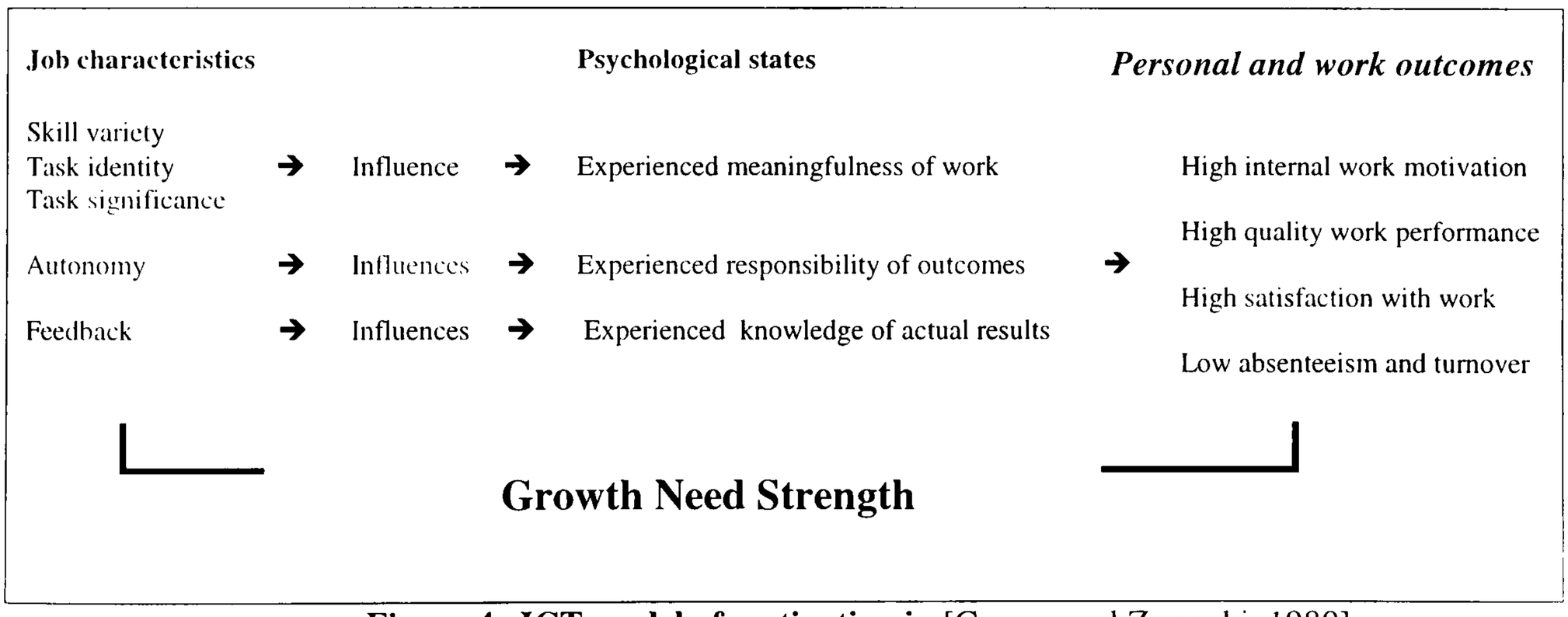


Figure 4: JCT model of motivation in [Couger and Zawacki, 1980]

However, the extent to which these five job dimensions motivate practitioners is dependent on their personal growth need strengths (GNS) [Hackman and Oldman, 1976]. A practitioner's GNS is defined by their need for personal growth and development. The theory is that practitioners with high GNS will respond better to high motivational potential in a job than those with low GNS [Couger and Zawacki, 1980].

The motivational potential in a job, is denoted by a single index referred to as the Motivational Potential Score (MPS). The MPS is calculated from the five core dimensions [Hackman and Oldman, 1976]:

$$MPS = [(Skill\ Variety + Task\ Identity + Task\ Significance)/3] \times [Autonomy] \times [Feedback]$$

Where each core characteristic is scored on a scale of 1 to 7 (where 7 is high and 1 low)

A practitioner's GNS is also depicted by a single measure on a scale of 1 to 7.

The basic tenet of the JCT is that practitioners will experience internal motivation and satisfaction if their GNS's are matched by the MPS of the jobs they do. This implies that practitioners with low GNS will be satisfied with low MPS in a job, in much the same way as practitioners with high GNS will need high MPS in a job. Optimum internal motivation and satisfaction is achieved when practitioners' GNS's are matched with the appropriate MPS's in a job.

The data collection tool developed from this concept, the Job Diagnostics Survey (JDS) [Hackman and Oldman, 1976] was later adapted for the software engineering industry by Couger and Zawacki and has since been used on several studies on the motivation of software practitioners. For example, [Couger et al., 1991; Couger and Ishikawa, 1995; Khalil et al., 1997]. This tool, the Job Diagnostic Survey for Data Processing personnel (JDS/DP) collects five sets of measurements [Couger and Zawacki, 1980]:

The Core Job Dimensions

- Skill variety
- Task identity
- Task significance
- Autonomy
- Feedback from the job
- Feedback from supervisors

- Goal clarity
- Goal difficulty
- Goal acceptance
- Goal setting participation
- Feedback on Goal accomplishment

Internal motivators

- Experienced meaningfulness
- Experienced responsibility
- Knowledge of results

Measures of satisfaction

- General satisfaction
- Satisfaction with co-worker
- Satisfaction with supervisors
- Satisfaction with pay

Growth Need Strength

Social Need Strength

There have been several studies in the last twenty years that test the JCT on software practitioners. For example, Couger and Zawacki conducted a series of studies involving over 2,500 software practitioners in the US that measured their GNS and the MPS of their jobs [Couger and Zawacki, 1980]. These studies generally found that the MPS of most software practitioners' jobs was higher than that in the normal populace and that software practitioners have relatively high GNS [Couger and Zawacki, 1980].

By exploring the JCT in this review, this research is trying to identify the characteristics of software practitioners' motivators for SPI. In so doing, this research will ascertain whether these motivators represent sufficient factors that contribute towards favourable

MPS of SPI jobs. This is because according to the JCT, if jobs have high MPS, then practitioners will be better motivated to do them. Therefore reviewing the JCT is part of the process of understanding how SPI can be more motivating to software practitioners.

- Stimulus Response Theory

Stimulus response theory is about the activities that modify behaviour [Skinner, 1976]. These activities are termed stimuli. According to stimulus-response theory there are two types of stimuli: punitive and rewarding stimuli. The theory explains that punitive stimuli are easier to apply and do have the effect of producing the required responses in the short term. However, rewarding stimuli, which are more difficult to apply and require more ingenuity to devise, tend to have a longer term effect in inducing the correct responses from subjects [Skinner, 1976].

Again, exploring the stimulus response theory enables this research to characterise the motivators for SPI cited by practitioners. Understanding the characteristics of these motivators in terms of the SRT improves the understanding of SPI motivators that are more likely to receive long term positive support from practitioners. For example, some of the SPI case studies suggest that companies use both punitive and rewarding stimuli to gain practitioners' support for SPI. Understanding the SRT provides a rich insight into the effectiveness of the motivators suggested by the literature.

- Equity theory

The equity theory is concerned with how to make employees feel 'equitably treated' in an organisation [Couger and Zawacki, 1980]. It computes a set of inputs and outputs that must be in balance to make employees feel 'equitable'. In a nutshell, this theory translates as: the inputs that people bring into an organisation, that is their experience, education, skills and seniority, should be matched by the outputs (that is what they get from the organisation), which are salary, recognition, opportunity for achievement *etc*.

According to the equity theory, practitioners are not necessarily satisfied by the balance between their own set of output and inputs, but will continue to compare that balance with that of other practitioners within their department, company or industry [Couger and Zawacki, 1980]. So that invariably, practitioners' sense of equity is derived from the relative rewards they receive for their jobs.

The equity theory is useful in this research to identify how the motivation of different staff groups for SPI can be influenced by their perception of what other staff groups receive for their role in SPI. For example, software developers' motivators may be influenced by the technical skills of their project managers. On the other hand, project manager's motivation may be influenced by the responsibilities allocated to developers within SPI. In this research, it will be important to understand how this concept of equity affects the motivators for SPI within and across the three staff groups.

2.3.2 Content theories

Content theories perceive motivation as "at a single point in time" [Couger and Zawacki, 1980]. This research provides an overview of two content theories: Need theory [Maslow, 1954] and Motivation-hygiene theory [Herzberg et al., 1959].

- Need theory

Maslow's hierarchy of needs roughly translates that people are motivated by different types of needs at different stages in their lives [Maslow, 1954]. Such needs manifest in a hierarchy where physical needs are at the bottom of the hierarchy and self-actualisation comes at the top (Figure 5). Maslow's theory suggests that people pursue these needs in a sequence so that, for example, peoples' social acceptance needs will not dominate them until most of their security needs are met [Mata Toledo and Unger, 1985]. This way, as people's needs are satisfied, new ones emerge to motivate their behavior.

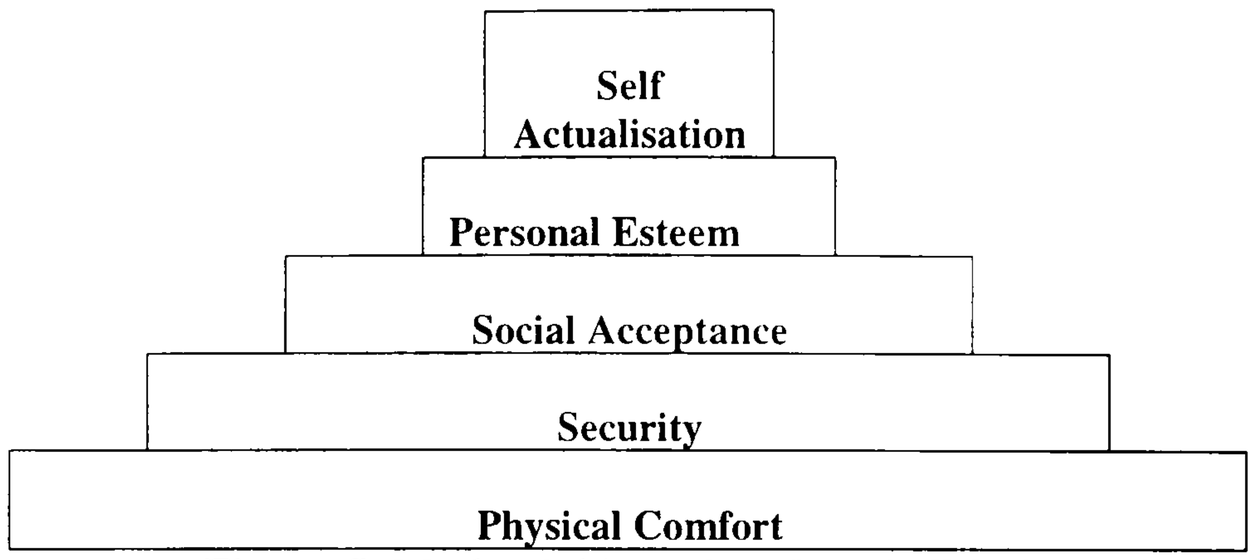


Figure 5: Maslow's hierarchy of needs [Maslow, 1954]

Studies done on software practitioners' motivators using Maslow's hierarchy of needs theory explain the reasons for how software practitioners rate aspects of their work. For example [Mata Toledo and Unger, 1985] suggest that software practitioners' essential needs are mainly personal esteem and self-actualisation because the high salaries of the software industry tends to satisfy their basic needs. This indicates that factors like pay and job security are bound to be less motivating for practitioners.

A review of Maslow's needs theory provides understanding of software practitioner's need for growth. Such understanding is important in identifying the kind of motivators that are applicable to software practitioners at various stages of their career. It can also explain differences in what motivates different hierarchical groups of software practitioners. Most importantly, this understanding can help to identify and explain why certain motivators may be more effective in gaining software practitioner's support for SPI than others. For example, Maslow's hierarchy of needs theory can explain why reward schemes and process ownership may have varying impacts on practitioners' support for SPI.

- Motivation-Hygiene Theory

Herzberg's motivation hygiene theory classifies factors that motivate practitioners into two distinct sets: Extrinsic factors and intrinsic factors.

- Extrinsic factors are those that are external to the job that practitioners do. For example peer relationships, company policy and pay. Herzberg suggests that these factors are necessary in order to stop practitioners from feeling dissatisfied with their work, but do not on their own motivate practitioners internally. They just maintain practitioners in their jobs.
- Intrinsic factors, on the other hand, are the primary determinants of motivation and satisfaction. These are the factors that are directly intrinsic to the work practitioners do, for example the job itself, responsibility, recognition and achievement. These factors motivate practitioners, internally, in their jobs.

All together, Herzberg suggests sixteen factors that are most important to practitioners' motivation. Ten are extrinsic and six intrinsic. Table 1 provides a summary of these factors. In a study to test this theory, Herzberg found that practitioners generally ranked the intrinsic motivators higher than the extrinsic factors.

Extrinsic factors	**Intrinsic factors**
Pay	Achievement
Interpersonal relations, subordinate	Recognition
Status	The work itself
Interpersonal relations with superior	Responsibility
Interpersonal relation with peers	Possibility of growth
Technical supervision	Advancement
Company policy and administration	
Work conditions	
Personal life	
Job security	

Table 1: Herzberg's extrinsic and intrinsic factors in:[Couger and Zawacki, 1980]

The essence of Herzberg's theory is that to motivate practitioners to produce over and above what they normally do, then there should be sufficient intrinsic factors in their jobs. The extrinsic conditions in a job, alone, will not do that.

Since the late 1970s many studies have tested the motivation hygiene theory on various aspects of software practitioners. For example, in 1977 Fitz-Enz surveyed the opinion of over 1,500 US software practitioners, asking them to rank sixteen motivation factors about their work in order of importance [Fitz-Enz, 1978]. Couger replicated the same study ten years later [Couger, 1988]. Both studies confirmed that, overall, software practitioners found factors that are intrinsic to the jobs practitioners do, more motivating.

There are also more recent studies that test motivation-hygiene theory with software practitioners. For example, [de-Souza, 1998] investigates the extrinsic and intrinsic motivators for using microcomputers in the work place. Findings from this study confirm that managers are motivated by both extrinsic factors (e.g. ease of use) and intrinsic factors (e.g. enjoyment) to use information technology in the work place.

Herzberg's motivation-hygiene theory is especially important to this research because it helps to understand and identify the group of motivators that can actually satisfy software practitioners. According to Brooks Jr. [1995] these are the motivators that the software engineering industry has found difficult to implement properly. This research suggests that if practitioners are to be motivated to support SPI long term without being constantly 'encouraged' to do so then these motivators – those that are intrinsic to the jobs that practitioners do – ought to be paid more attention.

2.3.3 Summary of motivation theories

The five theories discussed describe different characteristics of motivators. However, they do not describe mutually exclusive concepts, so that, for example, motivators can be intrinsic, as well as rewarding.

Also, the two basic categories of motivation: content and process theories are not mutually exclusive. It is widely recognised that the JCT - a process theory - is made up of two content theories (Herzberg and Maslow). So that the motivation potential described in the JCT model are the intrinsic factors in Herzberg's motivation-hygiene theory. On the other hand, the factors that militate against the outcome of a job's motivation potential on an individual is the individual's own GNS. That is the individuals' need to advance themselves. Maslow's hierarchy of needs theory determines the extent of an individual's growth need at any given point in time. Furthermore, the early steps of Maslow's needs theory can be shown to be extrinsic motivators whereas the latter steps are intrinsic motivators.

Nevertheless, the JCT seems to be the most comprehensively tested of the five theories described here not only because it is the most recent of the theories to be developed, but also because it provides quantifiable measures by which motivation within jobs can be assessed.

Several of the case studies on motivation done in the preceding twenty years on software engineers have tested the JCT, the Motivation Hygiene theory and Hierarchy of Needs theory in one form or another. As a result there is a substantial body of evidence to validate the above theories in the context of software engineering. In the following section, this research discusses these findings, and how they inform research into motivators for SPI.

2.4 Motivating software engineers

This section presents a review of the key studies done on motivating software practitioners and discusses the major motivators identified in these studies. It also indicates no studies have been done which directly examine the motivators and de-motivators of software practitioners for SPI.

2.4.1 Case studies on motivators of software engineers

Since the late 1970s many studies have been conducted to examine the factors that motivate software engineers. Studies on motivators have tested different motivation theories on groups of software practitioners in a variety of environments. A review of these studies indicates that factors that motivate software practitioners vary only marginally with respect to social or political environments. However, significant differences exist between what motivates software practitioners and what motivates other practitioner groups. The following are some of the key motivating factors identified in studies about software practitioners.

- The work itself

Many studies done on software practitioners' motivators show that software practitioners are motivated most by the nature of the job itself. Findings from these studies show that because software practitioners have a higher than average need for growth, they are motivated by challenging jobs that provide them with this opportunity to grow.

In studies that used Herzberg's instrument for ranking motivators, findings show that software practitioners from several different socio/political backgrounds in different time periods ranked the work itself as one of the most important motivators in a list of up to fifteen possible motivators [Fitz-Enz, 1978; Couger, 1988; Couger and Adelsberger, 1988; Couger et al., 1991; Couger and O'Callaghan, 1994]. In many of these studies, software practitioners' growth need strength was also assessed to be higher than any other practitioner group. In fact [Couger and Zawacki, 1980] established that out of about 500 different professions in the US, software practitioners had the highest growth need strength. This high need for growth therefore means that software practitioners are only truly motivated by factors that are intrinsic to the job that they do [Fitz-Enz, 1978; Mata Toledo and Unger, 1985; Couger, 1988].

- Opportunity for achievement

When practitioners cite achievement as a motivator, they refer to the factors that determine successful conclusion of a job, being able to provide solutions to problems and also being able to see the results of a job [Herzberg et al., 1959]. Achievement can therefore be seen as part of the characteristics of a job that provides practitioners with knowledge of the results of their work activities, i.e. feedback. Mata Toledo [1985] suggests that software practitioners must see their work as important, challenging and must receive high feedback on performance in order to make them highly motivated. When practitioners do not receive feedback for their jobs it reduces their perception of their need for excellence and their jobs become less meaningful to them which in turn makes the job less challenging [Mata Toledo and Unger, 1985].

In the early studies using the motivation-hygiene theory on software practitioners, achievement was the highest motivator [Fitz-Enz, 1978]. In fact Herzberg's original study identified achievement as the top motivator too [Herzberg et al., 1959]. However, lately, this motivator has become less important than the work itself even though it still remains amongst five motivators for all software practitioners.

Like the work itself, achievement is important to practitioners with high growth needs because it serves as an indicator by which they can assess their growth.

- Opportunity for advancement and growth

The opportunity for advancement and growth refers to an increase in status and the progress that practitioners make in their companies [Herzberg et al., 1959]. It also refers to an individual's personal growth and development [Hackman and Oldman, 1976].

Over the years, software practitioners' high need for advancement and growth has remained unchanged. In studies done in the last decade, in several countries, across at least three continents, findings point to the same high growth need. So that Egyptian software practitioners have high growth needs [Khalil et al., 1997] in much the same way as software practitioners in Japan [Couger and Ishikawa, 1995], the US, Spain and Finland [Couger and O'Callaghan, 1994]. The general consensus is that the computer industry attracts people with a high need for growth and achievement thus any mismatches that occur between software practitioners' growth need and the motivating potential of their jobs is likely to be because their jobs are not sufficiently challenging for them [Couger et al., 1991].

- Pay and benefits

Pay and benefits are classic examples of extrinsic motivators as described by the motivation-hygienists. There needs to be sufficient amounts in order to stop practitioners from being dissatisfied with their work. However, on their own, pay and benefits do not

provide the internal satisfaction needed to motivate practitioners to work better for a sustainable period of time. In particular, pay and benefits are not good for software practitioners because a raise in pay may indicate to software practitioners that they are being overworked [Mata Toledo and Unger, 1985]. Also, as people, generally, get older pay motivates them less. According to Maslow's hierarchy of needs theory, because the software profession is relatively well paid and prestigious, software practitioners achieve their fundamental needs early. Thereafter software practitioners tend to turn their attention very quickly to achieving personal growth [Mata Toledo and Unger, 1985]. Which implies that pay may only motivate practitioners in the short term. As a result, long-term motivation must be derived from more intrinsic factors.

Early studies of software practitioners' motivators support this notion that software practitioners consider pay and benefits as less important than intrinsic factors like recognition and even less important than other extrinsic factors like technical supervision and interpersonal relations [Fitz-Enz, 1978]. However later studies reveal that pay has become more important to software practitioners as a motivator. Software practitioners in the US and in Europe consistently rank pay amongst the top four motivators, ahead of achievement [Couger and O'Callaghan, 1994]. Couger suggested that a change in economic circumstances was a possible reason for the change in importance of pay and benefits [Couger, 1988]. Couger also asserts that motivators that are not being adequately met may be ranked higher than those that are being met. In fact this assertion can be supported by how software practitioners rate their pay satisfaction levels. For example, in [Couger and O'Callaghan, 1994], pay satisfaction levels in the US, Finland and Spain are relatively lower than general satisfaction ratings and also lower than rating for motivation potential of jobs. The issue of dissatisfaction with pay is not only confined to the west. Egyptian software practitioners are even less satisfied with pay than their counterparts in the west [Khalil et al., 1997]. Khalil et al suggests that pay plays a greater significance in developing environments like Egypt than it would in the west. Studies done in the UK also show that software quality practitioners are more concerned by pay than they used to be [Nicholson et al., 1995; Warden and Nicholson, 1995].

- Recognition

Recognition as a motivator refers to being noticed, praised (or possibly blamed) for doing something [Herzberg et al., 1959]. Practitioners receive recognition from superiors, peers or subordinates. Recognition can be assessed from the level of feedback received from jobs practitioners do. So that software practitioners may not feel sufficiently recognised if they do not receive sufficient feedback on their work. In [Warden and Nicholson, 1995] software quality practitioners complain that they do not feel that the work they do is valued by their companies. This research suggests that one of the underlying reasons for this perception is that these software quality practitioners receive very little feedback from their work.

Earlier studies on software practitioners' motivators showed that recognition was one of the most important motivators for software practitioners [Fitz-Enz, 1978]. However, recent studies have shown that recognition has become less important [Couger and O'Callaghan, 1994; Couger and Ishikawa, 1995]. Couger indicates that there are temporal influences in the way software practitioners rank motivators. So that motivation factors that are not being satisfied over the years tend to be higher ranked and vice versa [Couger, 1988]. Consequently software practitioners may be ranking recognition lower in studies over the years because they may be enjoying better recognition for their work.

- Increased responsibility

Unlike recognition, responsibility has never been highly ranked by software practitioners. Software practitioners find it more important than most of the extrinsic motivators, but usually the least important amongst the intrinsic motivators. Couger's concept of the temporal effects on rankings does not apply to responsibility. However, responsibility may be more affected by social environments than any other factor. In 1988, Couger and Adelsberger [1988] conducted a study to compare the effect of socio/political environments on software practitioners motivations. Overall, this study revealed that US and Austrian software practitioners were similar in their ranking of motivators but the

biggest variation in ranking occurred in their ranking of responsibility, where Austrian software practitioners ranked it far less important [Couger and Adelsberger, 1988]. In a similar study in 1994 involving practitioners from three socio/political environments. Finnish software practitioners perceived responsibility as much less important than practitioners from Spain and the US [Couger and O'Callaghan, 1994].

Overall, the literature suggests that even though responsibility may be intrinsic to practitioners' motivation, software practitioners do not perceive it as very important.

- Technical supervision

Earlier studies on software practitioners' motivators shows that technical supervision is mildly important to software practitioners. In fact technical supervision is the only extrinsic motivator higher than any intrinsic motivators [Fitz-Enz, 1978]. Over the years, technical supervision has become less important but not significantly so.

Egyptian software practitioners in particular are dissatisfied by the level of supervision they receive [Khalil et al., 1997]. They identified the cause of poor supervision as a result of inadequate feedback received from supervisors. Couger et al suggest that because software practitioners have low social needs strength, they are poor at activities that requires them to interact with other practitioners to provide feedback [Couger et al., 1991]. Hence poor supervision is attributable to poor social needs of software practitioners.

Couger and O'Callaghan suggest that it is possible to improve the interactive skills of software practitioners if it is made clear to software practitioners that interactive skills are necessary for better performance of their jobs. This is because software practitioners' high growth need will motivate them to learn the necessary behavioral skills needed to achieve growth, and in so doing overcome their low social needs [Couger and O'Callaghan, 1994].

Also, Warden and Nicholson [1995] showed that when software quality practitioners found their jobs were lacking in feedback from their supervisors it made it difficult for them to experience sufficient knowledge of the results of their work. This situation impacted negatively on their satisfaction levels. The literature suggests that technical supervision can be useful to software practitioners in more than one way. It can also provide them with feedback of their progress.

- Interpersonal relations

Studies done on software practitioners' motivators indicate that software practitioners do not rate interpersonal relations highly. This is because, as discussed earlier, software practitioners have notoriously low needs to socialise with others. In many studies on motivation, software practitioners social need strength has been measured as the lowest of most professions irrespective of the socio/political context [Couger and Zawacki, 1980; Couger and O'Callaghan, 1994; Couger and Ishikawa, 1995; Nicholson et al., 1995; Khalil et al., 1997]. In these same studies, software practitioners have consistently ranked interpersonal relations in the lower third of important motivators. For example in [Couger and O'Callaghan, 1994] interpersonal relations is ranked 10^{th} in importance out of eleven motivators. These are strong indications that software practitioners do not rate interpersonal relations highly.

Couger and O'Callaghan [1994] suggest that this low need for social interactions is detrimental in an industry where there is a need for strong social skills in order to effectively liaise with clients and also work effectively in groups. The authors recommend that software practitioners' low social need strength can be improved, indirectly, by appealing to their high growth need as described above.

As many SPI practices recommend some degree of interaction with peers, for example, the inspection and review process, it is imperative in this research to explore whether interpersonal relations can motivate software practitioners to support SPI.

- Job security

Software practitioners do not perceive job security as very motivating. In studies on motivation from the late 1970's to date, starting with [Fitz-Enz, 1978], software practitioners have consistently ranked job security in the lower third of a list of fifteen factors. The notable exception is [Couger and O'Callaghan, 1994] where Spanish practitioners ranked job security higher than intrinsic factors like opportunity for achievement and increased responsibility. Apart from this particular finding, software practitioners have generally not considered job security an important motivator.

However, more recently, dedicated software quality improvement positions in companies have come under some pressure. In [Warden and Nicholson, 1995] software quality practitioners indicate that they are concerned about job security. Their perception is that management will shift their commitment from SPI if resources become constrained. This research suggests that if software quality improvement positions are under pressure, it can have a detrimental effect on software practitioners' motivation to support quality improvement programmes in the first place. On the other hand if quality improvement positions are perceived to be secure by software practitioners, then it is likely that they will support them. It is therefore important to investigate how the whole issue of job security motivates practitioners to support SPI.

- Work conditions

Work conditions refer to the physical conditions of the work, the quality of work and the tools available to carry out the work [Herzberg et al., 1959].

Defined as an extrinsic factor, work conditions do not necessarily motivate software practitioners to work better, however they need to be adequate to prevent practitioners from being dissatisfied at work. Most case studies on software practitioners' motivators indicate that software practitioners rank work conditions as one of the least important of their motivators [Fitz-Enz, 1978; Couger and O'Callaghan, 1994].

There have been technology advances in the software industry over the years. With such increasing change in technology, especially over the last decade, it will be important to investigate the role that work conditions play on software practitioners' motivators generally.

- Company policy and administration

Like work conditions, company policies are one of the least important motivators identified by software practitioners. This research suggests that company policy and administration are very similar to SPI in that they are a series of processes and procedures that detail how companies function. It is possible to speculate that how motivated software practitioners are by company policies may indicate the extent to which they will react to SPI generally.

- Senior management support

In a study on what motivates software quality practitioners, Warden and Nicholson [1995] found that software quality practitioners had major problems with motivation. They experienced low satisfaction levels with their jobs. One of the reasons that software quality practitioners gave for this low satisfaction was that they thought senior management did not support their improvement efforts.

Senior management support has been identified as critical to the success of several SPI projects. There are several accounts of the importance of senior management support for SPI success. In [Mellis, 1998] senior management support is shown to be the most important factor to SPI success. [Willis et al., 1998] state that senior management's strong support in pursuing SPI in Hughes helped to stimulate support for SPI throughout the organisation. Diaz and Sligo [1997] show that senior management support was critical to Motorola's SPI success. Ahuja [1999] and Pitterman [2000] indicate that senior management support was vital to gaining acceptance for Telcordia's SPI programme.

2.4.2 Motivating software engineers for SPI

Despite the many studies investigating the motivation of software practitioners discussed above, there have been no empirical studies that have actually looked at the motivation of practitioners for SPI. Few studies have concentrated on the motivation of software practitioners in quality improvement initiatives, generally. These studies have replicated research methods that have been tried and tested in previous general studies on software practitioners' motivators. For example, [Nicholson et al., 1995; Warden and Nicholson, 1995], is one of the few studies to directly report on the motivation of software practitioners for quality improvement initiatives and they used the Job Diagnostic Survey for DP personnel JDS/DP.

In 1995, Nicholson et al surveyed the motivators of software quality practitioners in forty-five UK companies [Nicholson et al., 1995; Warden and Nicholson, 1995]. The findings of this study indicated that software quality practitioners experienced 'serious' motivational problems. Nicholson et al reported alarm at the state of these motivational problems. As agents for change, software quality practitioners' motivation for quality improvement programmes is fundamental to the success of such programmes [Nicholson et al., 1995]. Also as custodians of improvement initiatives, it is important that quality practitioners themselves are motivated. These findings raise concern for the state of quality improvement initiatives in UK companies.

Also, this survey found that software quality practitioners were less motivated than other software practitioners in their companies [Warden and Nicholson, 1995]. This is a surprising finding in view of the generally accepted fact that software practitioners have a high growth need and tend to be highly motivated by the challenges of their work. However, the study indicated that software quality practitioners' high growth need was being poorly matched by the appropriate motivational potential of their jobs. The study found that the motivation potential strength of the jobs that quality practitioners did were either too high or too low and failed to achieve optimum motivation in practitioners

[Warden and Nicholson, 1995]. This therefore accounted for the low motivation experienced by these quality practitioners.

Warden and Nicholson suggest that some of the quality improvement roles that practitioners found too demanding were probably new roles that had evolved out of recent quality management systems. Warden and Nicholson argue that because these roles are new, they may not have been sufficiently tested to establish how conducive they are to the softer aspects of software management. This is in contrast with traditional software engineering practices, which, as Brooks Jr. [1995] suggests have reduced, over the years, the environmental factors that may impede software practitioners' ability to work properly [Brooks Jr., 1995]. For example, basic motivational factors like technical supervision and work conditions may have been overlooked. Are quality practitioners being provided the appropriate tools for their work?

A similar argument can be made as to why software practitioners may find their jobs too mundane or undemanding. Are quality improvement jobs less motivating than other software developing jobs? If, as Warden and Nicholson [1995] report, software quality practitioners are failing to attain high levels of satisfaction in their jobs, then what are the factors that can make these jobs more satisfying and in effect more motivating to software practitioners?

Overall, Warden and Nicholson suggest that the problems with motivation amongst software quality practitioners can be related to the fact that the quality programmes they work on have not been sufficiently tailored to take care of certain human factors. They recommend a comprehensive examination of the factors that affect practitioners motivation for quality improvement practices.

This review shows that sufficient empirical studies have not been done in the area of the motivators and de-motivators for software quality improvement practices. This indicates a need for such a study to elicit from practitioners themselves what they identify as motivating and de-motivating to quality improvement, generally and SPI in particular.

2.4.3 Gap in the research

Findings from [Nicholson et al., 1995; Warden and Nicholson, 1995], hitherto the most comprehensive study on software practitioners' motivation for quality improvement initiatives, confirms some of the previous findings on the motivation of software practitioners generally. For example it confirms that software practitioners have high growth needs. However this study also presents findings which raise concern about software practitioners' motivators. The study shows that software quality practitioners do not feel that their jobs are valued, are worried by job security and are de-motivated by pay. These are specific concerns about software quality practitioners' motivators which previous studies have not identified.

This research suggests that there may be other explicit factors relating to software practitioners' motivators and de-motivators for SPI that may not have been highlighted by previous studies on motivation. This is because:

- No studies have specifically investigated software practitioners' motivators for SPI.
- Studies that have addressed software practitioners' motivators for other quality management approaches have replicated research methods which measure or rank preset motivating factors. These studies do not make it possible to uncover new factors that may be peculiar for SPI.

This research suggests that to understand software practitioners' motivators and de-motivators in order to improve support for SPI, it is necessary to elicit such factors from software practitioners without the use of a pre-emptive data collection instruments, for example, questionnaires that test a set of pre-determined motivation attributes. This approach to the research, however, has the advantage of uncovering issues which may previously have been overlooked by other tried and tested models like the JDS/DP [Hackman and Oldman, 1976] and Herzberg's sixteen factor motivation-hygiene ranking instrument [Herzberg et al., 1959]. Also, such an approach provides the basis for independently verifying the factors that have been established as important to software

practitioners' motivators. This research suggests that it helps to determine, for example, if the same factors identified in software practitioners' motivators apply to their motivators for SPI, too. For example, are software practitioners motivated by recognition to support SPI?

2.5 Staff group perspective of motivators and de-motivators for SPI

Few studies that examine the motivators and de-motivators of software practitioners do so from the perspective of hierarchical staff groups. The series of studies that have looked at software practitioners' motivation from particular staff groups have reported that particular groups have differences in the way they perceive their motivators [Couger and O'Callaghan, 1994; Khalil et al., 1997]. For example, Khalil et al report that even though software practitioners generally have a low inclination to social interaction, there are variations in their social needs across staff levels so that managers and operations staff have higher social needs strength than programmer/analysts [Khalil et al., 1997]. This suggests that since social need strength has been shown to correlate positively to feedback [Couger and O'Callaghan, 1994], programmer/analysts are the least likely to either provide or respond positively to feedback. Such findings hint at differences in software practitioners' responses to key motivation issues.

Indeed, Hall and Wilson [1997] found that different staff groups of software practitioners have different experiences and perceptions of SPI and that these differences can impact differently on how practitioners behave towards SPI. This implies that different staff groups of software practitioners can have different motivators and de-motivators for SPI which will impact differently on their support for SPI.

Presently, there are no empirical studies that analyse software practitioners' motivators and de-motivators for SPI from the perspective of different staff levels.

2.6 Software practitioners' perception of their role in SPI

In order to properly address the differences in motivators and de-motivators for SPI, it is useful to understand the nature of the differences, that is, it is important to understand the different responses to SPI of the different staff groups. Personal construct theorists propose that to understand how individuals or small groups behave in or respond to a given situation, it is important to examine how they perceive themselves within that given situation [Kelly, 1970]. Kelly suggests that it is necessary to *"stand in others' shoes to see the world as they see it"* [Fransella and Bannister, 1977]. It can be similarly inferred that in order to appreciate the differences in software practitioners' motivators and de-motivators for SPI, it is useful to understand how different groups of software practitioners perceive their role in SPI programmes in relation to the role of other staff groups.

2.7 SPI managers' perceptions of SPI

SPI implementation requires competence in change management. Paulk et al [1994a] and Curtis in [ESEPG, 1999] suggest that companies that excel at SPI are those that are effective at change management, amongst other things. One of the critical success factors of change management is the vision and drive of change agents [McCalman and Paton, 1992]. If change agents are not committed to the programmes through a lack of vision or lack of motivation, then such programmes are bound to be less effective [McCalman and Paton, 1992]. Change agents can also be good indicators of how well programmes are doing in companies.

SPI managers are change agents in SPI programmes [Moitra, 1998]. This research suggests that as a result, it is possible to assess the state of SPI practice in companies by understanding the perception of SPI managers. No studies have actually examined SPI managers' perception of SPI. However, [Nicholson et al., 1995] and [Warden and Nicholson, 1995] investigated the perception of quality improvement personnel. Findings from these studies suggest that the agents of change themselves are de-motivated in their jobs. Nicholson and Warden suggest that such a finding presents a bleak prospect for quality improvement programmes in companies. This finding, by implication, presents a bleak picture of the prospects of SPI, also.

Three important reasons for understanding SPI managers' perception of SPI are:

- First, it serves as a check up on the motivation and experiences of people running SPI in companies.
- Secondly it provides a barometer for the state of SPI practice in companies
- It sets into context the state of software practitioners' motivators and de-motivators for SPI.

The SPI literature reveals certain fundamental issues about SPI. To understand the state of SPI practice in companies through the perception of SPI managers, it is necessary to understand their perception on the fundamental issues. These fundamental issues, some of which have already been discussed in previous sections, can be summarised into the inputs, implementors and outputs of SPI according to systems theory. The following is an overview of these issues.

- Experiences of SPI

Implementing quality programmes of any sort is about people [DeMarco and Lister, 1987]. Implementing SPI programmes, in particular, requires close attention to the experiences and expectations of the practitioners who manage and carry out the SPI practices. This research suggests that it is important to survey experiences of SPI managers to establish if it impacts on their opinions of SPI. Other studies have suggested that practitioner experiences forge attitudes and subsequent behaviour towards SPI [Humphrey, 1998].

- Expectations of SPI

It can be argued that practitioners' expectations of SPI also contribute towards their perceptions of SPI. Practitioner expectations of SPI may be formed by several factors, but most importantly, experiences of SPI play a significant part in forming such expectations. Humphrey explains that practitioners tend to have negative expectations of an

UNIVERSITY OF HERTFORDSHIRE LRC

improvement programme if their experiences of that programme are negative, too [Humphrey, 1998]. It is therefore useful in this research to understand the expectations of SPI managers for SPI to assess whether such expectations have been formed from their experiences.

- Product quality improvement

The fundamental ethos of SPI is improving software quality through focussing on the development processes [Deming, 1986; Humphrey, 1989], as discussed in Section 2.2.2. Which means that SPI success can be judged through the quality improvement achieved in the product. Although some recent empirical studies on the impact of SPI report substantial quality improvements, for example in [Paulk et al., 1994b; Krasner, 1997; Ahuja, 1999; Fitzgerald and O'Kane, 1999], an overwhelming body of evidence has not yet been established to support this claim. SPI managers' perception of the impact of SPI on product improvement can offer some insight into the extent of product improvement.

- Cost-effectiveness

Crosby's assertion that *"quality is free"* [Crosby, 1979] has often been quoted by companies embarking on SPI programmes to justify the substantial start–up costs of such programmes. Indeed, high profile SPI success stories like Hughes Aircraft Company [Willis et al., 1998] and Raytheon [Haley, 1996] report long term cost gains. Some companies have reported substantial increases in the Return On Investment (ROI) of between 5:1 to 9:1 [Krasner, 1997]. Overall, reports from companies practising SPI in the last ten years indicate long term sustainable cost benefits [Herbsleb et al., 1994; Haley, 1996; Fox and Frakes, 1997; Willis et al., 1998; Krasner, 1999].

However, in the UK there is a lack of independent evidence, anecdotal or otherwise, to either support or counter published accounts of the cost effectiveness of SPI as in, for example [Krasner, 1997], where companies like Hewlett Packard report *"savings of*

$20Million in one financial year alone due to inspections". SPI managers' perception can provide some evidence of cost benefits in UK companies.

- Senior management support

Many case studies consider senior management support critical to the success of SPI programmes [Krasner, 1999; Wilson et al., 2001]. Telcordia's SPI success reports visible senior management support as critical to this success [Pitterman, 2000]. Laporte and Trudel [1998] suggest that by showing understanding and a full commitment to process issues, and displaying this through their day to day activities, senior managers are sending a 'positive' signal to middle and lower ranked practitioners about their commitment to SPI. Such a signal is instrumental in gaining the buy-in of other practitioner groups for SPI [Laporte and Trudel, 1998].

However, with the exception of the handful of successful case studies, there are few independent accounts reporting on the level of senior management support, generally, for SPI programmes. This research suggests that understanding SPI managers' perception on this issue can provide some indication of the level of senior management support for SPI in practice.

- Developer buy-in

Successful SPI accounts like [Paulk et al., 1994b; Goldenson and Herbsleb, 1995; Herbsleb and Goldenson, 1996; Krasner, 1997; Willis et al., 1998; Paulk, 1999; Paulk et al., 2000] suggest that to achieve high maturity in development processes, it is necessary to transfer ownership of such processes to the people who actually perform the functions. This indicates that it is vital to let the practitioners who conduct the improvement effort to have ownership of those processes. However practitioners must buy-in to SPI first before this stage can be reached. The successful case studies report that buy-in can be achieved through consultation with practitioners where their views are encouraged and incorporated into company-specific improvement initiatives.

Understanding SPI managers' perceptions of developer buy-in in companies can provide a strong indication of the level of potential grassroots support for SPI in companies.

- Implementation approaches

Recent accounts of SPI programmes in companies question the merits of the classical management approaches to implementing SPI [Hammock, 1999; Paulk, 1999]. In fact there are increasing calls to software engineering managers to move away from top-down and bottom-up approaches of implementing programmes to more adaptive forms of implementation. For example, inside-out or *"growing programmes in situ"* [Hovenden et al., 1996]. There is also an increasingly popular assertion that whole improvement initiatives should be run from within, not fostered by external agencies [Krasner, 1997]. SPI managers' perception of the favoured implementation approaches can provide insight into how SPI is being implemented in practice.

2.8 Outputs of literature review

This review of the literature identifies four main themes that underpin software practitioners' motivation for SPI. The review also suggests that various guidelines can be distilled from the disparate literature which are focused on improving software practitioners' motivators and de-motivators for SPI. This section provides a summary of these two outputs.

2.8.1 Four themes underpinning software practitioners' support for SPI

The following four themes have been identified from the literature as underpinning software practitioners' motivation for SPI

i. SPI change agents

Paul et al [1994a] and Curtis in [ESEPG, 1999] suggest that companies that excel at SPI are those that are effective at change management. One of the critical success factors of change management is the vision and drive of change agents [McCalman and Paton, 1992]. SPI managers serve as the change agents in SPI programmes. Therefore their perception of the factors that influence software practitioner's support for SPI is worth assessing in order to understand the state of software practitioners' support for SPI in companies.

ii. SPI motivators

One of the most important factors to deploying defined software processes is motivation [Kaltio and Kinnula, 2000]. According to Kaltio and Kinnula [2000] SPI has a higher chance of success in companies where practitioners experience high motivation for it. Humphrey [1995] suggests that software practitioners will support well-defined processes if the problems with motivation are identified and solved. Understanding what motivates software practitioners for SPI can show how to increase their support for SPI.

iii. SPI de-motivators

A variety of reasons inhibit software practitioners' support for SPI. According to Moitra [1998] one such factor is the unwillingness of practitioners who actually use the practices to take them up. Software practitioners' experiences of SPI can also de-motivate them from supporting SPI [Humphrey, 1998]. Understanding what de-motivates software practitioners for SPI can show how to increase their support for SPI.

iv. Different SPI motivators and de-motivators for different groups of SPI

Hall and Wilson [1997] suggest differences in software practitioners' group attitudes to SPI. These differences impact negatively on their responses to SPI. There are also differences in the motivation of different staff groups of software practitioners [Couger and O'Callaghan, 1994; Khalil et al., 1997]. Implying that these differences may exist in

their motivations for SPI also. Personal Construct Theory (PCT) [Kelly 1970] suggests that the differences in the way people react to situations are related to the way they perceive themselves in those situations. The RGT [Fransella and Bannister, 1977] helps to understand how people perceive themselves in particular situations. Using the RGT to explore the perceptions of software practitioners for SPI can help to explain the nature of their different motivators and de-motivators. Such an understanding can help to properly address the staff-group specific differences in these motivators and de-motivators for SPI.

2.8.2 Guidelines for increasing software practitioners' support for SPI

The following guidelines are recommended by the literature for improving software practitioners' motivators and de-motivators for SPI. A summary of the guidelines is provided in Appendix A.

I. Senior management commitment

There are several accounts of the importance of senior management support for SPI success. In an analysis of SPI success factors involving 56 case studies and reports [Mellis, 1998] shows that senior management support is the most important factor to SPI success.

Describing the SPI experience of Hughes Aircraft Company (now merged with Raytheon), [Willis et al., 1998] state that senior management's strong support in pursuing SPI in more of Hughes companies, helped to stimulate support for SPI throughout the Hughes organisation. Senior managers in Hughes manifested this strong support by linking practitioners' incentives to organisational goals and also by providing resources for SPI such as, people dedicated to SPI tasks, funding and tools.

[Diaz and Sligo, 1997] show that senior management support was critical to Motorola's SPI success. In this programme, senior management success was demonstrated through the provision of funds and resources for SPI activities and by rewarding practitioners for their SPI effort.

Ahuja [1999] and Pitterman [2000] indicate that senior management support was vital to gaining acceptance for Telcordia's SPI programme. Senior management's visible commitment and support for SPI made SPI acceptable for other practitioner groups too [Ahuja, 1999; Pitterman, 2000].

Other accounts of senior management support and commitment are reported in Herbsleb's study of the effect of SPI efforts in five companies [Herbsleb et al., 1994]. Senior management commitment was found to be critical to SPI success in three of these companies. In one of these companies, Oklahoma City Air Logistics Centre, management involvement was reported to motivate practitioners because it demonstrated to them that their efforts were valued [Herbsleb et al., 1994].

In Dyba's formation of a prescription for successful SPI, a panel of experts and academics agree upon six key facilitating factors of SPI success, one of which is senior management involvement [Dyba, 2000].

II. Practitioner buy-in

Gaining practitioners' involvement in SPI is another of the six factors identified as critical to SPI success in [Dyba, 2000]. Also, Mellis and Stelzer [1998] report staff involvement as the second most important factor to SPI success

Lessons learnt from the deployment of SPI by Bull HN Information Systems Inc. indicate that it is not only sufficient to receive senior management support for SPI, but it is also important to secure middle management involvement and the buy-in of the technical community [Herbsleb et al., 1994]. Also, the experience of Schlumberger shows that support of middle management is crucial because they are often expected to make provision for SPI whilst meeting project deadlines. This can make them hostile to SPI. So that by working with middle managers from the beginning, their realistic expectations of time and resources are reflected into the estimates that are given to senior managers. This

process makes them less hostile to SPI, as reported in Schlumberger [Herbsleb et al., 1994].

Motorola's account of SPI shows that management commitment is needed from all levels of the company. In addition to senior managers, project managers and other managers too need to be committed to SPI [Diaz and Sligo, 1997].

In effect, securing the buy-in of all practitioners for SPI is critical to its success.

III. Process ownership

Process ownership is widely reported as instrumental in empowering practitioners and motivating them to support SPI in ways that few of the other success factors do. One of the most successful process maturity stories, the Space Shuttle Onboard Software Project, used process ownership as a key factor to improving process and software quality. In the 1990's process ownership teams were set up in nine key areas, from requirements evaluation to control boards [Krasner et al., 1994]. These teams were charged with studying, improving and providing education on these nine key processes. The knowledge and subsequent improvements made on processes were disseminated throughout the project and organisation. Overall, process teams have helped to speed up the maturity of the project's processes [Paulk et al., 1994b], [Billings et al., 1994].

Krasner's account of the SPI success of Raytheon Equipment Division describes how practitioners are empowered by the ownership of processes because SPI was run from within the ranks of the company. This is similar to the case of the Space Shuttle Onboard Software Project.

In Motorola's Cellular infrastructure division in Cork, management hold the view that *"process ownership and development are best placed with those closest to the processes"* [Fitzgerald and O'Kane, 1999]

Haley [1996] describes one of the factors of Raytheon's SPI success as the ownership of processes and products felt by task managers and line engineers - the people closest to the processes - because they performed majority of the work [Haley, 1996].

IV. Evidence / Visible success

The effectiveness of using evidence of SPI benefits and success to overcome resistance to SPI has been reported in many successful SPI accounts. The Space Shuttle Onboard Software Project provided evidence of SPI benefits to overcome initial resistance to the improvement programme [Billings et al., 1994; Krasner et al., 1994; Paulk et al., 1994b].

BMW used evidence of the benefits of the *ASCET* method of code development - a seamless conversion of abstract code to production code - to overcome initial resistance to its introduction. BMW simulated a baseline project and run it alongside the original project [VASIE, 1999]. Project leaders then compared and contrasted the two separate projects. The result of this exercise was that BMW were able to win apathetic practitioners over by showing them the benefits from the simulation.

Krasner presents three main reasons why evidence of benefits is crucial to SPI [Krasner, 1997]:

- External evidence of benefits is needed to justify embarking on SPI at a time when the internal SPI initiative is not old enough to produce such results.
- Internal evidence is needed to show how improvements within processes impact on other areas of the business.
- Evidence of benefits is needed to validate the SPI programme when it comes under threat of financial constraints

Above all, Krasner also identifies that it is important for SPI to demonstrate some early internal benefits to win practitioners over. This was the case in the SPI programme of Motorola India Electronics Ltd. [Krasner, 1997].

The concept of displaying early small benefits is evidenced in the SPI programme of Telcordia Technologies [Ahuja, 1999; Pitterman, 2000]. Managers in Telcordia were converted to using metrics when they were presented with the benefits of managing with metrics from a small project. In this case, a small project was chosen because it could provide early benefits.

Both [Haley, 1996] and [Krasner, 1997] mention the importance of visible success in the SPI effort of Raytheon Electronic Systems and Raytheon Equipment Division, respectively. These accounts indicate that it is imperative that visible benefits are continually demonstrated.

V. Training

Training appears as an important factor of SPI success. Case studies recount the importance of training all practitioner groups in SPI skills. Hughes Aircraft has a formal training programme that constitutes about 60 formal courses to train engineers on SPI. When Hughes adopted the Hatley-Pirbhai real time structured analysis methodology as its approach to requirements development, a course was set up to train systems and software engineers on the use of the methodology and the associated tools that were purchased for the methodology [Willis et al., 1998]. Hughes training programme is perceived as one of the assets of its SPI initiative. Hughes also emphasise the importance of providing SPI awareness training for senior managers. SPI awareness training for senior managers can make them overcome their inertia for SPI by making them aware of how SPI can make companies meet their business objectives [Willis et al., 1998]. Krasner implies that one such approach to SPI awareness can be by using the Cost of Software Quality (CoSQ) approach to make management aware of the cost effectiveness of SPI [Krasner, 1999].

At Xerox, it was acknowledged that providing training for all levels of practitioners was critical to SPI success [Fowler et al., 1999]:

"In most cases managers involved in SPI had limited exposure to SPI concepts and experience because of time constraints".

Fowler *et al* [1999] suggest that training of these managers helps both the SPI effort and themselves.

There are several other accounts of the critical importance of training to SPI success in [Billings et al., 1994; Krasner et al., 1994; Paulk et al., 1994b; Laporte and Trudel, 1998; Ahuja, 1999; Nolan, 1999; Pitterman, 2000].

VI. Mentoring

Paulk's assessment of high maturity organisations report that most high maturity organisation have, in addition to training programmes, mentoring schemes also [Paulk, 1999]. Paulk [1999] implies that these schemes are important to gain software practitioners' support for SPI. Indeed, the Space shuttle programme [Billings et al., 1994] reported that a mentoring scheme was instrumental in gaining software practitioners' support for SPI.

Goldenson and Herbsleb [1995] found that the absence of a mentoring scheme tended to inhibit success of SPI in companies. Hughes' highly successful SPI training programme included a mentoring scheme also [Willis et al., 1998].

VII. Standardisation

Providing a common platform for the practice of SPI is reported as important to SPI success in the SPI initiative of Oerlikon Aerospace, a defense contractor specialising in integrating laser guided systems [Laporte and Trudel, 1998]. Standardisation in Oerlikon took the form of providing a common vocabulary and vision for SPI.

VIII. Sharing best practice

In a review of 56 case studies and reports, sharing best practice is identified as one of ten critical success factors to change management in SPI [Stelzer and Mellis, 1998]. This is achieved through encouraging communication and collaboration within the organisation.

In Hughes, sharing best practice is implemented by having a change process where any member of the organisation can suggest improvement to processes. When the suggested changes have been validated, they are reflected in the corporate SPI programme thereby making them available to everyone in the company [Willis et al., 1998].

Hantos and Gisbert [2000] suggest that best practice can be shared within disciplines as well as across disciplines. They suggest 'cross pollination' where practitioners learn form practitioners of other fields within the company [Hantos and Gisbert, 2000].

IX. Prioritising

One of the important lessons learned from deploying SPI in Oklahoma City Air Logistics Centre is that SPI should be treated with equal importance as projects [Herbsleb et al., 1994].

X. Dedicated resources

The experience of Oklahoma City Air Logistics Centre indicates that is crucial to dedicate resources to SPI [Herbsleb et al., 1994]. Software practitioners must not be expected to do SPI in their own time.

In Motorola GED, senior managers were able to demonstrate their commitment to SPI by providing dedicated resources, in terms of funding and time to SPI activities [Diaz and Sligo, 1997].

XI. Relevant and realistic objectives

In [Stelzer and Mellis, 1998] it is suggested that it is important to set objectives for SPI which are both relevant and realistic. SPI objectives must reflect what companies are trying to achieve and be reasonably achievable.

Advanced Information Services showed that relevant objectives must reflect the aspiration of practitioners too[Ferguson et al., 1999]. In their SPI programme, employees were surveyed to elicit their views on the key process areas that should be concentrated upon [Ferguson et al., 1999]. These views were incorporated into the overall SPI programme resulting in improved practitioner support for SPI.

The experience of Oerlikon Aerospace is that companies should work towards aligning the quality views of everybody involved in the improvement programme towards the programme goals, which should not be dis-similar to the overall company goals [Laporte and Trudel, 1998].

XII. Vision through internal leadership

SPI experiences in Raytheon [Haley, 1996], Motorola GED [Diaz and Sligo, 1997], Telcordia [Ahuja, 1999; Pitterman, 2000] and Texas Instruments [Herbsleb et al., 1994] attest to the importance of internal leadership and vision to SPI success. [Stelzer and Mellis, 1998] suggest that it is vital to provide this leadership both at corporate level and grassroots level. A lack of leadership and vision is reported as one of the key problems of SPI that can result in resistance from practitioners [Moitra, 1998].

XIII. Overcoming internal resistance

There are several solutions offered to combat resistance to SPI programmes from all levels of practitioners. Most of these solutions have already been covered in preceding sections. One such solution is to make the proposed changes introduced by SPI meaningful to grassroots practitioners [Moitra, 1998]. The suggestion here is that

practitioners will fail to see the relevance of supporting SPI if the perceived benefits are projected as benefits for SPI manager's only [Moitra, 1998].

The importance attached to combating resistance to SPI is derived from many accounts that suggest that resistance is one of the biggest hurdles to SPI success [Diaz and Sligo, 1997].

XIV. Rewarding SPI effort

Rewarding SPI effort in Raytheon was found to be important to the success of the SPI programme [Haley, 1996]. The experience in Raytheon showed that not only were practitioners motivated by such rewards, but that it also demonstrated senior management's commitment and support for the programme.

XV. SPI forum

The creation of a change process in Hughes helped to incorporate practitioners' decisions about processes into the corporate SPI programme [Willis et al., 1998]. It also indirectly created a forum for SPI that allowed all the parts of the corporate organisation to provide feedback on any proposed changes. Such a forum was important for the growth of the improvement programme in Hughes.

XVI. Well respected SPI people

Goldenson and Herbsleb [1995] found that when the people involved in SPI are well respected, it could motivate other practitioners to support SPI. In their analyses of factors that influence SPI success in companies, Goldenson and Hebsleb [1995] identified respect for SPI people as one of six most important factors.

XVII. Implementing SPI from within

One significant factor that affects how practitioners support SPI is related to how SPI is implemented. The experience of Raytheon Equipment Division is that SPI run from within the ranks of the company can give practitioners ownership of the processes and empower them [Krasner, 1997]. Earlier sections have discussed the importance of ownership and empowerment to SPI success. Also, Haley [1996] considers running SPI from within the ranks of Raytheon Electronic Systems as the most important factor to the company's success.

Motorola's experience is that the processes in a SPI programme should be defined by "practitioners and task leaders" [Diaz and Sligo, 1997].

In the following chapters, this research will describe four studies that are conducted to examine the four themes identified in the review. The results of these will be used to validate the guidelines recommended by the literature. The next chapter will describe the research process adopted in these four studies.

Chapter Three: Methodology

3.1 Introduction

This Chapter describes the research approach and methods. In particular, it describes the use of social science research methods to elicit and analyse practitioners' perceptions. Although it is unusual to use these methods in the study of software engineering they are widely used and have been thoroughly validated in other disciplines. For example, in the field of health care, the methods described here have been successfully applied to improve quality [Edwards and Browne, 1995].

The rest of this chapter is structured as follows:

Section 3.2 provides an overview of empirical research. Section 3.3 discusses qualitative and quantitative research methods. In Section 3.4, this chapter presents the research design, describing the triangulation approach adopted. This section also describes the data collection and analysis processes used in the studies. Finally, section 3.5 discusses other research approaches that could have been used in this research.

3.2 Empirical research

3.2.1 Definition

This research adopts an empirical approach defined as [Black, 1999]:

> *"The information, knowledge and understanding gathered through experiences and direct data collection"*

According to Lehman and Brady [1976], and Harrison et al [1999] because empirical research is based on observation and experience, it reflects the world more fully than other research approaches. In software engineering research, empirical approaches enable

researchers to examine the inherent multidisciplinary aspects of this area [Harrison and Wells, 1999]. Empirical approaches also offer the opportunity to build and verify theories and to provide better understanding of the software engineering discipline [Lehman and Belady, 1976]. Through replicated studies, using an empirical approach, general theories can be pursued that can help software engineering in many areas [Harrison et al., 1999].

3.2.2 Relevance of empirical methods

Much has been said about the need for rigorous, quantitative experimentation in the field of software engineering research [Hetzel, 1995]. There have been loud calls from the software engineering community for more scientific and better-controlled research projects [Hetzel, 1995]. Software engineering researchers have been encouraged to conduct research where the hypotheses and subsequent results are based on representative subjects and situations [Fenton et al., 1994]. The empirical approach provides a framework for addressing these concerns.

Empirical research offers the opportunity to build and verify theories and therefore provides a way for better understanding of software engineering [Lehman and Belady, 1976]. It also enables rigorous experimentation by encouraging multiple analysis, from multiple perspectives using different techniques [Harrison et al., 1999].

However the time and effort needed to conduct these controlled experiments is often a disincentive. This problem is further compounded by the fact that sometimes companies are unwilling to participate in such studies, as they are reluctant to reveal information about their operations and their products. This is because such information may reflect badly on the company, or may be commercially sensitive. Which means that, overall, there have been relatively few empirical studies conducted in some key and important areas of software engineering, like software practitioners' motivations for SPI. Adopting an empirical approach, therefore, can provide valuable insight into this important area of software engineering.

Empirical research often produces both quantitative and qualitative results. The results can help evaluate models and tools that software practitioners use in order to improve processes and products [Harrison et al., 1999]. The results from this research can help explore the motivators and de-motivators of SPI in an attempt to provide similar insights about SPI motivations.

3.3. Qualitative and quantitative research methods

This research uses a combination of both qualitative and quantitative research methods to explore software practitioners' motivators and de-motivators for SPI. According to Seaman [1999] qualitative and quantitative research methods in empirical research are both equally appropriate in the process of providing evidence to answer research questions. However, the research process is usually more fruitful when the qualitative and quantitative methods are combined [Seaman, 1999]. This research uses a combination of qualitative and quantitative data collection and analysis techniques because the individual techniques turned out to be the most appropriate for the purposes to which they were applied.

Qualitative research methods are designed to study the complexities of human behaviour like motivation, communication and understanding [Seaman, 1999]:

> *"They delve into the complexity of problems as opposed to abstracting them".*

This makes qualitative research methods particularly useful for this research because the issues of motivators and de-motivators are complex and any proper study of such issues requires a deeper understanding of this underlying complexity.

Seaman argues that human behaviour is among very few phenomena that are so complex as to require qualitative research methods to study. All other phenomena can be adequately studied by quantitative research methods [Seaman, 1999].

Quantitative research methods, however, include all the numeric-based computational analysis, including statistical data analysis [Chong, 2001].

Software engineering is intrinsically multidisciplinary [Harrison and Wells, 1999], combining socio-technical issues. As a result the combination of both qualitative and quantitative research methods is an approach that lends itself well to research in software engineering. Combining qualitative and quantitative research methods in software engineering takes advantage of the strengths of both sets of research methods [Seaman, 1999].

3.3.1 Differences between qualitative and quantitative research methods

It is difficult to label research methods exclusively as quantitative or qualitative, as the nature of data collected by the method and the way the data is interpreted by the method are what determines whether a research method can be termed qualitative or quantitative. Gilgun [1992] describes qualitative data as data represented in words and pictures, not numbers [Gilgun, 1992]. Seaman [1999] describes quantitative data as that represented by numbers or discrete categories. It is possible, however, for a research method to be either qualitative or quantitative without necessarily being exclusively so. For example, participant observation [Taylor and Bogdan, 1984] is more appropriate for the collection of qualitative data, than quantitative data. This makes it a qualitative research method in a study where the data collected is qualitative.

Therefore the use of the terms qualitative and quantitative to describe the methods in this research are principally derived from the type of data collection and analysis that these methods are used for in this research.

3.3.2 Subjective and objective data

Qualitative data is often assumed to be subjective whilst quantitative data is assumed to be objective [Seaman, 1999]. However, it is the way the original data was collected that determines whether it is subjective or objective.

- Subjective data

Subjective data is data that is collected as an opinion or perception of the participant of a study or the researcher in a study. For example, if a participant in a survey cites that the ideal number of faults in a new software system should be at most two (2) in the first year of the systems' implementation, then the data item '2' though quantitative, remains subjective. In a similar manner if a researcher in observing a group of software developers records what they perceive as the ideal number of lines of source code that developers should be made to write in a day, then that number denoting the ideal lines of code is subjective data too.

- Objective data

Objective data is the recording of an actual event. For example, the names of the ten participants in a focus group discussion, though qualitative data by nature, is objective. In a similar manner the monthly average fault density in the first year of a new software system is objective and quantitative.

Even though the nature of the data can change from qualitative to quantitative and vice versa, the issue as to whether the data is subjective or objective remains the same. This is because in research where a combination of both qualitative and quantitative research methods are being used, the process of coding data collected in research can transform qualitative data to quantitative data and vice versa [Seaman, 1999]. It is therefore important to recognise this factor when interpreting findings. For example, the results of a study can offer the following two statements:

1. *"Four out of five software developers in the UK use the PSP"*

2. *"Most SPI managers perceive that four out of five UK developers use the PSP"*

Even though both statements have presented a quantitative variable to support their finding, the former is objective whilst the later is subjective.

In this research, a combination of subjective and objective data is collected from software practitioners. In the description of the research design that follows, some qualitative data is transformed to quantitative data in order to carry out some statistical analysis. However, throughout the reporting of the results, the integrity of the original data - subjective or objective - is maintained.

3.4 Research design

The design of this research is influenced by the concept of triangulation. Triangulation advocates that the inherent biases in the research process, which can be attributable to using only a particular researcher, method and data collection source, can be reduced by combining these researchers, methods and data collection sources with others. Triangulation advocates that a variety of research methods in a variety of settings using different data sources be used in order to reduce bias by arriving at the same results through different routes [Harrison et al., 1999]. According to Seaman [1999] this process of enquiry in empirical research strengthens confidence in the evidence being reported.

The studies in this research collect data from different sources in different settings, applying a variety of methods for both data collection and analysis.

Even though triangulation is a major reason for combining research methods, other reasons have been suggested for combining research methods. For example Greene et al suggest that research methods are combined in a single study for the following purposes [Greene et al., 1989]:

- Triangulation - to seek convergence of results
- Complementary - exploring overlapping and different aspects of the same issue
- Developmentally - using the first method to inform the second method
- Initiation - exploring contradictions and fresh perspectives that emerge
- Expansion - whereby the mixed methods add scope and breadth to the study

In this research, a triangulation approach is adopted to increase confidence of the research findings. This is done through the different studies:

- SPI managers' opinion of the motivators and de-motivators of SPI is collected using a questionnaire survey and the results analysed by frequency analysis. This forms the first stage in the research process and serves as the method that informs the subsequent data collection methods.
- Data is collected on software practitioners' motivators and de-motivators for SPI using focus group discussions. The data is analysed using content analysis techniques of coded categories. The coded categories are further analysed using Multidimensional Scaling to establish the degree of association between motivators and de-motivators. This second stage of the research process provides deeper insight into the problems with practitioners' motivators and de-motivators for SPI.
- Software practitioners' perception of their role in SPI is collected using the Repertory Grid technique. This data is analysed using Grid analysis technique. Results from these analysis helps to qualify the findings of software practitioners' motivators and de-motivators for SPI conducted in earlier studies. They also help to explain the differences in perception of motivators and de-motivators by different practitioner groups.

A matrix of the research findings from the four studies is provided in Appendix C, as recommended by [Neff, 1987]. The research process is described in more detail as follows.

3.4.1 Data collections methods

The data collection method significantly influences the data analysis process that can be used in the research. The selection of the data collection process needs to be carefully considered since its impact on the rest of the research process in significant. This research uses questionnaire survey, focus groups and the Repertory Grid technique for data

collection. These methods are used because they are best suited for the nature and type of data that this research analyses.

- Questionnaire surveys

Questionnaires are one way of conducting a survey. Interviews and opinion polls are other ways. A questionnaire was used in the study of SPI managers' perceptions of the motivators and de-motivators for SPI discussed in Chapter Four.

Overview of questionnaires

Questionnaires are most appropriate when a survey aims to collect data from a large sample. As a method of collecting primary data, questionnaires are more convenient than most survey methods [Berdie and Anderson, 1974]. Well designed questionnaires collect less biased data as the respondent is not influenced by the attitude or opinion of an interviewer or *vice versa*. Administering questionnaires overcomes the inherent problems in replication because all respondents receive the same set of questions.

Questionnaires are not appropriate in all circumstances. For fear of the researcher's questions being misinterpreted by the respondents, sometimes questionnaires only ask shallow questions which are unable to capture the real issue under investigation [Berdie and Anderson, 1974]. Questionnaires also suffer from low response rates [Berdie and Anderson, 1974]. This is predominantly because the respondent is usually under no obligation to respond to the questionnaire. According to Berdie and Anderson [1974] questionnaires also suffer from reliability problems regarding the accuracy of the data collected. This is because it is often difficult to know who answered the questionnaire. However, this particular problem can be overcome by directly targeting sample respondents. In this research, sample respondents were directly targeted.

Questionnaire administration

Questionnaires can be grouped into mailed questionnaires, self-administered and researcher-administered. When a survey is aiming to reach a large sample across a wide geographic span, mailed questionnaires are usually the most appropriate [Heather and Stone, 1984]. Mailed questionnaires also have the advantage of eliciting very frank answers from respondents.

Designing the questionnaire

The task of designing an effective questionnaire that adequately captures the data the researcher is interested in is very difficult. There can be several problems with questionnaire design the most prominent of which is the extent to which respondents understand the questions being asked. Czaja and Blair [1996] suggests that the most effective way of testing a questionnaire is by piloting it on people who were not involved in its construction [Czaja and Blair, 1996].

Application of questionnaires in this study

This research uses questionnaire survey to collect both subjective data - SPI managers' perception of SPI in their companies - and objective data - demographic and background information about SPI managers.

A sample of SPI managers was identified using public domain information about software development companies. This information included relevant mailing lists and conference attendance lists. Questionnaires were mailed to SPI managers at one thousand companies and two hundred replies were received of which eighty were fully completed questionnaires. A response rate of 20% is normally considered acceptable for a survey of this nature, however, considering many UK software companies have no formal SPI programme, a total response of two hundred questionnaires could, in this case, be considered very good.

Questionnaire responses were measured on a Likert scale [Likert, 1932] of 1 to 5, where 1 indicates strong disagreement with a statement and 5 strong agreement. Appendix D presents a copy of the complete questionnaire.

Pilot study

To test the effectiveness of the questionnaire used in this research, it was piloted on SPI managers in two UK companies. Feedback from the managers in these companies suggested that the questionnaire would be more effective if additional questions were added to elicit the personnel profile of the respondent. These were added later.

Overall, the use of surveys, in general, in software engineering research is relatively common. For example, Britton et al [1997] used two survey methods in a study that set out to understand the development practices in multi-media systems. In this study, postal questionnaires were used to supplement structured interviews in order to focus the area of research, and at the same time, expand the target sample for the study [Britton et al., 1997].

- Focus groups

Focus groups is another data collection method used in this research. Focus groups were used in studies of software practitioners' motivators and de-motivators for SPI discussed in Chapters Five and Six.

Overview of focus groups

Focus groups are a type of interviewing technique commonly used to elicit group perspectives about a given situation. They are basically research interviews where the researcher acts as the moderator of a group of subjects and relies on interaction between the subjects [Morgan, 1997]. In that respect focus groups differ from normal interviews in which the researcher poses the questions and the subject responds. The data from focus

group interviews is predominantly from the interaction between the members in the group as directed by the researcher [Morgan, 1997].

Morgan [1997] and Krueger and Casey [2000] suggest that the advantage of focus group interviews is that the existence of subjects in a group of peers allows them to be more open about issues discussed than they will in, for example, individual interviews.

Limitation of focus groups

A weakness of focus groups is that the method can be less naturalistic than say participant observation because it relies on the topic of interest and focus of the researcher [Morgan, 1997]. Morgan argues that this reliance on the researcher's focus and interest can make subjects less likely to talk about issues that are pertinent to them [Morgan, 1997].

Application of focus groups in this research

This research uses focus groups to collect data from software practitioners about their motivators and de-motivators for SPI. Focus groups are used to uncover some of the complicated issues regarding practitioner support for SPI, which other methods like questionnaires do not cover. Focus groups allowed this research to investigate a larger sample of software practitioners than one to one interviews would have.

In this research focus groups were separated into three groups of practitioners:

- The first group was made up of software developers/ testers /designers. Referred to as "developers"
- The second group was made up of project managers/middle managers/team leaders. Referred to as "project managers"
- The third group comprised senior managers.

This research collected data from forty-nine focus groups from thirteen companies. In the focus group sessions, practitioners were asked the following questions:

Motivators for SPI

- *What are the motivators for SPI in your company?*
- *What would motivate you to support SPI?*

De-motivators for SPI

- *What are the obstacles to SPI in your company?*
- *What would stop SPI from happening in this company?*

All focus group sessions were audio recorded. Recording focus group sessions enabled the sessions to run smoothly without the 'start-and-stop' practice that can occur if the sessions are hand written. However, notes were taken of the key issues that were raised in the discussions.

- Repertory Grid Technique (RGT) interviews

Another method of data collection used in this research was the Repertory Grid Technique (RGT). It was used in the study of software practitioners' perception of their SPI roles discussed in Chapter Seven.

Overview of RGT

RGT is a suite of techniques available to researchers studying human behaviour and attitude. RGT allows the exploration of the construct system of participants without interference from the researcher. It is based on work done in psychoanalysis by George Kelly in the 1950's on personal construct theory (PCT) [Bannister and Fransella, 1986]. RGT was initially designed for use in psychoanalysis but has subsequently been generalised for broader use.

RGT is also an investigative data collection technique which removes the observer's frame of reference from what is being observed [Bannister and Fransella, 1986]. It attempts to nullify the influence of the observer's own perspective on the observed. RGT examines and records personal constructs: that is, an individual's concepts about a given situation [Stewart et al., 1981]. These constructs are formed through the individual's experiences. The aim of RGT is to allow subjects to reveal their constructs without any influences from the investigator.

RGT enables researchers to understand how the subjects have arrived at their perspective on a given situation. It enables the researcher to uncover the "building blocks" of an opinion. Once this has been established, the aim is for the researcher to be able to manipulate these building blocks to alter the subject's attitude towards the given situation or set of circumstances.

Even though initially developed and used in psychoanalysis, RGT currently has a broader appeal and is used in fields as far afield as education and market research. Indeed in software engineering, RGT has been used as a method of requirements engineering [Maiden and Rugg, 1996]. One of the many purposes in requirements for which RGT can be used is in the selection of a commercial system. For example, RGT can be used to guide the selection criteria for purchasing a software package. RGT can also be used in acquiring domain knowledge in requirements engineering. As a knowledge elicitation tool, RGT is useful for acquiring knowledge in the form of behaviour, process and data [Maiden and Rugg, 1996].

The basic tenet of RGT is that individuals' construct systems are formed by their experiences [Kelly, 1970]. The construct system becomes the medium by which individuals perceive a given situation. It is also the medium by which expectations of a given situation are formed. Through the construct system, perceptions influence expectations and expectations influence perception [Stewart et al., 1981]. This makes the construct system something that develops throughout an individual's lifetime. The basic components of construct system are the elements and constructs.

The following are definitions for constructs and elements:

Elements: In RGT, elements are usually the subject of analysis. Stewart et al describe elements as "people, objects, events and activities" [Stewart et al., 1981]. Elements should refer to specific people or specific objects. For example, "the ideal practitioner" cannot represent an element, whereas "project manager" refers to a specific type of practitioner.

Suggested guidelines for selecting elements include [Stewart et al., 1981]:

- Must be discrete
- Must be homogeneous
- Should not be subsets of other elements
- Should not contain implied values

In this research the elements chosen are the three discrete groups of software practitioners as defined in Chapter One:

- Developers,
- Project managers
- Senior managers

Constructs: Stewart et al [1981] describe an individual's construct system as their system of hypotheses by which they interpret the world. These interpretations are built upon their experiences of the world. A construct is the perception that one has of a given issue - in this case an element - based upon experience with respect to that element. For example, developers may describe project managers as technically minded. This means that developers have used *"technically minded"* as a construct of the element *project managers.*

Bi-Polar Constructs: Constructs are essentially bi-polar. According to Kelly this bi-polarity enables one to "*make sense out of our world by*

simultaneously noting the likenesses and differences" [Fransella and Bannister, 1977]. It is by showing the contrast that the meaning of the construct is made clearer. Therefore in most RGT research bi-polar constructs are used to distinguish between elements. Table 2 illustrates a bi-polar construct used in the context of three staff groups. Table 2 implies that developers and project managers are technically minded whereas senior managers are not. The bi-polar constructs are 'technically minded', and "not technically minded", which make a distinction between developers and project managers on one hand and senior managers on the other:

Technically minded	*Not technically minded*
Developers	Senior managers
Project managers	

Table 2: Example of a bi-polar construct

Application of RGT in this research

This research used RGT to collect attitudinal data on the perceptions of the three groups of practitioners on their role in SPI. The aim was to understand how each group of software practitioners considers their role in SPI and how this contrasts with that of the other practitioner groups.

To achieve this aim, this research asked a set of questions that were aimed at eliciting bi-polar constructs from the three groups of practitioners.

The questions used in these RGT sessions were:

"Think about developers, project managers and senior managers in the context of software process improvement. What do you think the paired groups have in common that differentiates them from the single group."

1. *What do senior managers and project managers have in common that differentiates them from developers?*

2. *What do developers and project managers have in common that differentiates them from senior managers?*

3. *What do developers and senior managers have in common that differentiates them from project managers?*

Practitioners' responses to these questions were audio recorded. The audiotapes were transcribed into bi-polar grids. Each transcript contained a list of bi-polar responses to each of the three questions above. Figure 6 presents an example of a transcribed bi-polar grid:

1. *Developers*	*Project managers* *Senior managers* Spend a lot of time in meetings Have a wider perspective Talk about things that affect us but don't tell us Protective information goes on here
2. *Developers* *Project managers* Visibly in the same team Know what is going on *"Us and them"* divide	*Senior managers*
3. *Developers* *Senior managers* These conduct their roles for a considerable time They are stuck in their positions	*Project managers* Whereas these lot come and go

Figure 6: Example of a transcribed bi-polar grid

3.4.2 Data analysis methods

The methods of data analysis chosen by the researcher are strongly determined by the type of data collected. The following subsections outline the data analysis methods used by this research. The methods outlined are determined by the qualitative nature of the data collected in this research.

- Frequency analysis

One of the first ways of organising raw data is to group scores or values into frequencies [Black, 1999]. Frequency analyses are useful for reporting descriptive information from research. Frequency tables are used to report numbers of occurrence of each data variable. These frequencies can then be presented either in tallies or in percentages. Frequencies are useful for comparing and contrasting within groups of variables or across groups of variables.

Frequency analyses can be used for both nominal/ordinal data and also numeric data. Frequency analyses can also be used to conduct elementary statistics on both subjective and objective data.

Application of frequency analysis in this research

Frequency analyses were used to analyse data collected from the questionnaire survey of SPI managers' perception of SPI. This is discussed in Chapter Four.

Frequency analyses were also used to summarise data categories in the study of motivators and de-motivators for SPI. These analyses are discussed in Chapters Five and Six.

- Correlation analyses between pairs of responses.

Correlation analyses determine the strength of the relationship between two or more variables [Rowntree, 1981]. Correlation analyses between a pair of variables establishes the extent to which changes in one variable are reflected by corresponding changes in the other.

The strength of association is represented by the correlation coefficient, which is usually denoted by '***r***'. ***r*** takes the value of between -1 to + 1, where -1 and 1 represent a perfect relation between the two variables and 0 represents no relationship at all. Negative ***r*** represents a reciprocal relation between two variables. That means positive changes in one variable are accompanied by corresponding negative change in the other variable and *vice versa*. Whereas positive ***r*** represents a positive change in one variable accompanied by a corresponding positive change.

Correlation analyses can be used to establish the relationship between both quantitative and qualitative variables. It is possible to conduct correlation analysis between pairs of the following types of data:

- Quantity variables, where the values are either measurements (continuous) or counts (discrete). For example, age of company (continuous) and number of staff (discrete)
- Ranked/Scaled ordinal data, where the underlying data is qualitative, but it is denoted by a numeric rank. For example, software process maturity levels as denoted by the CMM
- Nominal data, where the underlying variable is category and presented as category. For example, stages in software development: requirements capture or testing.

Correlation coefficients for quantity and ranked or scaled ordinal data are calculated in the same way, using, usually, the Pearson's product-moment correlation coefficient. This correlation coefficient is calculated by [Rowntree, 1981]:

"taking into account the amount by which each value differs from the mean of its own distribution, the deviation of the two deviations and the number of pairs of values"

For nominal data, however, ***r*** is calculated using a rank correlation coefficient. Rank correlation coefficients calculate the closeness in the ranking of two variables by pairs of observations. The closer the ranking, the higher the value of ***r***.

Significance levels for ***r***

The coefficient of correlation is said to be significant if the strength of the relationship that it suggests between two variables can be inferred for the whole population. That is, if it can be said that, within a certain probability, the strength of the relationship established between two variables has not occurred by chance.

There are two factors used for determining the significance of the correlation coefficient. These are the absolute value of r and the size of the sample used [Rowntree, 1981]. The closer the value of r is to 1 or -1 and the larger the sample of data used, the more reliable the correlation results are. For example, ***r***-value of 0.600 is not significant at the confidence level of 1% if the size of the sample happens to be ten (r critical is 0.794). However, for a sample size of twenty, the same ***r***-value is significant (r critical is 0.570) [Coolican, 1990].

Applying correlation analyses in this research

In this research, correlation analyses were used to establish the strength of the relationship between SPI managers' responses in the questionnaire. In this questionnaire, scalar values are attached to responses. Responses were calibrated on a scale of 1 to 5, where 1 represents strong disagreement with a statement and 5, a strong agreement.

Correlation analyses were also used to investigate the strength of the relationship between the responses given by SPI managers and the maturity of the processes in their

companies. SPI managers are asked to provide the maturity of their company's development processes according to CMM levels.

In both instances, this research used correlation analyses to measure the relationship between two ordinal variables.

- Content analysis

The process of data analysis described here takes into account the content analysis methods described by [Krippendorf, 1980] and the principles of analysing qualitative data from the works of [Bryman, 1990] and [Silverman, 2000]. The analysis process described here is used in studies on software practitioners' motivators and de-motivators for SPI discussed in Chapters Five and Six.

Once data has been collected in the form of transcripts from focus group discussions, next is to develop categories to be used in comparing and contrasting results [Creswell, 1994]. This analysis process can take several forms. According to Tesch [1990], there is no rigid format to analysing qualitative data. The process can be eclectic, containing several analysis procedures [Creswell, 1994]. The analysis process described in this research is two staged: coding categories of the qualitative data and interpreting results of the coded categories.

Coding categories in qualitative data analysis

One of the most important stages in analysing qualitative data is identifying the categories that appear in the data [Creswell, 1994]. Coding is a common technique used to establish data categories in qualitative data. Coding helps to re-organise and re-arrange the data so it can be subjected to more rigorous analysis [Seaman, 1999]. Different writers describe slightly different processes for establishing categories for qualitative data through coding. Burnard describes the following steps for categorising qualitative data obtained in the form of interview transcripts [Burnard, 1991]:

i. Make notes immediately after interviews

ii. Skim read through transcript to identify general themes that appear.

iii. Browse through each transcript in detail to generate categories for relevant themes.

iv. Categories are analysed to identify clusters. Similar categories are grouped together under a higher category heading.

v. The process is repeated until a tighter distinct set of categories is obtained.

vi. The category system is verified to improve its validity and reduce researcher bias. This is done through generating two other independent category systems with the same data. The three category systems are then cross-checked against each other for differences and similarities. Through adjustments a final category system is produced that truly reflects all three systems. This list of categories must be able to account for any issue raised in the transcripts. But they must not be too broad to put more than one meaning into one category.

vii. Each transcript is therefore read through again and adjustments are made to the final agreed list of categories.

viii. Assign each category a colour and mark sections of the transcripts with the relevant colour that reflects the category in question.

ix. All marked sections from the transcripts are taken out.

Burnard [1991] describes five more stages where categories are isolated, validated and verified with participants of the research. After this process of validation and verification the categories are then linked into the transcripts from where they were taken to ensure that their meaning reflects the context from which they were taken.

Burnard's process is elaborate and most suitable for cases where the researcher conducts the whole analysis process qualitatively and manually. Burnard's process is adequate when the researcher is not seeking to identify frequencies of occurrence of category issues.

This research, however, seeks to identify the frequency of occurrence of motivators and de-motivators from focus group sessions. The frequency analysis therefore requires a system for coding the motivators and de-motivators as they appear in the focus group sessions. Tesch describes eight important steps necessary for coding textual data [Tesch. 1990]:

i. Read through all the transcripts and write down ideas as they come to mind
ii. Pick one transcript and go through. Decipher what the key points are. What is the underlying meaning of this transcript? Write your thoughts in the margin
iii. Repeat the process for several transcripts and make a list of all the topics. Cluster together similar topics. Form an array of major topics, unique topics and leftovers
iv. Take the list of topics and abbreviate the topics as codes. Apply the codes to segments of the transcripts. Through this process, find out if new categories and codes emerge
v. Find the most descriptive word for the topics and turn the topics into categories by grouping together the topics that relate to each other
vi. Make a final decision on the abbreviation for each category and put the codes in alphabetic order
vii. Assemble the data material belonging to each category in one place and perform a preliminary analysis
viii. If it becomes necessary after the preliminary analysis, re-code the existing data

According to Seaman [1999] coding in empirical research is one method of extracting quantitative data from qualitative data in order to perform some statistical or quantitative analysis on the data. In this research, data from focus group discussions is categorised and coded to enable quantitative analysis of the data in terms of frequency of occurrence, and also to enable some comparative and contrasting analysis of motivators and de-motivators within and between staff groups.

The steps adopted in this analysis process are an amalgamation of the steps described in the literature on categorising qualitative data. In this research, the focus group sessions

with software practitioners discussing the motivators and de-motivators to SPI have been transcribed. The qualitative data in the transcripts is the basis for this categorisation process. The following is the categorisation process adopted in this research:

i. Note important issues

During the focus group session, notes were made about the topics discussed and the issues raised.

ii. Identify themes from transcripts

All the transcripts were read to identify the major themes that appeared in the focus group sessions. The themes were noted down and compared to the notes made during the focus group sessions. This two step process serves as reassurance that the transcripts being analysed are indeed a true reflection of the discussions in the focus group session. It also verifies that the process of transcription has not diluted the original data generated in the sessions. Finally, it sets the scene for the researcher, by making the researcher aware of the themes that will appear in the analysis.

iii. Generate categories

All focus group transcripts were read again to generate categories for responses. Categories identified were recorded. Because there were two issues being discussed in each focus group session, i.e. motivators and de-motivators for SPI, two sets of categories were created for each transcript.

iv. Group similar categories together

In each of the two sets, categories were analysed and similar categories are grouped together under a higher category heading. For example, under motivators for SPI, *time for SPI, money for SPI* and *people for SPI,* were grouped under the higher category *resources for SPI.*

v. Generate aggregate list of categories

The process was repeated until distinct sets of categories were obtained. These two sets of categories became the categories of software practitioners' motivators and de-motivators for SPI.

vi. Assign codes to categories

The transcripts were read through again, this time in conjunction with the list of categories. Adjustments are made to ensure that the categories correctly reflect issues in the transcripts. The process was repeated for each transcript. A code was then assigned for each category in the final list of categories.

The subsequent stages in this qualitative analysis process are to do with preparing the data for quantitative analysis and verification of the categorisation process. This next step involved the construction of data matrices for each of the two sets of categories. In this research because the analysis is also concerned with how motivators and de-motivators are cited across practitioner groups, it becomes necessary to develop data matrices for each of the three practitioner groups under each set of categories. This procedure is described in detail when the individual studies are reported in Chapters 5 and 6.

vii. Construct data matrix for categories

A data matrix is constructed that reflects how categories appear in each transcript. The process is repeated until all the transcripts have been accounted for. The result is two master matrices: one for motivators and another for de-motivators.

viii. Verify categories

Next in the process is the use of verification procedures and tests aimed at reducing researcher bias in recording categories. At this stage, the triangulation approach is again adopted in the context of comparing the analyses of two independent researchers [Coolican, 1990]. This is a low-level application of the concept whereby an independent researcher is used to audit a random selection of research findings [Creswell, 1994].

A blank matrix is prepared which shows all the categories under each set, but not how they are cited in the transcripts. Four random transcripts were chosen and an independent researcher, not involved with this research, was asked to record the issues that practitioners discussed in the transcripts under the categories provided. The procedure is repeated for both sets of categories: motivators and de-motivators. An inter-rater reliability index is calculated to establish the degree of agreement between the two researchers. The process is described below.

ix. Adjust categories to produce final list

Where disagreements between researchers are too many, and the inter-rater reliability index indicates a very low disagreement, it becomes necessary for adjustments to be made to the category sets to reflect mutual concerns of both researchers. Strong disagreements requiring the inclusion or the omission of some categories, are followed by a repetition of stages vii to ix. This process is repeated until there is mutual agreement from both researchers on a common set of categories.

x. Record transcripts into matrices

Data matrices are then recorded with the final agreed set of categories. Under each set of categories, three sub matrices are recorded reflecting developers', project managers' and senior managers' motivators and de-motivators.

xi. Begin frequency / MDS analysis

At this stage the data is ready to be analysed by frequency analysis or MDS analysis.

Calculating inter-rater reliability index

The process of calculating agreement index between two independent researchers is described as follows:

The four random transcripts for de-motivators are labelled A, B, C and D. The second batch of transcripts for motivators are labelled E, F, G and H. Appendix E shows how the two researchers have identified motivators and de-motivators in the transcripts.

A comparison is made between how this research and the independent researcher have identified motivators and de-motivators as they appeared in the transcripts. This is effectively an exercise to test the reliability of the categories derived for the motivators and de-motivators. Cohen's kappa (κ) statistic is used to test this reliability. According to SPSS [1999]:

> "Cohen's kappa measures the agreement between the evaluations of two raters when both are rating the same object. A value of 1 indicates perfect agreement. A value of 0 indicates that agreement is no better than chance. Kappa is only available for tables in which both variables use the same category values and both variables have the same number of categories"

In this research, a kappa statistic is calculated for how the two researchers translate each transcript. It is unusual to calculate more that one kappa statistic to evaluate the same set of categories, however, this approach has been adopted by Theodore when rating a 34 variable content dictionary from 101 offender profiles [Theodore, 1999]. As in Theodore's [1999] work, this research calculates separate kappa statistics for each transcript because individual transcripts do not necessarily identify the same number of category values. The aim of this exercise is therefore to test the accuracy of the sets of categories as opposed to that of the data recorded.

Table 3 shows the inter-rater agreement frequencies. 1 in both columns means that both researchers selected the same motivator or de-motivator from the same category for a transcript.

	De-motivators		
	Rater 1	Rater 2	Freq.
Script A	1	1	3
	1	2	3
	2	1	4
	2	2	19
Script B	1	1	2
	1	2	3
	2	1	4
	2	2	20
Script C	1	1	2
	1	2	0
	2	1	3
	2	2	24
Script D	1	1	4
	1	2	2
	2	1	4
	2	2	19

	Motivators		
	Rater 1	Rater 2	Freq.
Script E	1	1	2
	1	2	3
	2	1	0
	2	2	27
Script F	1	1	2
	1	2	6
	2	1	0
	2	2	24
Script G	1	1	1
	1	2	3
	2	1	0
	2	2	28
Script H	1	1	2
	1	2	3
	2	1	0
	2	2	27

Key: "1" indicates category selected, "2" indicates not selected.

Table 3: Inter-rater agreement frequencies

The test of inter-rater reliability shows statistically significant results (95% confidence) in six out of the eight randomly selected transcripts, with low to moderate kappa statistics, ranging from 0.216 to 0.529 [Landis and Koch, 1977]. Table 4 provides an overview of the kappa statistics and significance results.

De-motivator Scripts	**kappa**	**Sig.**
A	.307	.096
B	.216	.241
C	.525	.001
D	.439	.016
Motivator scripts	**kappa**	**Sig.**
E	.529	.001
F	.333	.011
G	.368	.007
H	.529	.001

Table 4: Overview of the kappa statistics and significance results

Table 4 shows that the Kappa statistic for the scripts to test motivators are all significant at the 0.05 significance level. This indicates that it is unlikely that any two researchers can rate any two unrelated sets of categories and come up with the same selection as presented in Appendix E. So that even though the level of agreement between two researchers is only low to moderate, the kappa statistics are statistically significant.

However, Table 4 also shows that the kappa results for two of the de-motivator scripts - 'A' and 'B' are not significant. Which means that the selections for de-motivators between

those two researchers can occur by chance. Nevertheless, the overall exercise in reliability shows some statistical significant results albeit not necessarily high agreements between researchers.

The next stage in this research process is to analyse the categorised coded data by frequencies to determine occurrences within software practitioner groups and also across groups. After frequency analysis, the association between motivators and de-motivators is examined using the MDS technique of SSA.

- Multidimensional scaling

Multidimensional Scaling is a data analysis technique that represents the relationship between variables within geometric space. It is defined as:

> A set of procedures that allow for information contained in a set of data to be represented by a set of points in space, arranged in such a way that the distances between the points reflect the empirical relationship [Guttman, 1968].

Traditionally, MDS has been used in conjunction with facet theory [Guttman, 1968]. Facet theory is an empirical research design and analysis technique that Guttman and Greenbaum [Guttman and Greenbaum, 1998] describe as:

1. Having a definitional framework for a universe of observations in a study.
2. Establishing an empirical structure of observations within this framework.
3. Searching for correspondence between the definition framework and the empirical structure.

Facet theory proposes ways of exploring and understanding a concept, for example attitude, intelligence or motivation in this instance, by depicting the inter-relationship between the components of that concept [Shye et al., 1994]. For example, according to Herzberg's Motivation-Hygiene theory, motivation is made up of intrinsic and

extrinsic factors [Herzberg et al., 1959]. According to facet theory, in order to understand motivation better, one needs to understand the inter-relationship between the intrinsic components and the extrinsic components of motivation. Such understanding is achieved through empirical observation of several motivator variables within these components.

Multidimensional scaling is the technique for representing and analysing the observed empirical variables of a concept.

Application of MDS in this research

MDS is used in two studies in this research to analyse the association between the motivators of SPI and also between the de-motivators of SPI. The technique is applied to gain better insight into software practitioners' motivators and de-motivators for SPI by showing the interaction between the factors that constitute the regions/component/facets of the two concepts.

In this analysis the concepts are motivators and de-motivators for SPI. The components or facets are the classification of motivators according to classic motivation theory and the classification of de-motivators as either organisational, process or personal issues. The variables are the motivators and de-motivators for SPI as cited by three groups of software practitioners.

Overview of MDS

A basic tenet of MDS is that the stronger the statistical relationship between variables, the closer they will appear in geometric space. Table 5 shows the statistical relationship between three variables, A, B and C.

	A	B	C
A	*	.8	.5
B		*	.1
C			*

Table 5: Multivariate correlation - 3 variables

Table 5 shows that variables A and B have the highest correlation (.8). Variables A and C show a weaker correlation than between A and B but a stronger correlation than that between B and C. These correlations can be represented in one dimensional space as in Figure 7.

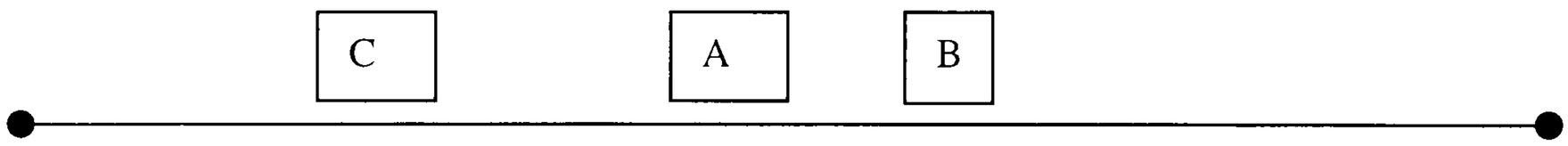


Figure 7: One-dimensional representation of statistical relationship

The limitation of this one-dimensional representation becomes apparent if Table 5 is expanded to include a fourth variable. Table 6 shows a four variable correlation matrix.

	A	B	C	D
A	*	.8	.5	.3
B		*	.1	.6
C			*	.4
D				*

Table 6: Multivariate correlation - 4 variables

Representing the relationship between all four variables as distances is not possible to do on a one-dimensional scale. This is because it is not possible to place D anywhere in Figure 6 to truly reflect its statistical relationship with the other three variables. Representing the statistical relationships will also be difficult in two dimensions, i.e. in an area. However, conceptualising the space in which these variables occur in three dimensions (within a cube) makes it possible to adequately present this statistical relationship.

Multidimensional scaling comes from a need to represent multiple variables that are related to one another. As the examples in tables 5 and 6 show, when there are more than three variables, this representation becomes difficult to show either on one or two-dimensional scales. A three dimensional scale, therefore, becomes the best way of representing the statistical relation between several variables.

Smallest Space Analysis (SSA)

SSA is one of three MDS techniques recommended by Guttman (1968). It is a non-parametric MDS technique, used for cases where the data being analysed is nominal or categorised ordinal data. The type of data collected in the focus group studies in this research makes the non-parametric technique of SSA the most appropriate MDS technique to apply.

The technique is termed *smallest* space analysis because it uses the ranks of the correlation, as opposed to the actual correlation to plot the closeness of points in geometric space. This ensures that the smallest space possible is used in the depiction of relationships.

Traditionally, correlation between quantitative variables is measured by the degrees of change over cases. Correlation is measured in non-parametric data by the presence or absence of variables over a series of cases.

The following is a summary of the steps involved in SSA:

1 Establish a Content Category Dictionary of all variables.
2 Create a data matrix based upon how many occurrences of the variables in the content dictionary appear in each 'case'.
3 Run the SSA tool to plot the variables and their relationships.
4 Partition the SSA plot into regions. The researcher should be able to argue the inclusion of variables in the region they appear.

When a SSA plot is produced, a coefficient of alienation is used to measure the "fit" between the geometric representation and the original correlation matrix. A coefficient of 0 represents a perfect fit and anything up to 0.25 is considered good.

Illustrating SSA using CMM maturity levels

The following is an illustration of how a SSA plot is derived from non-parametric data, using process maturity levels as an example:

> Consider the data in Table 7. Assume that these are the results of a survey conducted on six companies using the Capability Maturity Model (CMM) [Paulk et al., 1994a]. SPI managers in each of the six companies have indicated those Key Performance Areas they are currently concentrating on. So that overall, four KPAs have been indicated between the six SPI managers. The similarity between the responses offered by these six SPI managers determines the relationship amongst the four KPAs. Using a SSA plot, such a relationship is depicted by the geometric representation of the KPAs. In Table 7, KPAs depicted similarly by the respondents will appear closer to each other in a SSA plot. Overall, the greater the similarity between two variables, the greater their proximity in the corresponding geometric space.

	Key Process Areas (KPA's)			
	A	B	C	D
Company 1	0	1	0	1
Company 2	1	0	1	0
Company 3	1	0	0	0
Company 4	1	0	1	0
Company 5	0	1	0	1
Company 6	0	0	1	0

A = Quantitative process management
B = Integrated software management
C = Software quality management
D = Software product engineering

Table 7: Example of a data matrix of non-parametric variables

The geometric representation in a SSA plot is achieved by using the data matrix - the 0's and 1's in, say Table 7 - to derive a multivariate correlation matrix with

quantitative coefficients of between 0 and 1. This matrix is then plotted in multi-dimensional scaling to show the likelihood of variables co-occurring to each other. In this illustration, a SSA plot is used to demonstrate the relationship between the four KPAs. That is, the relative likelihood of the four KPAs co-occurring to each other. Figure 8 represents this information geometrically.

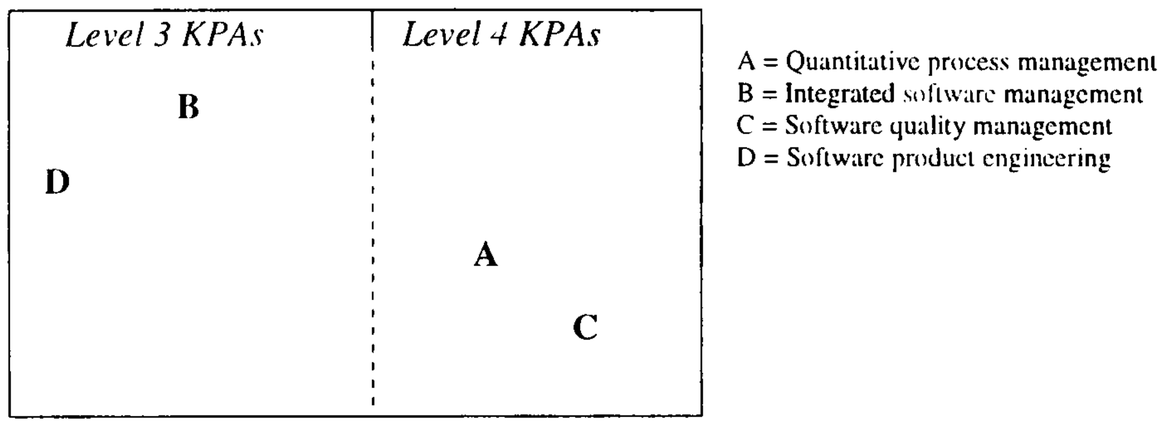


Figure 8: An illustration of a SSA plot using CMM KPAs

Figure 8 shows two clustering of KPAs: A & C and B & D. A further analysis of Figure 8 reveals that the SSA plot has two broad regions in accordance with CMM levels and the closest KPAs happen to co-occur around each other within these regions. These broad regions are CMM level 3 and CMM level 4.

The SSA plot illustrated by Figure 8 can now provide some insight into the relationship between the four KPAs. One such insight is that, according to the six SPI managers, companies focusing on *quantitative process management* (A) are more likely to be also focusing on *software quality management* (C) than they are *integrated software management* (B) or *software product engineering* (D). Even though this example is contrived, it nevertheless, shows how the technique can be used to investigate real relationships between variables.

Applying SSA in this research

In this research, focus group sessions take the place of "Companies" in the previous illustration and "KPAs" are the motivators and de-motivators that practitioners cited in focus groups.

The partitioning of SSA's is done in relation to particular "facets". In this research "facets" are the motivation theories and de-motivation issues.

The regions in a SSA plot are the components (i.e. the facets) of the concept under investigation. Partitioning takes place according to how the variables represent themselves in the plot. When partitioning a SSA, the researcher is trying to show how components of a concept are mapped out on the plot and how variables are represented within the components. Established theory defines the components of each concept and the variables within that component. It should therefore be possible to defend the inclusion of any variable in a particular region/component/facet of a plot with theory.

In this research the "facets" are the characteristics of the motivators, and the categories of the de-motivators. So that, for example, a SSA plot of motivators may be divided into the *punitive* and *rewarding* regions, whereas a similar plot of de-motivators is divided into *organisational*, *process* or *project* regions.

The choice of partitioning SSA's is strictly arbitrary, but the partitioning should be done such that the inclusion of every variable in a particular region is supported by theory suggesting that the variable can be so defined.

Limitation of MDS analyses

Although MDS is a technique that represents data in 3D, the plots that are produced from MDS analyses are usually in 2D. Therefore the SSA plots used in the analyses in this research are from 2D representations of the 3D relationships depicted. This means that it is not always easy to see some of the relationships depicted by these plots.

- Grid analysis techniques

This research uses grid analyses techniques to analyse the data collected from forty-six RGT sessions in a study of software practitioners' perception of SPI roles discussed in Chapter Seven.

There are six suggested techniques for analysing grid data, four are manual methods whilst two are computer analysis [Stewart et al., 1981]:

Manual analysis:

- Frequency counts
- Content Analysis
- Performance appraisal
- Visual focusing

Computer analysis:

- Principal-component programs
- Dentritic analysis

This research uses visual focusing to analyse grid data elicited from three practitioner groups of developers, project managers and senior managers.

Visual focusing

After the grid interview, a full grid is constructed which cross references constructs with elements. Full grids are constructed for each of the three elements of developers, project managers and senior managers. It has been necessary to construct separate grids for each group of practitioners so that separate analysis can be conducted of the particular groups' perspectives.

The following are the steps involved in grid analysis, using visual focusing:

1. A raw grid comprising elements and constructs represented by 0's and 1's is constructed. Table 8 presents an example.

	E1	E2	E3	E4	E5
C1	0	0	1	1	1
C2	1	0	1	0	1
C3	1	1	0	1	1
C4	1	1	0	0	0
C5	0	1	0	1	0

Table 8: Example of a raw grid showing constructs against elements

2. Pairs of elements are compared to each other. An agreement score (0 and 0 or 1 and 1) is computed for each pair of elements. Table 9 shows an example.

E1	E2		
0	0	=	1
1	0	=	0
1	1	=	1
1	1	=	1
0	1	=	0
	Total	=	3

Table 9: Eliciting element to element agreement

3. A matrix showing the agreement scores between every possible pair of elements is constructed. Table 10 shows an example.

	E1	E2	E3	E4	E5
E1	*	3	2	1	3
E2		*	0	3	1
E3			*	2	4
E4				*	3
E5					*

Table 10: Example of an element by element construct

4. The matrix is inspected, looking for the high agreement scores. Where there is a high score between two elements, it implies there is a close understanding between those elements.

5. Next is to return to the original grid and place elements that have high agreement scores next to each other. In this research, because there are only three elements, this process will not show much of a difference in the original grid. The next stage, involving constructs, however, may show greater differences.
6. Stages 1 to 5 are repeated, this time using the constructs.

It may be possible to reverse particular constructs if the researcher wants to achieve maximum agreement between constructs. This is made possible because the constructs used in this analysis are bi-polar. For example, the construct "have short term objectives" can be reversed to mean, "do not have short term objectives". By doing this, the 0's in the construct over element matrix can be changed to 1's and vice versa. Tables 11 and 12 illustrate this:

C1	0	0	1	1	1
C5	0	1	0	1	0
	1	0	0	1	0

Total = 2

Table 11: Illustrating a construct by construct agreement

However if construct C5 were to be reversed, then the agreement between C1 and C5 will become higher, as shown by Table 12:

C1	0	0	1	1	1
C5R	1	0	1	0	1
	0	1	1	0	1

Total = 3

Table 12: Illustrating a construct by construct agreement with a reversed construct

At the end of this exercise, the original grid is re-arranged to place similar constructs - that is, construct with high agreement scores - next to each other. At this stage, the reconstructed grid is ready for interpretation.

Applying visual focusing in this research

After the RGT interviews, grids were constructed for each of the three staff groups. Each grid cross-referenced constructs with elements, with bi-polar constructs represented by binary notation where '1' represents a construct and '0' its polar opposite. There are three elements in all three grids. For each staff grid, the analysis processes produced:

- A practitioner group by practitioner group agreement matrix
- A construct by construct matrix
- A reconstructed grid showing similar constructs next to each other

Limitations of grid analysis techniques

One main limitation of grid analysis is that the data collected from participants can appear "flat", without much structure [Maiden and Rugg, 1996]. This can hide some of the complexities of the results reported.

There are also limitations in representing elements and constructs on scales. For example, when the number of constructs or elements are too few, then the agreement scales derived for either, may not adequately reflect the subtle differences between groups of elements and or constructs.

3.4.3 Verifying research results

Reliability and validity tests are two ways of determining, accuracy, reproducibility and generalise-ability of the results of a study [Creswell, 1994]. According to Creswell [1994] there is no consensus on verifying qualitative research. Over the years several approaches to attaining reliability and validity have evolved [Creswell, 1994]. These are different from the well-established set of procedures in quantitative research.

Establishing reliability and validity in qualitative research is less straightforward than in quantitative research. This difficulty is discussed by Krippendorf [1980] who suggests that the process of defining categories in context analysis makes it unrealistic to subject category data to the same tests of accuracy as quantitative data. Nevertheless, Krippendorf [1980] recommends that the processes involved in producing the categories, should at least be reproducible. That is, different researchers at different times should be able to arrive at similar results following the same research process.

Most of the steps taken to ensure internal validity in this research have been discussed in relation to the particular research process. In this section, this research discusses general issues relating to data reliability and verification issues relating to the external validity of research findings.

- Data reliability

One area of internal reliability relates to the nature of the data collected in research. According to Krippendorf [1980] the ultimate aim for assessing the reliability of the data used in research is to ascertain whether the findings that are reported can be accepted as fact. In the case of this research it means how are the findings being reported valid to the software engineering community? To establish this data reliability, it is often necessary to conduct some statistical tests. Such statistical tests are often difficult to conduct with qualitative data. This difficulty is faced in this research since most of the data used is qualitative: In particular, category data.

One of the most widely used tests of reliability for category data is the Chi Squared test. Chi Squared determines the extent to which differences reported in the data can be said to be real differences as opposed to differences that have occurred by chance. In effect, the process of calculating a Chi Squared statistic does test whether differences in the sample of data being reported are big enough to indicate that the samples have come from different populations. In the same vein, the Chi Squared statistic can be used to confirm that the differences in samples used in research are not big enough to render the samples as significantly different. That is, as having come from two different populations. Appendix F demonstrates the calculation of Chi Squared using similar data as collected in this research.

Unfortunately, this research is unable to test for reliability in the data collected mainly through focus group sessions using the Chi Squared test. This is because the assumption of independence that underlies the use of the test on category data does not apply for the mainly focus group data used in this research. These assumptions are that [Iversen and Gergen, 1997]:

- Each participant does not cite more than one category item.
- The total number of category items cited should equal the total number of participants.

Nevertheless, this research has confidence in the data reported, even though there are no independent indices to show for such reliability. It is notoriously difficult to derive any reliability indices when working with focus group data. This research suggests that the reliability precautions taken in the data collection and analysis process should compensate for the lack of this stand alone reliability index.

In fact Krippendorf suggests that insisting on standards for data reliability should not be accepted as ad hoc, but should relate to the requirements of validity that are imposed upon the research results [Krippendorf, 1980]. So that if the costs of drawing strong results from the research are very high, for example, life threatening, then rigid standards should be imposed. In this research, even though the research findings can be very important to the practice of SPI in companies, the absence of reliability index should not invalidate the findings.

- External validity

External validity in research is the ability to generalise the findings of the research to the community being researched as a whole. In this research some significant issues are identified that can be generalised to the entire community of SPI practising software companies.

However, two factors determine how valid the research results are externally [Creswell, 1994]:

- How representative is the sample of participants of the population being researched
- How possible is it to replicate the same research to arrive at the similar results.

Sampling validity

Sample validity in this research is explored from two positions. How representative is the sample of companies used in the research to the population of UK software companies in

general. Also, how representative are the groups of practitioners sampled in the companies of software practitioners in those companies in general.

How representative is sample within software industry? - The extent to which the sample of participants in a research adequately represent the target population, gives the results of that research external validity [Krippendorf, 1980]. The target population in this research is SPI practising companies.

This target population was identified using a SPI mailing list. The thirteen companies used in this research are those among the companies that responded to the initial invitation to participate. By volunteering to participate this sample becomes a self-selecting sample.

Self-sampling, as opposed to random sampling, though more practical, is often prone to bias [Krippendorf, 1980]. In this research because the sample of companies has been extracted from an original self-selected group, it is important that it is not biased through the over representation of one particular category [Coolican, 1990]. This research addresses this issue of over representation by using a sample of companies of varying operational complexity, size, nature of business, type of applications, etc from the initial group of companies.

Sample size is another source of bias. There are thirteen companies in this sample of participating companies. It is important to ascertain whether this sample is large enough to minimise the likelihood of bias. Generally, the larger the sample the less likely the sampling bias [Coolican, 1990]. However, it is difficult to establish the exact size of the UK software industry. This is because software operations in the UK are varied. For example, there are companies that are dedicated to software development only, whereas other companies have dedicated software development departments. Again this research suggests that the variety in company type, size, age, operational complexity, etc can limit sample bias which may be possible due to the uncertainty about size of the software industry.

A profile of companies involved in this research is presented in Appendix B.

How representative are samples of practitioners within companies? - It is also important that the practitioners sampled within companies are representative of practitioners in those companies as a whole. In this research, samples of between four to six practitioners were chosen for each staff group. For most of the companies in this research, these samples have been randomly chosen within the stratified groups of developers, project managers and senior managers. Random sampling is where each member of the target population has an equal chance of being selected [Coolican, 1990].

Random sampling is achieved for all developers and project manager groups. However for some senior manager groups, the samples are self-selecting as there are usually too few senior managers to make up for the sample size.

Overall, this research suggests that the process followed to select participants for this research makes the sample used fairly representative of software companies and software practitioners in the UK, as a whole. According to Coolican [1990], a truly representative sample is an abstract ideal. One that is realistically impossible to attain. As a result, the researcher should focus on removing as much of the sample bias as possible [Coolican, 1990]. The precautions taken in this research are sufficient to make the sample of companies and practitioners fairly representative of the groups they represent.

Replication

The findings of a research are reliable when they can be repeated [Coolican, 1990]. According to Coolican, to satisfy conditions of replication, it should be possible for the research process to be followed exactly by another researcher. This can be achieved by standardising the research process, standardising the research circumstances and applying the research procedure equally to all participants in the research [Coolican, 1990].

Based on the above definition, the main findings of this research can be replicated. This is because the process of research is standardised throughout. Therefore given the same staff groups, it is highly probable that another researcher can reproduce the concerns about SPI identified in this research.

It may, however, not be entirely possible to reproduce, in great detail, all the research findings. This is because time can influence software practitioners' concerns about SPI. For example, at the time of data collection, the state of the UK software industry, with respect to job security, was different from what it currently is. As a result, some of the finer research findings that relate to job security may not be the same if this same research is conducted today.

Overall, there are sufficient external validity and reliability issues addressed in this research that make the findings generalise-able. However, the real strength of the findings appears to be the interpretation of the findings as they relate to the thirteen UK companies and eighty SPI managers involved. According to Merriam [1988] the intent of qualitative research is more to form a unique interpretation of events, than to generalise the findings. Also, any limitations in verifying this research are likely to be more external than internal. This is a common limitation with most qualitative research which are usually better at addressing internal validity issues than external ones [Creswell, 1994].

3.4.4 Limitation of research design

- The data collected in this research characterises practitioners' perceptions. These perceptions have not been verified directly. This may imply that what practitioners say motivates or de-motivates them may not necessarily be what actually motivates or de-motivates them. Furthermore, practitioners' perceptions may not be accurate. However, opinion data is widely used in software engineering research. For example, in [Dyba, 2000], Dyba describes the use of opinion data from 120 software and

quality managers to develop an instrument for measuring key factors of success in SPI.

- Although over 200 practitioners were involved in the studies on motivators and de-motivators, the data collection points used were the forty-nine focus groups. This means that the data sets used for these studies are relatively small. This research chose not to scale up the data points to reflect participants rather than focus groups, but this means that it is sometimes difficult to generate significant relationships with a relatively small data set.

3.5 Other research approaches

There are other research approaches that could have been used in this research. These approaches were considered but discarded. In this section, these approaches are discussed and reasons are presented for why they were not adopted.

3.5.1 Grounded theory

Grounded theory is described as the discovery of theory from data through the process of constant comparison [Glaser and Strauss, 1967]. The Grounded theory approach to research advocates that data relevant to a particular phenomenon be collected and studied to find out if any theories or hypotheses can be derived from the data [Tesch, 1990]. In this sense Grounded theory offers an alternative to other traditional research approaches whereby researchers collect data with the aim of verifying or rejecting an existing theory. According to Tesch [1990] Glaser and Straus suggested this approach to research because they became increasingly concerned about the lack of the human input in traditional research approaches that relied too much on the verification of theories, through the use of statistics.

Theories or hypotheses are formed with the Grounded theory approach based on the key elements of concepts, categories and propositions. Concepts are the basic analysis of the data from which theory is built, categories are the higher abstract classes into which concepts can be grouped and propositions indicate the relation between categories and their concepts and also the relationship between discrete categories [Pandit, 1996]. The theory building process involves five analytic phases which are research design, data collection, data ordering, data analysis and literature comparison [Pandit, 1996].

The application of grounded theory in software engineering research is varied, innovative and useful. For example, Pandit [1996] applied grounded theory to generate theoretical frameworks for depicting corporate turnaround. The research process in Pandit's study is described in nine stages [Pandit, 1996]:

i. A review of the technical phases
ii. Selection of cases
iii. Developing a data collection protocol
iv. Entering the field
v. Data ordering
vi. Analysing data relating to the first phase
vii. Theoretical sampling
viii. Reaching closure
ix. Compare emergent theory with existing literature

The last of these nine steps illustrates a major emphasis of grounded theory approach to research. That is, the emergence of theory from the data collected in the research. So that whereas this research attempts to explore themes in existing theory in order to prove the research hypothesis, grounded theory involves a process whereby the research itself produces new theory to support a hypothesis by comparing that new theory with existing literature.

The grounded theory approach has not been used in this research for the following reasons. First, this research is investigating specific issues on software practitioners' motivators and de-motivators for SPI and thus requires a research approach that has more structure and offers more certainty than a grounded theory approach. Grounded theory is iterative and contains long periods of uncertainty [Pandit, 1996]. Secondly, a grounded theory approach is time consuming and therefore becomes inappropriate for this research due to the time constraints.

3.5.2 Ethnography

Ethnography is an observational research approach, which can be considered to have unobtrusive participant observation as its core method. In ethnographic research the researcher becomes part of the participating sample without influencing the outcome of their activities or processes. Ethnography has evolved out of anthropology where the researcher observes without participating.

> "The ethnographer participates in people's daily lives for an extended period of time, watching what happens, listening to what is said, asking questions: in fact collecting whatever data are available to throw light on the issues with which he or she is concerned"
>
> [Hammersley and Atkinson, 1983]

The use of ethnography in software engineering is a novel concept and most of the pioneering work has been conducted by Hovenden et al in their study of software quality practices in companies [Hovenden et al., 1996].

An ethnographic approach was considered for this research but discarded for the following reasons. Even though the ethnographic approach enhances the collection of data on participants' perceptions, the major subject of data collection is the behaviour of participants. This research is indirectly interested in the behaviour of software practitioners, but the main focus of data collection is software practitioners' perceptions of what motivates and de-motivates them for SPI. As a result, ethnography does not seem the most ideal research approach.

Also, there are inherent difficulties in collecting behavioural data on such a large-scale, which would require longitudinal studies in the software companies. There are also logistical complexities with analysing such a large amount of data,

Pragmatically, an ethnographic approach would be difficult to adopt for this research because of the time and effort constraints.

The following chapter presents the first of four studies conducted in this research. This study examines SPI managers' perceptions of the motivators and de-motivators for SPI.

Chapter Four: SPI managers' perception of motivators and de-motivators for SPI

4.1 Introduction

This chapter presents a study of SPI managers' perceptions of the motivators and de-motivators for SPI. The study was conducted using a questionnaire survey to investigate experiences and opinions of eighty SPI managers from a wide range of UK companies. The results of the study are presented in simple statistical analysis.

4.1.1. Study aims

The aim of this study is to answer the research question:

> What perceptions do SPI managers have of the motivators and de-motivators for SPI?

This study identifies what perceptions SPI managers have of the motivators and de-motivators for SPI by investigating their experiences and opinions of SPI. Such a study sets the overall research into context. It shows what SPI managers, who are the beneficiary of the overall research, perceive of software practitioners' support for SPI. It also highlights the expectations that SPI managers have of SPI programmes. Overall, this study provides a 'snapshot' of the issues that underpin software practitioners' motivators and de-motivators for SPI in UK companies.

4.1.2 Rationale and background for investigating SPI managers' perceptions of SPI

The rationale for this study is that:

A study of SPI managers' perceptions provides insight to SPI practice in companies. Such a study can, in particular, provide insight into how SPI is being supported in companies.

and so contribute some assessment of the effectiveness of SPI uptake in a sample of UK companies.

Also, a study of SPI managers' perceptions identifies the experiences and expectations of people with responsibility for managing SPI in companies. This research suggests that such information can assist SPI managers, overall, to design better strategies for SPI. This is because such information can highlight areas where SPI managers experience frustrations with SPI. In particular it can establish whether developers and senior managers are being supportive of SPI. These areas then form the basis of investigation for the overall study.

The issues underpinning motivators and de-motivators for SPI explored in this study can be classified into three areas of SPI: the inputs for, implementors to and outputs of SPI. This research has classified these three areas in terms of the traditional systems model [Curtis, 1989] of inputs, processes and outputs.

SPI managers' experiences and expectations of SPI serve as the inputs to SPI. Whereas SPI managers' perception of senior management support for SPI, developers' buy-in to SPI and the approaches to implementing SPI in companies represents an exploration of the implementors to SPI. Finally this study examines the outputs to SPI by exploring SPI managers' perception of the impact of SPI on software quality and also the cost effectiveness of SPI.

Questionnaire survey is used as the instrument of data collection. A copy of the questionnaire in Appendix D was mailed to SPI managers at 1000 companies. Two hundred replies were received of which eighty were fully completed questionnaires.

Overall, the findings from this study provide context for the whole research.

This rest of this chapter is structured as follows:

Section 4.2 presents a profile of the sample of SPI managers and explains some of the organisation of the analysis. Section 4.3 presents analysis of survey data. Section 4.4 discusses the analyses. Section 4.5 summarises the results of this study and concludes this chapter.

4.2 Sample profile

This section presents a profile of SPI managers and explains the use of sub-sample analysis to support findings.

4.2.1 Respondent profile

This section presents some demographic features of SPI managers who responded to this survey.

- Job titles

Table 13 shows that people who assume SPI management roles in companies have a variety of job titles. However, throughout this research, they are referred to as SPI managers.

Job title	No. in sample	% in sample
Quality director/manager/co-ordinator	28	35
Development director/manager/co-ordinator	12	15
SPI managers	10	12.5
Software manager/director	6	7.5
Managing Director/General manager/Partner	6	7.5
Project director/manager	4	5
Senior staff/software engineer	4	5
Technical manager/specialist	4	5
Programme manager	2	2.5
Other	4	5
Total	**80**	**100**

Table 13: People with SPI responsibility

- Career history

Figure 9 shows that more than half of respondents in this study have over fifteen years experience in the software industry. Figure 9 also shows that many have not been with their present companies for longer than five years. This suggests that most SPI managers are experienced computing professionals who have made the transition to SPI later in their career. The late up-take of SPI responsibilities could be attributed to the fact that the concept of SPI is a relatively recent phenomenon.

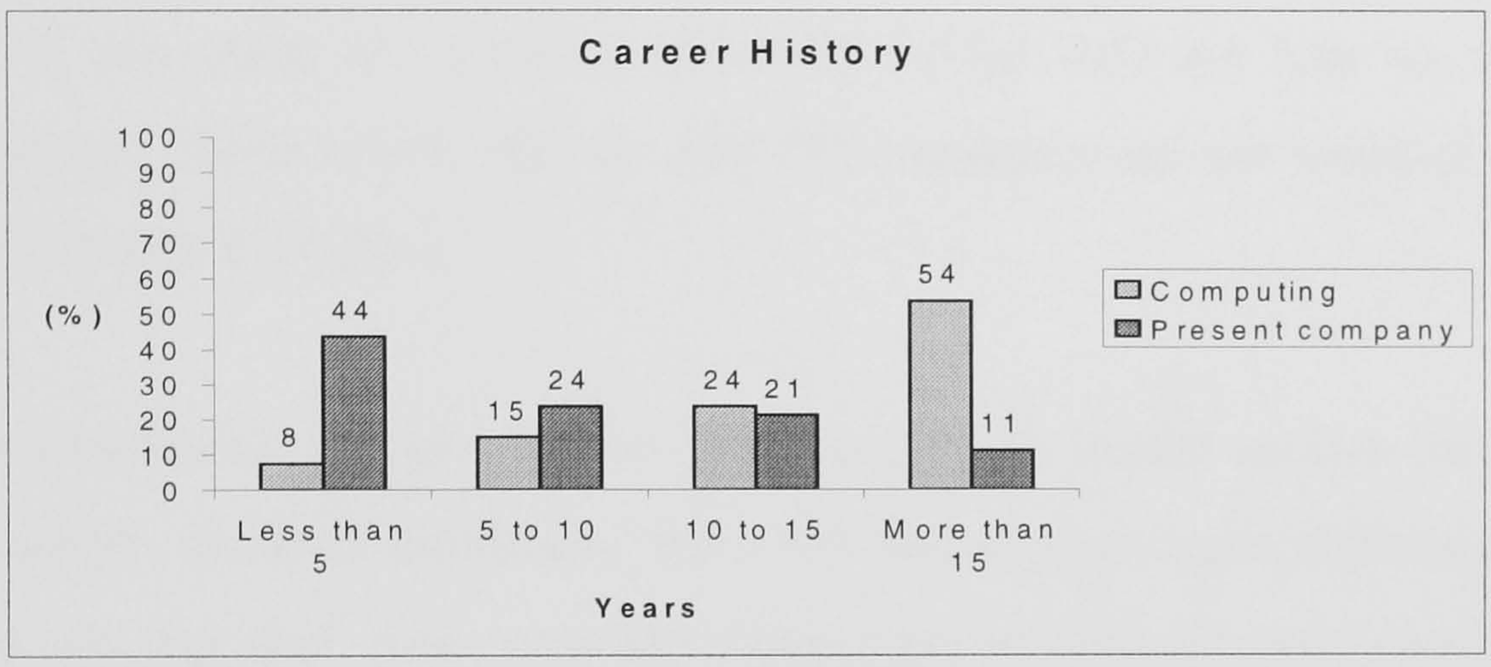


Figure 9: Distribution of years served in computing and present company

- Education.

More than 80% of SPI managers have obtained at least Bachelors degree, but only 35% of these degrees are from computing disciplines. A majority of the remaining 65% had degrees from other science disciplines with engineering and the natural sciences being most predominant. Studies described in later chapters show that some practitioner groups are de-motivated by managers who do not have a technical background.

- Professional body membership

Overall, 50% of respondents were affiliated to professional technical bodies. This research suggests that this may indicate that SPI managers' educational background may be technical too.

4.2.2 Sub-Sampling

In this study, further analysis is conducted on the perceptions surveyed by separating the responses of practitioners who report that they are highly familiar with SPI from those who report a low familiarity. This research suggests that software practitioners' familiarity with SPI qualifies their opinions on SPI, implying that respondents who are more familiar with SPI are likely to have more reliable opinions. Consequently, 62% of the sample in this study are familiar with SPI whilst 18% are less so, with the rest remaining neutral. Some of the reasons why SPI managers are not familiar with SPI are discussed in Section 4.3 below.

Despite this breakdown, the perceptions presented in the results section are generally of the whole sample, because familiarity with SPI makes significant differences to only a few of the results reported. As a result sub-sample perceptions are only presented in cases where the difference between familiar and unfamiliar managers is significantly different.

The three sets of opinion: from overall sample, 'familiar managers' and 'unfamiliar managers' are presented in a comparative table as Appendix G.

4.3 Analysing SPI managers' perceptions

Analyses of SPI managers' perceptions are presented in the context of the existing conditions within a company that influence SPI success, the conditions that are needed for successful SPI and the measure of the outcome of successful SPI. In short the inputs, implementors and outputs.

4.3.1 Analysis of the inputs

The following are a summary of responses to questions B1, B2 and B9 of section ii of the questionnaire (see Appendix D).

- Familiarity with SPI

Figure 10 shows that the majority of managers with SPI responsibility are familiar with implementing SPI initiatives. 62% of respondents indicated either strong or very strong familiarity with implementing SPI programmes. However, 18% said they were not familiar with SPI. Closer analysis shows that these 'unfamiliar managers' are not necessarily new to their positions. For example, some managers who report least SPI familiarity had spent between fifteen and twenty-seven years in their present jobs. Often these managers do not have official SPI job titles despite having responsibility for SPI. This may mean that they have recently been given SPI remits within an existing role. This research speculates that this may be related to other factors like the lack of senior management support that these managers also report in later sections. Indeed one respondent emphasised this by saying…

> *"I have found that many SPI managers are "lumbered" with the position and are expected to cope rather than being given the support and resources to make it worthwhile"*

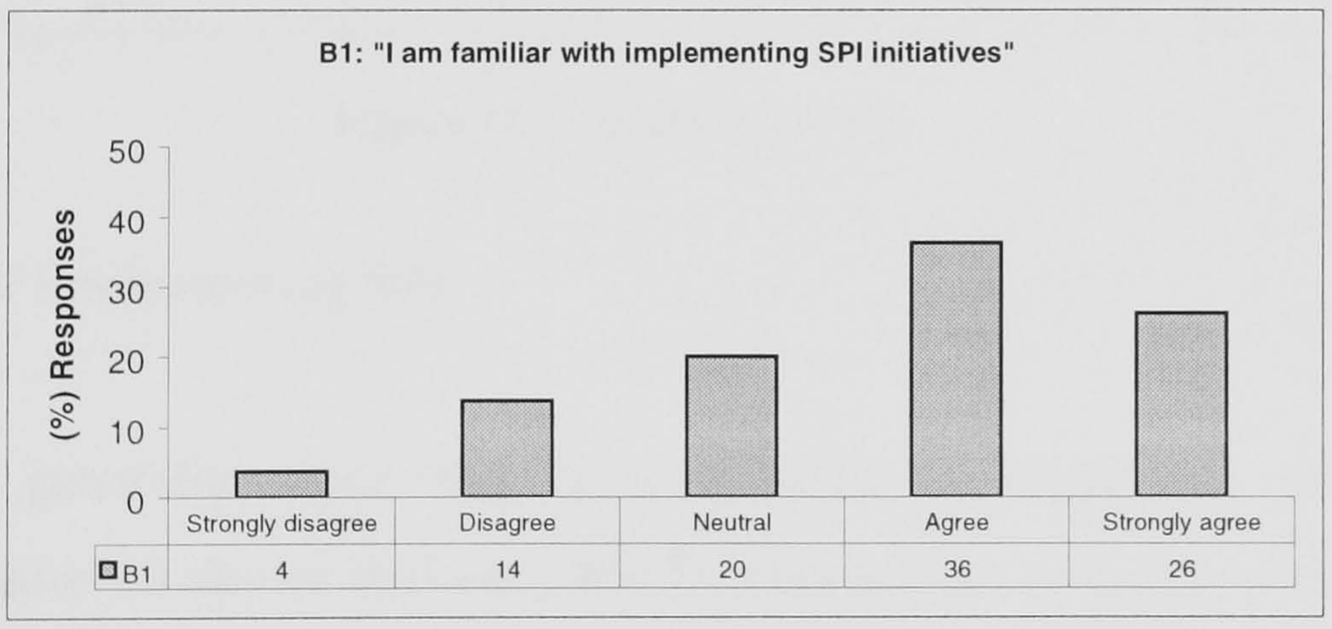


Figure 10: Familiarity with SPI

- Experience of SPI

More than 65% of respondents reported positive experiences of SPI. Figure 11 shows that the percentage of negative experiences of SPI was negligible at 7%. Furthermore, 82% of managers who are familiar with SPI report positive experiences of SPI, whilst 14% of

managers who are unfamiliar with SPI report negative experience of SPI. Overall, these results suggest that SPI managers' general experience of SPI is positive.

However, a few managers who responded positively to also had issues to raise about SPI. For example, one manager said:

"My experience of championing SPI has been mixed. Initially we had an excellent SPI programme. However changes in the company have made implementing this far harder."

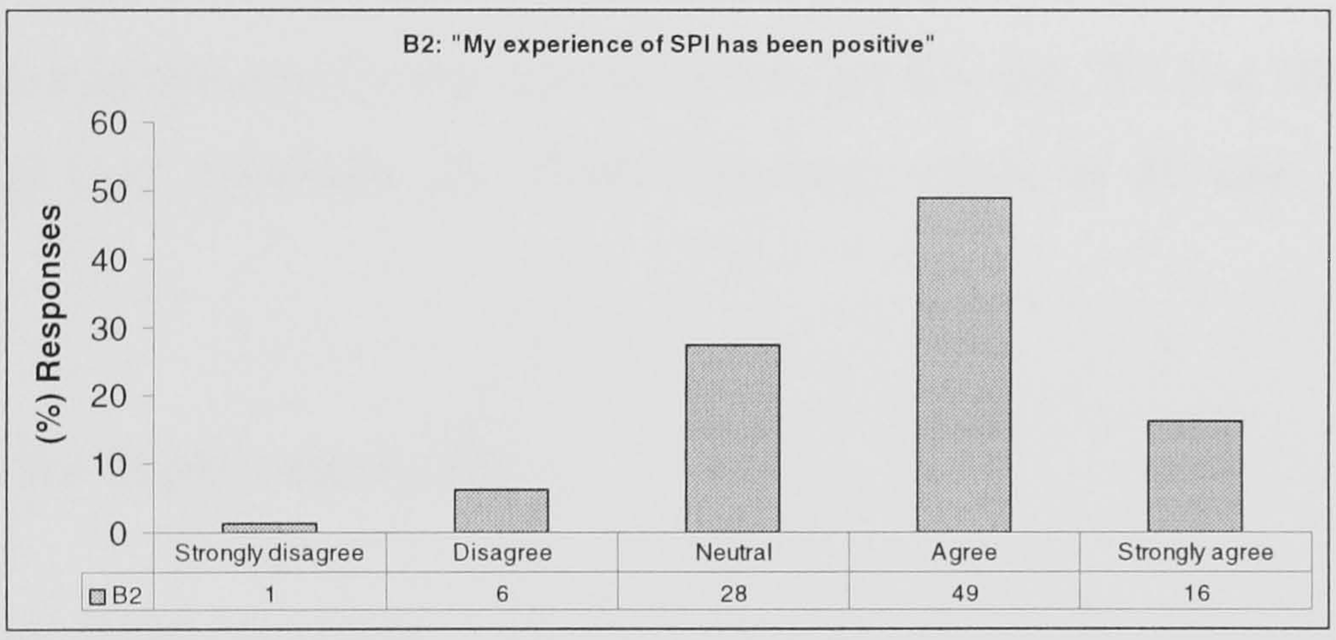


Figure 11: Experiences of SPI

- Difficulty of implementing SPI

SPI managers generally agree that it is difficult to implement an effective SPI programme. Figure 12 shows that only 8% of respondents disagree with this statement. Sub-sample analysis based on reported familiarity with SPI indicates that both familiar and unfamiliar managers agree that SPI programmes are difficult to implement, with 80% of familiar managers in agreement with the statement. This finding indicates that even though a strong majority of respondents have reported positive SPI experiences, they are still objective about how difficult it is to implement SPI and therefore have realistic expectations of SPI.

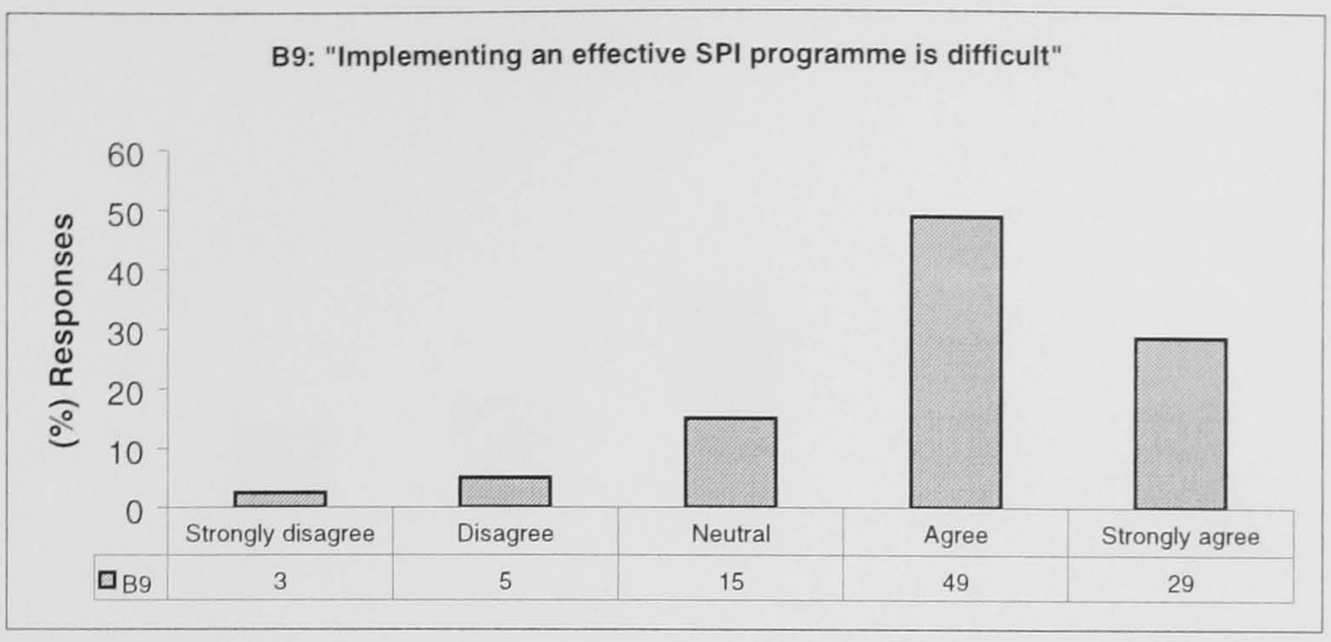


Figure 12: The difficulty of implementing SPI

4.3.2 Analysis of *implementors*

The following is a summary of responses to questions B4, B5, B7 and B8 of section ii of the questionnaire (see Appendix D). These responses relate to the *implementors* to SPI programmes.

- Approaches for implementing SPI

Figures 13 and 14 show that SPI managers' perceptions of approaches to implementing SPI are varied. Less than 45% agree that SPI should be implemented by a top-down approach. The largest single percentage of responses, 30%, was neutral. Indicating that there is a sizeable proportion of SPI managers who are undecided as to whether top down is the appropriate approach to implementing SPI or not.

Further analysis of responses from managers familiar with SPI reveals varied opinions. 48% agree that SPI should be implemented top down, 30% disagree and 22% remained neutral. Again, this implies that SPI managers' opinions are not overwhelmingly strong as to whether SPI should be implemented top down or not.

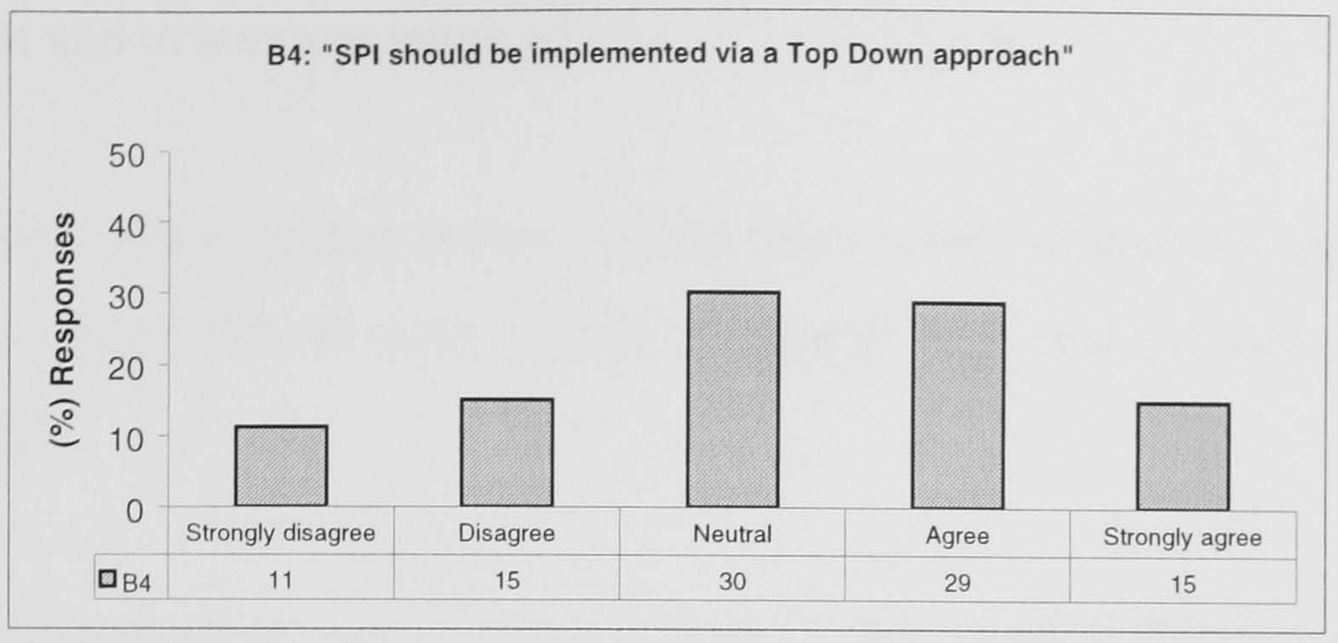


Figure 13: Top-down implementation of SPI

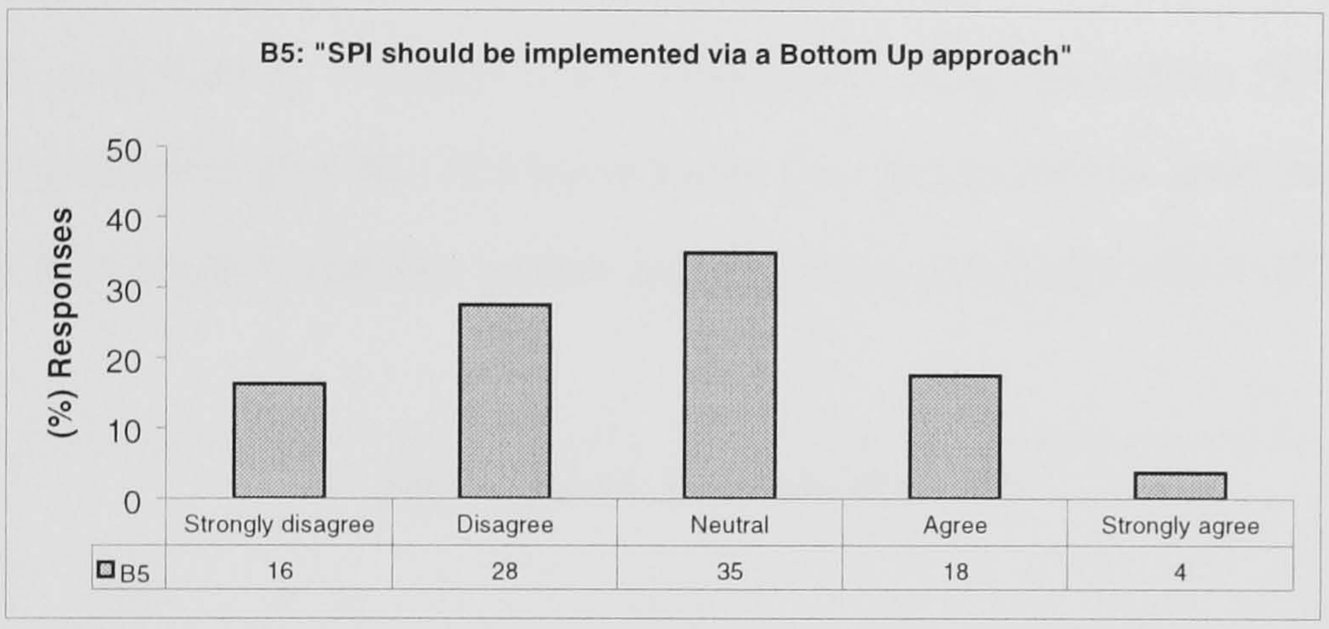


Figure 14: Bottom-up implementation of SPI

SPI managers are also undecided about bottom-up approaches to implementing SPI. Only 4% of respondents strongly agreed that such an approach should be used, with the single highest percentage, 35%, remaining neutral.

This result may reflect a problem with the way the statements were presented in the questionnaire. SPI managers may prefer a combined top-down/bottom-up approach to implementing SPI. In fact this is substantiated by annotations provided by some respondents:

"I think you need to use both approaches".

"It depends on the culture. No 'right' answer".

- Management and grassroots support

SPI managers' perception is that senior management is not supportive of SPI. Figure 15 shows that less than 50% of respondents indicated that their senior managers were supportive of SPI.

This finding is important as successful accounts of SPI suggest that senior management support and commitment for SPI is critical.

This research suggests that senior management motivation for SPI may often be short term and profit motivated, whereas SPI managers may perceive SPI as long term investment. This potential conflict between these two perspectives may explain why most SPI managers in this study consider senior managers as not supportive of SPI.

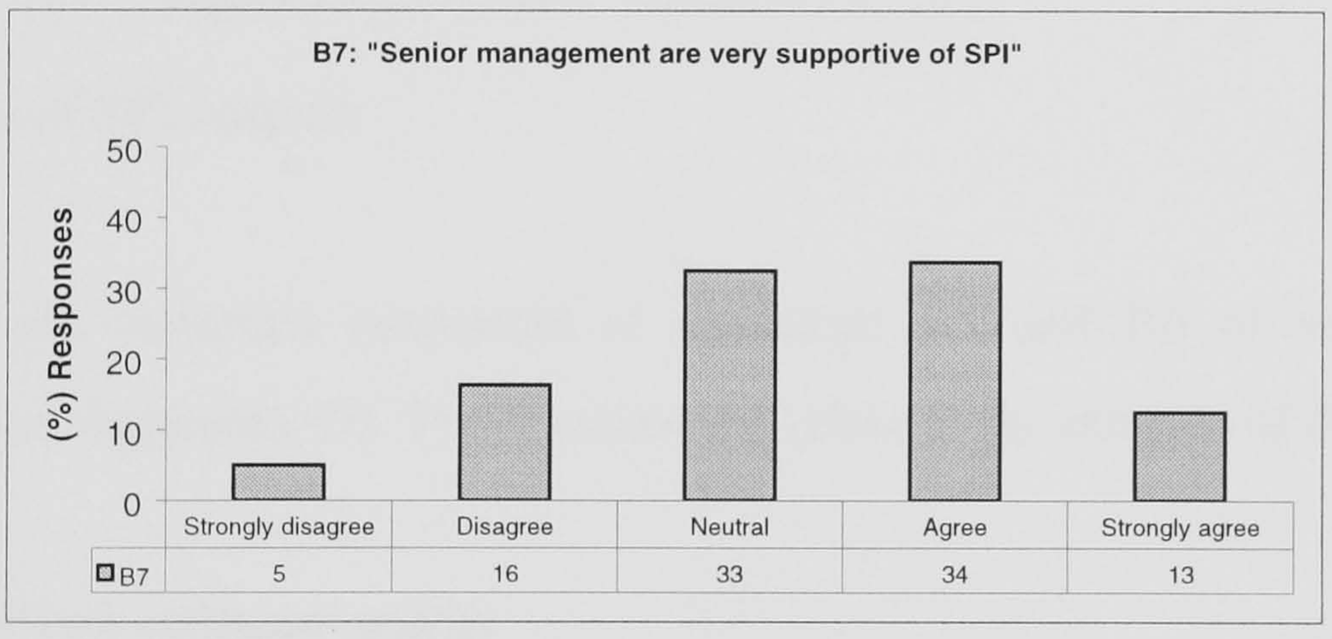


Figure 15: Senior management support of SPI

Figure 16 shows that developers' support for SPI is weak. Less than 40% of respondents felt that grassroots practitioners were enthusiastic about SPI. One respondent added that developers were *"sceptical if it means more work, but want to see improvement"*.

This is another important finding in view of what the literature and published case studies advocate about grassroots support and buy-in for SPI. This research suggests that this finding indicates problems with grassroots support for SPI in UK companies.

This research suggests that because senior management commitment and grassroots support for SPI are two critical factors for SPI success in companies, the above findings make it necessary to conduct more studies to establish why senior managers and grassroots practitioners appear not to be supporting SPI. This research suggests that these findings on senior management commitment and grassroots support are major justifications for the rest of the studies in this research.

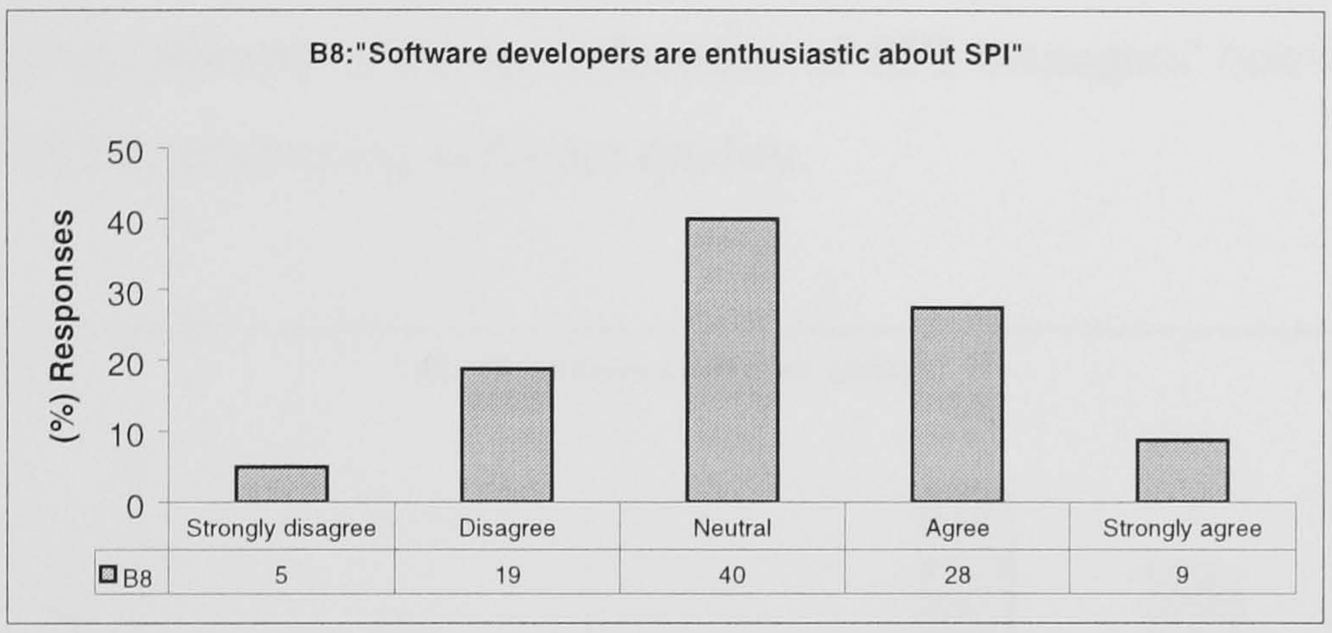


Figure 16: Developers' enthusiasm for SPI

4.3.3 Analysis of SPI outputs

The following are summary responses to questions B3 and B6 of section ii of the questionnaire (see Appendix D). These responses relate to the outputs of SPI.

- Impact of SPI on software quality

Figure 17 shows that most SPI managers acknowledge that SPI is an effective approach to improving the quality of software. This is one of the strongest positive responses in the study. More than four out of five respondents say that SPI leads to better software quality. The proportion of negative responses is negligible and is not from the same respondents as reported negative experiences of SPI in question B2. This shows that even managers whose experience of SPI has not been positive, still believe that SPI can lead to better quality software. It suggests that the negative experiences reported in question B2 are not negative evaluations of SPI as such, but rather of particular implementation of

SPI. It is also interesting because it supports the view that quality problems in software development can be solved by focusing on the processes [Humphrey, 1989].

Further analysis reveals that 90% of managers who reported familiarity with SPI agree that SPI leads to improved software quality, with a further 64% of unfamiliar managers also agreeing with the statement.

Overall, this finding presents a strong indication of SPI managers' conviction about the effectiveness of SPI in improving software quality.

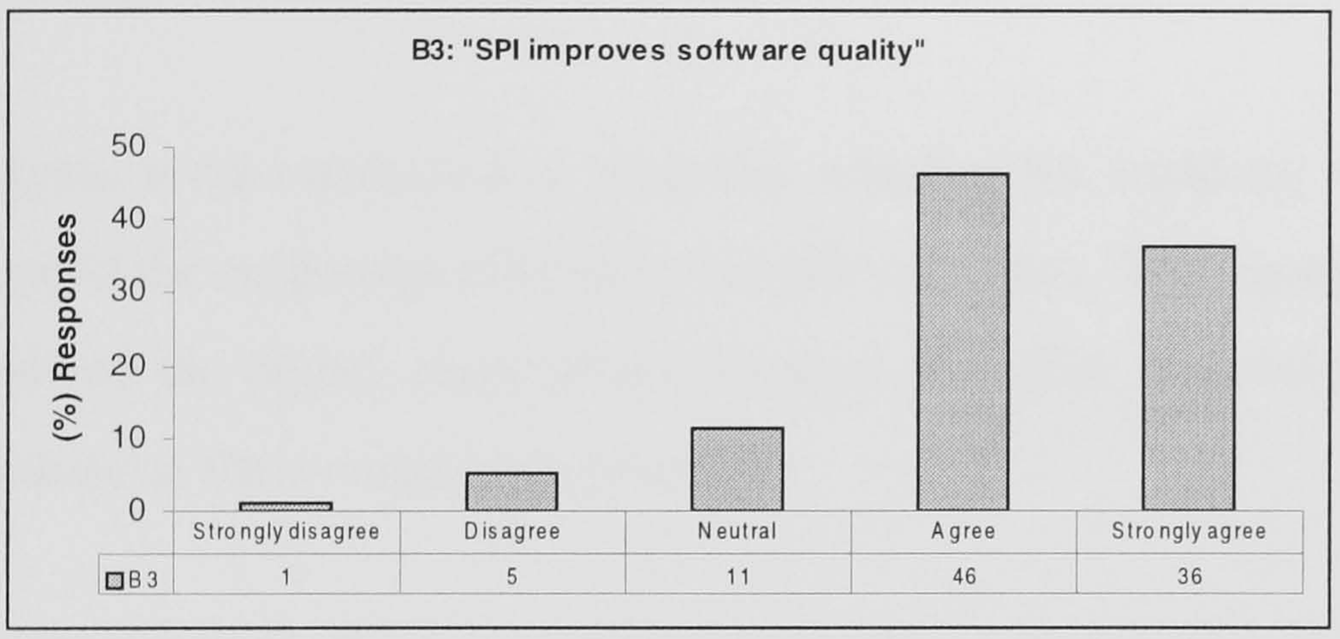


Figure 17: Software quality and SPI

- Cost/Benefits of SPI

Figure 18 shows that 74% of SPI managers acknowledge that SPI has benefits that outweigh costs. Only 7% of respondents disagree with this.

This is an important finding for two reasons. Firstly, because it supports what companies undertaking successful SPI programmes have been reporting over the years. In fact many successful companies report a healthy return on investment. Secondly, it is a view that reflects the general literature on cost effectiveness of SPI programmes.

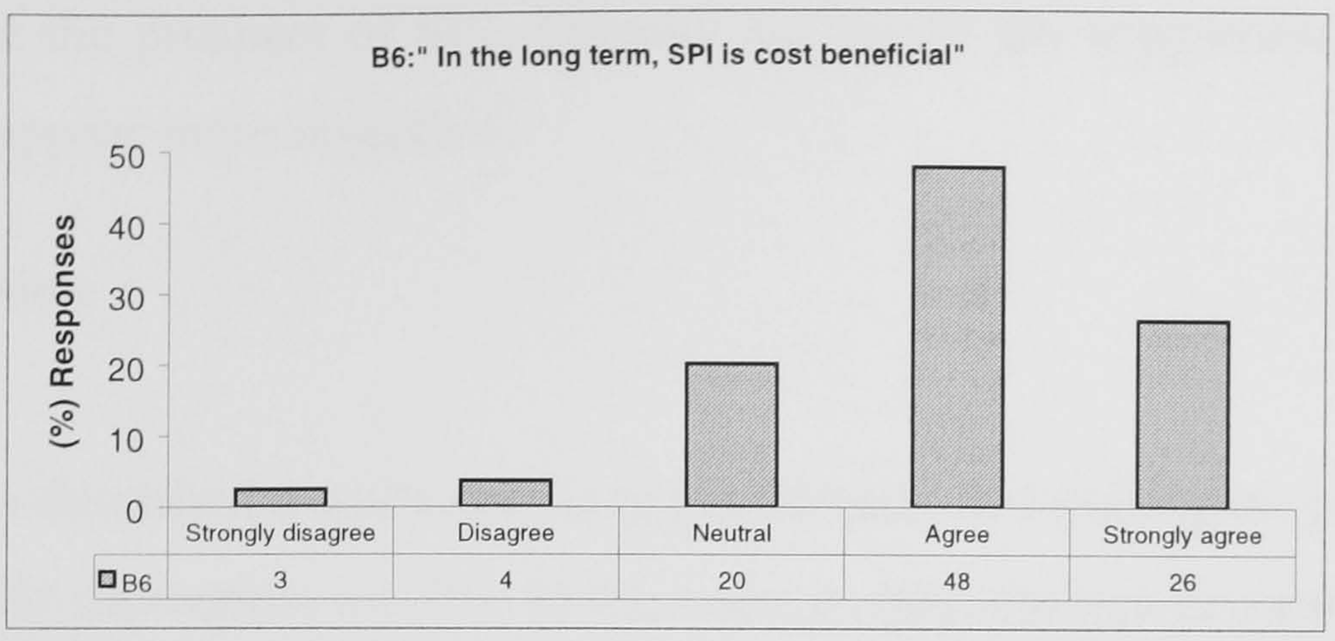


Figure 18: Cost benefits of SPI

4.3.4 Process maturity

Correlation analyses were conducted to establish whether the maturity of a company's processes influenced the responses offered by its SPI managers. This analysis was carried out with fifty out of the eighty respondents because the other respondents declined to estimate the maturity of their company's processes.

Table 14 shows a correlation matrix of responses B1 to B9 against company process maturity.

	Correlation between process maturity and responses (n=50), df = 48 Responses B1 to B9								
	B1	**B2**	**B3**	**B4**	**B5**	**B6**	**B7**	**B8**	**B9**
Estimated CMM	0.211	0.233	-0.010	-0.171	-0.161	0.070	0.159	0.208	-0.270

(where r critical is 0.273, for df=50, at $p<0.05$)

Table 14: Correlation matrix of process maturity and questionnaire responses

Table 14 shows that none of the correlation coefficients are significant. This result indicates that SPI managers' responses to questions do not relate to the maturity of the processes in their companies. This may further suggest that the success or otherwise of SPI in individual companies has not biased the perceptions collected in this study in one way or the other. So that SPI managers from more successful companies are not being more optimistic and neither are managers from the less successful companies being less

optimistic about the prospect of SPI. Thereby leading to the suggestion that the results being reported appear more objective.

4.4 Conclusion

This chapter has described a study conducted to identify SPI managers' perception of the motivators and de-motivators for SPI. In this section, this research presents a summary of the study findings and shows how these findings answer the research question:

> What perceptions do SPI managers have of the motivators and de-motivators for SPI?

4.4.1 Summary of findings

- Experience of SPI

SPI managers generally report positive experiences of SPI. The findings also show that SPI Managers with a high familiarity of SPI are more positive about their experiences of SPI.

On the whole, opinions of SPI managers who are familiar with SPI appear stronger than the opinions of the overall sample. Managers who are familiar with SPI consistently confirmed the general view of the whole sample by reporting stronger percentages. This shows some consistency in the perceptions elicited in this study and also affirms the notion that SPI managers' experiences and familiarity with SPI informs their perception of SPI.

- Expectations of SPI

SPI managers are optimistic about the success of SPI. They generally have positive expectations of SPI, but acknowledge that there are significant difficulties with managing SPI programmes. Some SPI managers indicate that some of the difficulties with SPI are

caused because software projects often take priority over improvement practices. On the whole, SPI managers have the perception that there are effective ways of managing SPI programmes successfully.

- SPI impact on software quality

SPI managers' positive expectations of SPI generally also translate into their perceptions of the impact of SPI on software quality. This study shows that SPI managers perceive that SPI leads to improved product quality. This represents a positive reflection of SPI managers' expectations of SPI.

- Cost benefits

SPI managers hold the view that SPI has cost benefits. This finding is encouraging because justifying the cost for SPI is often the most difficult hurdle to overcome in implementing SPI. This research suggests that if SPI managers are convinced of the cost benefits of SPI, then it is likely to make the process of convincing senior management easier. Again, this finding indicates that SPI managers have positive expectations of SPI.

- Senior management support

In this study, SPI managers do not think that senior management is supportive of SPI. Less than half of respondents indicated that senior management was supportive of SPI. This research suggests that this is an important finding because of how critical senior management support has been reported to be to SPI success.

- Developers' support for SPI

SPI managers also indicate that developers are not enthusiastic about SPI. This finding shows that developers are not supportive of SPI. This is another important finding because of the importance of grassroots support for SPI success.

Overall, SPI managers' perceptions indicate that there is insufficient support for SPI from practitioners at both grassroots and senior management levels of their companies. These finding are important indicators for the remaining studies in this research.

- Process maturity

Overall, SPI managers' responses bear no relation to the current maturity of the processes in their companies. This finding may imply that the responses provided by SPI managers are drawn from their longer-term experience of SPI as opposed to what their current circumstances with SPI may reflect. As a result, this research suggests that this finding gives more confidence to the data provided by SPI managers.

4.4.2 Questions raised from findings

Two findings from this study on SPI managers' perceptions of the motivators and de-motivators of SPI are of significant importance to the overall research. These findings relate to support for SPI from senior managers and developers.

1 Senior managers are not supportive of SPI

2 Developers are not enthusiastic about SPI

This research suggests that these two findings are the strongest indication of the state of motivators and de-motivators for SPI discerned from this study. They indicate that software practitioners of all levels are not sufficiently motivated to support SPI and are being de-motivated from supporting SPI. These findings suggest that:

- Factors that can improve practitioners' support for SPI have not been sufficiently addressed.
- There may be factors that are hindering practitioners from supporting SPI

It is therefore important to identify, from software practitioners, what the factors are that can improve their support for SPI. In Chapters Five and Six, this research describes studies conducted to identify these factors: software practitioners' motivators and de-motivators for SPI.

Chapter Five: Analysis of motivators

5.1 Introduction

This chapter presents a study of software practitioners' motivators for SPI. The study was conducted using focus group discussions to collect practitioners' perception of their motivators. Over two hundred software practitioners from thirteen UK companies took part in this study. Practitioners were divided into three staff groups of developers, project managers and senior managers.

5.1.1 Study aim

The aim of this study is to answer the research question:

What SPI motivators do software practitioners report?

This study identifies what motivators for SPI developers, project managers and senior managers report. It also identifies the differences and similarities in these motivators across the staff groups. Finally, it explores the relationship between the motivators reported by software practitioners.

5.1.2 Study rationale

The literature review identified that to increase software practitioners' support for SPI it is important to improve software practitioners' motivators for SPI. In order to improve such motivators, it is necessary to know and understand what these motivators are. For example, the literature review also established that the motivators for software practitioners may vary across staff groups, as a result, it is important to understand how these motivators vary with relation to the different staff groups in the study. The rationale for this study, therefore, is to explore the motivators for SPI, empirically, in order to establish what software practitioners report as motivating them in practice. The findings

from this study will then be used to refute or confirm a set of guidelines, extracted from the literature that suggest how to improve software practitioners' support for SPI.

The following is how this chapter is structured:

Section 5.2 describes the data collection and analysis process used in this study. Section 5.3 presents results of the analysis. Section 5.4 discusses the findings and Section 5.5 summarises and concludes this chapter.

5.2 Data collection and analysis process

The following is the data collection and analysis process used in this study.

5.2.1 Data collection via focus groups

In this study, forty-nine focus group sessions were held made up of twenty-one developer, sixteen project manager and twelve senior manager sessions. Each group comprised between four to six members. Each focus group lasted approximately ninety minutes.

In each session practitioners were asked to discuss the following questions:

- What are the motivators to SPI in your company?
- What would motivate you to support SPI?

Practitioner responses were audio recorded and the recordings were transcribed.

5.2.2 Data analysis: frequency analysis and smallest space analysis (SSA)

All motivators cited in response to the above questions were categorised. Inter-rater reliability was calculated to increase confidence in this categorisation process. A data matrix was constructed for each staff group to record how motivators appeared in the

groups. From the data matrix, frequencies were calculated for the occurrence of each motivator category. The data matrices were also subsequently used to populate the SSA application [Shye, 1992] to produce SSA plots showing similarities between motivators.

Below is a fuller explanation of the analysis procedure.

Stage 1 - Categories of motivators are produced using steps denoted in Chapter 3. After inter-rater reliability checks are performed to account for researcher bias an agreed list of all motivator categories is produced. Appendix H presents a definition of all motivators categorised. This list also serves as the content category dictionary to be used for the subsequent SSA plots.

Stage 2 - A data matrix is constructed mapping motivators cited to each of the three practitioner groups. Appendix I shows the data matrices for developers project managers and senior managers respectively.

Stage 3 - Each motivator is weighted for each practitioner group by frequency. Frequencies are converted into percentages. So, for example, if out of a total of twenty-one developer groups *motivator x* was identified in six of them, *motivator x* is assigned an occurrence weighting of 28.6%.

Stage 4 - Data matrices in Stage two are used to plot SSAs for developers, project managers and senior managers.

Stage 5 - The SSA plots for developers and project managers are partitioned into regions defined by the Stimulus Response Theory of Skinner [Skinner, 1976]. These plots show two regions: "rewarding" and "punitive". The plot for senior managers is partitioned into "tangible" and "intangible" regions.

5.3 Findings of practitioner motivators

This section presents findings from the frequency and smallest space analysis of practitioner motivators. The tables presented show the motivators cited by practitioner groups and the frequency they occurred. The percentage shows the proportion of practitioner groups that cited a particular motivator.

5.3.1 Developer motivators

Table 15 shows the list of motivators cited by developers. It shows that nearly half of developer groups cited visible success as a motivator. This suggests that developers wanted evidence of the success of SPI before committing to SPI. Closer examination of the data indicates that developers want evidence of success both generally and, more particularly, within their development environment.

Motivator	Occurrence in focus groups (n=21)	
	Freq.	%
Automation	2	10
Bottom-up initiatives	5	24
Communication	4	19
Compulsory	2	10
Critical mass	1	5
Eliminates bureaucracy	2	10
Feedback	4	19
Job satisfaction	4	19
Maintainable processes	1	5
Phased introduction	2	10
Process ownership	4	19
Resources	5	24
Reward schemes	3	14
Shared best practice	4	19
SPI forum	3	14
Standardisation	1	5
Top-down commitment	5	24
Training	2	10
Visible success	9	43

Table 15: Developer motivators

Table 15 also shows that a quarter of developers cite bottom-up initiatives, resources and top-down commitment as motivators. The following quotes illustrate developers' views of bottom-up initiatives and top-down commitment:

"I think it is the fact that practitioners know that they can make an input: it has been made apparent that practitioners can actually input into changing the procedures. Previously, that hasn't been made apparent."

"I think the commitment from the top is important. Top Down commitment. When a decision is put forward, it should be emphasised that it is actually good to follow through and see it happen"

Table 15 also shows that only five percent of developers cite critical mass, maintainable processes and standardisation as motivators. This indicates that these factors may not be prominent motivators for developers.

5.3.2 Project manager motivators

Table 16 presents project managers' motivators. It shows that nearly a third of project managers cite visible success and resources as motivators for SPI. Like developers, project managers also want evidence of the success of SPI. More importantly, they want the resources to enable them to carry out SPI. Other relatively highly ranked motivators were empowerment and process ownership.

Motivator	**Occurrence in focus groups (n=16)**	
	Freq.	%
Autonomy	1	6
Communication	2	13
Easy processes	3	19
Empowerment	4	25
External audits	1	6
Knowledgeable team leaders	3	19
Maintainable processes	3	19
Process ownership	4	25
Reduced admin	1	6
Resources	5	31
Reward schemes	3	19
Saleability	1	6
Top-down commitment	1	6
Visible success	5	31

Table 16: Project manager motivators

Table 16 also shows that autonomy, external audits, reduced administration, saleability and top-down commitment are the motivators least cited by project managers. Again, this shows that these particular factors may not be the most important motivators as far as project managers are concerned.

5.3.3 Senior managers' motivators

Table 17 shows motivators cited by senior managers. It shows that, like developers and project managers, the most popular motivator for senior managers was visible benefits. However, another strong motivator for senior managers was meeting targets, if SPI helps meet business targets, then senior managers will be more motivated to support SPI. One senior manager said:

> *"The fact that people are getting their portion of the product in on time and to the costs they were set is a real motivating factor. As long as we can show that process improvement is the vehicle which has enabled them to achieve what they set out to achieve at the beginning, then there is motivation in that."*

Motivator	Occurrence in focus groups (n=12)	
	Freq.	%
Career prospects	1	8
Cost beneficial	2	17
Feedback	1	8
Justifiable benefits	1	8
Maintainable Processes	1	8
Meeting targets	3	25
Process ownership	2	17
Resources	2	17
Reward schemes	2	17
Taller hierarchy	1	8
Task forces	1	8
Visible success	3	25

Table 17: Senior managers' motivators

Table 17 also shows that nearly one fifth of senior managers cite as motivators cost beneficial, process ownership, resources and reward schemes.

The motivators cited by senior managers were unevenly spread across the twelve focus groups. Indeed one senior manager focus group cited half of all motivators. Furthermore, career prospects, taller hierarchy and task forces occurred only once in the senior manager sessions. This suggests that the motivators cited by senior managers may not be that common across the senior manager group as a whole.

5.3.4 Motivators across practitioner groups

- Common motivators

Table 18 presents motivators cited by more than one practitioner group. It shows that the motivators cited by all three staff groups closely relate to SPI critical success factors reported in the literature. For example, all groups of practitioners cite five motivators: maintainable processes, process ownership, resources, reward schemes and visible success as motivators.

Motivators	Developers	Project managers	Senior managers
Communication	√	√	
Feedback	√		√
Maintainable processes	√	√	√
Process ownership	√	√	√
Resources	√	√	√
Reward schemes	√	√	√
Top Down commitment	√	√	
Visible success	√	√	√

Table 18: Common motivators

Table 18 also shows that project managers and developers cite top-down commitment and communication as motivators, whereas senior managers do not.

- Different motivators across practitioner groups

Table 19 shows motivators cited by only one type of practitioner group. It shows, for example, that only project managers cite autonomy, only developers cite bottom-up initiatives and only senior managers cite cost beneficial. Initial interpretation from this

finding suggests that whereas developers' motivators appear to address specific SPI issues, managers cite motivators that are more related to the organisation or to projects.

Motivators	Developers	Project managers	Senior managers
Automation	√		
Bottom Up initiatives	√		
Compulsory	√		
Critical mass	√		
Eliminates bureaucracy	√		
Job satisfaction	√		
Phased introduction	√		
Shared best practice	√		
SPI forum	√		
Standardisation	√		
Training	√		
Autonomy		√	
Empowerment		√	
External audits		√	
Knowledgeable team leaders		√	
Reduced admin		√	
Saleability		√	
Career prospects			√
Cost beneficial			√
Justifiable benefits			√
Meeting targets			√
Taller hierarchy			√
Task forces			√

Table 19: Motivators unique to particular practitioner groups

- Managers' perception of developer motivations

When manager groups cited their motivators, they also cited factors that they perceived would motivate developers to get involved in SPI. Table 20 shows the relationship between manager perceptions of developer motivators and what developers say motivates them.

Motivators	Developers	Project managers	Senior managers
Feedback	√		√
Maintainable processes	√	√	√
Process ownership	√	√	√
Reward schemes	√	√	√
Autonomy		√	
Career prospects		√	
Saleability			√
Taller hierarchy			√

Table 20: What managers perceive motivates developers

Table 20 shows that the developer motivators that managers know about include feedback, maintainable processes, process ownership and reward schemes. It also shows that managers inaccurately believe that other motivation factors like autonomy, career prospects and saleability motivate developers.

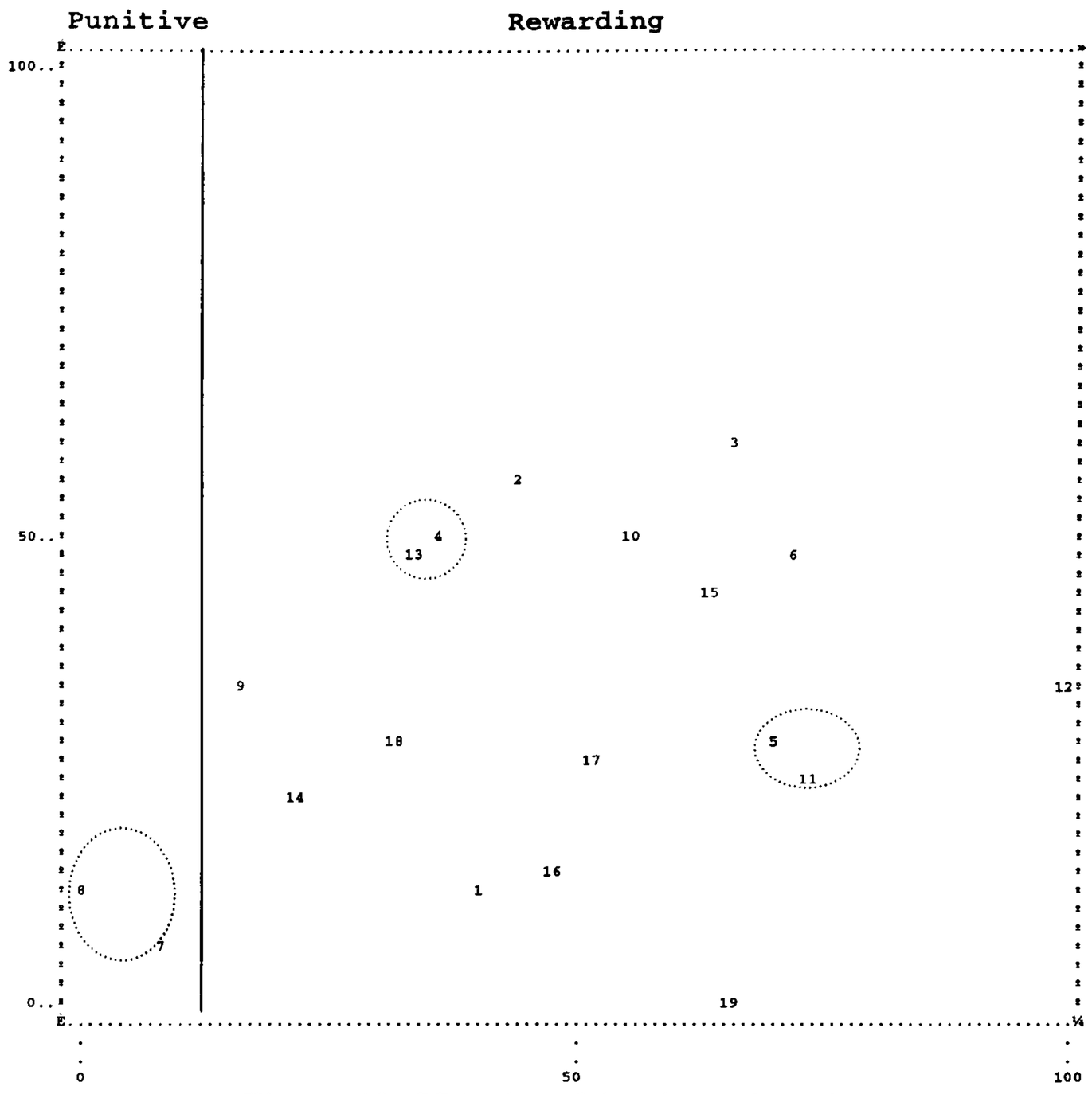
Managers' perceptions of developers' motivations seem more accurate regarding tangible motivators like feedback and reward schemes. They are less accurate on intangibles like career prospects and saleability.

5.3.5 Relationship between motivators

This section describes the relationship between motivators cited by practitioners. It presents the SSA plot for each practitioner group's motivators.

- SSA: Developer motivators

Figure 19 shows a SSA plot of developers' motivators. Coefficient of alienation for this plot is 0.13156, indicating that the plot is a good representation of the data matrix.



Code	Motivator
D1	Visible success
D2	Feedback
D3	Bottom-up initiatives
D4	Top-down commitment
D5	Shared best practice
D6	Training
D7	Compulsory
D8	Standardisation
D9	Phased introduction
D10	Critical mass
D11	Job satisfaction
D12	Eliminates bureaucracy
D13	Process ownership
D14	Reward schemes
D15	SPI forum
D16	Communication
D17	Resources
D18	Automation
D19	Maintainable processes

Figure 19: SSA plot of developers' motivators

Figure 19 shows that developer motivators can be partitioned into two main regions: punitive motivators and rewarding motivators, according to the SRT. The plot shows that there are only two motivators in the punitive region and that these are closely associated: compulsory(7) and standardisation(8). This means that it is highly likely that developers will cite these two closely associated motivators together. The vast majority of motivators are in the rewarding region of the plot and the closest association is between top-down commitment(4) and process ownership(13). These are also highly likely to be cited by developers together. The plot shows that other close relationships are between shared best practice(5) and job satisfaction(11) and visible success(1) and communication(16).

Figure 19 shows that some of the most distant associations are those between:

- Eliminates bureaucracy(12) and reward schemes(14)

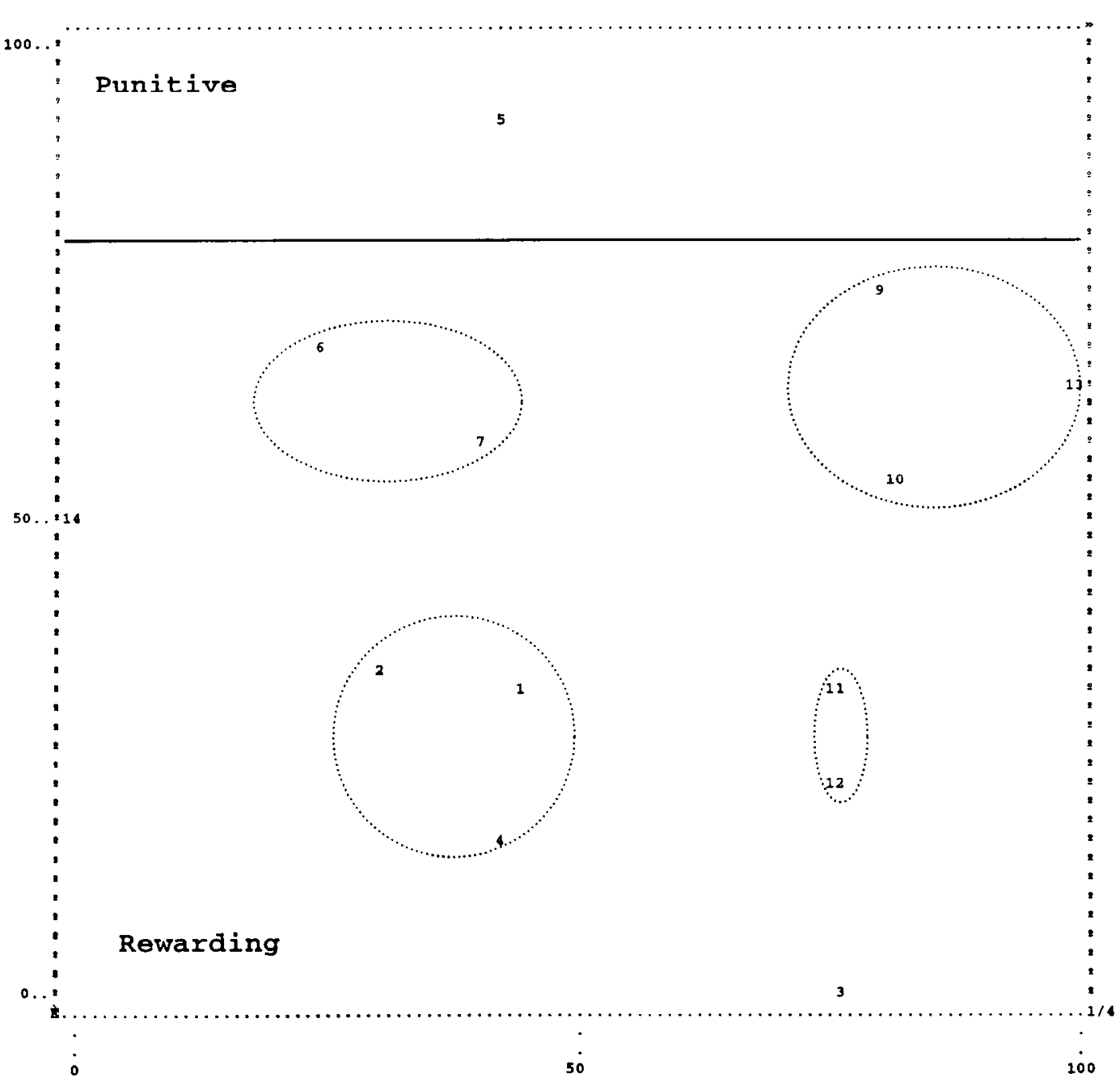
- Bottom-up initiative(3) and maintainable processes(19)
- Phased introduction(9) and eliminates bureaucracy(12)

These distant relationships imply that developers are less likely to cite these motivators together.

Overall, the developer motivators are plotted closer together than the motivators in the other two SSA's. This suggests that developers have stronger agreement about what motivates them than the other two groups of practitioners.

- SSA: Project manager motivators

Figure 20 shows a SSA plot of project managers' motivators. Coefficient of alienation is 0.05325, indicating a good fit.



Code	Motivator
P1	Resources
P2	Maintainable processes
P3	Reduced admin
P4	Empowerment
P5	External audits
P6	Easy processes
P7	Communication
P8	Autonomy
P9	Visible success
P10	Knowledgeable team leaders
P11	Reward schemes
P12	Process ownership
P13	Top down commitment
P14	Saleability

Figure 20: SSA plot of project managers' motivators.

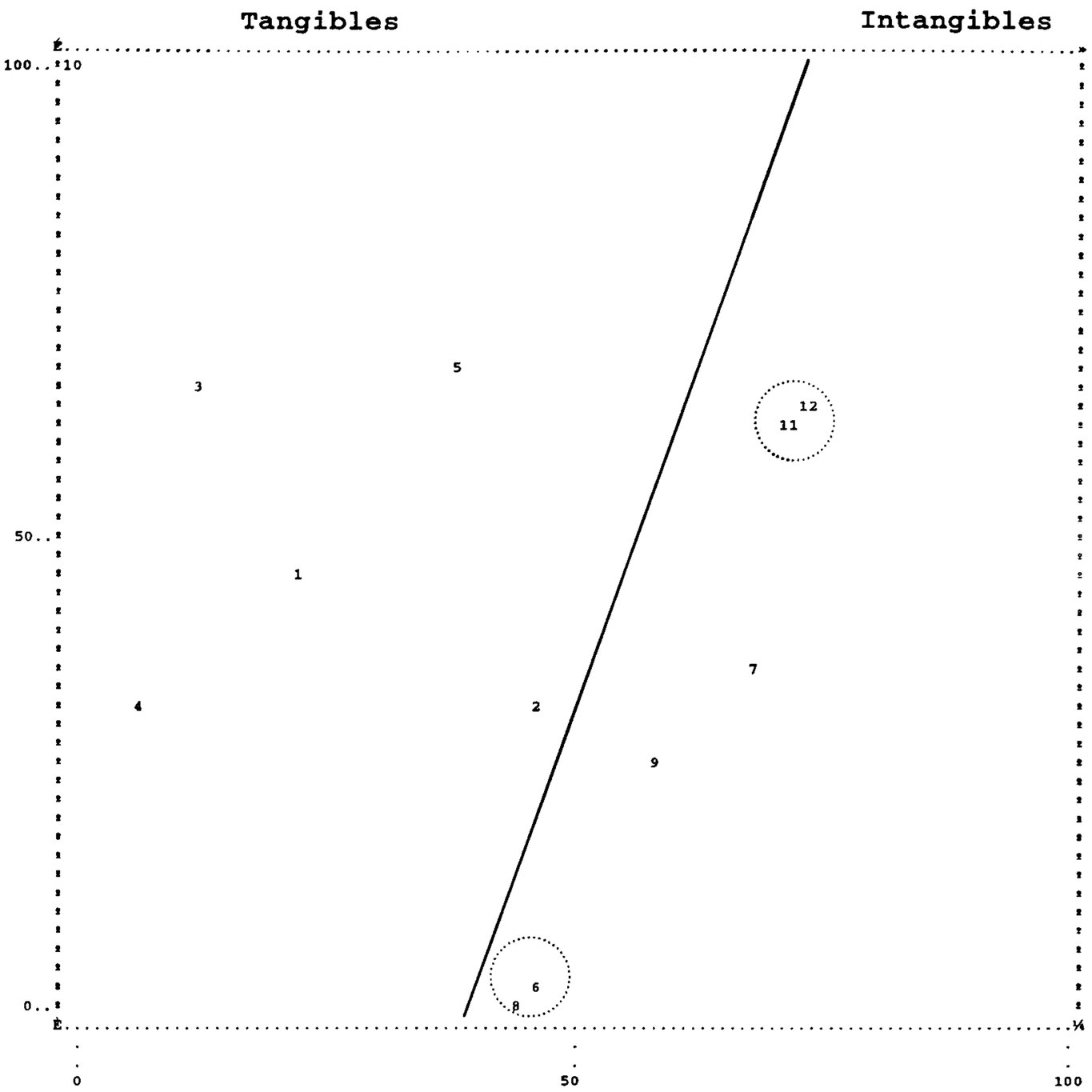
It shows that project manager motivators can be partitioned into punitive and rewarding regions. Figure 20 shows that external audits(5) is the only motivator in the punitive region, with the rest of the motivators classified as rewarding. The plot also shows that there are generally fewer close associations between the motivators cited by project managers. However autonomy(8) and top-down commitment(13) do seem very closely related. The two motivators have exactly the same profile, hence end up as a perfect fit (though autonomy is shielded from view on the plot being directly behind top-down commitment). This indicates that these two motivators are highly likely to be cited together by project managers. Other associations are between:

- Reward schemes(11) and process ownership(12)
- Resources(1), maintainable processes(2) and empowerment(4)
- Visible success(9), knowledgeable team leaders(10), top-down commitment(13) and autonomy(8)
- Easy processes(6) and communication(7)

However, generally, most project manager motivators are sparsely spread. The most divergent project manager motivators were top-down commitment(13) / autonomy(8)and saleability(14). Indicating that these motivators are least likely to be cited together by project managers.

- SSA: Senior manager motivators

Figure 21 shows a SSA plot of senior managers' motivators. Coefficient of alienation is 0, indicating a perfect fit.



Code	Motivator
S1	Justifiable benefits
S2	Visible success
S3	Cost beneficial
S4	Reward schemes
S5	Meeting targets
S6	Taller hierarchy
S7	Task forces
S8	Career prospects
S9	Process ownership
S10	Resources
S11	Feedback
S12	Maintainable Processes

Figure 21: SSA plot of senior managers' motivators

Figure 21 shows that with the exception of one 'outlier' (task forces) senior manager motivators divide into tangible and intangible regions. Task forces is an outlier because even though it is clearly a tangible motivator, it appears in the region of intangible motivators. Nevertheless, this occurrence does not invalidate the plot of senior managers' SSA or indeed the partitioning of the SSA into the said regions.

This plot shows that taller hierarchies(6) and career prospects(8) have a close association. In fact the plot co-ordinates indicate that they are a near perfect fit - suggesting that they have very similar profiles and thus are highly likely to be cited together by senior managers. The plot also shows a close association between maintainable processes(12) and feedback(11).

Resources(10) appears in the region of tangible motivators. However, it seems removed from the other motivators. This suggests that resources(10) has a different profile to the

other motivators and indicates that it is highly unlikely to be cited together with the other tangible motivators.

5.3.6 Limitation of SSA findings of practitioners' motivators for SPI

Overall, the findings reported from these SSA analysis should be viewed as exploratory findings. This research suggests that because of the very small number of observations and the low frequencies recorded for some of the observations, it will be misleading to attach too great a significance to the findings from these analyses. However, as a categorical data analysis technique, the use of SSA in this research has shown how future studies can apply this technique to effectively analyse the relationship between variables within a concept like motivators for SPI. The discussion of the motivators, therefore uses very little of the findings from the SSA plots. Only in cases where the frequencies are significantly high, are the SSA findings used to qualify some of the earlier findings.

5.4 Discussion of motivators

This section discusses analysis of practitioners' motivators for SPI.

5.4.1 Similarities across practitioner groups

There are eight motivators common to more than one staff group. Focusing initially on these motivations can deliver immediate cost benefit because a small number of motivators can be implemented that are known to have wide appeal.

Five motivators are common to all practitioner groups. Each of these appear to be a rewarding stimulus [Skinner, 1976]. For example, reward schemes and process ownership as opposed to say compulsion. It is interesting that all three groups agree on rewarding motivators, this means that rewarding stimuli are most likely to receive widespread support.

The following are the common motivators across staff groups:

- Ownership and top-down commitment

Results from this study suggest that many of the intangible motivators cited in this study are to do with empowering practitioners to take responsibility for the processes they work with. Across all groups, practitioners suggest that process ownership will be a motivation for SPI. Developers want grassroots input into processes. Project managers want autonomy so they can 'mould' the processes around their present practices. Both developers and project managers want to feel they have the support of senior management. These empowering motivators are reported as crucial to the success of SPI [Paulk et al., 1994b; Goldenson and Herbsleb, 1995; Stelzer and Mellis, 1998; Pitterman, 2000].

However, empowerment seems to be difficult to implement and does not seem widespread. In classic motivation theory, such motivators can be characterised as rewarding stimuli: difficult to implement even though their effects are longer lasting [Skinner, 1976].

Ownership motivators are also intrinsic to SPI. They are concerned with SPI processes themselves and how practitioners feel about the processes they work with. These motivators can 'satisfy' practitioners and can make them 'happy' to support SPI. [Herzberg et al., 1959].

Also, the SSA analyses show that there is a close association between the empowerment motivators cited by developers and project managers. Developers relate top-down commitment to process ownership; project managers relate top-down commitment to autonomy. These findings suggest that these empowerment motivators are perceived similarly by these practitioner groups and are therefore highly likely to motivate them similarly. The relevance of this set of motivators means that SPI managers need to address these in SPI implementation strategies.

- Visible success and communication

One of the most powerful motivators across all practitioner groups seems to be the provision of evidence to show that SPI can be successful. The fact that this motivator heads the list for all three groups of practitioners indicates its widespread importance. It implies that there are insufficient independent accounts of SPI success in industry that are accessible to practitioners. It also suggests that the case for SPI has not been sufficiently made. According to Glass [1998] quantitative evidence of the impact of SPI on software companies and their products is not easy to find. This suggests that companies could benefit significantly from providing evidence to practitioners. Indeed case studies report on the effectiveness of providing evidence [VASIE, 1999].

Evidence is also about communicating improvements to software practitioners. The SSA analyses show that where developers cite visible success they are highly likely to cite increased communication, too. This research suggests that there is a strong relationship between communication and evidence of improvement for developers in particular. Therefore managers should consider addressing communication motivators together with motivators about evidence.

- Resources

Resources are the time, tools and human effort that are dedicated to SPI. Thus when all three practitioner groups cite *resources* as a motivator for SPI, they seem to suggest that the presence of all or a combination of these tangible factors will motivate them to practice SPI. Herbsleb and Goldenson support this by suggesting that "people must be funded to engage in SPI rather than engaging in it as a spare time activity" [Herbsleb and Goldenson, 1996].

Resources encompass many factors and therefore may also motivate practitioners in different ways. For developers, resources can be both an 'extrinsic factor' [Herzberg et

al., 1959] and a 'rewarding stimulus'[Skinner, 1976]. For example, allocating time in a project for SPI may be an extrinsic factor, whereas funding developers to attend SPI courses may be rewarding.

Project managers, on the other hand, want resources allocated specifically to SPI so that project deliverables can still be met. This is particularly true in companies where SPI is sponsored on a project basis, i.e. SPI is not funded directly, but projects are expected to absorb SPI effort. In a sense resources are not directly an incentive to SPI practice, but the absence of dedicated resources can be a dis-incentive. Again resources 'maintain' project managers' participation in SPI. That is, resources are an extrinsic factor. However, unlike developers, resources do not necessarily 'reward' project managers for SPI.

Senior managers, too, cite resources as a motivator for SPI. They also cite it in conjunction with cost benefits and justifiable benefits and are the only practitioner group which cite these two motivators. This research speculates that senior managers do not perceive resources as a factor that motivates them *per se,* but rather as one that motivates an overall SPI programme. This is in line with their high-level view of the company and also because often they tend to be the same people in control of the 'purse strings'. Their particular motivator, however, may seem to be from the benefits that committing resources to SPI brings to the company as a whole. That is, the return on the resources. This is a different position to the other practitioner groups.

Indeed, this point is shown by the SSA analysis of senior managers' motivators for SPI. In that resources appear as the motivator with the least association with any of the other motivators. The SSA analyses suggest that senior managers are least likely to cite resources together with any of the other motivators.

5.4.2 Differences in motivators across practitioner groups

There are more differences in the motivators cited by practitioner groups than similarities. Understanding these differences may provide a long-term opportunity of designing highly effective implementation strategies. Investigating these differences offers insight on how to tailor SPI to particular practitioner groups and 'market' SPI more effectively to each group.

Results from this study show that only developers consider bottom-up initiatives as a motivator. This is despite the importance of 'bottom-up initiatives' according to the SPI literature [Paulk et al., 1994b; Krasner, 1997; Willis et al., 1998; Hammock, 1999]. In Paulk's analysis of the success of the Space Shuttle Project he identifies that quality measures were developed from the bottom up to ensure that they were accepted by the practitioners responsible for performing the processes [Paulk et al., 1994b]. Therefore the results from this study raise some concern by suggesting that manager groups, in practice, do not appreciate the importance of bottom-up initiatives.

Analyses of common motivators across practitioner groups indicate that most of the motivators reported by the literature as critical to SPI have been cited in practice. However, in the analysis of differences only developers seem to have cited most of the published SPI specific motivators. Managers, especially senior managers, seem to cite motivators that could be considered generally organisational. This seems to suggest that managers are focusing on the top-level organisational motivators and failing to identify the lower level, and more SPI-specific motivators. This research suggests that since it is developers who generally work with processes, it will be beneficial if managers paid greater attention to their specific motivations to SPI.

Study results also show that most of the motivators that developers cite can be classified as 'intrinsic' to the job [Herzberg et al., 1959]. For example, shared best practice, a SPI forum, phased introduction and bottom-up initiatives are all factors that will make practitioners 'happy' to practice SPI. Manager groups, on the other hand, mostly cite motivators that can be classified as 'maintenance' factors. The literature suggest that to maximise practitioner support for SPI, managers should be aware and be prepared to

implement the 'intrinsic factors that developers have suggested. This is because it is the intrinsic factors that really motivate [Herzberg et al., 1959].

The SSA analyses also indicate that, overall, the motivators cited by developers have the strongest relationship between them. Indicating that there are higher similarities between developer motivators than between the set of motivators cited by manager groups. This indicates that, as a group, developers are more in agreement about what motivates them to support SPI than the manager groups.

5.4.3 Predicting developer motivations

Results from this study also show that managers are not very accurate at predicting developers' motivators for SPI. Although managers show a reasonable understanding of the tangible motivators, i.e. feedback, maintainable processes and reward schemes, they show less understanding of the intangibles like autonomy, career prospects and saleability. This suggests that managers are not fully aware of developers' SPI motivations and will find it difficult to maximise developer support for SPI.

Managers also show better understanding of the extrinsic rather than the intrinsic factors that developers cite. With the exception of process ownership, all the other motivators that managers cite for developers are maintenance factors. Managers inaccurately believe that other motivation factors like autonomy, career prospects and saleability motivate developers.

These study results indicate that if SPI managers are to motivate developers to better support SPI it is imperative that they implement more of the intrinsic factors cited by developers. The literature suggests that when companies implement more intrinsic motivators, practitioners will support SPI better as they become genuinely motivated and not because they are being coerced to do so.

5.5 Conclusion

This chapter has described a study conducted to identify the motivators for SPI that software practitioners report in practice. In this section, this research presents a summary of the findings and shows how these findings answer the research question:

What SPI motivators do software practitioners report?

This section also summarises the differences in motivators reported across software practitioner groups.

5.5.1 Summary of findings

- The work itself / Job satisfaction

Software practitioners are motivated by the work itself to support SPI. However, only developers cite the job itself as a motivator to support SPI. Developers indicate that job satisfaction is a motivator for them to support SPI. Even though manager groups cite other work related motivators, it is developers only who cite the job itself as a motivator for supporting SPI.

- Opportunity for achievement / Empowerment

Software practitioners are also motivated by the opportunity for achievement to support SPI. Project managers indicate that if practitioners are empowered, they will be motivated to support SPI. In this finding, project managers seem to refer to empowerment for themselves as well as for developers. Therefore opportunity for achievement is a motivator for supporting SPI, but only for the lower level practitioners.

- Opportunity for advancement and growth / Training

Software practitioners are motivated by the opportunity for advancement and growth. Training to support SPI motivates developers and senior managers cite taller hierarchy and career prospects as motivators for supporting SPI. These motivators offer practitioners opportunities for advancement and growth.

- Reward for SPI

All staff groups of software practitioners are motivated by reward schemes to support SPI. All three groups of practitioners indicate that reward schemes will motivate practitioners to support SPI.

- Feedback and recognition

Developers and senior managers indicate that receiving feedback for their SPI work motivates practitioners to support SPI. Feedback indicates to practitioners that their support for SPI is being recognised. Practitioners also indicate that they are motivated by the recognition they receive for doing SPI work to support SPI.

- Increased responsibility / Process ownership

All three groups of software practitioners are motivated by increased responsibility. All software practitioners indicate that giving practitioners ownership of the processes they work in motivates them to support SPI.

- Technical knowledge / Knowledgeable team leaders

Project managers report that knowledgeable team leaders motivate practitioners to support SPI. They state that practitioners are motivated to support SPI if team leaders are knowledgeable of their field. This finding shows project managers' perception of the importance that developers attach to their own technical proficiency. It is an

acknowledgement from project managers that they can motivate developers to support SPI if they themselves are technically knowledgeable.

- Work conditions and company policies

Findings from this study show that several types of environmental factors, in terms of work conditions and company policies, motivate software practitioners to support SPI. All staff groups are motivated by maintainable SPI process. In addition, automation and practices that eliminate bureaucracy motivate developers.

- Senior management support

Software practitioners are motivated by senior management support to support SPI. Developers and project managers, in particular, indicate that top down commitment motivates them to support SPI.

5.5.2 Differences in motivators across practitioner groups

Findings from this study show that the common motivators reported by the three staff groups have been generally reported by the literature as critical to the success of SPI. These are motivators that relate to process ownership, top-down commitment, evidence, communication and resources.

There are, however, some differences in motivators across practitioner groups. The nature of these differences is that developers' motivators are more SPI-specific whereas managers' motivators tend to be organisational. Also, managers are better at predicting tangible developer motivators, than the intangible ones.

In Chapter Seven, this research will present a study that explores the reason behind these differences.

Chapter Six, meanwhile, presents a study that identifies the de-motivators that software practitioners report in practice.

Chapter Six: Analysis of de-motivators

6.1 Introduction

This chapter presents a study of software practitioners' de-motivators for SPI. This study uses focus group discussions to collect practitioners' perception of their de-motivators. Over two hundred software practitioners from thirteen UK companies took part in this study. Practitioners were divided into three staff groups of developers, project managers and senior managers.

6.1.1 Study aim

The aim of this study is to answer the research question:

What SPI de-motivators do software practitioners report?

This study identifies what de-motivators for SPI developers, project managers and senior managers report. It also identifies differences and similarities in these de-motivators across staff groups of developers, project managers and senior managers. It also explores the relationship between the de-motivators reported by software practitioners.

6.1.2 Study rationale

The review of the literature for this research identified four themes that underpin software practitioners' support for SPI. One of these themes is software practitioners' de-motivators for SPI. The review suggests that to improve software practitioners' support for SPI, it is important to address their de-motivators for SPI. However, in order to address these de-motivators, it is necessary to understand them. The rationale for this study is to find out what de-motivators for SPI software practitioners are reporting in practice so that these findings can be used to refute or confirm a set of guidelines, extracted from the literature, that suggest how to improve software practitioners' support

for SPI. Overall, this study, together with the previous studies described in Chapters Four and Five, attempts to validate the guidelines suggested by the literature for improving support for SPI and provide them a stronger empirical basis.

This chapter is structured thus:

Section 6.2 describes how data was collected and analysed. Section 6.3 presents the results of the analysis of practitioners' de-motivators. Section 6.4 discusses the findings of this study. Section 6.5 summarises and concludes this chapter.

6.2 Data collection and analysis process

6.2.1 Data collection

The data collection process in this study on practitioner de-motivators is the same as that in the previous study on practitioner motivators. Data collection involved forty-nine focus group sessions made up of twenty-one developer, sixteen project manager and twelve senior manager sessions. Each group comprised between four to six members. Each focus group lasted approximately ninety minutes. All groups were asked the following questions:

- What are the obstacles to SPI in your company?
- What would stop SPI from happening in this company?

Practitioner responses to the questions were audio recorded and the audio recordings were transcribed.

6.2.2 Data analysis: frequency analysis and smallest space analysis (SSA)

The data analysis process is, again similar to the data analysis process used in the previous study of motivators. De-motivators cited in response to the above questions

were categorised. Inter-rater reliability was calculated to increase confidence in this categorisation process. A data matrix was constructed for each staff group to record how de-motivators appeared in the groups. From the data matrix, frequencies are calculated for the occurrence of each de-motivator category. The data matrices are also subsequently used to populate the SSA application, [Shye, 1992] to produce SSA plots showing similarities between de-motivators.

This analysis procedure is the same at that adopted for the study on motivators, described in Chapter Five.

6.3 Findings of practitioner de-motivators

Table 21 provides a summary of all de-motivators identified in this study. It shows how de-motivators occurred across all three practitioner groups. Frequencies show the number of occurrence across all forty-nine groups and the percentages reflect the proportion of groups the de-motivators occurred in.

De-motivators	Occurrence in focus groups (n=49) Freq.	%
Time pressure/constraints	27	55
Inertia	19	39
Lack of resources	13	27
Commercial pressures	11	22
Lack of overall support	9	18
Budget constraints	8	16
Cumbersome processes	8	16
Lack of evidence of direct benefits	7	14
Negative / Bad experience	7	14
Inadequate communication	5	10
Imposition	4	8
Inexperienced staff	4	8
Lack of mgt direction/commitment	4	8
Lack of SPI management skills	4	8
Low process priority	4	8
Personality clashes	3	6
Workload	3	6
Lack of feedback	2	4
Lack of standards	2	4
Staff turnover	2	4
Customers	1	2
Fire fighting	1	2
Inadequate metrics	1	2
Irrelevant objectives/deliverables	1	2
Isolated best practice	1	2
Large-scale programmes	1	2
Organisational changes	1	2
PM's lack of technical knowledge	1	2
Reduced creativity	1	2

Table 21: Summary of de-motivators in study across all groups

Table 21 shows that time and resource issues, inertia and difficulties with obtaining practitioner support and buy-in for SPI are perceived as the major de-motivators for SPI by practitioners in this study.

6.3.1 Developers' de-motivators

De-motivators	Occurrence in focus groups (n=21) Freq.	%
Time pressure/constraints	13	62
Inertia	9	43
Budget constraints	5	24
Cumbersome processes	5	24
Commercial pressures	4	19
Lack of mgt direction/commitment	4	19
Imposition	3	14
Workload	3	14
Lack of standards	2	10
Lack of feedback	2	10
Personality clashes	2	10
Inadequate communication	1	5
Customers	1	5
Inexperienced staff	1	5
Isolated best practice	1	5
Lack of overall support	1	5
Negative/Bad experience	1	5
PM's lack of technical knowledge	1	5
Reduced creativity	1	5

Table 22: De-motivators cited by developers

Table 22 shows that 62% of developers find time pressures de-motivating to SPI. When this percentage is combined with 14% of developers who cited workload and 24% citing budget constraints as de-motivators, this research suggests that time and resources for SPI seem very hard to find. Nearly half of developers also think that inertia caused by practitioners' resistance to change is an obstacle to SPI. A quarter of developers also cited cumbersome processes as de-motivators. These are the only de-motivators cited by more than a quarter of developer groups.

Eight of the nineteen de-motivators cited by developers were cited by single focus groups (frequency =1). A few of these seemed obscure, for example, reduced creativity. It is possible to suggest that such de-motivators are specific to the particular company in the study. However, many are related to other de-motivators cited by other groups. For example, only one group of developers indicates that customers are de-motivating to SPI. At the same time, this issue of customers is linked to some of the views expressed about commercial pressures and external pressures on SPI. Also, other de-motivators like inadequate communication and negative experience are widely shared by other practitioner groups even though they are cited in only one developer session.

The following are additional comments made by developers about some de-motivators:

Commercial pressures: *"If you want to spend a little bit of time fixing up one of your areas and there's a customer offering to write a cheque for a new piece of development, the piece of development would be done in preference to the niceties of something else."*

Inertia: *"You are always going to come across people who will say, "oh I have done this for the last twenty years, I am going to carry on doing it". And I think when it comes to CMM, they are the ones that are causing us problems."*

Negative experience: *"I think the biggest problem is getting people that have been de-motivated in the past, re-motivated. They will say "we have done this in the past, nothing happened. I gave you those volumes, nothing happened, we are still doing the same old thing, why should I bother? This has to be the biggest challenge I think."*

6.3.2 Project managers' de-motivators

De-motivators	Occurrence in focus groups (n=16) Freq.	%
Time pressures/constraints	7	44
Lack of evidence of direct benefits	6	38
Lack of resources	5	31
Commercial pressures	4	25
Inertia	4	25
Low process priority	4	25
Cumbersome processes	2	13
Inadequate communication	2	13
Lack of overall support	2	13
Negative/Bad experience	2	13
Staff turnover	2	13
Fire fighting	1	6
Imposition	1	6
Inadequate metrics	1	6
Irrelevant objectives/deliverables	1	6
Large-scale programmes	1	6

Table 23: De-motivators cited by project managers

Table 23 shows that time pressures and lack of evidence of direct benefits were cited by around 40% of project managers. Other de-motivators were cited by over 25% of project managers, for example commercial pressures, inertia, lack of evidence and low process priority.

Project managers cite more de-motivators than other groups. Project managers are often perceived by developers and senior managers as a hurdle to SPI implementation in companies. The fact that they cite more de-motivators may explain this perception that they are a barrier to SPI.

Here are some additional comments offered by project managers to describe their de-motivators for SPI:

Fire fighting: "When we are 'fire fighting' we can't step back and think about things that can be improved. "

Low process priority: "Engineers get very disillusioned when they start working on a process activity and they have to stop and finish the project work they are doing because they know it`s a low priority. Process work is not recognised as much as project work. If they don't manage to get process work done it's very much don't worry, try and do it next year."

Time pressures: "The biggest problem that I have is that I ask people to change the way they work when they are already working flat out in order to deliver a product. The biggest issue is that people do not have time to understand a change and the benefit that that change will give them so they are resistant to change."

"Officially we've been told that you can build in time for PI but as soon as you hit any deadline the first thing to go is that."

6.3.3 Senior managers` de-motivators

De-motivators	Occurrence in focus groups (n=12) Freq.	%
Lack of resources	8	67
Time pressures/constraints	7	58
Inertia	6	50
Lack of overall support	6	50
Bad/Negative experience	4	33
Lack of SPI management skills	4	33
Budget constraints	3	25
Commercial pressures	3	25
Inexperienced staff	3	25
Inadequate communication	2	17
Cumbersome processes	1	8
Lack of evidence of direct benefits	1	8
Organisational changes	1	8
Personality clashes	1	8

Table 24: De-motivators cited by senior managers

Table 24 shows that, overall, there is strong agreement between senior managers about their de-motivators, with nearly two out of three de-motivators cited by over 25% of senior managers. Table 24 shows that 67% of senior managers cited lack of resources as a de-motivator and 58% cited time pressures. Inertia and lack of overall support are cited by 50% of senior managers. Furthermore 33% of senior managers cited lack of SPI management skills as a de-motivator with 25% citing inexperienced staff. This indicates senior managers have a particular concern about the shortage of SPI skills.

This is how senior managers described some of their de-motivators:

> **Bad experience:** "We have had a situation in the past where we have put a lot of effort into something but haven't got the end result and people remember that. They always remember the things that didn't go well, not the things that did."

> **Lack of resources:** "The whole areas of giving the reward, the incentive and the bandwidth to engineers to go and do it. The best engineers at PI also happen to be the best engineers at getting the product out of the door, so it's very hard to free up enough to concentrate on the process stuff."

6.3.4 De-motivators across practitioner groups

- Common de-motivators

Table 25 reproduces de-motivators cited by more than one practitioner group. The percentages represent de-motivators within particular groups.

De-motivators	Cited by (%)		
	Dev	PM	SM
Commercial pressures	19	25	25
Cumbersome processes	24	13	8
Inadequate communication	5	13	17
Inertia	43	25	50
Lack of overall support	5	13	50
Negative/Bad experience	5	13	33
Time pressure/constraints	62	44	58
Lack of resources	0	31	67
Lack of evidence of direct benefits	0	38	8
Budget constraints	24	0	25
Inexperienced staff	5	0	25
Personality clashes	10	0	8
Imposition	14	6	0

Table 25: Common de-motivators across practitioner groups

Half of the de-motivators are common to more than one practitioner group. In fact Table 25 shows that over half of these common motivators are common to all the three practitioner groups. This provides an indication of the level of agreement that practitioners across all staff groups have about what de-motivates them from practising SPI.

These results suggest that if all staff groups cite the same de-motivator then they may actually be identifying important de-motivators. It also indicates a good understanding of the problems in SPI.

- Different de-motivators across practitioner groups

Table 26 shows de-motivators that were cited within only one practitioner group.

De-motivator	Cited by: (Frequencies)		
	Dev	PM	SM
Lack of mgt direction/commitment	4		
Workload	3		
Lack of feedback	2		
Lack of standards	2		
Customers	1		
Isolated best practice	1		
PM's lack of technical knowledge	1		
Reduced creativity	1		
Low process priority		4	
Staff turnover		2	
Fire fighting		1	
Inadequate metrics		1	
Irrelevant objectives/deliverables		1	
Large-scale programmes		1	
Lack of SPI management skills			4
Organisational changes			1

Table 26: De-motivators cited in only specific groups

Table 26 shows that both developers and project managers cite more group specific de-motivators than senior managers. The proportion of group specific de-motivators for developers and project managers are approximately 0.4 respectively, whereas senior manager specific de-motivators are less than .2. Indicating that the issues that de-motivate senior managers are less group specific.

On the one hand, the 'group specific' de-motivators cited by developers and project managers are more likely to be company-specific than group-specific. Company specific de-motivators are those that are more likely to be cited in only one focus group within a staff group as opposed to several times within the same staff group. For example, four out of eight developer-specific de-motivators can be considered company specific. This is because developers cited customers, isolated best practice, project managers' lack of technical knowledge and reduced creativity in only one focus group session. Indicating that these issues may only be critical in those particular companies.

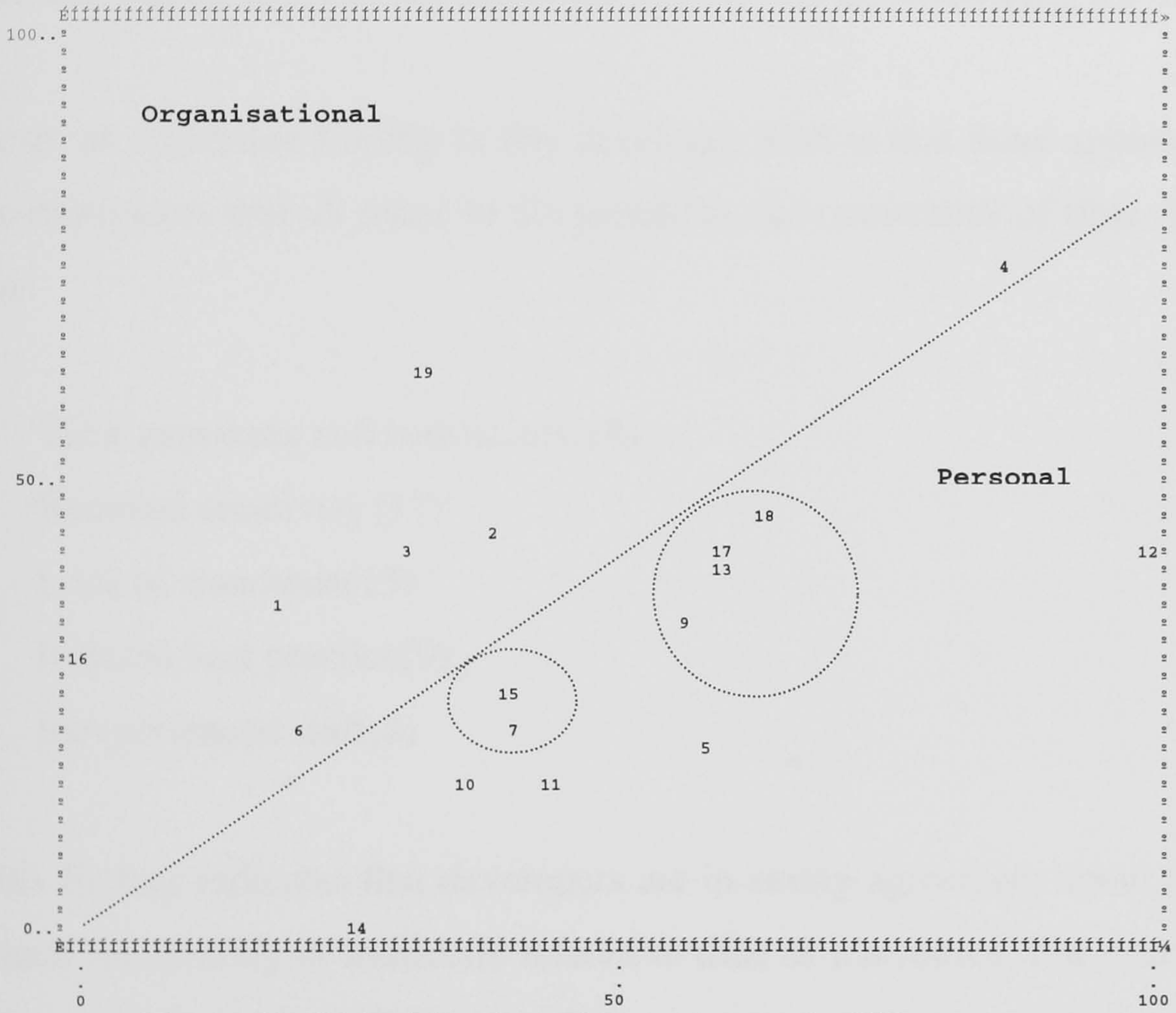
On the other hand, there are other de-motivators that were cited across several companies but within the same practitioner group. For example, senior managers cited lack of SPI management skills whilst project managers cited low process priority and staff turnover. Such de-motivators appear to be specific to the particular practitioner groups and merit further investigation as to why they are not acknowledged by other groups. These de-

motivators are probably directly related to specific functions within particular staff groups.

6.3.5 Relationship between de-motivators

The following are SSA plots showing the relationship between de-motivators cited by developers, project managers and senior managers.

- SSA: Developers de-motivators



D1	Budget constraints
D2	Commercial pressures
D3	Cumbersome processes
D4	Customers
D5	Imposition
D6	Inadequate communication
D7	Inertia
D8	Inexperienced staff
D9	Isolated best practice
D10	Lack of feedback
D11	Lack of mgt direction/commitment
D12	Lack of overall support
D13	Lack of standards (different platforms)
D14	Negative/Bad experience
D15	Personality clashes
D16	PM's lack of technical knowledge
D17	Reduced creativity
D18	Time pressure/constraints
D19	Workload

Figure 22: SSA of developers' de-motivators for SPI

Figure 22 presents a SSA plot of developers de-motivators for SPI. Coefficient of alienation is 0.07245, suggesting a very good fit. It shows that even though there is a loose association between a majority of the de-motivators, depicted by a sparse cluster at the centre of the plot, there are a few close associations. In fact, three sets of developer

de-motivators appear very tightly close together. Inexperience staff(8) and isolated best practice(9) are a perfect fit - appearing on the same spot. Lack of standards(13) and reduced creativity(17) are also very closely shown together. Finally, inertia(7) and personality clashes(15) seem closely associated.

These results show that the likelihood of developers citing these sets of de-motivators together is high. That is: if developers indicate that they are de-motivated by inexperienced staff, they are highly likely to indicate that they are de-motivated by isolated best practices, too. Also, developers who indicate that reduced creativity de-motivates them from supporting SPI are also likely to indicate that a lack of standards de-motivates their SPI effort. Similarly, developers who report inertia de-motivating for SPI are also highly likely to report personality clashes de-motivating for SPI.

Also, an important finding in this developer SSA is that there appears another cluster of de-motivators that all relate to the pressures and constraints of time. These de-motivators are:

- Time pressures and constraints(18)
- Reduced creativity (17)
- Lack of standards(13)
- Isolated best practice(9)
- Inexperienced staff(8)

This finding indicates that developers are in strong agreement about these de-motivators, which are directly or indirectly related to time as a resource. For example, time pressures may result in using staff for jobs they are not properly trained for. Also, time constraints can frustrate the evolution and application of standards within companies. This is because practitioners are more likely to ignore standards and procedures when under pressure.

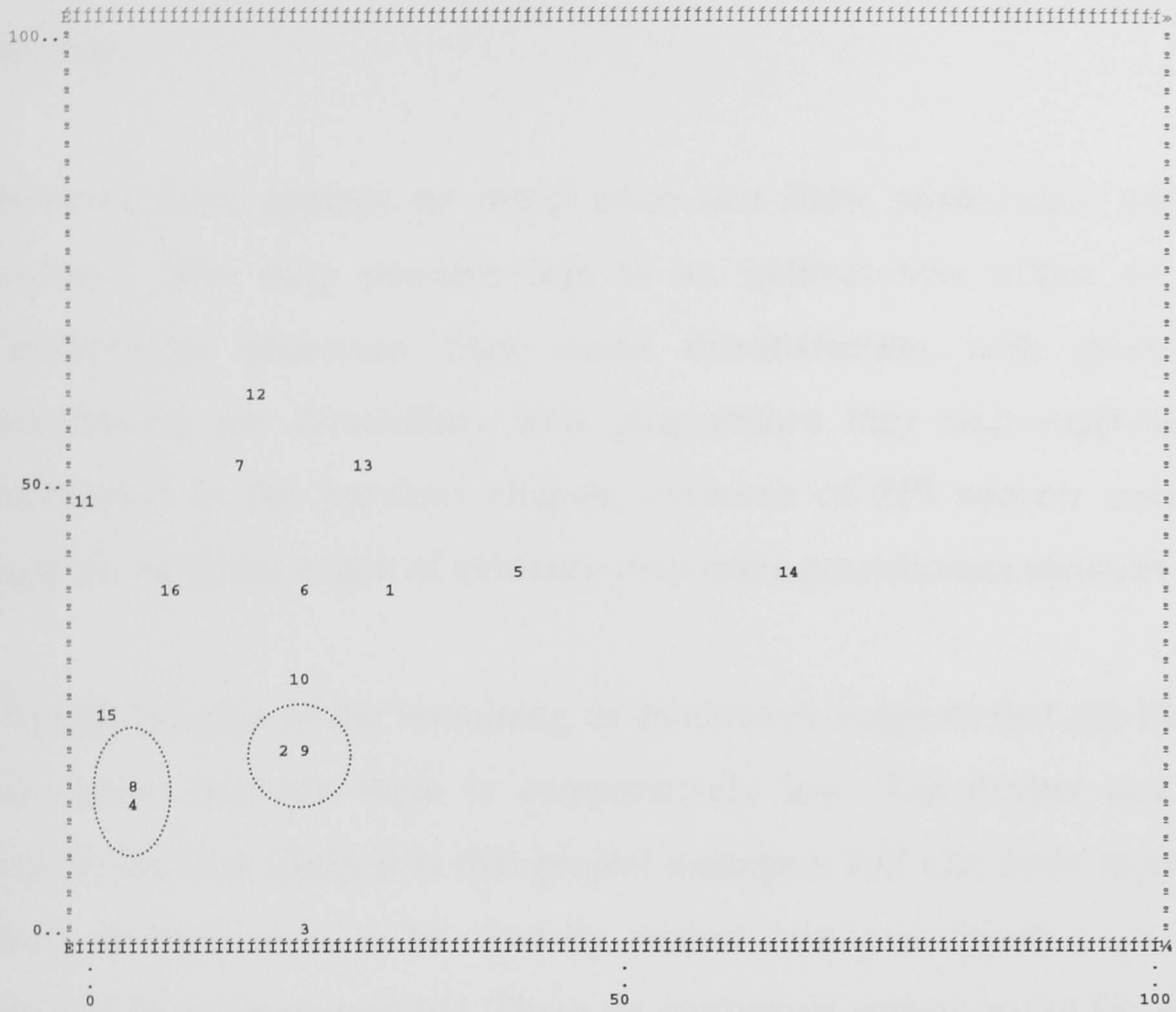
However, the likelihood of co-occurrence of most of the de-motivators in the plot is very low because most of the de-motivators appear sparse in the plot. For example, the

likelihood of developers citing *inertia* in conjunction with *imposition* is less than that of citing *inertia* in conjunction with *personality clashes*. This likelihood is even less when the comparison is made with *inertia* and *lack of overall support*.

Furthermore Figure 22 also shows that there are some developers' de-motivators that appear far removed from any of the other de-motivators. These de-motivators appear on the outskirts of the plot. These de-motivators are customers(4), lack of overall support(12), negative/bad experience(14), project managers' lack of technical knowledge(16) and workload(19). Their positions in the plot indicate that they have little association with each other and also with the remaining de-motivators that are situated in the middle of the plot.

Overall, it is difficult to establish issues that bind many of the very closely associated de-motivators together. For example, even though Figure 22 shows that inexperienced staff and isolated best practice appear as identical plots, it is difficult to establish a particular issue that binds them together. The only concept that sufficiently partitions de-motivators in Figure 22 is one that categorises de-motivators as either organisational or personal. Hence the partition employed.

- SSA: Project manager's de-motivators



P1	Commercial pressures
P2	Cumbersome processes
P3	Fire fighting
P4	Imposition
P5	Inadequate communication
P6	Inadequate metrics
P7	Inertia
P8	Irrelevant objectives/deliverables
P9	Lack of evidence of direct benefits
P10	Lack of overall support
P11	Lack of resources
P12	Large-scale programmes
P13	Low process priority
P14	Negative/Bad experience
P15	Staff turnover
P16	Time pressures/constraints

Figure 23: SSA of project managers' de-motivators for SPI

Figure 23 shows a SSA plot of project managers' de-motivators of SPI. This plot has a coefficient of alienation is 0.0662, indicating a very good fit. This figure shows that most of the de-motivators cited by project managers appear in the same region of the plot. With the exception of negative/experience(14) and fire-fighting(3), most de-motivators form a cluster in one region of the plot. This cluster, however, is not a very close one even though some individual de-motivators appear very close to each other. Among these there are imposition(4) and irrelevant objectives(8) and cumbersome processes(2) and lack of evidence(9).

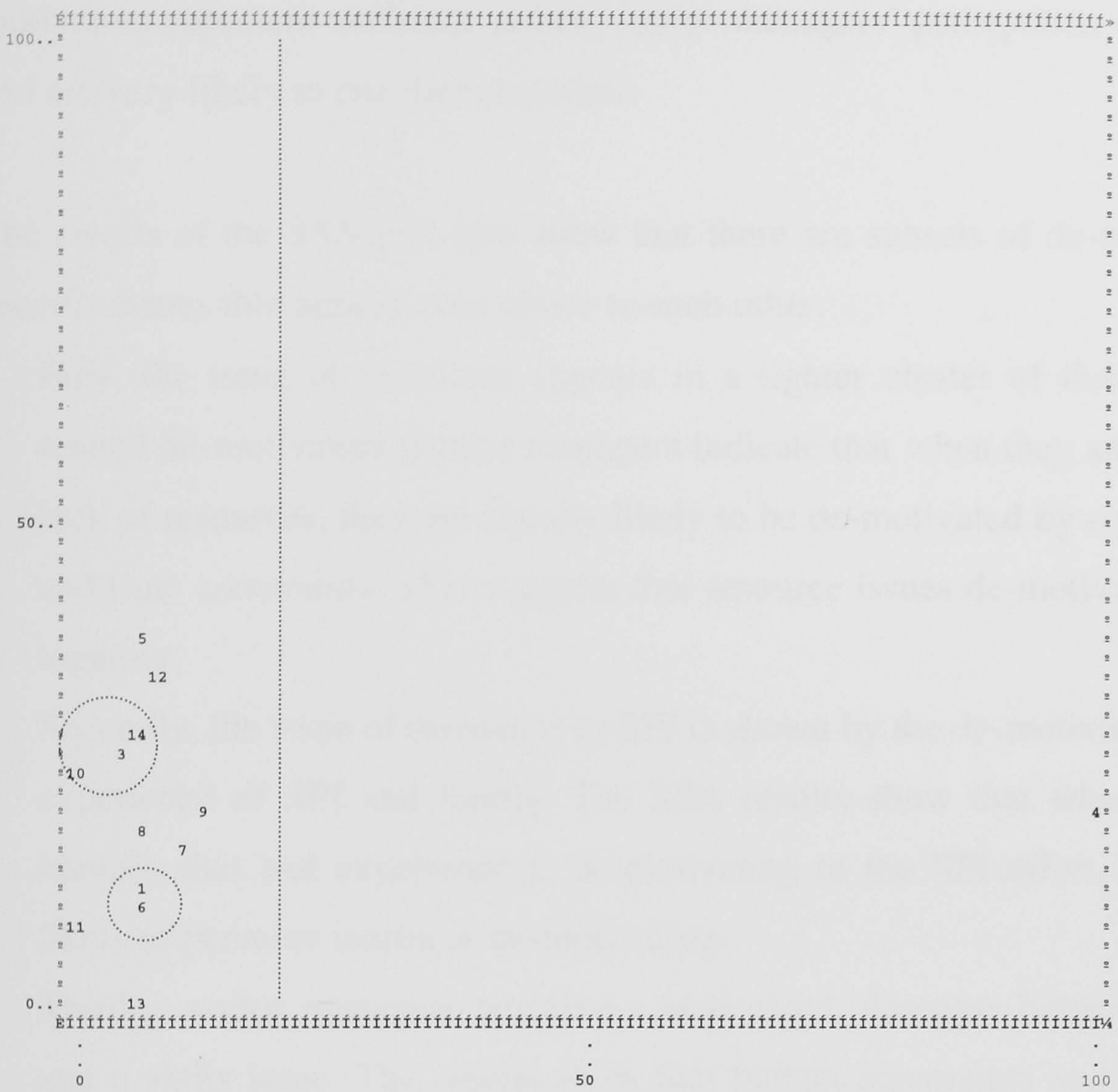
This plot shows that where project managers perceive imposition as de-motivating to their SPI effort, they are also highly likely to cite irrelevant objectives as de-motivating. Likewise, where a lack of evidence de-motivates project managers from supporting SPI, they are highly likely to be de-motivated by cumbersome SPI processes.

The strong association between *irrelevant objectives* and *imposition* suggest that project managers' dissatisfaction with how SPI is implemented is highly likely to be cited together.

However there appears no direct issue that links *cumbersome processes* and *lack of evidence*. The only possible link is an indirect one related to resistance to SPI. Cumbersome processes may cause dissatisfaction with practitioners and when practitioners are dissatisfied with programmes they stop supporting them. Also, as established in the previous chapter, evidence of SPI success encourages practitioner support, therefore a lack of evidence may make practitioners resistant to SPI.

Overall, the plot of the remaining de-motivators suggests that the likelihood of a strong association between them is comparatively low. The further any two de-motivators appear, the less likely it is that project managers will cite them together. The two least likely de-motivators to be cited by project managers together are fire-fighting(3) and negative/bad experience(14). These de-motivators appear on the fringes of the plot of de-motivators. These two de-motivators also seem the least associated with any of the other de-motivators.

- SSA: Senior managers' de-motivators



S1	Bad/Negative experience
S2	Budget constraints
S3	Commercial pressures
S4	Cumbersome processes
S5	Inadequate communication
S6	Inertia
S7	Inexperienced staff
S8	Lack of evidence of direct benefits
S9	Lack of overall support
S10	Lack of resources
S11	Lack of SPI management skills
S12	Organisational changes
S13	Personality clashes
S14	Time pressures/constraints

Figure 24: SSA of senior managers' de-motivators for SPI

Figure 24 presents a SSA plot of senior managers' de-motivators for SPI. This plot has a coefficient of alienation of 0.02924, which indicates a very good fit. Figure 24 shows that most of de-motivators appear close to each other in the plot. In fact all the de-motivators, bar one, appear in one region of the SSA plot. Initial indication from this plot is that most of these de-motivators have close associations with each other. Further investigation of Figure 24 shows that within this cluster of de-motivators, some are very close to each other. For example, bad/negative experience (1) and Inertia (6) are very close to each other. Similarly, lack of resources(10), commercial pressures(3) and time pressures/constraints(14) are very close to each other. Budget constraints(2) and inexperienced staff(7) are so much close together that they appear as one plot.

The results from senior managers' SSA indicate that senior managers perceive a strong relationship between their de-motivators for SPI. Even though collectively these de-motivators represent different issues, senior managers' perceptions of them are similar and are very likely to cite them together.

The results of the SSA plot also show that there are subsets of de-motivators related to specific issues that appear even closer to each other.

- First, the issue of resources appears in a tighter cluster of three specific resource related de-motivators. Senior managers indicate that when they are de-motivated by a lack of resources, they are equally likely to be de-motivated by commercial pressures and time constraints. This suggests that resource issues de-motivate senior managers together.
- Secondly, the issue of resistance to SPI is shown by the de-motivators of bad/negative experience of SPI and inertia. The SSA results show that where senior managers identify that bad experience is de-motivating to the SPI effort, they are also most likely to perceive inertia as de-motivating.
- Thirdly, senior managers perceive a near exact closeness between a resource issue and a skills issue. The results show that budget constraints and inexperienced staff both share a common profile. This profile suggests that where senior managers perceive budget constraints as de-motivating to the SPI effort, they are likely also to see inexperienced staff.

Figure 24 also shows that one de-motivator, only, cumbersome processes(4) is far removed from the other de-motivators cited by senior managers. This indicates that this particular de-motivator possesses little in common with the other de-motivators and is unlikely to be cited in tandem with these other de-motivators.

This research suggests that cumbersome processes does not appear closely related to the other de-motivators cited by senior managers because it is a low level de-motivator whereas the other de-motivators are higher level de-motivators. Also, the other de-motivators cited by senior managers are related to peripheral issues with the processes,

whereas this particular de-motivator addresses the processes directly. This research suggests that since senior managers tend to abstract their concerns at a higher level than the technical level of the nature of the processes, it becomes improbable that they will be citing cumbersome processes together with the other de-motivators for SPI.

6.3.6 Limitation of SSA findings of practitioners' de-motivators for SPI

As already discussed in Chapter Five, the use of MDS in this research has been as an exploratory approach to analysing categorical data. As a result strong conclusions cannot be drawn from the findings of these SSA analyses of software practitioners' de-motivators for SPI. This is because the number of data points used in these analyses is low. However, the use of the technique in this research pilots the use of MDS in software engineering research and lays a step for future software engineering research to apply this technique to gain a richer understanding of other software engineering concepts.

The following discussion, therefore, makes little use of these findings except in instances where the frequency of observations are high and the findings corroborate what has already been established in the larger content analysis phase.

6.4 Discussion

This section discusses the major findings from this study. It also discusses similarities and differences in de-motivators across staff groups.

6.4.1 Major de-motivators

The following are the major de-motivators identified by practitioners from our findings.

- Resources for SPI

Findings from this study indicate that all practitioner groups are concerned about the effect of resource constraints on motivations for SPI. Developers and senior managers

perceive these in terms of the constraints of project budgets whilst project managers view it as a general lack of resources. Practitioners generally view resource constraints de-motivating to the SPI effort.

Practitioners also indicate that pressure from the market to keep up with competition in terms of maintaining a company's position in the market frustrates support for SPI. This research suggests that such pressures come from the need to satisfying time to market demands and meet customer demands. Practitioners indicate that such commercial pressures make it difficult to actively devote resources to SPI. Practitioners find that in an environment where the commercial imperative is so high, incentives for SPI are compromised.

Overall, software practitioners perceive a close relationship between resource related de-motivators. In particular, developers and senior managers cite these de-motivators together. This indicates that when managing practitioners' de-motivators for SPI, effective improvements can be made when resource related de-motivators are addressed together.

- Resistance to SPI

This study also explores the issue of inertia, bad experiences and general lack of support for SPI on software practitioners' resistance to SPI. Findings show that inertia, negative previous experiences and lack of overall support de-motivates practitioners' support for SPI.

Practitioners may resist SPI due to their negative experiences of previous SPI programmes. As Humphrey describes in his studies of the Personal Software Process (PSP), past experiences can make practitioners think that the new processes will not improve their output [Humphrey, 1998]. Such negative experiences can create resistance to SPI, which can prevent practitioners from supporting SPI.

Practitioners will also not support SPI if they are unwilling to give up established ways of working. Hovenden et al recount the actions of the 'maverick practitioners' who frustrate the quality improvement effort in companies as they fear that it will threaten their established way of working [Hovenden et al., 1996]. Humphrey also indicates that practices become established in practitioners themselves, thereby creating intransigence to new ways of working. Such intransigence results in practitioners failing to support SPI.

Lack of support for SPI becomes de-motivating for other practitioners who may not have necessarily resisted SPI, but become frustrated by the apathy that others have for SPI. Findings from this study show that all practitioner groups find that overall lack of support for SPI de-motivating.

Overall, this study shows that software practitioners are resistant to SPI. This study also shows that senior managers, in particular, perceive a strong association between the de-motivators related to resistance: Inertia and bad /negative experience. So that addressing all of senior managers' de-motivators related to resistance together, is likely to yield more success.

- Evidence for SPI

The results of this study show the effect of lack of evidence on software practitioners' de-motivators for SPI.

Project managers and senior managers indicate that the lack of direct evidence showing the benefits of SPI is de-motivating. Developers are the only group that does not cite a lack of evidence as de-motivating for their support of SPI. This indicates that the issue of evidence is more prominent for manager groups than it is for developers.

- Skills for SPI

Senior managers indicate that a lack of SPI management skills is a de-motivator for SPI. They also indicate that it is critical to have people with the expertise to drive SPI programmes. Absence of such skill de-motivates the SPI effort. This perception is supported indirectly by developers and project managers.

Developers and senior managers say that working with inexperienced staff de-motivates the SPI effort. Project managers on the other hand highlight staff turnover as de-motivating. This research suggests that both sets of de-motivators are related to a lack of SPI skills. Companies may have inexperienced staff because of high staff turnover. However, having inexperienced staff could also be due to company policy on training - where little to nothing is spent on training staff for SPI.

- Imposed SPI initiatives

Practitioners cited the following de-motivators that are directly related to the way SPI is implemented in companies:

- Imposition
- Inadequate communication
- Irrelevant objectives

Project managers and developers find that imposing SPI without prior consultation is de-motivating. Even though senior managers do not perceive imposition, particularly, as a de-motivator, they do acknowledge that inadequate communication is de-motivating for SPI.

For project managers, particularly, there is a close association between imposition and irrelevant objectives. This suggests that quick successes can be made in project managers' support for SPI when these implementation de-motivators are addressed together.

Overall, all practitioner groups suggest that SPI initiatives that do not involve practitioners, through consultation and communication and do not have objectives that are relevant to practitioners, are de-motivating and unlikely to be supported fully by practitioners.

6.4.2 Spread of de-motivators across practitioner groups

Findings from this study show that there are both similarities and differences in de-motivators across software practitioner groups.

- Common de-motivators

Out of these, nearly 45% are common to more than one practitioner group. These common motivators can be categorised into the following broad areas:

- Resource related
- Commercial pressures
- The actual process constraints
- Implementation issues
- Personnel factors

- Group specific de-motivators

Within the group-specific de-motivators, developers and project managers have cited, proportionately, twice the number of de-motivators as senior managers. This may suggest that whereas developers and project managers may be aware of many senior manager de-motivators, the same cannot be said of senior managers' awareness of developers' and project managers' de-motivators. This finding raises concern about senior managers' awareness of the issues that de-motivates other practitioner groups for SPI.

Findings from this study also show that the group specific motivators are very much related to the roles that practitioners play in their companies.

Developer specific motivators are:

- Implementation factors
- Excessive workload
- Loss of creativity
- Customers
- Lack of management commitment and know how

Project manager specific de-motivators are:

- Lack of measures for controlling projects
- Lack of planning
- Staff turnover

Senior manager de-motivators:

- Organisational changes
- Lack of SPI management skills

6.5 Conclusion

This chapter has described a study conducted to identify the de-motivators for SPI that software practitioners report in practice. In this section, this research presents a summary of the findings of the study and shows how these findings answer the research question:

What SPI de-motivators do software practitioners report?

This section also summarises the differences in de-motivators reported across software practitioner groups.

6.5.1 Summary of findings

- Lack of resources and commercial pressures

Findings from this study show that software practitioners are de-motivated from supporting SPI by the lack of resources dedicated to SPI. Practitioners find time and budget restraints de-motivating. They also find that the pressure from commercial commitments compromises the incentive for SPI.

- Software practitioners' resistant to SPI

The study findings show that software practitioners are resistant to SPI for a variety of reasons. Previous negative experiences of SPI prevent software practitioners from supporting SPI and software practitioners are resistant to SPI because they are unwilling to change their tried and established practices. This finding indicates that the underlying factors that make software practitioners resistant to SPI also de-motivate them from supporting SPI.

- Lack of evidence

This chapter has shown that software practitioners are de-motivated by lack of evidence of the direct benefits of SPI to their practices. Findings from the study reported in this chapter show that practitioners are more likely to support SPI if they know how it will benefit them directly.

- Imposed initiatives

Study findings indicate that software practitioners are more receptive to consultative initiatives. Therefore they find SPI initiatives that are imposed upon them de-motivating.

- Lack of appropriate skills

The study findings show that software practitioners are de-motivated from supporting SPI when there is a lack of appropriate SPI skills in their companies. Developers and senior managers find it de-motivating when project managers lack SPI skills. Project managers on the other hand see the problem of shortage in SPI skills as an indirect result of high staff turnover.

6.5.2 Differences in de-motivators across staff groups

The study described in this chapter has also shown that there are common issues that de-motivate all groups of practitioners. These issues are varied and cover factors relating to resources, processes, implementation and people.

There are, also, differences in the de-motivators for SPI amongst different staff groups of practitioners. These differences are often related to the role that practitioners play in software development.

Overall, these findings from this study of software developers' de-motivators for SPI provide useful empirical account of the factors that frustrate support for SPI in practice. The next chapter, Chapter Seven, presents a study that investigates the perceptions of developers, project managers and senior managers about their role in SPI. This study will explain the differences in software practitioners' de-motivators for SPI, reported here, and also the differences in motivators for SPI reported in Chapter Five.

Chapter Seven: Practitioners' perception of their role in SPI

7.1 Introduction

This chapter presents a study of software practitioners' perceptions of their role in SPI. The study uses the Repertory Grid Technique (RGT) to explore the views of three staff groups of practitioners. Over two hundred software practitioners from thirteen UK companies took part in this study. Software practitioners were divided into three staff groups of developers, project managers and senior managers.

7.1.1 Study aim

The aim of this study is to answer the research question:

> What are the differences in the SPI motivators and de-motivators that senior managers, project managers and developers report?

This study explores the different perceptions that software practitioners have of their role in SPI in order to understand the nature of the differences in their motivators and de-motivators for SPI.

7.1.2 Rationale for study on practitioner perceptions of SPI

The review of the literature in this research identified that different practitioner groups tend to have different responses to SPI. These differences then tend to impact negatively on the support for SPI in companies. The review suggests that support for SPI can be improved if the differences are identified and their nature understood. To understand these differences, the literature suggests that it is important to understand the different perceptions that software practitioners have about the role they have in SPI, because often the differences in responses are related to the differences in perception of that role.

In Chapters Five and Six, this research described two studies that showed what motivators and de-motivators software practitioners reported in practice. These studies showed that there were differences in the motivators and de-motivators reported by the different staff groups. The rationale of this study, therefore, is to establish whether there are differences in software practitioners' perception of their role in SPI, too. Then to identify whether these differences relate to the differences in software practitioners' motivators and de-motivators for SPI.

Overall, this study aims to provide an empirical basis for the suggestion by the literature that different practitioner groups have different motivators and de-motivators for SPI. This study also confirms or refutes the suggestion that the nature of the differences is related to software practitioners' perceptions of their role in SPI. Finally, the study identifies the nature of the differences in the motivators and de-motivators for SPI that developers, project managers and senior managers' report in practice.

This chapter is structured as follows:

Section 7.2 discusses how data was collected and analysed in this study. Section 7.3 presents results of data analysis. Section 7.4 discusses findings of the study. Section 7.5 summarises and concludes this chapter.

7.2 Data collection and analysis process

7.2.1 Data collection

This study uses RGT to collect data from developers, project managers and senior managers. Forty-six RGT sessions were conducted with software practitioners in groups of four to six. In each RGT session the following questions were asked:

> *"Think about developers, project managers and senior managers in the context of software process improvement. What do you think the paired group have in common that differentiates them from the single group?"*

Senior managers

Project managers *Developers*

Developers

Project managers *Senior managers*

Developers

Senior managers *Project managers*

7.2.2 Data analysis

The data collected in this study were analysed using the manual RGT analysis method of Visual Focusing [Stewart et al., 1981] discussed in Chapter Three.

7.3 Study results

7.3.1 Developers' perceptions

Altogether, forty bi-polar constructs were elicited from developers (see Appendix L). Table 27 shows the element matrix derived from these developer constructs. The matrix shows the agreement scores for the three staff groups. The maximum agreement score is 40 and minimum is 0.

	Dev	PM	SM
Dev	*	14	6
PM	*	*	20
SM	*	*	*

Table 27: Developers' element by element agreement matrix

Table 27 shows that developers view project managers and senior managers as having the most in common whereas they see the two sets of practitioners with least in common as themselves and senior managers.

Table 28 presents the construct by construct matrix for developers. There are forty constructs altogether with scores between 0 and 3. Constructs with a rating of 3 are the most similar pairs. These are re-arranged next to each other to reflect this similarity. Whereas constructs with an agreement score of 0 are the least similar.

	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16	C17	C18	C19	C20	C21	C22	C23	C24	C25	C26	C27	C28	C29	C30	C31	C32	C33	C34	C35	C36	C37	C38	C39	C40
C1	*																																							
C2	0	*																																						
C3	3	0	*																																					
C4	2	1	2	*																																				
C5	0	3	0	1	*																																			
C6	3	0	3	2	0	*																																		
C7	3	0	3	2	0	3	*																																	
C8	0	3	0	1	3	0	0	*																																
C9	1	2	1	0	2	1	1	2	*																															
C10	2	1	2	3	1	2	2	1	0	*																														
C11	2	1	2	3	1	2	2	2	0	3	*																													
C12	2	1	2	1	1	2	2	2	2	1	1	*																												
C13	2	1	2	3	1	2	2	2	0	3	3	1	*																											
C14	2	1	2	3	1	2	2	2	0	3	3	1	3	*																										
C15	0	3	0	1	3	0	0	2	2	1	1	1	1	1	*																									
C16	2	1	2	1	1	2	2	1	2	1	1	3	1	1	1	*																								
C17	0	3	0	1	3	0	0	2	2	1	1	1	1	1	3	1	*																							
C18	2	1	2	3	1	2	2	1	0	3	3	1	3	3	1	1	1	*																						
C19	2	1	2	3	1	2	2	2	0	3	3	1	3	3	1	1	1	3	*																					
C20	1	2	1	2	2	1	1	1	1	2	2	0	2	2	2	0	2	2	2	*																				
C21	0	3	0	1	3	0	0	3	2	1	1	1	1	1	3	1	3	1	1	2	*																			
C22	0	3	0	1	3	0	0	3	2	1	1	1	1	1	3	1	3	1	1	2	3	*																		
C23	0	3	0	1	3	0	0	3	2	1	1	1	1	1	3	1	3	1	1	2	3	3	*																	
C24	0	3	0	1	3	0	0	3	2	1	1	1	1	1	3	1	3	1	1	2	3	3	3	*																
C25	0	3	0	1	3	0	0	3	2	1	1	1	1	1	3	1	3	1	1	2	3	3	3	3	*															
C26	2	1	2	3	1	2	2	1	0	3	3	1	3	3	1	1	1	3	3	2	1	1	1	1	1	*														
C27	2	1	2	3	1	2	2	2	0	3	3	1	3	3	1	1	1	3	3	2	1	1	1	1	1	3	*													
C28	2	1	2	3	1	2	2	2	0	3	3	1	3	3	1	1	1	3	3	2	1	1	1	1	1	3	3	*												
C29	2	1	2	1	1	2	2	2	2	1	1	3	1	1	1	3	1	1	1	0	1	1	1	1	1	1	1	1	*											
C30	2	1	2	1	1	2	2	2	2	1	1	3	1	1	1	3	1	1	1	0	1	1	1	1	1	1	1	1	3	*										
C31	0	3	0	1	3	0	0	2	2	1	1	1	1	1	3	1	3	1	1	2	3	3	3	3	3	1	1	1	1	1	*									
C32	0	3	0	1	3	0	0	3	2	1	1	1	1	1	3	1	3	1	1	2	3	3	3	3	3	1	1	1	1	1	3	*								
C33	0	3	0	1	3	0	0	3	2	1	1	1	1	1	3	1	3	1	1	2	3	3	3	3	3	1	1	1	1	1	3	3	*							
C34	2	1	2	3	1	2	2	1	0	3	3	1	3	3	1	1	1	3	3	2	1	1	1	1	1	3	3	3	1	1	1	1	1	*						
C35	2	1	2	3	1	2	2	2	0	3	3	1	3	3	1	1	1	3	3	2	1	1	1	1	1	3	3	3	1	1	1	1	1	3	*					
C36	2	1	2	1	1	2	2	2	2	1	1	3	1	1	1	3	1	1	1	0	1	1	1	1	1	1	1	1	3	3	1	1	1	1	1	*				
C37	0	3	0	1	3	0	0	2	2	1	1	1	1	1	3	1	3	1	1	2	3	3	3	3	3	1	1	1	1	1	3	3	3	1	1	1	*			
C38	3	0	3	2	0	3	3	0	1	2	2	2	2	2	0	2	0	2	2	1	0	0	0	0	0	2	2	2	2	2	0	0	0	2	2	2	0	*		
C39	0	3	0	1	3	0	0	2	2	1	1	1	1	1	3	1	3	1	1	2	3	3	3	3	3	1	1	1	1	1	3	3	3	1	1	1	3	0	*	
C40	1	2	1	0	2	1	1	2	3	0	0	2	0	0	2	2	2	0	0	1	2	2	2	2	2	0	0	0	2	2	2	2	2	0	0	2	2	1	2	*

Table 28: Construct by construct matrix - developers

Table 28 shows that developers perceive the following constructs as similar:

- more practical (C1)
- more pragmatic expectations of SPI (C3)
- use processes (C6)
- focus more on technical aspects of SPI (C7)
- do 'real' work (C38)

Developers use these constructs to differentiate themselves from the other two groups.

Table 29 shows the re-arranged grid of developers perceptions. Table 29 suggests that according to developers, practitioners who directly use processes tend to concentrate on the technical aspects of SPI. It also implies that these practitioners tend to be more practical in their views on SPI and have pragmatic expectations of SPI. It implies that these practitioners see themselves as those who actually do 'real' work.

No.	Constructs (Rating = 1)	Elements			(Rating = 0) Constructs
		Dev	PM	SM	
C1	More practical	1	0	0	Less practical
C3	More pragmatic expectations of SPI	1	0	0	Idealistic expectations of SPI
C6	Use processes	1	0	0	Scarcely use processes
C7	Focus more on technical aspects of SPI	1	0	0	Focus on cost and time
C38	Do 'real' work	1	0	0	Do not do 'real' work
C2	Delegate work	0	1	1	Do not delegate work
C5	Define and suggest processes	0	1	1	Do not define processes. Just follow processes
C8	Have a wider perspective of SPI	0	1	1	Have a narrower perspective of SPI
C15	Communicate via reports and presentations	0	1	1	Do not communicate via reports and presentations
C17	Possess overall view of SPI	0	1	1	Possess a limited view of projects
C21	Interact and interface with customers	0	1	1	Little customer interaction. Isolated from customers
C22	Aware of business and strategic issues. Have access to long-term info	0	1	1	Less aware of business and strategic issues. No access to long term info
C23	Higher accountability	0	1	1	Lower accountability
C24	Higher salaries	0	1	1	Lower salaries
C25	Knowledge of cross fertilisation	0	1	1	Do not hear about other projects
C31	Have authority	0	1	1	Do not have authority
C32	Can see bigger picture	0	1	1	Do not see bigger picture
C33	Providing driving/guiding force	0	1	1	Do not provide/guiding force
C37	Consult on decisions affecting developers	0	1	1	Are not consulted on developer decisions
C39	C39: Blame developers for problems	0	1	1	Do not blame developers for problems
C4	Possess a project focus	1	1	0	Lack of project focus
C10	Closer to actual production	1	1	0	Not close to actual processes
C11	Interested in nature of problems	1	1	0	More interested in the effects of problems than the nature of them
C13	Technical background, technically skilled, technical focus	1	1	0	Non-technical background. Not technically skilled. Not worried about technical issues
C14	Subject to day-to-day interruptions	1	1	0	Not subject to day to day interruptions
C18	Close to work environment and possess great knowledge of work	1	1	0	Not very close to the work environment
C19	Good knowledge of product	1	1	0	Poor knowledge of software
C26	Closer domain knowledge	1	1	0	Knowledge of domain not close
C27	Variable workload	1	1	0	Consistent workload
C28	Functions in teams	1	1	0	Function in isolation. Not in any given team with others
C34	Short term objectives	1	1	0	Do not work with short term objectives
C35	Grasp on 'reality'	1	1	0	Lack a grasp on reality
C9	Closer to senior management. Aware of overall goals	0	0	1	Far from senior management. Unaware of overall company goals
C40	People management skills	0	0	1	Lack of people management skills
C12	Very enthusiastic about SPI	1	0	1	Less enthusiastic about SPI. Most likely to 'kick it into touch
C16	Possess long term view of SPI	1	0	1	Possess a project term view of SPI
C29	Realistic about problems	1	0	1	Not realistic about problems
C30	Apply pressure on project managers	1	0	1	Do not apply pressure on project managers
C36	Have long-term roles (positions)	1	0	1	Have short term roles/positions
C20	Understand deadlines	0	1	0	Do not understand deadlines

Table 29: Reconstructed grid with similar elements and constructs next to each other (developers)

Developers also seem to be indicating that being *closer to senior management* and being *aware of overall company goals* is very similar to *possessing senior management skills.* This suggests that practitioners who are closer to senior management and are aware of overall company goals also posses management skills.

There are, on the other hand, constructs that do not appear either similar nor dissimilar to any other constructs. For example, *understand deadlines* does not appear to share any strong similarity or dissimilarity with any other construct. However, when it is reversed, to 'do not understand deadlines', it becomes similar to *have long term roles* and other similar constructs. In this study of only three elements, it is easier to envisage this similarity without having to reverse the constructs.

7.3.2 Project managers' perceptions

Thirty-two bi-polar constructs were elicited from project managers (see Appendix M). Table 30 shows an element by element matrix derived from project managers' constructs. The score range is between 0 and 32

	Dev	PM	SM
Dev	*	10	9
PM	*	*	13
SM	*	*	*

Table 30: Project managers' element by element agreement matrix

Table 30 shows that the strongest similarity between elements is, again, between project managers and senior managers. Developers and senior managers appear to have the least in common, however the extent of their differences is lower than from developer constructs.

Table 31 presents the construct by construct matrix of project managers' perceptions.

	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16	C17	C18	C19	C20	C21	C22	C23	C24	C25	C26	C27	C28	C29	C30	C31	C32
C1	*																															
C2	3	*																														
C3	1	1	*																													
C4	1	1	3	*																												
C5	3	3	1	1	*																											
C6	1	1	1	1	1	*																										
C7	1	1	3	3	1	1	*																									
C8	1	1	3	3	1	1	3	*																								
C9	1	1	1	1	1	3	1	1	*																							
C10	3	3	1	1	3	1	1	1	1	*																						
C11	3	3	1	1	3	1	1	2	1	3	*																					
C12	1	1	1	1	1	3	1	0	3	1	1	*																				
C13	0	0	2	2	0	2	2	2	2	0	0	2	*																			
C14	1	1	3	3	1	1	3	3	1	1	1	1	2	*																		
C15	1	1	1	1	1	3	1	1	3	1	1	3	2	1	*																	
C16	2	2	2	2	2	0	2	2	0	2	2	0	1	2	0	*																
C17	3	3	1	1	3	1	1	2	1	3	3	1	0	1	1	2	*															
C18	1	1	1	1	1	3	1	0	3	1	1	3	2	1	3	0	1	*														
C19	2	2	2	2	2	0	2	2	0	2	2	0	1	2	0	3	2	0	*													
C20	1	1	3	3	1	1	3	2	1	1	1	1	2	3	1	2	1	1	2	*												
C21	1	1	3	3	1	1	3	3	1	1	1	1	2	3	1	2	1	1	2	3	*											
C22	3	3	1	1	3	1	1	1	1	3	3	1	0	1	1	2	3	1	2	1	1	*										
C23	0	0	2	2	0	2	2	1	2	0	0	2	3	2	2	1	0	2	1	2	2	0	*									
C24	3	3	1	1	3	1	1	1	1	3	3	1	0	1	1	2	3	1	2	1	1	3	0	*								
C25	3	3	1	1	3	1	1	2	1	3	3	1	0	1	1	2	3	1	2	1	1	3	0	3	*							
C26	3	3	1	1	3	1	1	2	1	3	3	1	0	1	1	2	3	1	2	1	1	3	0	3	3	*						
C27	2	2	2	2	2	0	2	3	0	2	2	0	1	2	0	3	2	0	3	2	2	2	1	2	2	2	*					
C28	2	2	0	0	2	2	0	1	2	2	2	2	1	0	2	1	2	2	1	0	0	2	1	2	2	2	1	*				
C29	0	0	2	2	0	2	2	1	2	0	0	2	3	2	2	1	0	2	1	2	2	0	3	0	0	0	1	1	*			
C30	1	1	3	3	1	1	3	3	1	1	1	1	2	3	1	2	1	1	2	3	3	1	2	1	1	1	2	0	2	*		
C31	1	1	1	1	1	3	1	1	3	1	1	3	2	1	3	0	1	3	0	1	1	1	2	1	1	1	0	2	2	1	*	
C32	2	2	0	0	2	2	0	0	2	2	2	2	1	0	2	1	2	2	1	0	0	2	1	2	2	2	1	3	1	0	2	*

Table 31: Construct by construct matrix - project managers

Table 31 shows that project managers perceive the following constructs as similar:

- Want to use metrics (C1)
- Want to measure improvements (C2)
- Focus on big picture (C5)
- Design plan and monitor SPI programmes (C10)
- Lead and enable processes (C11)
- Sell SPI (C17)
- Appreciate financial imperatives of projects (C22)
- Possess a customer focus (C24)

Project managers use these constructs to describe the manager groups.

Table 32 presents a re-arranged grid of all project managers' constructs.

No.	Constructs (Rating = 1)	Elements			(Rating = 0) Constructs
		Dev	PM	SM	
C1	Want to use metrics to monitor developers	0	1	1	Do not want to use metrics to monitor developers
C2	Want to measure improvement	0	1	1	Want to see improvement
C5	Focus on big picture	0	1	1	Focus on specific deliverables
C10	Design, plan and model SPI programmes	0	1	1	Carry out the actual SPI work
C11	Lead and enable processes	0	1	1	Follow and do processes
C17	Sell SPI	0	1	1	Buy SPI
C22	Do appreciate financial imperatives of projects	0	1	1	Do not appreciate financial imperative of projects
C24	Posses more of a customer focus, hence commercial minded	0	1	1	Possess less of a customer focus, less commercial minded
C25	Have more responsibilities	0	1	1	Have less responsibilities
C26	Have people management responsibilities	0	1	1	Do not have people management responsibilities
C3	Good understanding of day to day development issues	1	1	0	Poor understanding of day to day development issues
C4	More in touch with processes	1	1	0	Less in touch with processes
C7	Short to medium term view / goals of SPI	1	1	0	Long term (strategic) view / goal of SPI
C8	Perceive SPI in terms of product and project	1	1	0	Perceive SPI in an organisational context
C14	Possess a project aim of SPI	1	1	0	Possess a strategic aim of SPI
C20	Work to schedules: to life cycles beginning to end	1	1	0	Do not work to schedules
C21	More technically minded and technical background	1	1	0	Less technically minded, less technical background
C30	Deliver objectives	1	1	0	Set objectives
C6	Complain to project managers	1	0	1	Do not complain to project managers
C9	Abdicate responsibility for SPI changes	1	0	1	Do not abdicate responsibility for SPI changes
C12	Share personal, long term aspirations for SPI	1	0	1	Do not have personal nor long term aspirations for SPI
C15	Have a feeling of owning the development processes	1	0	1	Do not feel they own the development processes
C18	Are more focused on key milestones	1	0	1	Less focused on key miles, more focused on project milestones
C31	Clear role definition	1	0	1	Role not defined
C13	Posses a technical understanding of processes	1	0	0	Non-technical understanding of SPI
C23	Motivations are mostly product orientated	1	0	0	Motivations are not solely product orientated
C29	Perceive that manager groups have a hidden agenda	1	0	0	Do not perceive hidden agenda
C16	Made more accountable for software development	0	1	0	Are not made more accountable for software development than PMs
C19	Possess a better view of the whole project	0	1	0	Posses a poor view of overall project. More likely to focus on single issues
C27	More responsible for project failures	0	1	0	Less responsible for project failures
C28	Empowered to make changes	0	0	1	Not empowered to make changes
C32	Main interest is financial	0	0	1	Main interest is not financial

Table 32: Reconstructed grid with similar elements and constructs next to each other (project managers)

7.3.3 Senior managers' perceptions

Thirty bi-polar constructs were elicited from senior managers (see Appendix N). Table 33 shows an agreement matrix of these constructs. This is on a score range of between 0 and 30.

	Dev	PM	SM
Dev	*	11	4
PM	*	*	15
SM	*	*	*

Table 33: Senior managers' element by element agreement matrix

Table 33 shows that, according to senior managers, the strongest similarity between the three elements is that between project managers and senior managers. Table 33 also shows that the set of elements with the least in common is developers and senior managers. This similarity score from senior manager constructs is similar to that from developer constructs. This research suggests that it may mean that these staff groups are

less optimistic and probably more realistic about what they have in common than project managers.

A construct by construct matrix derived from senior managers' perceptions is presented in Table 34.

	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16	C17	C18	C19	C20	C21	C22	C23	C24	C25	C26	C27	C28	C29	C30
C1	*																													
C2	3	*																												
C3	0	0	*																											
C4	3	3	0	*																										
C5	1	1	2	1	*																									
C6	1	1	2	1	3	*																								
C7	1	1	2	1	1	1	*																							
C8	3	3	0	3	1	1	1	*																						
C9	3	3	0	3	1	1	1	3	*																					
C10	0	0	3	0	2	2	2	0	0	*																				
C11	1	1	2	1	3	3	1	2	1	2	*																			
C12	2	2	1	2	2	2	0	1	2	1	2	*																		
C13	3	3	0	3	1	1	1	3	3	0	1	2	*																	
C14	3	3	0	3	1	1	1	3	3	0	1	2	3	*																
C15	1	1	2	1	3	3	1	1	1	2	3	2	1	1	*															
C16	1	1	2	1	3	3	1	2	1	2	3	2	1	1	3	*														
C17	1	1	2	1	3	3	1	2	1	2	3	2	1	1	3	3	*													
C18	3	3	0	3	1	1	1	2	3	0	1	2	3	3	1	1	1	*												
C19	3	3	0	3	1	1	1	3	3	0	1	2	3	3	1	1	1	3	*											
C20	2	2	1	2	0	0	2	2	2	1	0	1	2	2	0	0	0	2	2	*										
C21	1	1	2	1	1	1	3	1	1	2	1	0	1	1	1	1	1	1	1	2	*									
C22	1	1	2	1	1	1	3	2	1	2	1	0	1	1	1	1	1	1	1	2	3	*								
C23	3	3	0	3	1	1	1	2	3	0	1	2	3	3	1	1	1	3	3	2	1	1	*							
C24	3	3	0	3	1	1	1	3	3	0	1	2	3	3	1	1	1	3	3	2	1	1	3	*						
C25	3	3	0	3	1	1	1	3	3	0	1	2	3	3	1	1	1	3	3	2	1	1	3	3	*					
C26	1	1	2	1	3	3	1	1	1	2	3	2	1	1	3	3	3	1	1	0	1	1	1	1	1	*				
C27	1	1	2	1	3	3	1	2	1	2	3	2	1	1	3	3	3	1	1	0	1	1	1	1	1	3	*			
C28	1	1	2	1	3	3	1	2	1	2	3	2	1	1	3	3	3	1	1	0	1	1	1	1	1	3	3	*		
C29	3	3	0	3	1	1	1	2	3	0	1	2	3	3	1	1	1	3	3	2	1	1	3	3	3	1	1	1	*	
C30	1	1	2	1	3	3	1	1	1	2	3	2	1	1	3	3	3	1	1	0	1	1	1	1	1	3	3	3	1	*

Table 34: Construct by construct matrix - senior managers

This table shows that the examples of similar constructs include:

- More focused on deliverables (C1)
- Good appreciation of customer's perspective (C2)
- More aware of the financial costs of SPI (C4)

These are constructs that senior managers use to describe the manager groups. The results suggest that managers use 'typical' manager objectives to describe their role in SPI.

Table 35 presents the re-arrangement of constructs and elements reflecting how similar constructs relate to particular sets of elements.

No.	Constructs (Rating = 1)	Elements			(Rating = 0) Constructs
		Dev	PM	SM	
C1	More focused on deliverables. Manage commitment to deliverables	0	1	1	Less focus on deliverables. Do work that focuses on deliverables
C2	Good appreciation of customer's perspective	0	1	1	Poor appreciation of customers' perspective
C4	More aware of the financial costs of SPI	0	1	1	Less aware of the financial costs of SPI
C8	Have responsibility for building environment for SPI	0	1	1	Do not have responsibility for building SPI environment
C9	Responsible for promoting SPI to others in company	0	1	1	Are not responsible for promoting SPI
C13	Have access to resources	0	1	1	Do not have access to resources
C14	Have power	0	1	1	Do not have power
C18	Have responsibility for the bottom line, for the margin	0	1	1	Do not have responsibility for the bottom lime, for the margin
C19	Have diverse set of responsibilities	0	1	1	Have limited set of responsibilities
C23	Have people management responsibilities	0	1	1	Do not have people management responsibilities
C24	Possess business acumen and business responsibilities	0	1	1	Do not have business responsibilities
C25	Look at the bigger picture	0	1	1	Do not look at the bigger picture
C29	Possess a business view of company	0	1	1	Do not possess a business view of the company
C3	More internally focused	1	0	0	More externally focused
C10	Generate ideas for SPI	1	0	0	Create environment for pushing forward generated ideas
C5	Implement SPI	1	1	0	Sponsor SPI
C6	Possess good visibility of projects	1	1	0	Possess poor visibility of projects
C11	Think it is up to Senior managers to make SPI work	1	1	0	Do not think that the responsibility for SPI lies solely on senior managers
C15	Technical background. Tend to do hands-on technical work	1	1	0	Do not necessarily come from technical background. Do not do hands-on technical work
C16	Share same technology	1	1	0	Do not share technology with other practitioner groups
C17	Share common interest in project performance	1	1	0	Do not share similar interest in project performance
C26	Less able to effect changes. Work within a framework	1	1	0	More able to effect changes. Do not work within framework
C27	More accepting of the status quo	1	1	0	Less accepting of the status quo
C28	More reluctant to change things	1	1	0	Less reluctant to change things
C30	Share common project goals	1	1	0	Do not share common project goals
C7	More aware of people issues (e.g. training development etc)	1	0	1	Less aware of people issues
C21	'Hate' project managers	1	0	1	Do not 'hate' project managers
C22	Have long term projects. Long term vision	1	0	1	Have short-term projects. Short term vision
C12	Face pressure from both technical and management processes	0	1	0	Do not face pressure from both technical and management processes
C20	Tend to manage a portfolio of things: have many objectives	0	0	1	Tend to have fewer objectives

Table 35: Reconstructed grid with similar elements and constructs next to each other (senior managers)

7.4 Discussion: Practitioners' perceptions of their role in SPI

The results of the RGT analysis helps to identify the differences and similarities in software practitioners' perceptions about their role in SPI and their concerns of SPI.

7.4.1 Understanding the SPI roles of practitioner groups

- Managers understand developers' role in SPI

Results from the RGT analysis show that the manager groups agree with the developers' position on SPI. There are few differences between the perception of developers' roles in SPI offered by developers themselves and that offered by managers. From this point of

view project managers and senior managers understand developers' position regarding SPI.

These results, however, suggest that developers are relegated to 'followers' of SPI whereas manager groups appear as the 'suggestors' and 'planners' of SPI. This may mean that despite all three staff groups agreeing to developers' role in SPI, the role itself is not helpful to the SPI effort.

The literature and published SPI case studies suggest that if companies want to achieve long term support for SPI from the people who work with the processes, then they should own the processes, and be in a position to maintain them and improve them. If, however, developers do not see themselves as 'suggestors' and owners of the processes, but rather as 'followers', then this may affect companies' capability for SPI success.

These results provide answers to the research question on the different perceptions of software practitioners of the role in SPI.

- Developers understand managers' role in SPI

Findings from this study show that developers agree with managers' perception of their role in SPI.

The RGT results show that most bi-polar constructs that developers use to distinguish their role in SPI from that of managers favour managers. For example, developers say that managers have a wider perspective of SPI, posses an overall view of SPI and are aware of both the business/strategic issues relating to SPI. This is a description that managers agree with. Also, even though developers are sometimes critical of management's position to SPI, they do generally appreciate that the higher level view of managers is beneficial. For example, developers say that managers are less close to the processes so their expectations of SPI are rather idealistic. They also say that managers

have 'knowledge of cross fertilisation', see a bigger picture and provide a driving/guiding force to SPI.

Overall, results from this study show that all three groups of software practitioners agree about their respective roles in SPI. These findings may seem reassuring, but suggest that SPI roles do seem derived from practitioners' line roles in software development. Since SPI is relatively new, there are no tried and tested roles for particular practitioner groups. As a result line roles have been borrowed from software development but may not be ideal for SPI. This research suggests that more work needs to be done to properly evaluate and design roles at all levels within SPI.

7.4.2 Similarities and differences in perception of SPI roles across practitioner groups

The study results show that there is varying similarity in the perception of SPI roles for the three practitioner groups. The two manager groups have the most in common. The results suggest that managers' development responsibilities seem to have been directly transferred to SPI.

Developers' and project managers' perceptions of their role in SPI arise out of the common experience they share in working very close to software processes. This perception may also be attributable to the common backgrounds and experiences that these two sets of practitioners bring to their roles.

Developers and senior managers have the least in common. However, the few concerns they have in common relate to the long-term aspirations for SPI and expectations of SPI. These issues are further discussed as follows:

- Time scales for SPI

All three staff groups agree that developers and senior managers see SPI as a long-term issue with both groups tending to be more enthusiastic about SPI. Developers, on the other hand, say that project managers tend to have a project lifetime view of SPI.

- Responsibility

All three groups define their role in SPI in relation to responsibility for SPI. Furthermore they all agree that manager groups have the responsibility for SPI.

Developers perceive managers as having responsibility for SPI because they define and suggest processes. Developers do not see themselves as doing these things. Project managers say that senior managers and themselves have responsibilities for leading and enabling SPI and have responsibility for planning and designing SPI programmes. Senior managers see manager groups as having responsibility for creating the conditions for SPI in their companies and also for promoting SPI to developers. Senior managers also see manager groups as being responsible for the business implications of SPI.

Again, this suggests that all staff groups appear to be defining their SPI roles in relation to what they already do in software development. This research suggests that if managers continue to view responsibility as solely theirs, they may find it difficult to pass such responsibility to developers. Also, developers' attitude towards responsibility can prevent them from assuming such roles. Overall, this can be detrimental to SPI because, as reported earlier in this research, making developers responsible for SPI is critical to SPI success.

- Empowerment and accountability

Developers see manager groups as having authority and therefore being accountable for SPI. Developers also see manager groups as having a wider perspective of SPI, leading to knowledge of other process areas within the company. Developers suggest that managers tend to see a bigger picture of processes than they do.

Senior managers say that manager groups have similar access to power and resources. They see managers, rather than developers, as empowered to make decisions related to processes. They also perceive that manager groups have access to information that developers do not have.

Empowering developers is reported as critical to SPI success [Billings *et al.*, 1994; Fitzgerald & O'Kane, 1999; Paulk et al., 1994b]. However, these study findings suggest that companies are not devolving power to developers. This research suggests that these findings can frustrate the SPI effort in companies.

- Technical proficiency

All staff groups agree that developers and project managers share similar technical backgrounds and experiences. Furthermore, developers and project managers share a project focus - they tend to deal with day-to-day activities, are task orientated and share common project goals.

All staff groups agree that common experience and background gives developers and project managers a better knowledge of the software process than senior managers.

- Customer interaction

Managers have a different perspective of the commercial imperatives of SPI from developers. All three staff groups suggest that this commercial imperative is formed from managers' interaction with customers. This suggests that managers' focus on customers' requirements makes their expectations of SPI more commercial. For example, some of managers' expectation of SPI would be, shorter cycle time, shorter time to market. On the other hand, because developers have little customer interaction, their expectations of SPI tend to be more technical, for example fault reduction, automating processes etc.

7.5 Conclusion

This chapter has described a study conducted to identify differences in the perception of software practitioners' role of SPI. The study was undertaken to understand the nature of the differences in the motivators and de-motivators for SPI which different groups of software practitioners report. In this section, this research presents a summary of the findings and shows how the findings answer the research question:

> What are the differences in the SPI motivators and de-motivators that senior managers, project managers and developers report?

7.5.1 Summary of findings

- Practitioners see responsibility for SPI as a management responsibility. However, since developers are closer to the processes and generally have better technical knowledge of these processes, they should be encouraged to share some of the responsibility for SPI.

- Developer groups do not perceive themselves accountable for SPI. Neither do they see themselves empowered for SPI. Findings from Chapters Five and Six suggest that these two issues can be directly addressed if developers were encouraged to share some responsibility for SPI. For example, if developers were encouraged to own some of the processes and also made responsible for changing and maintaining these processes, they will become accountable for these processes. Such accountability can then empower developers to support SPI better.

- Managers have a wide outlook towards SPI. Managers take into consideration both the commercial and technical imperatives of SPI (which is probably influenced by their interaction with customers), but developers have a greater knowledge of the technical processes and are therefore better placed to suggest improvements to the processes. Ideally, these two competencies should appear together. Unfortunately,

they seem to be attributes of two groups of practitioners with the least in common. This finding makes the need for consultative SPI programmes more compelling.

- Project managers have a shorter-term view of SPI than the other two groups. This implies that their support for SPI is relatively short-term, too, since it is usually tied in with project objectives. This finding suggests that these differences do not make SPI implementation in companies cohesive. Which means that project managers may concentrate more on SPI practices that show results within the lifetime of a project, whereas other practitioner groups may be concentrating on more long-term objectives for SPI.

7.5.2 Differences in SPI perceptions reflect differences in motivators and de-motivators

The findings from this study suggest that differences in software practitioners' perception of their role in SPI are related to the roles that staff groups play in software engineering, generally, but not in SPI particularly. The results suggest that developers may have become restricted by their development responsibilities, and may be unable to assume SPI responsibilities advocated by successful SPI case studies.

These differences are similar to the differences in software practitioners' motivators and de-motivators for SPI, which also show that the differences between practitioner groups relate to software practitioners' current responsibilities within software development.

Overall, this chapter has described a study that identified the differences in software practitioners' perceptions of their SPI roles. The findings explain the nature of the differences in motivators and de-motivators of software practitioners for SPI, across staff groups, that were established in Chapters Five and Six.

Chapter Eight presents a summary of the four studies conducted in this report and explains how these findings are used to validate the guidelines recommended by the literature for improving software practitioners' support for SPI.

Chapter Eight: Research findings

8.1 Introduction

This chapter sets out to present overall recommendations from this research. It uses findings from the four studies described in Chapters Four, Five, Six and Seven to validate the set of guidelines suggested by the literature for improving software practitioners' support for SPI. The study findings that confirm the literature guidelines are presented as empirically based recommendations of this research.

The following is how this chapter is structured:

Section 8.2 summarises the findings from the studies in Chapters Four, Five, Six and Seven. Section 8.3 shows how the research findings validate the literature guidelines. Section 8.4 concludes this chapter.

8.2 Research findings

The four studies conducted in this research investigate themes identified by the literature as underpinning software practitioners' motivation and support for SPI.

The first study focused on the perception SPI managers have of the motivators and de-motivators for SPI. This study provided a 'snap shot' of the issues that underpin software practitioners' support for SPI.

The second study identified the motivators for developers, project managers and senior managers. This study identified the issues that SPI managers should be improving to increase software practitioners' support for SPI.

The third study identified the de-motivators of developers, project managers and senior managers for SPI. This study identified the issues that SPI managers should be addressing to decrease software practitioners' de-motivation for SPI.

The fourth study identified the differences and similarities in the perception of SPI roles for developers, project managers and senior managers. This study established reasons for the differences in the motivators and de-motivators for SPI across staff groups, and the nature of these differences.

In this section, this research provides a summary of the findings of all four studies.

8.2.1 SPI managers' perception of the motivators and de-motivators for SPI

The following are a summary of the findings of SPI managers' perception of SPI.

- Figure 11 in Chapter Four shows that a majority of SPI managers generally have positive experiences of SPI. The most experienced managers tend to be most positive about SPI.

- A high number of SPI managers report optimistic expectations of SPI success. SPI managers are of the opinion that SPI does improve software quality. They are also of the opinion that SPI is cost beneficial.

- Figure 15 in Chapter Four suggests that the perception of a number of SPI managers is that senior managers are not supportive of SPI in their companies. Also, some SPI managers have the perception that developers are not enthusiastic about SPI. This evidence is suggested by Figire16 in Chapter Four. Overall, these SPI managers indicate that there is insufficient support for SPI in companies.

Overall, the findings from this study indicate that a majority of change agents for SPI have a realistic outlook on SPI. However, some are frustrated by the support for SPI that

they receive in companies. The findings show that senior management support and developers' enthusiasm for SPI are issues that merit further investigation. These are the subjects of the subsequent two studies: identifying what motivates and de-motivates support for SPI amongst software practitioners.

8.2.2 Motivators for SPI

The study of software practitioners' motivators for SPI identified the characteristics of practitioners' motivators. This study showed that there are differences in the motivators for SPI across staff groups. It also showed that some of the motivators cited by software practitioners have strong associations to each other and are therefore highly likely to be cited together by software practitioners.

The following is a summary of the study findings:

- A proportion of software practitioners seem to be motivated by SPI work itself. Table 15 in Chapter Five shows that developers, especially, mention job satisfaction as a motivator for SPI.

- A majority of software practitioners seem to support SPI if they are provided with opportunities for achievement. For example, project managers suggest that practitioners will be motivated to support SPI if they are empowered.

- A minority of software practitioners appear to be motivated by the opportunity for advancement to support SPI. Some developers are motivated by training opportunities to support SPI.

- Reward for SPI seems to motivate some software practitioners to support SPI.

- A proportion of software practitioners seem to be motivated to support SPI when they receive recognition for the work that they do on SPI. This recognition can be provided by providing practitioners with feedback.

- Increased responsibility seems to motivate some software practitioners to support SPI. Increased responsibility can take the form of giving practitioners ownership of the processes they work in as shown by Tables 15, 16 and 17 of Chapter Five.

- A minority of software practitioners seem to be motivated by the amount and quality of technical supervision they receive for SPI. If their leaders have knowledge of their field they will be motivated to support SPI.

- The conditions within which software practitioners work also seems to motivate some of them to support SPI. For example, some software practitioners are motivated by the use of tools to automate their processes. They are also motivated by processes that are easy to follow.

- A majority of software practitioners seem to be motivated by senior management support and commitment for SPI. Practitioners indicate that if they can see evidence of senior management support for SPI, then they are likely to be motivated to support SPI.

8.2.3 De-motivators for SPI

The study of software practitioners' de-motivators for SPI identifies the factors that frustrate practitioners' support for SPI. This study shows that there are differences in the de-motivators for SPI across staff groups. It also shows that there are strong associations between some of these factors, so that practitioners are highly likely to cite them together.

The following are the findings from this study:

- Table 21 in Chapter Six suggests that most software practitioners seem to be de-motivated by a general lack of resources for SPI. In particular, software practitioners are de-motivated by the constraints that a lack of resources imposes on their practice of SPI. For example, shortage of people to do SPI and a lack of time for SPI.

- A number of software practitioners seem to be generally resistant to SPI. This resistance is caused by negative experiences of SPI, fear of loosing control and a general unwillingness to adopt new practices. This resistance in turn frustrates software practitioners' SPI effort.

- Some software practitioners seem to be de-motivated by a lack of evidence of SPI success. Practitioners are de-motivated when they are unable to obtain evidence of the direct evidence of SPI.

- Shortage of SPI skills also seems to de-motivate the effort of a minority of software practitioners for SPI. Some practitioners indicate working with inexperienced staff de-motivates their SPI effort.

- Tables 22 and 23 in Chapter Six suggest that SPI initiatives that are imposed upon software practitioners seem to be de-motivating to some of them. These software practitioners find a lack of consultation and inadequate communication de-motivating.

8.2.4 Practitioners' perception of their roles in SPI

The two previous studies on software practitioners' motivators and de-motivators for SPI suggested differences across staff groups. This study of practitioners' perceptions of their role in SPI is conducted to investigate the reasons behind these differences. The results of this study suggest that software practitioners' perception of their role in SPI differs across practitioner groups too and this difference is also related to their software development responsibilities in companies.

The following is a summary of the findings from this study:

- Tables 29, 32 and 35 in Chapter Seven suggest that all practitioner groups seem to see responsibility for SPI as a management responsibility.

- Developers do not seem to see themselves accountable for SPI, nor do they see themselves responsible for SPI. This is suggested by construct analysis presented in Table 29 of Chapter Seven. As a result, they are not sufficiently empowered to support SPI.

- Managers seem to have a wider outlook of SPI than developers. Managers see a bigger picture that incorporates the commercial imperatives of SPI. Developers, on the other hand, tend to focus on the technical aspects of SPI, and have better knowledge of the technical processes than manager groups.

- Project managers have a shorter-term view of SPI and so may concentrate on short term goals of SPI, whereas developers and managers tend to concentrate on longer-term goals of SPI.

8.3 Validating literature guidelines with research findings

This section shows how the research findings from the four studies have been used to validate the guidelines suggested by the literature as critical to improving software practitioners' support for SPI.

Table 36 shows how the literature guidelines discussed in Chapter Two are supported by findings from this research. It shows that most of the guidelines from the literature have been supported by findings from the four studies. These guidelines are offered as

empirically supported recommendations from this research. The recommendations are expanded on and some are explained with examples.

		Research study findings															
Code	Literature guidelines for increasing software practitioner's support for SPI in companies	Visible management support for SPI	Secure practitioners' buy-in	Transfer ownership of processes to practitioners	Communicate SPI success	Provide SPI training	Standardise practices	Share SPI best practice	Visibly re-prioritise SPI	Dedicate resources to SPI	Make SPI objectives relevant to all	Provide internal leadership for SPI	Manage internal resistance	Reward SPI work	Encourage SPI forum	Initiate SPI from within the company
I	Senior management commitment	√														
II	Secure practitioner's buy-in		√													
III	Process ownership			√												
IV	Evidence / Visible benefits				√											
V	Train practitioners					√										
VI	Mentor practitioners	Not supported by study findings														
VII	Standardise						√									
VIII	Sharing best practice							√								
IX	Proper prioritising of SPI								√							
X	Dedicated resources to SPI									√						
XI	Setting relevant / realistic objectives										√					
XII	Internal leadership that provides vision											√				
XIII	Overcoming internal resistance to SPI												√			
XIV	Reward SPI effort													√		
XV	SPI forum														√	
XVI	Well respected SPI people	Not supported by study findings														
XVII	Run SPI from within															√

Table 36: How literature guidelines are supported by research findings

8.3.1 Visible management support for SPI

This research suggests that software practitioners' support for SPI can be improved if senior management shows visible support for SPI programmes. Practitioners are motivated by such support.

This research suggests that management support for SPI can be made visible in the following ways:

- Senior managers attending SPI meetings
- Resource allocation to SPI programmes are publicised in company bulletins
- Senior management mentioning SPI programmes in other company meetings

8.3.2 Secure practitioner buy-in

Securing buy-in from all levels of software practitioners improves the support for SPI. This research suggests that buy-in is achievable through consultation with practitioners. Through consultation, software practitioners' potential fears about SPI can be addressed. This research further suggests that the process of consultation can also enrich SPI programmes by providing a forum to include suggestions from software practitioners. Overall, software practitioners support programmes to which they have contributed.

Conversely, practitioners are de-motivated by SPI initiatives that are not consultative.

8.3.3 Transfer ownership of processes to practitioners

Giving practitioners ownership of the processes they work in motivates them to support SPI. This recommendation is a continuation of the consultation process. Practitioners are highly motivated by intrinsic aspects of their job and process ownership is intrinsic to the job that software practitioners do. This research suggests that for software practitioners, ownership of processes implies more responsibility. Increased responsibility, as this research has shown in earlier chapters, has a higher chance of gaining practitioners' support for SPI over a long period of time.

8.3.4 Communicate SPI success

Communicating SPI success to software practitioners can increase their support for SPI. A high number of software practitioners are motivated by evidence of SPI success. This research suggests that it is also important to communicate early benefits of SPI to software practitioners as it serves as feedback on their SPI effort. Findings from studies in this research have supported what the literature reports on motivating software practitioners by providing them with knowledge of the actual results of SPI. That is, a high number of software practitioners have reported that they are motivated to support SPI if the results of SPI are better communicated to them.

8.3.5 Provide SPI training to all practitioners

It is important to provide software practitioners with SPI training. Software practitioners are highly motivated by the opportunity for advancement and growth. This research suggests that training offers some software practitioners the opportunity to advance themselves and to grow professionally.

On the other hand, lack of training de-motivates some software practitioners for SPI. SPI skills shortage is a big de-motivator to SPI and some software practitioners' support for SPI can be improved is they are provided with the skills to do that work.

This research suggests that in this instance, training serves both as an intrinsic motivator - a factor that motivates software practitioners internally to support SPI - and also an external motivator, in that shortage of training can de-motivate software practitioners from supporting SPI.

8.3.6 Standardise SPI practices

Support for SPI can be improved if SPI practices are standardised throughout the company. A minority of practitioners indicate that they are motivated by standardised

practices because it provides a common platform and enhances communication. Making SPI compulsory can also motivate a number of software practitioners to support SPI.

Lack of standardisation, on the other hand de-motivates some software practitioners.

However, standardisation and compulsion are punitive stimuli that can motivate software practitioners to support SPI, but only for a short term. This research suggests that standardisation and compulsion should be implemented with the knowledge that their effect on SPI success can only be short-term. As a result, this research recommends that they be implemented in conjunction with other, preferably rewarding, motivators.

8.3.7 Share SPI best practice

Standardising SPI practices can encourage the sharing of best practice. A number of software practitioners indicate that they will be motivated to support SPI when best practices are shared across operations, teams or departments. This research suggests that these software practitioners may be indicating that sharing of best practices saves time and reduces incidents of 're-inventing the wheel'.

8.3.8 Visibly re-prioritise SPI

A number of software practitioners are de-motivated from supporting SPI when they get the impression that SPI has lower priority to other project work. This research suggests that it is therefore vital to show that SPI has equal priority to other project work through the visible and explicit support from senior management.

8.3.9 Dedicate resources to SPI

This research suggests that one of the many ways of increasing the priority of SPI is by allocating dedicated resources to SPI. This is because when project budgets explicitly account for SPI, it can indicate to software practitioners that SPI is important. In fact, dedicating resources to SPI is on its own a strong motivator for supporting SPI. This

research suggests that dedicated resources for SPI gets SPI work done, indicates senior management support for SPI and also shows software practitioners that SPI is important to the company. This research suggests that dedicated SPI resources can have several impacts on software practitioners' motivations to support SPI.

8.3.10 Provide internal leadership for SPI

Some practitioners' support for SPI can be improved through the provision of internal leadership for SPI programmes. Managers who are knowledgeable of SPI can motivate some software practitioners. On the other hand, some SPI practitioners are de-motivated by managers who lack technical knowledge of the development processes and also managers who lack direction. This research therefore suggests that to improve support for SPI, change agents must be knowledgeable about SPI and be able to provide the direction needed for SPI. This research suggests that this can be achieved by providing training for SPI managers and SPI teams.

8.3.11 Manage internal resistance

This research suggests that SPI support can be improved if practitioners' resistance is properly managed. SPI managers can reduce resistance to SPI by providing a majority of software practitioners with evidence of the benefits of SPI. This research suggests that these benefits can be at a personal, project or an organisational level. So, for example, managers can provide software practitioners with evidence of how SPI can assist in reducing faults or shortening coding time. At the same time, managers can show practitioners how SPI can assist them to improve project performances in terms of reduction in schedule slippage. At the organisational level, managers can show practitioners how SPI helps to improve customer satisfaction through better requirements capture and improved product quality. This research further suggests that the closer the evidence happens to be to practitioners' current operations, the better it can be at motivating software practitioners to support SPI.

8.3.12 Reward SPI work

Rewarding some software practitioners for SPI seems to directly motivate them. However, reward schemes generally tend to be extrinsic to the jobs that software practitioners do, therefore their effect, as motivators, tend to be limited. This is because software practitioners are mainly motivated by the intrinsic factors in a job. This research suggests that to make reward schemes more motivating, SPI managers should convert the nature of such schemes into factors that are intrinsic to the jobs practitioners do, for example, rewarding practitioners for SPI through the staff appraisal system can motivate them better than, say, increased remuneration.

8.3.13 Encourage SPI forum

SPI managers should encourage SPI forums in their companies. SPI forums allow some practitioners to contribute towards SPI both directly and indirectly. This research suggests that by contributing towards SPI in this manner, software practitioners can experience a greater sense of involvement in the SPI programme. A SPI forum can also serve as a medium for practitioners to receive feedback about SPI.

8.3.14 Initiate SPI from within the company

SPI programmes that are initiated from within a company seem to be able to improve software practitioners' support for SPI. Some software practitioners are de-motivated by programmes that are imposed upon them from the corporate level. This research suggests that these software practitioners are also de-motivated by SPI programmes that are led by external consultants.

8.3.15 Make SPI objectives relevant to all practitioners

Setting SPI objectives that are realistic and relevant to practitioners seems to improve support for SPI. Some software practitioners are motivated to support SPI programmes when these programmes have objectives that are relevant to their own circumstances. It

is, therefore, important that SPI objectives set by companies reflect the aspirations of software practitioners. This research suggests that this can be achieved through consultation with software practitioners at all levels of the companies. Findings from this research suggest that the process of consultation can reveal the differences in practitioners` motivation and de-motivation for SPI.

The literature recommends making SPI objectives relevant to all SPI groups and the study findings support this recommendation. The findings suggest that the differences in the motivators and de-motivators across staff groups can be addressed if they are perceived in the light of software practitioners' software development roles. For example, practitioners' support for SPI can be improved if the factors implemented to improve their motivators and reduce their de-motivators are related to their individual software development responsibilities. Managers will support SPI more if it improves their management responsibilities. Developers will be more enthusiastic about SPI if it meets their day-to-day operational objectives.

8.3.16 Guidelines not supported by study findings

As indicated in Table 36, two of the guidelines suggested by the literature were not supported by findings from the research studies. These are:

- Mentoring practitioners

Even though the literature suggests that mentoring software practitioners during SPI is an important factor for improving support for SPI, none of the studies conducted in this research support this claim. Mentoring involves social interaction, usually on a one-to-one basis. However, software practitioners have a notoriously low need for social interaction, as discussed in Chapter Two. It can therefore be suggested that the study findings have not confirmed this particular suggestion because software practitioners do not appreciate the social interaction aspect of mentoring.

- Well respected SPI people

There was little evidence from the research findings to support the suggestion that software practitioners are motivated to support SPI if the people engaged in SPI work are perceived to be well respected. Software practitioners report that knowledgeable SPI people motivate them to support SPI, but they do not imply that these people need to be well respected in order to motivate them to support SPI.

8.4 Conclusion

This chapter has shown how the guidelines suggested by the literature to be important to improving software practitioners' support for SPI have been supported by the four studies conducted in this research. The process of validation gives these guidelines a better empirical basis. The validated guidelines are offered as recommendation of this research to companies embarking upon SPI. This research suggests that these recommendations can give these companies better insight into their software practitioners' motivations for supporting SPI.

Chapter Nine: Conclusion

This chapter summarises the research programme. It begins by re-stating the research hypothesis and discussing the extent to which this has been established under the four themes developed in Chapters Four, Five, Six and Seven. It summarises the factors that the research suggests are particularly relevant to improving software practitioners' support for SPI. It discusses how the findings from this research can be useful to the software industry. It also offers a critique of the research methodology adopted and what could have been done differently if this research were to be done again. It also reflects on the research methodology both in terms of its success and its use in future research. Finally, it discusses other research that might be evolved out of this research.

9.1 Summary of research findings

The findings from this research largely confirm that the motivators and de-motivators of SPI suggested by the literature are supported, in practice, by software practitioners' opinions of their motivators and de-motivators for SPI. Whilst most of the guidelines suggested by the literature have been supported by the research findings, there were two guidelines for which there was insufficient evidence to support. These were the guidelines on mentoring software practitioners and ensuring that SPI people were well respected. Firstly, the findings suggest that because mentoring involves social interaction on an individual basis, software practitioners may not be particularly interested in it. This is because software practitioners have been reported to be notoriously adverse to social interaction. Secondly, software practitioners do not specifically report that SPI people need to be well respected to motivate them to support SPI even though SPI people with technical knowledge of SPI processes motivate software practitioners. The findings suggest that software practitioners do not necessarily equate knowledgeable people to well respected people.

The research findings are underpinned by four themes:

Theme 1: SPI managers' perception of the motivators and de-motivators for SPI

The perception of SPI managers as change agents on the motivators and de-motivators for SPI were examined under the first theme. This theme was supported by the literature suggestion that as change agents, the perception, experience and expectation of SPI managers, is important to SPI success. Also, that as change agents, SPI managers serve as a good 'barometer' of support for SPI in their companies. The results from this research showed that SPI managers perceived practitioners' support for SPI as inadequate. They perceived that senior management were not adequately committed to SPI and software practitioners were not sufficiently enthusiastic about SPI. These findings suggested that addressing senior managers' commitment and developers' buy-in to SPI was integral to improving overall support for SPI.

Themes 2 and 3 (amalgamated): Motivators and de-motivators for SPI

The motivators and de-motivators for SPI that software practitioners reported in practice were explored under the second and third themes. These themes were supported by the literature suggestions that software practitioners' motivators and de-motivators were integral to their support for SPI. The findings from this research showed that the factors that motivated software practitioners were senior management commitment, obtaining practitioners' buy-in, good communication within the SPI programme, high priority given to SPI, internal leadership for SPI, the standardisation of SPI across the company and reward for SPI work. The findings also showed that the factors that de-motivated software practitioners were mostly a lack of the motivators. However, internal resistance to SPI and imposed SPI initiatives were shown to be two specific de-motivators for SPI.

Theme 4: Differences in motivators and de-motivators across staff groups

The differences in software practitioners' perceptions of their role in SPI were examined under the fourth theme. This theme was underpinned by the literature suggestion that software practitioners have different responses to SPI and that exploring and addressing these differences was important to improving their support for SPI. The literature also

suggested that exploring peoples' perception gave insight into their responses. As a result, the study used the differences in perception to explain the differences in responses. Findings from this study showed that different staff groups have different perceptions of their role in SPI. It showed that these differences were related to the software development responsibilities that software practitioners have in their companies. These findings reflected the differences in software practitioners' motivators and de-motivators for SPI, which were related to their software development responsibilities too. Overall this study supported the literature guideline that suggests that in order to improve the support for SPI, the objectives of SPI should be designed to reflect practitioners' software development objectives too. By so doing, these objectives become more relevant to software practitioners and will more likely improve their support for SPI.

9.2 Research recommendations

On the basis of what the research themes have recommended, a set of factors has been drawn out as the most important for gaining software practitioners' support for SPI. Table 37 shows these factors.

Order of importance based upon evidence from studies	Factors that improve software practitioners' support for SPI
4th	Senior management commitment to SPI
5th (joint)	Secure software practitioners' buy-in
7th	Process ownership
2nd	Evidence of the benefits and successes of SPI
10th	Training practitioners in SPI skills
14th	Standardising SPI across the company
11th	Sharing best practice
12th (joint)	Proper prioritising of SPI
1st	Dedicated resources to SPI activities
15th	Setting relevant and realistic objectives for SPI
12th (joint)	Internal leadership that provides vision for SPI
3rd	Overcoming internal resistance to SPI
9th	Rewarding SPI effort
5th (joint)	SPI forum
8th	Run SPI from within

Table 37: Research recommendations

9.3 Application of research findings

The findings from this research provide an empirical basis for a set of guidelines that have been identified as critical to improving software practitioners' support for SPI. Even

though these guidelines have already been published in the literature, their appearance in the literature is disparate and the basis of what is published is not always convincing. In this research, these guidelines have been collated as a set and validated by independent studies that are based upon what software practitioners of all levels, including SPI managers, report in practice. The empirical basis provided to these guidelines through this validation process can give more confidence to the guidelines and persuade more companies to adopt them in the effort to secure practitioners' support for SPI.

If more companies adopt these empirically supported research recommendations, then both the literature and the reports from the practitioners' perceptions suggest that they are likely to improve support for SPI in their companies. As a result, there is likely to be more evidence of the positive impact of SPI on software quality. As suggested by the various accounts in the literature, especially work coming from the SEI, SPI does indeed improve software quality. The main concern of this research has been to identify ways of improving the support for SPI amongst software practitioners. Therefore if companies are more willing to adopt the guidelines suggested by this research due to the empirical evidence that supports them, then it is likely that the SPI experience of these companies will be more successful and the problem of software quality would be addressed better.

The recommendations of this research will be helpful to managers of companies embarking on SPI programmes. These recommendation can give them better insight into designing strategies for SPI that improve software practitioners' support for SPI by addressing the motivators and de-motivators and tackling the staff group-specific differences between these factors.

Overall, these recommendations can help to address some of the important people issues of SPI. According to Trudel and Laporte [1998], it is important to have knowledge of the motivation factors in order to effectively manage the human dimensions of software process improvement [Laporte and Trudel, 1998]:

> *"Managing people issues is important to the success of technical changes"*

By paying greater attention to these people issues, the software community will be addressing the problems of software quality from the areas where these problems emanate and not necessarily applying technical solutions to non-technical problems because technical solutions appear to be better understood. In the words of DeMarco and Lister [1987] the software community will look for its keys where it lost them and not "*in the adjacent street because the light is better there*".

9.4 Critique of methodology

This section presents the limitations identified in this research. It identifies issues that would be done differently if this research were to be done again.

9.4.1 The use of perception data only

Data collected in this research characterises opinions data that not been verified directly. It is therefore possible that what software practitioners perceive to motivate or de-motivate them is not actually supported by their behaviour. In an ideal world the perception data would be supplemented by behaviour data collected through observation.

The collection of behaviour data on such a large scale would have been very difficult to achieve. Also, because of the highly sensitive nature of the operations of some of the companies in this research, access to behaviour data would have been difficult to attain. Without access to observe all the companies in these studies, the reporting of behaviour data would have been prejudiced and less representative of the population of companies and indeed software practitioners. This is because in order to make qualitative research of this nature replicable, it is necessary to adopt the same research process for all the members in the study sample. Failure to follow this process can skew the research results towards or away from particular members of the sample. In so doing, the results being reported then become less representative of the population, and by implication, less generalise-able to the population.

9.4.2 One to one interviews as opposed to focus group discussions

Even though over 200 software practitioners were involved in this research the data points used for analysis were less because focus groups were adopted in the data collection. One to one interviews would have resulted in a much larger data size and potentially sharpened the validity of the results being reported.

Also, it could be argued that the use of focus groups compromises the independence of the data being reported. This is because within a focus group, one issue could be mentioned several times by different people and still be recorded once whilst, several issues may be mentioned by the same person and be recorded several times. This may have masked the weighting of the issues reported. One to one interviews may have reflected more accurate weightings of the issues discussed.

One to one interviews, however, are very time-consuming so would not have enabled the number of practitioners that took part in this study. Also, one to one interviews tend to isolate interviewees, thus making them less able to express the issues that they tend to do within the pseudo anonymity of a group.

9.4.3 Structured interviews in place of SPI managers' questionnaire survey

Using questionnaires to survey SPI managers' perception of the motivators and de-motivators for SPI had the disadvantage of pre-empting the issues reported in that study, even though those issues were correctly identified by other studies as relevant to the theme. The use of structured interviews would have made the validation of the study findings and the research findings stronger for two reasons. Firstly it would have shown that SPI managers cited the issues independently without any suggestion from the researcher. This independence could have provided more confidence in the study findings. Secondly it would have demonstrated that all four distinct studies arrived at the same set of complementary findings that were used to validate the literature guidelines.

This would have provided a better demonstration of the application of triangulation in this research.

However, the use of structured interviews would have decreased the data size, as it would have been near impossible to have arranged and conducted structured interviews with the eighty SPI managers that took part in this study. A questionnaire survey made it possible to sample a larger group of SPI managers than one to one interviews would have.

9.4.4 Other themes underpinning support for SPI

The literature identifies other themes, apart from those explored in this research, that underpin software practitioners' support for SPI. For example, skills and attitudes. Exploring these other themes in this research could have provided a more rigorous basis for validating the guidelines suggested by the literature. This is because if a study of software practitioners' attitudes towards SPI programmes had produced findings that also confirmed the literature guidelines, then the recommendation made from the guidelines would have been made from a stronger process of validation. Even if the study of attitudes had refuted some of the literature guidelines the resultant set of reduced recommendations would have also been subjected to validation from an extra set of study findings. So that either way, the recommendations would have been derived from a more rigorous process of validation.

Investigating such issues like support for SPI is always very difficult due to the many intangible factors that need to be considered. It is therefore not always possible to address all the underlying themes. Not all the themes that under-pin software practitioners' support for SPI have been exhausted in this research. The four themes identified were those found to be most relevant to improving support for SPI. However, the fact that some themes have not been investigated should not de-value the research. As in most empirical research, this research has attempted to make a contribution towards understanding how to improve support for SPI and thereby improve software quality.

9.5 Success and use of methodology in future application

This research has conducted four separate studies to validate a set of literature recommended guidelines. An empirical approach was adopted for these studies because it made it possible to validate what was reported in the literature with observations from practice. The use of an empirical approach has been very successful in this research since it is the only possible approach for investigating the issues set out by the aims of the research.

A triangulated data collection and analysis process was adopted in order to improve the confidence of the findings being reported through convergence. The use of triangulation in this research has been beneficial since it has been possible to show convergence of some findings. This has, in the process, helped to increase confidence in the results reported. This research suggests that future studies that aim to examine similar concepts in software engineering would benefit from such a triangulated approach because it builds checks and balances into the research process and overall, gives more confidence to the results of studies.

Overall, the particular study methods used were mainly social science research methods. The use of such methods in this research provided a rich and deep understanding of software practitioners' motivators and de-motivators. The use of these methods, both in terms of data collection and analysis has been useful. However, there are some minor difficulties, which are discussed below:

- Questionnaire surveys

Surveys are successful in generating a lot of data because they can target a larger sample. However their success is very much limited to how tightly the questions are focused on the issues being researched. The result is that, even though the data analysis process can be relatively straightforward, the resultant analysis can seem flat. Of all three methods of data collection used in this research, questionnaires represented the source of the least

interesting data. They are however the most effective route to reaching large samples of participants.

- Focus groups

The use of focus groups in this research has been successful for generating large amounts of in-depth qualitative data. This data has helped to gain further insight into the issues being investigated, through rich anecdotal accounts. However, the amount of data collected from focus groups can be overwhelming and this can hamper the data analysis process. Future application of focus groups for data collection should concentrate on in-depth collection of data from a fewer number of focus groups than was used in this research. This is because the process of categorising the data from focus groups can become laborious and in the process the researcher may experience "diminishing returns" if too many focus groups are used.

- The Repertory Grid Technique

The use of the repertory grid technique for both data collection and analysis represents a structured and easily replicable data collection and analysis process. RGT has proved very useful in this research both in terms of data collection and analysis. The format of data collection and analysis in RGT makes replication easy, thereby making it a useful technique for repeated studies.

- Multidimensional scaling

As discussed in Chapters Five and Six, the use of multidimensional analysis in this research did not particularly add any significant insight to the findings reported due to the small number of data points involved. However, as a tool for investigating future software engineering concepts, it is a novel technique and promises to be very useful for understanding the complexities of software engineering concepts.

9.6 Future work

The following provides an overview of future research that might be evolved out of this research.

9.6.1 Predictive model of software practitioners' motivations

One area of future work is the development of a predictive model of software practitioners' motivation.

Having set out to understand the factors that influence software practitioners' motivation for SPI in this research, it would be useful to be able to predict the consequences of these motivation factors on software practitioners' support. Predicting the consequences of motivations follows the general trend identified by Shepperd *et al* of applying prediction systems to software engineering [Shepperd et al., 2000]. Some prominent examples are the introduction of COCOMO [Boehm, 1984] in the 1980's to more recent and sophisticated approaches using Bayesian Belief Networks (BBNs) [Fenton et al., 2001] and Systems Dynamics [Lehman et al., 2000].

The research process for such work will set out to calibrate the effect of recommendations made in this research on software practitioners' support for SPI. This will involve collecting two sets of measurements for each recommendation. The first will be a measurement of practitioners' support for SPI before the recommendations are implemented and the second will be the measurements after the recommendations have been implemented. The measurements can be a combination of both perception data, where software practitioners indicate their level of support for SPI on a predetermined scale and objective measures, for example average number of faults introduced into the software in that time period. The collection of both perception and objective data in the second set of measurements should enable the analyses of both the impact of the recommendation on practitioners' support for SPI and also on the quality of the software product. Through the collection of these sets of data, the relationship between the

attributes and effects of motivation can be modelled using either BBN's or other prediction techniques. The aim will then be to be able to predict the effect of a motivator on software practitioners' support for SPI based upon the attribute of the particular motivator applied.

9.6.2 Extended Multidimensional scaling analysis of motivators and de-motivators for SPI

Another area of future work could be a more detailed study of the relationship within the motivators and de-motivators for SPI.

In this research, the concept of MDS was used to investigate the relationship between motivators and de-motivators for SPI in order to understand practitioners' motivators and de-motivators better. Even though some important relationships were identified from these analyses, the size of the data sets were too small to allow for the emergence of prominent relationships, from which new concepts could be suggested. Future work could expand the size of the sample by using questionnaire survey to target a larger group of participants. This work will survey participants using the list of motivators and de-motivators identified in the individual studies of this research. Participants could be asked to indicate whether they considered a particular factor to be a motivator (or de-motivator) to them or not. This should make it possible for participants' responses to be coded as binary responses and therefore make them ready to be used in the binary MDS analyses techniques available like the Smallest Space Analysis technique.

The advantages of using a questionnaire with pre-defined motivators and de-motivators in such work is that it will bypass the problems of data reliability that can be introduced through the coding of transcript data. It will also allow the targeting of a larger sample of participants. Finally, it will make it easier to replicate the research, so that repeated studies can be carried out to compare results. These advantages should enable better and more focused analysis and also improve confidence in the results.

Overall, such work will be useful in confirming and expanding some of the findings made in this research about the association between software practitioners' motivators and de-motivators for SPI. Through repeated work, general theories about the association between software practitioners' motivations may be established.

9.6.3 A survey of the findings of this research

Another area of future work is a large-scale survey of the research findings.

In this research fifteen factors have been recommended as a set of guidelines to improve software practitioners' support for SPI. Future work will set out to measure how software practitioners rate these factors. This work could be conducted using a postal questionnaire that targets software practitioners in UK companies. Data collected in this survey will include some personnel information so that analysis of results can be qualified by certain criteria like, staff groups, length of service and educational background. The questionnaire design will aim at a simple and short questionnaire in order to encourage a high response rate.

Overall, the aim of this work will be to establish a rating system for the findings of this research. It will also be to validate, again, these findings using a larger sample of software practitioners. The process will enable analysis of differences in software practitioners' rating of the findings based upon their personnel characteristics. For example, it will be possible to evaluate whether newly recruited software practitioners rated training higher than those who were more established in their positions.

9.6.4 Investigating SPI skills that improve software practitioners' support for SPI

A further area of future work could be an analysis of skills that improve software practitioners' support for SPI.

One of the findings of this research indicates that training software practitioners and providing them with appropriate skills can increase their support for SPI. Future work will investigate the type of skills that are important to this support for SPI.

This work will adopt a case study approach, involving one software company. A set of one-to-one interviews could be used with groups of software practitioners from three staff groups of senior managers, project managers and developers to ascertain the type of skills that software practitioners cite as important to their support for SPI. The one-to-one interviews will be supported by behaviour data. Behaviour data will be collected through participatory observation where software practitioners will be observed undertaking SPI work. For example, software practitioners will be observed during a software project review to ascertain the skills they appear to be using.

The two processes of data collection will be used to establish whether the SPI skills that software practitioners cite as important to their support for SPI are the same skills that practitioners appear be applying in their SPI.

Overall, the aim of this work will be to establish a set of core skills that have been confirmed to be important to practitioners' support for SPI. These results can give managers' in companies a better understanding of where to focus their SPI training programmes.

References:

Ahuja S (1999). Process Improvement In A Rapidly Changing Business And Technical Environment. *Fourth Annual European Process Group Conference*. Amsterdam, Holland. c303.

Axtell CM, Holman DJ, Unsworth KL, Wall TD and Waterson PE (2000) Shopfloor Innovations: Facilitating The Suggestion And Implementation Of Ideas. *Journal Of Occupational And Organisational Psychology* **73**:265-285.

Baddoo N and Hall T (2002a) De-motivators Of Software Process Improvement: An Analysis Of Practitioners' Views. *Journal Of Systems & Software* **accepted awaiting publication**.

Baddoo N and Hall T (2002b) Motivators Of Software Process Improvement: An Analysis Of Practitioners' Views. *Journal Of Systems & Software* **accepted awaiting publication**.

Baddoo N and Hall T (in review-a) Practitioner Roles In Software Process Improvement: An Analysis Using Grid Technique. *Software Process - Improvement and Practice*.

Baddoo N and Hall T (in review-b) Using Experiences And Opinions To Measure Software Process Improvement. *Information And Software Technology*.

Bannister D and Fransella F (1986) *Inquiring Man: The Psychology Of Personal Constructs*. Croom Helm, London.

Berdie DR and Anderson JF (1974) *Questionnaires: Design And Use*. The Scarecrow Press, Metuchen.

Billings C, Clifton J, Kolkhorst B, Lee E and Wingert WB (1994) Journey To A Mature Software Process. *IBM Systems Journal* **33 (1)**:46-61.

Black TR (1999) *Doing Quantitaive Research In The Social Sciences: An Integrated Approach To Research Design, Measurement And Statistics*. Sage.

Boehm BW (1984) Software Engneering Economics. *IEEE Transactions On Software Engineering* **10 (1)**:4-21.

Britton C, Jones S, Myers M and Sharif M (1997) A Survey Of Current Practice In The Development Of Multimedia Systems. *Information And Software Technology* **39 (10)**:695-705.

Brooks Jr. FP (1995) *The Mythical Man Month: Essays In Software Engineering*. Addison-Wesley, Reading, MA.

Bryman A (1990) *Research methods and organisational studies*. Unwin Hyman.

Burnard P (1991) A Method Of Analysing Interview Transcripts In Qualitative Research. *Nurse Education Today* **11**:461-466.

Chong HY (2001) *Misconceived Relationships Between Logical Positivism And Quantitative Research: An Analysis In The Framework Of Ian Hacking* in: http://seamonkey.ed.asu.edu/~alex/computer/sas/positivism.html 15th August 2001

Coolican H (1990) *Research Methods And Statistics In Psychology*. Hodder & Stoughton, London.

Couger JD (1988) Motivators And Demotivators In The IS Environment. *Journal Of Systems Management* **39 (6)**:36-41.

Couger JD and Adelsberger H (1988) Comparing Motivation Of Programmers And Analysts In Different Socio/Political Environments: Austria Compared To The United States. *Computer-Personnel* **11 (4)**:13-16.

Couger JD, Halttunen V and Lyytinnen K (1991) Evaluating The Motivating Environment In Finland Compared To The United States: A Survey. *European Journal Of Information Systems* **1 (2)**:107-112.

Couger JD and Ishikawa A (1995). Comparing Motivation Of Japanese Computer Personnel Versus Those Of The United States. *Proceedings Of The Twenty-Eigth Hawaii International Conference On System Science*. Los Alamitos, CA. 1012-1019.

Couger JD and O'Callaghan R (1994) Comparing The Motivations Of Spanish And Finish Computer Personnel With Those Of The United Stated. *European Journal Of Information Systems* **3 (4)**:285-291.

Couger JD and Zawacki RA (1980) *Motivating And Managing Computer Personnel*. John Wiley and Sons.

Creswell JW (1994) *Research Design : Qualitative And Quantitative Approaches*. Sage, Thousand Oaks.

Crosby PB (1979) *Quality Is Free*. McGraw-Hill, New York.

Crosby PB (1986) *Quality Without Tears*. McGraw-Hill.

Crosby PB (1996) *Philip Crosby's Reflections On Quality*. McGraw-Hill.

Curtis B, Hefley WE and Miller S (1995) *Maturity Model - People Capability Maturity Model*. SEI. CMU/SEI-95-MM-02.

Curtis B, Hefley WE and Miller S (2001) *People Capability Maturity Model - Version 2* in: http://www.sei.cmu.edu/cmm-p/version2/ 13/08/2001 2001

Curtis G (1989) *Business Information Systems: Analysis, Design and Practice*. Addison-Wesley Publishing Company, London.

Czaja R and Blair J (1996) *Designing Surveys: A Guide To Decisions And Procedures*. Pine Forge Press.

DeMarco T and Lister T (1987) *Peopleware - Productive Projects And Teams*. Dorset House.

Deming WE (1986) *Out Of Crisis*. MIT Press, Cambridge, Mass.

de-Souza DD (1998) Managers' Motivation For Using Information Technology. *Industrial Management + Data Systems* **98 (7)**:338-342.

Diaz M and Sligo J (1997) How Software Process Improvement Helped Motorola. *IEEE Software* **September/October**:75-81.

Dyba T (2000) An Instrument For Measuring The Key Factors Of Success In Software Process Improvement. *Empirical Software Engineering* **5 (4)**:357-390.

Edwards S and Browne M (1995) *Quality In Information Services*. Library And Information Science Research.

El Emam K and Briand L (1997) *Costs And Benefits Of Software Process Improvement*. Internatiol Software Engineering Research Network. ISERN-97-12.

El Emam K, Fusaro P and Smith B (1999) Success Factors And Barriers For Software Process Improvement, in *Better Software Practice for Business Benefit: Principles and Experience* (Messnarz R and Tully C eds) pp 355-371, IEEE Computer Society, Los Alamitos, CA.

El Emam K, Goldenson D, McCurley J and Herbsleb J (2001) Modelling The Likelihood Of Software Process Improvement: An Exploratory Study. *Empirical Software Engineering* **6 (3)**:207-229.

ESEPG (1999). Conference Proceedings. *Proceedings Of The Fourth Annual European Software Process Group Conference*. Amsterdam, Holland. June 6-10

Fenton N, Krause P and Neil M (2001) *A Probabilistic Model for Software Defect Prediction* in: http://www.dcs.qmul.ac.uk/~norman/papers/fenton_krause_neil_IEEE.pdf October 2001

Fenton N, Pfleeger SL and Glass RL (1994) Science And Substance: A Challenge To Software Engineers. *IEEE* **4 (July)**:86-95.

Ferguson P, Leman G, Perini P, Renner S and Seshagiri G (1999) *Software Process Improvement Works!* SEI. CMU/SEI-99-TR-027.

Fitz-Enz J (1978) Who Is The DP Professional? *Datamation* **September**:125-128.

Fitzgerald B and O'Kane T (1999) A Longitudinal Study Of Software Process Improvement. *IEEE Software* **May/June**:37-45.

Fowler P, Middlecoat B and Yo S (1999) *Lessons Learned Collaborating On A Process For SPI At Xerox*. SEI. CMU/SEI-99-TR-006.

Fox C and Frakes W (1997) The Quality Approach: Is It Delivering? *Communications of the ACM* **40 (6)**:25-29.

Fransella F and Bannister D eds (1977) *A Manual For Repertory Grid Technique*. London, Academic Press.

Gilgun JF (1992) Definitions, Methodologies And Methods In Qualitative Family Research, in *Qualitative Methods In Family Research*, Sage, Thousand Oaks.

Glaser BG and Strauss AL (1967) *The Discovery Of Grounded Theory: Strategies For Qualitative Research*. Aldine de Gruyter.

Glass RL (1998) Head To Head: Defining Quality Intuitively. *IEEE Software* **May/June**:103-104,107.

Glass RL (1999) The Reality Of Software Technology Payoffs. *Communications Of The ACM* **42 (2)**:74-99.

Goldenson DR and Herbsleb JD (1995) *After The Appraisal: A Systematic Survey Of Process Improvement, Its Benefits And Factors That Influence Success*. Software Engineering Institute. CMU/SEI-95-TR-009 ADA302225. http://www.sei.cmu.edu/publications/documents/95.reports/95.tr.009.html

Gray EM and Smith WL (1998) On The Limitation Of Software Process Assessment And The Recognition Of A Required Re-orientation For Global Process Movement. *Software Quality Journal* **7**:21-34.

Greene JC, Caracelli VJ and Graham WF (1989) Towards A Conceptual Framework For Mixed Method Evaluation Designs. *Education Evaluation And Policy Analysis* **11** (**3**):255-274.

Guttman L (1968) A General Nonmetric Technique for Finding the Smallest Coordinate Space for a Configuration of Points. *Psychometrika* **33**:469-506.

Guttman R and Greenbaum CW (1998) Facet Theory: It's Development And Current Status. *European Psychologist* **3 (1)**.

Hackman JR and Oldman GR (1976) *Motivation through the design of work : test of a theory*. Academic Press, New York.

Haley JH (1996) Software Process Improvement At Raytheon. *IEEE Software* **November**:33-41.

Hall T (1995) What Do Developers Really Think About Software Quality, in *Quality Management III* (Ross MM, Brebbia CA, Staples G and Stapleton J eds) pp 359-368, Computational Mechanics Publications.

Hall T and Fenton N (1997) Implementing Effective Software Metrics Programmes. *IEEE Software* **March**.

Hall T and Wilson D (1997) Views Of Software Quality: A Field Report. *IEE Procs On Software Engineering* **114 (2)**:111-118.

Hammersley M and Atkinson P (1983) *Ethnography: Principles In Practice*. Tavistock Publications, New York.

Hammock N (1999). Critical Success Factors In Software Process Improvement. *Proceedings Of The Fourth Annual European Process Group Conference*. Amsterdam, Holland. C304a. June 6-10

Hantos P and Gisbert M (2000) Identifying Software Productivity Improvement Approaches And Risks: Construction Industry Case Study. *IEEE Software* **17** (**1**):48-56.

Harrison R, Baddoo N, Barry E, Biffl S, Parra A, Winter B and Wuest J (1999) Directions and Methodologies for Empirical Software Engineering Research. *Empirical Software Engineering* **4 (4)**:405-410.

Harrison R and Wells MJ (1999). Multidisciplinary And Interdisciplinary Research Methods And Their Impact On Software Systems Design. *Workshop On Empirical Studies Of Software Maintenance*. Oxford, UK.

Heather P and Stone S (1984) *CRUS Guide: Questionnaires*. University Of Sheffield.

Herbsleb JD, Carleton A, Rozum J, Siegel J and Zubrow D (1994) *Benefits Of CMM-Based Software Process Improvement: Initial Results*. SEI Technical Report. CMU/SEI-94-TR-013.

Herbsleb JD and Goldenson DR (1996). A Systematic Survey Of CMM Experience And Results. *18th International Conference on Software Engineering - ICSE*. Berlin, Germany. 323-330. 25-29 March

Herzberg F (1987) One More Time: How Do You Motivate Employees? *Harvard Business Review* **September-October**:109-120.

Herzberg F, Mausner B and Snyderman BB (1959) *The Motivation To Work*. John Wiley & Sons Inc, New York.

Hetzel B (1995) The Sorry State Of Software Practice, Measurement And Evaluation, in *Software Quality Assurance And Metrics. A Worldwide Perspective* (Fenton N, Whitty R and Iizuka Y eds), Thompson Computer Press.

Hovenden FM, Sharp HC, Walker SD and Woodman M (1996) Building Quality Into Scientific Software. *Software Quality Journal* **March (5)**:25-32.

Humphrey WS (1989) *Managing The Software Process*. Addison Wesley.

Humphrey WS (1995) *A Discipline For Software Engineering*. Addison-Wesley.

Humphrey WS (1998) Why Don't They Practice What We Preach? *Annals Of Software Engineering* **6 (4)**:201-222.

Humphrey WS (1999a) *Introduction To The Personal Software Process*. Addison Wesley.

Humphrey WS (1999b). Putting Engineering Into Software. *Proceedings Of The Fourth Annual European Process Group Conference*. Amsterdam. C403. June 6-10

Ince D (1994) *ISO 9001 And Software Quality Assurance*. McGraw-Hill.

ISO (1991). International Standard ISO/IEC 9126. Information Technology -- Software Product Evaluation -- Quality Characteristics And Guidelines For Their Use. Geneva.

ISO (1999) *Software Process Assessment - Part 5: An Assessment Model And Indicator Guidance*. International Standards Organisation.

Iversen GR and Gergen M (1997) *Statistics: The Conceptual Approach*. Springer-Verlag, New York.

Juran J (1988) *Quality Control Handbook*. McGraw-Hill, New York.

Kaltio T and Kinnula A (2000) Deploying The Defined Software Process. *Software Process - Improvement and Practice* **5**:65-83.

Kelly GA (1970) Behaviour Is An Experiment, in *Perspectives In Personal Construct Theory* (Bannister D ed), Academic Press, London and New York.

Khalil OEM, Zawacki RA, Zawacki PA and Selim A (1997). What motivates Egyptian IS Managers And Personnel: Some Preliminary Results. *SIGCPR 97*.

Krasner H (1997) *Accumulating The Body Of Evidence For The Payoff Of Software Process Improvement -1997* in: http://www.utexas.edu/coe/sqi/archive/krasner/spi.pdf May 1999

Krasner H (1999). Using The Cost Of Quality Approach For Software. *Proceedings Of The 12th International Software Quality Week Conference*. San Jose, California. 1132-1154. May 24-28

Krasner H, Pyles J and Wohlwend H (1994) *A Case History Of The Space Shuttle Onboard Systems Project*. SEMATECH. 94092551A-TR.

Krippendorf K (1980) *Content Analysis: An Introduction To Its Methodology*. SAGE Publications Ltd, London.

Kuilboer JP and Ashrafi N (2000) Software Process And Product Improvement: An Empirical Assessment. *Information And Software Technology* **42 (2000)**:27-34.

Landis J and Koch G (1977) Measurement Of Observer Agreement For Categorical Data. *Biometrics* **33**:159-174.

Laporte CY and Trudel S (1998) Addressing The People Issues Of Process Improvement Activities At Oerlikon Aerospace. *Software Process - Improvement And Practice* **4**:187-198.

Lehman M, Kahen G and Ramil JF (2000). Role And Impact Of Feedback And System Dynamics In Software Evolution Processes And Their Improvement. *EuroSPI 2000*. Denmark, Copenhagen. 7-9 Nov

Lehman MM and Belady LA (1976) A model of a large system development. *IBM Systems Journal* **15 (3)**:225-252.

Leung HK (1999) Slow Change Of Information System Development Practice. *Software Quality Journal* **8 (3)**:197-210.

Likert R (1932) A Technique For The Measurement Of Attitudes. *Archives Of Psychology* **22 (140)**.

Lizotte R (1994) NQA And ISO9000. *Software Quality Management* **Autumn (23)**:29-33.

Maiden NAM and Rugg G (1996) Selecting Methods For Requirements Acquisition. *Software Engineering Journal* **11 (3)**:183-192.

Maslow A (1954) *Motivation And Personality*. Harper & Row, New York.

Mata Toledo RA and Unger EA (1985) Another Look At Motivating Data Processing Professionals. *Computer-Personnel* **10 (1)**:1-7.

McCalman J and Paton RA (1992) *Change Management: A Guide To Effective Implementation*. Paul Chapman Publishing, London.

McDermid JA and Bennett KH (1999) Software Engineering Research: A Critical Appraisal. *IEE Procs On Software Engineering* **146 (4)**:179-186.

McGuinness E (1999). Climbing The CMM: How Long Can You Tread Water - Do We Need An Evolution Or A Revolution? *Proceedings Of The Fourth Annual European Software Engineering Process Group Conference*. Amsterdam, Holland. C408c. June 6-10

Mellis W (1998) Software Quality Managment In Turbulent Times - Are There Alternatives To Process Oriented Software Quality Management? *Software Quality Journal* **7**:277-295.

Moitra D (1998) Managing Change For Software Process Improvement Initiatives: A Practical Experience-Based Approach. *Software Process - Improvement And Practice* **4**:199-207.

Morgan DL (1997) *Focus Groups As Qualitative Research*. Sage Publications.

Neff BE (1987) *The Impact Of Prescriptive Planning Models On Preservice English Teachers' Thought And On The Classroom Environements They Create*. Doctoral Dissertation. Virginia Polytechnic Institute And State University. Blacksburg

Nicholson I, Holmes D, Richards J and Warden R (1995) Motivation Of Quality Practitioners, in *Quality Management III* (Ross MM, Brebbia CA, Staples G and Stapleton J eds) pp 347-357, Computational Mechanics Publications.

Nolan AJ (1999) Learning From Success. *IEEE Software* **16 (1)**:97-105.

PAC (1999) *ANNEX A: Information Technology Projects Examined By The Committee Of Public Accounts And The Comptroller And Auditor General* in: 24/10/2000 2000

Pandit NR (1996) The Creation of Theory: A Recent Application of the Grounded Theory Method. *The Qualitative Report* **2 (4)**.

Paulk M (1999). Practices Of A High Maturity Organisation. *Proceedings Of The Fourth Annual European Software Engineering Process Group Conference*. Amsterdam, Holland. C405c. June 6-10

Paulk M, Chrissis M, Curtis B and Weber C (1994a) *The Capability Maturity Model: Guidelines For Improving The Software Process*. Addison-Wesley.

Paulk M, Weber C, Curtis B and Chrissis M (1994b) A High Maturity Example: Space Shuttle Onboard Software, in *The Capability Maturity Model: Guidelines For Improving Software Process*, Addison-Wesley.

Paulk MC, Goldenson D and White DM (2000) *The 1999 Survey of High Maturity Organizations*. Software Engineering Institute, Carnegie Mellon University. CMU-SEI-2000-SR-002.

Pitterman B (2000) Telcordia Technologies: The Journey To High Maturity. *IEEE Software* **17 (4)**:89-96.

Reiblein S and Symons A (1997) SPI: "I Can't Get No Satisfaction" - Directing Process Improvement To Meet Business Needs. *Software Quality Journal* **6 (2)**:89-98.

Rifkin S (2001) Why Software Process Innovations Are Not Adopted. *IEEE Software* **18 (4)**:112-111.

Rokeach M (1968) *Beliefs Attitudes And Values: A Theory Of Organisation And Change*. Jossey-Bass, London.

Rowntree D (1981) *Statistics Without Tears: A Primer For Non Mathematecians*. Penguin Books, London.

Seaman CB (1999) Qualitative Methods In Empirical Studies Of Software Engineering. *IEEE Transactions on Software Engineering* **25 (4)**:557-572.

Shepperd M, Cartwright M and Kadoda G (2000) On Building Prediction Systems For Software Engineers. *Empirical Software Engineering* **5 (3)**:175-182.

Shye S (1992) Faceted small space analysis. DOS computer program version, in, Israel Institute for Applied Social Research, Jerusalem.

Shye S, Elizur D and Hoffman M (1994) *Introduction to facet theory*. Sage. Thousand Oaks, CA.

Silverman D (2000) *Doing Qualitative Research: A Practical Handbook.* Sage Publications.

Skinner BF (1976) *Walden Two*. Macmillan, New York.

SPSS Inc. (1999) SPSS For Windows: System Reference Guide, in, Chicago.

Stelzer D and Mellis W (1998) Success Factors Of Organisational Change In Software Process Improvement. *Software Process - Improvement and Practice* **4 (4)**:227-250.

Stewart V, Stewart A and Fonda N (1981) *Business Applications Of Repertory Grid.* McGraw-Hill.

Taylor SJ and Bogdan R (1984) *Introduction To Qualitative Research Methods*. John Wiley and Sons, New York.

Tesch R (1990) *Qualitative Research: Analysis Types And Software Tools*. Falmer, New York.

Theodore D (1999) *Offender Profiles: Do Their Contents Equate To Cold Reading? Potential Issues Regarding The Interpretation Of Profiles.* M.Sc Dissertation. Department Of Psychology. The University Of Liverpool. Liverpool

Thompson HE and Mayhew P (1997) Approaches To Software Process Improvement. *Software Process - Improvement and Practice* **3**:3-17.

VASIE (1999) *Application Of The ASCET Method, The Business Benefits Of Software Best Practice: Case Studies* in: http://www.esi.es/ESSI/Reports/All/24151/Download.html#48 August 1999

Warden R and Nicholson I (1995) IT Quality Initiatives At Risk. *Software Quality Management* **New Year (24)**:24-27.

Weigers KE (1998) Head To Head: Read My Lips: No New Models! *IEEE Software* **September/October (5)**:10-13.

Wernham B (1992) Letter From America. *Software Quality Management* **Spring (14)**:20-21.

Willis RR, Rova RM, Scott MD, Johnson MI, Ryskowski JF, Moon JA, Shumate KC and Winfield TO (1998) *Hughes Aircraft's Widespread Deployment OF A*

Continuously Improving Software Process. Software Engineering Institute. Canergie Mellon University. CMU/SEI-98-TR-006.

Wilson D, Hall T and Baddoo N (2001) A Framework For Evaluation And Prediction Of Software Process Improvement Success. *Journal of Systems and Software* **59** (**2**):135-142.

Zeibe T (1996) *Software Process Improvment (SPI) Guidelines For Improving Software Release 5:0*. SEMATECH. http://www.sematech.org/public/docubase/document/3188aeng.pdf

Appendices

Appendix A: Guidelines for improving practitioners' support for SPI extracted from the literature

Code	Literature guidelines for increasing software practitioner's support for SPI in companies	Case studies / Technical reports /Journal accounts of SPI in companies																						
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**
I	**Senior management commitment**				√	√	√	√		√	√	√	√					√	√			√			
II	**Secure practitioner's buy-in**				√		√				√											√			
III	**Process ownership**	√	√	√		√		√							√										
IV	**Evidence / Visible benefits**	√	√	√	√	√		√		√			√					√	√						
V	**Train practitioners**	√	√	√						√		√		√	√	√	√	√	√	√			√		
VI	**Mentor practitioners**	√	√	√						√													√		√
VII	**Standardise.**											√													
VIII	**Sharing best practice**				√					√	√										√				
IX	**Proper prioritising of SPI**				√																				
X	**Dedicated resources to SPI**				√		√																		√
XI	**Setting relevant / realistic objectives**										√	√		√										√	√
XII	**Internal leadership that provides vision**				√	√	√		√		√							√	√						
XIII	**Overcoming internal resistance to SPI.**	√	√	√			√		√		√														
XIV	**Reward SPI effort**						√																		√
XV	**SPI forum**									√															
XVI	**Well respected SPI people**																								√
XVII	**Run SPI from within**					√	√	√																	

Key

No.	**Case studies / Technical reports / Journals etc**
1	A High Maturity Example: Space Shuttle Onboard Software [Paulk et al., 1994b]
2	Journey To A Mature Software Process [Billings et al., 1994]
3	A Case History Of The Space Shuttle Onboard Systems Project [Krasner et al., 1994]
4	Benefits Of CMM-Based Software Process Improvement: Initial Results [Herbsleb et al., 1994]
5	Software process improvement at Raytheon [Haley, 1996]
6	How Software Process Improvement helped Motorola [Diaz and Sligo, 1997]
7	Accumulating the body of evidence for the payoff of Software Process Improvement [Krasner, 1997]
8	Managing Change For Software Process Improvement Initiatives: A Practical Experience-Based Approach [Moitra, 1998]
9	Hughes Aircraft's widespread deployment of a continuously improving software process [Willis et al., 1998]
10	Success factors of organisational change in Software Process Improvement [Stelzer and Mellis, 1998]
11	Addressing the people issues of process improvement at Oerlikon Aerospace [Laporte and Trudel, 1998]
12	Application Of The ASCET Method, The Business Benefits Of Software Best Practice [VASIE, 1999]
13	AIS: Software Process Improvement works! [Ferguson et al., 1999]
14	Motorola Cork: A longitudinal study of Software Process Improvement [Fitzgerald and O'Kane, 1999]
15	Lessons learned collaborating on a process for SPI at Xerox [Fowler et al., 1999]
16	Using the Cost of Quality approach for software [Krasner, 1999]
17	People improvement in a rapidly changing business and technical environment [Ahuja, 1999]
18	Telcordia Technologies: The journey to a high maturity [Pitterman, 2000]
19	Learning From Success [Nolan, 1999]
20	Identifying Software Productivity Improvement Approaches And Risks: Construction Industry Case Study [Hantos and Gisbert, 2000]
21	An Instrument For Measuring The Key Factors OF Success In SPI [Dyba, 2000]
22	Practices of High Maturity Organisations [Paulk, 1999]
23	Why Software Process Innovations Are Not Adopted [Rifkin, 2001]
24	After The Appraisal: A Systematic Survey Of Process Improvement, Its Benefits And Factors That Influence Success [Goldenson and Herbsleb, 1995]

Appendix B: Companies in focus group and RGT studies

Company number	HW/SW Producer	UK or Multi-national?	Size (people)	SE size (people)	Age (yrs)	SW type
A1	HW/SW	UK	100-500	<10	20-50	EM
A2	SW	MN	>2000	>2000	10-20	RT
A3a	HW/SW	MN	>2000	500-2000	>50	RT/EM
A3b	HW/SW	MN	>2000	500-2000	>50	RT/EM
A4	SW	MN	>2000	>2000	10-20	RT
A5	SW	MN	>2000	>2000	20-50	Packs
A6	SW	UK	100	40	10-20	Bus
A7	HW/SW	MN	>2000	>2000	>50	RT/EM
A8	SW	MN	>2000	10-100	10-20	Sys/EM
A9	SW	UK	10-100	10-100	5-10	Bus
A10	SW	MN	10-100	10-100	10-20	RT/EM
A11	HW/SW	MN	500-2000	11-25	20-50	RT/EM
A12	SW	UK	100-500	100-500	20-50	Bus

Key: RT = real time; EM = Embedded; Bus = Business systems; Packs = Packages; Sys = Systems software

Appendix C: Matrix of research findings by data sources, methods and types

Study description	Perception of SPI	Motivators for SPI	De-motivators for SPI	Perception of role in SPI
Data source	SPI managers	Developers/ project managers and senior managers	Developers/ project managers and senior managers	Developers/ project managers and senior managers
Data collection method	Survey	Focus groups	Focus groups	RGT
Type of data collected	Quantitative and qualitative (ordinal)	Qualitative text	Qualitative text	Qualitative text
Research findings				
Senior management commitment	√	√	√	
Grassroots practitioners' buy in	√	√	√	√
Process ownership		√		√
Evidence of SPI success		√	√	
Training		√	√	
Standardisation		√	√	
Communicating best practices		√	√	
Prioritisation		√		
Resources		√	√	
Relevant objectives			√	√
Internal resistance	√		√	
Rewards for SPI		√		
SPI forum		√		
Organic SPI initiatives		√	√	

Appendix D: Sample questionnaire - SPI managers' perception of SPI

Thank you for dedicating some time to completing this questionnaire. We guarantee that all the information given will be treated in the strictest confidence.

Your name: _____

Your job title: _____

E-mail address: _____

Company: _____

Address: _____

_____

_____

Date: _____

Section i : Personal Background

This section is concerned with information about your background and experience as a software process improvement manager. This information will be treated in the STRICTEST CONFIDENCE and any publication of this study will present information in aggregate form and such information will be anonymous and unattributable to individual organisations or individual respondents.

A.1 How long have you worked in this company? _____ years

A.2 How long have you worked in computing/software engineering/IT? _____ years

A.3 What is your educational background?
(You may tick more than one)

a. Bachelors degree [ ]
b. Masters degree [ ]
c. Doctorate degree [ ]
d. None of the above [ ]
e. Other [ ]

Please specify _____

A.4 Are any of your educational qualifications in the following areas?

a. Computer science [ ]
b. Software engineering [ ]
c. Information Systems [ ]
d. Information Technology [ ]
e. Other [ ]

Please specify _____

A.5 Of which of the following professional bodies are you a member?
(You may tick more than one)

a. British Computer Society (BCS) [ ]
b. Institute of Electronic Engineers (IEE) [ ]
c. Institute of Electronic and Electrical Engineers (IEEE) [ ]
d. Association for Computer Machinery (ACM) [ ]
e. Institute for the Management of Information Systems [ ]
f. Association of Chartered Engineers [ ]
g. None of the above [ ]
e. Other [ ]

Please specify _____

Section ii: Personal Opinions

This section is concerned with information about your opinions as a software process improvement manager. This information will be treated in the STRICTEST CONFIDENCE and any publication of this study will present information in aggregate form and such information will be anonymous and unattributable to individual organisations or individual respondents.

In response to the following statements, please indicate your level of agreement by circling the appropriate number on the scale: where 5 indicates a strong agreement with the statement made and 1 indicates a strong disagreement.

i.e. 5 = Strongly agree
4 = Agree
3 = Neutral
2 = Disagree
1 = Strongly disagree

B.1 I am familiar with implementing Software Process Improvement initiatives.

Agree 5 ---------- 4 ---------- 3 ---------- 2 ---------- 1 Disagree

B.2 My experience of Software Process Improvement has been positive.

Agree 5 ---------- 4 ---------- 3 ---------- 2 ---------- 1 Disagree

B.3 Software Process Improvement is an effective approach to improving the quality of the software product.

Agree 5 ---------- 4 ---------- 3 ---------- 2 ---------- 1 Disagree

B.4 Software Process Improvement should be implemented via a Top Down approach.

Agree 5 ---------- 4 ---------- 3 ---------- 2 ---------- 1 Disagree

B.5 Software Process Improvement should be implemented via a Bottom Up approach.

Agree 5 ---------- 4 ---------- 3 ---------- 2 ---------- 1 Disagree

B.6 In the long term, the cost of setting up an SPI programme is compensated for by the cost savings made elsewhere in the development process.

Agree 5 ---------- 4 ---------- 3 ---------- 2 ---------- 1 Disagree

B.7 Senior managers are very supportive of SPI.

Agree 5 ---------- 4 ---------- 3 ---------- 2 ---------- 1 Disagree

B.8 Software developers are enthusiastic about SPI.

Agree 5 ---------- 4 ---------- 3 ---------- 2 ---------- 1 Disagree

B.9 Implementing an effective SPI programme is difficult.

Agree 5 ---------- 4 ---------- 3 ---------- 2 ---------- 1 Disagree

Appendix E: Inter-rater reliability test - Software practitioners' motivators and de-motivators for SPI

	Group A		Group B		Group C		Group D			Group E		Group F		Group G		Group H	
	R1	R2	R1	R2	R1	R2	R1	R2		R1	R2	R1	R2	R1	R2	R1	R2
De-motivator									**Motivators**								
Budget constraints	1	1					1	1, 1	Automation								
Commercial pressures			1	1		1	1		Autonomy								
Cumbersome processes						1		1, 1	Bottom-up initiatives	1		1					
Customers				1					Career prospects								
Fire fighting									Communication			1	1	1			
Imposition		1			1	1		1	Compulsory								
Inadequate communication		1					1	1	Cost beneficial							1	
Inadequate metrics									Critical mass	1	1						
Inertia			1		1	1	1	1, 1	Easy processes					1			
Inexperienced staff	1	1, 1							Eliminates bureaucracy								
Irrelevant objectives/deliverables									Empowerment								
Isolated best practice									External audits					1	1		
Lack of evidence of direct benefits									Feedback								
Lack of feedback		1							Job satisfaction	1							
Lack of mgt direction/commitment							1	1, 1	Justifiable benefits							1	
Lack of overall support	1								Knowledgeable team leaders								
Lack of resources	1			1				1	Maintainable processes								
Lack of SPI management skills									Meeting targets							1	1
Lack of standards(different platforms)									Phased introduction								
Large-scale programmes			1			1			Process ownership	1							
Low process priority			1	1, 1				1	Reduced admin								
Negative / Bad experience	1			1		1, 1			Resources			1					
Organisational changes									Reward schemes			1				1	1
Personality clashes		1					1		Saleability								
PM's lack of technical knowledge								1	Shared best practice			1					
Reduced creativity									SPI forum			1					
Staff turnover									Standardisation								
Time pressure/constraints	1	1	1						Taller hierarchy								
Workload				1					Task forces								
									Top-down commitment			1	1				
									Training			1					
									Visible success	1	1			1		1	
None of the above				1,1					None of the above						1		

NB: Even though rater 2 may see more than one instance of the same de-motivator in a transcript, it does not skew the results of the agreement frequencies. This is because when a motivator or de-motivator is mentioned in a focus group session, it is recorded once irrespective of how many times it is recorded or how many people within the focus group mention it. For the process of testing the sets of categories, it shows that both sets of raters have correctly identified a particular category in the same transcript.

Appendix F: Demonstrating the calculation of Chi Squared

Table I shows two types of motivators identified by three groups of software practitioners.

Actual values	Developers	Project managers	Senior managers	Total
SPI specific motivators	58	29	12	99
Organisational motivators	5	8	8	21
Total	63	37	20	120

Table I: Table of actual values

A typical chi squared test determines whether the differences in the way different practitioner groups have cited motivators can tell whether the are likely (95% confidence) or highly likely (99% confidence) to have come from the same population.

To do this, an expected table of the above observations, based upon staff group would be constructed. The expected value will record the SPI specific and organisational motivators based upon the percentage of developers, project managers and senior managers. So for example, 99 SPI specific motivators will be 'apportioned' in the following ratio: 0.525 X 99 for developers, 0.308 X 99 for project managers and 0.167 X 99 for senior managers. Table II shows a table of expected values.

Expected values	Developers	Project managers	Senior managers	Total
SPI specific motivators	51.955	30.525	16.5	99
Organisational motivators	11.025	6.475	3.5	21
Total	63	37	20	120

Table II: Table of expected values

The next step in the analysis is to compare the aggregate difference between expected values and the actual values to ascertain whether the sample of three staff groups are likely or highly likely to have come from the same population.

Chi squared statistic is calculated using SPSS [1999] statistical package. If Chi squared falls below a certain figure, then it is not possible that the samples have come from the same population. However if Chi squared falls above a certain figure, then it implies that the differences are significant enough to imply that the samples have come from different populations. The value for this Chi squared benchmark - known as Chi critical - is based upon the degrees of freedom (df). DF is calculated by (number of rows -1)(number of columns-1). In this example, df = (2-1)(3-1) = 2.

In the above example, Chi squared calculated from [Laporte and Trudel, 1998] is 11.439 See Table III.

	Value	df	Asymp. Sig. (2-sided)
Pearson Chi-Square	11.439	2	.003
Likelihood Ratio	10.811	2	.004
Linear-by-Linear Association	11.257	1	.001
N of Valid Cases	120		

Table **III:** Chi-square tests

Chi critical for df = 2 at 99% confidence is 9.21 [SPSS Inc., 1999]. It can therefore be said that it is highly likely (99%) that the three staff groups do not come from the same population. That the differences represented in their choice of motivators is real and can only occur by chance in 3 out of 1000 chances (sig 0.003).

Appendix G: Sub-Sample percentages based on reported familiarity with SPI

Questionnaire statements	Whole Sample (%) Responses					Familiar Managers (%) Responses					Unfamiliar Managers (%) Responses				
	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
B1: "I am familiar with implementing SPI initiatives"	4	14	20	36	26	0	0	0	58	42	21	79	0	0	0
B2: "My experience of SPI has been positive"	1	6	28	49	16	0	4	14	60	22	7	7	72	14	0
B3: "SPI improves software quality"	1	5	11	46	36	0	6	4	44	46	7	0	29	43	21
B4: "SPI should be implemented via a Top Down approach"	11	15	30	29	15	14	16	22	28	20	14	14	43	22	7
B5: "SPI should be implemented via a Bottom Up approach"	16	28	35	18	4	16	26	34	18	6	21	36	36	7	0
B6: "In the long term, SPI is cost beneficial"	3	4	20	48	26	0	2	16	50	32	14	0	29	43	14
B7: "Senior management are very supportive of SPI"	5	16	33	34	13	2	14	36	32	16	14	14	36	29	7
B8: "Software developers are enthusiastic about SPI"	5	19	40	28	9	2	20	38	30	10	14	14	43	29	0
B9: "Implementing an effective SPI programme is difficult"	3	5	15	49	29	0	4	16	46	34	14	0	14	65	7

NB: Percentage figures have been rounded up for presentation purposes

Appendix H: Definition of motivators (Content Category Dictionary)

Plot codes			Motivator	Definition
D18			Automation	Tools to eliminate paper work.
	P8		Autonomy	Enables practitioners to carry on present roles without prescribing specific roles for them.
D3			Bottom-up initiatives	Developers have input into the design and planning of SPI
		S8	Career prospects	Improves career prospects.
D16	P7		Communication	Improved communication about SPI
D7			Compulsory	SPI practice is made mandatory
		S3	Cost beneficial	Favourable Cost/Revenue ratio through SPI
D10			Critical mass	The presence of sufficient number of people who want to see SPI happen
	P6		Easy processes	Processes that are easy to understand and follow.
D12			Eliminates bureaucracy	Eliminates spending time on bureaucratic processes.
	P4		Empowerment	Practices within the SPI programme that empower staff to take decisions on changing processes.
	P5		External audits	Stipulation by some external body to maintain SPI practices. For example, certification bodies.
D2		S11	Feedback	Feedback, both from management and from customers.
D11			Job satisfaction	Practitioners get job satisfaction from producing good quality process and high quality products.
		S1	Justifiable benefits	The ability to justify the long-term benefits of SPI.
	P10		Knowledgeable team leaders	Having team leaders who know about software engineering. i.e. posses technical backgrounds.
D19	P2	S12	Maintainable processes	Processes that are changeable and maintainable.
		S5	Meeting targets	SPI practice doesn't deviate the company from meeting commercial and project goals.
D9			Phased introduction	SPI is introduced through small and incremental implementation.
D13	P12	S9	Process ownership	Practitioners own and therefore are able to change processes
	P3		Reduced admin	SPI leads to reduced administration.
D17	P1	S10	Resources	Sufficient time and resources allocated to SPI
D14	P11	S4	Reward schemes	Practitioners are rewarded for SPI work.
	P14		Saleability	The perception that SPI will lead to more saleable job market skills.
D5			Shared best practice	Best practice is shared across teams and departments in companies.
D15			SPI forum	Creating a forum where SPI ideas can be discussed.
D8			Standardisation	SPI makes practitioners work in a standardised way.
		S6	Taller hierarchy	Taller company hierarchies which create more opportunity for promotion.
		S7	Task forces	Using task forces to drive improvement.
D4	P13		Top-down commitment	Visible senior management support for SPI
D6			Training	Training provided to practitioners in SPI practices.
D1	P9	S2	Visible success	Evidence of the benefits of SPI.

Appendix I: Data matrices for developers, project managers and senior managers' motivators

Company	Group	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13	D14	D15	D16	D17	D18	D19
A1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A2	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0
A3a	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	0	0
	2	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A3b	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A4	1	1	0	0	0	0	0	1	0	0	0	0	0	1	1	0	0	0	0	0
A5	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A6	1	1	1	1	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0	0
	2	0	0	1	1	1	1	0	0	0	0	0	0	0	1	1	1	1	0	0
A7	1	1	1	1	0	1	1	0	0	0	0	0	1	0	0	1	0	0	0	0
	2	0	1	0	1	0	0	0	0	0	0	0	0	0	1	0	1	1	1	0
A8	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	2	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A9	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A10	1	1	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0
	2	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A11	1	1	0	0	0	1	0	0	0	0	0	1	0	0	0	0	1	0	0	1
A12	1	0	1	1	1	1	0	0	0	1	0	0	0	1	0	1	1	1	0	0
	2	1	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0

Data matrix of developers' motivators

Company	Group	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14
A1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A2	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0
A3a	1	1	0	0	0	0	0	0	0	0	1	0	1	0	0
	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A3b	1	0	0	1	1	0	0	0	0	0	0	1	1	0	0
	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A4	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A5	1	0	0	0	0	1	1	1	0	1	0	0	0	0	0
A6	1	0	1	0	1	0	1	0	0	0	0	0	0	0	0
A7	1	1	0	0	1	0	0	1	0	0	0	0	1	0	0
	2	1	1	0	0	0	0	0	0	0	0	1	0	0	0
A8	1	0	0	0	0	0	0	0	1	1	0	1	0	1	0
	2	0	1	0	0	0	1	0	0	0	0	0	0	0	1
A9	1	1	0	0	0	0	0	0	0	1	1	0	0	0	0
A11	1	0	0	0	0	0	0	0	0	1	1	0	1	0	0
A12	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0

Data matrix of project managers' motivators

Company	Group	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12
A1	1	0	0	0	0	0	0	0	0	0	0	0	0
A2	1	0	0	0	0	0	0	0	0	0	0	0	0
A3a	1	0	0	0	0	0	0	0	0	0	0	0	0
A3b	1	0	0	0	0	0	0	0	0	0	0	0	0
A4	1	0	0	0	0	1	0	0	0	0	1	0	0
A5	1	0	1	0	0	0	0	1	0	0	0	0	0
A6	1	0	0	0	0	0	0	0	0	0	0	0	0
A7	1	1	1	1	1	1	0	0	0	0	0	0	0
A8	1	0	1	0	0	0	1	0	1	1	0	0	0
A9	1	0	0	1	0	0	0	0	0	0	1	0	0
A10	1	0	0	0	0	1	0	0	0	1	0	1	1
A12	1	0	0	0	1	0	0	0	0	0	0	0	0

Data matrix of senior managers' motivators

Appendix J: Definition of de-motivators (Content Category Dictionary)

Plot codes			De-motivator	Definition
D1		S2	Budget constraints	Budgets do not allocate resources specifically to SPI, therefore SPI work becomes a drain on overall budget
D2	P1	S3	Commercial pressures	Pressure to satisfy commercial/financial objectives of company
D3	P2	S4	Cumbersome processes	Processes that are bureaucratic and difficult to implement
D4			Customers	Direct interference from customers
	P3		Fire fighting	A policy of tackling problems as they occur as opposed to a proactive long term strategy for tackling problems
D5	P4		Imposition	Imposing SPI as a dictate, without prior consultation with practitioners
D6	P5	S5	Inadequate communication	Lack of communication between different levels in a company and between different functional areas
	P6		Inadequate metrics	Not collecting sufficient metrics to guard improvement
D7	P7	S6	Inertia	Resistance to practices new
D8		S7	Inexperienced staff	New and temporary staff who are not sufficiently knowledgeable of company processes
	P8		Irrelevant objectives/deliverables	SPI objectives are not tailored to "real" needs that practitioners can identify with
D9			Isolated best practice	Best practices are kept within departments/teams/groups and not shared within the company
	P9	S8	Lack of evidence of direct benefits	Practitioners do not have or are not provided evidence of the success of SPI.
D10			Lack of feedback	Practitioners are not given feedback of the SPI outcomes, or of contributions they make towards SPI
D11			Lack of mgt direction/commitment	Senior management do not demonstrate understanding nor commitment to SPI
D12	P10	S9	Lack of overall support	SPI is not overwhelming supported by the practitioner involved in it. There is apathy amongst certain groups
	P11	S10	Lack of resources	The company does not have the resources -staff, time, tools - to properly fund SPI. This is dissimilar to *Budgets.*
		S11	Lack of SPI management skills	There are insufficient personnel with the appropriate skills to drive (manage) SPI in the company
D13			Lack of standards (different platforms)	The software development function operates across different platforms. There are no overall standards to SW development
	P12		Large-scale programmes	The SPI initiative is too big for the company. To many facets going on at the same time. It creates co-ordination problems
	P13		Low process priority	SPI is given low priority with respect to other project activities
D14	P14	S1	Negative / Bad experience	Previous negative experiences of SPI negates against SPI uptake amongst practitioners
		S12	Organisational changes	Organisational changes that imply re-allocation of staff and responsibilities impact negatively on ongoing SPI programmes
D15		S13	Personality clashes	Individuals and personal politics frustrates the SPI effort
D16			PM's lack of technical knowledge	Project managers do not possess technical knowledge of SW production hence are unable to appreciate the merits of SPI
D17			Reduced creativity	Practitioners perception of SPI procedures is that it takes away their individual creativity and flair
	P15		Staff turnover	High staff turnover frustrates the nurturing of SPI culture. Ever having to teach new people the established processes
D18	P16	S14	Time pressure/constraints	Pressure to deliver product on time frustrates SPI initiative
D19			Workload	Practitioners have too much work, thus are unable to devote sufficient effort to SPI

Appendix K: Data matrices for developers, project managers and senior managers' de-motivators

Company	Group	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13	D14	D15	D16	D17	D18	D19
A1	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
A2	1	1	1	0	0	0	1	1	0	0	0	1	0	0	0	1	0	0	0	0
	1	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	1
A3a	2	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
	1	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	1	0
A3b	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
A4	1	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0
A5	1	0	1	0	0	0	0	1	1	1	0	1	0	1	0	1	0	0	1	0
	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
A6	2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
	1	0	0	0	0	0	0	1	0	0	1	0	0	0	1	0	0	0	0	0
A7	2	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
	1	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
A8	2	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
	1	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	1	1	0
A9	2	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0
	1	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0
A10	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
A11	1	1	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	1
	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
A12	2	1	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	0

Data matrix of developers' de-motivators

Company	Group	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	P16
A1	1	0	0	0	1	0	0	0	1	0	1	0	0	0	0	0	0
A2	1	1	1	0	0	0	0	1	0	1	0	0	0	0	0	1	1
	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
A3a	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1
A3b	2	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0
A4	1	0	0	0	0	0	0	1	0	1	0	0	0	1	0	0	0
A5	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A6	1	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0
	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
A7	2	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
	1	1	0	0	0	0	0	0	0	1	0	0	0	1	0	0	1
A8	2	1	0	0	0	0	0	1	0	0	0	0	1	1	0	0	1
A9	1	0	0	0	0	0	0	0	0	1	0	1	0	0	0	1	1
A11	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1
A12	1	1	0	0	0	1	1	0	0	1	1	1	0	1	0	0	1

Data matrix of project managers' de-motivators

Company	Group	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14
A1	1	0	0	0	0	0	1	0	0	0	0	0	0	1	0
A2	1	0	0	1	0	0	0	0	0	0	1	0	0	0	1
A3a	1	1	1	0	0	0	1	1	0	1	1	1	0	0	0
A3b	1	1	1	0	0	0	1	1	0	1	1	1	0	0	0
A4	1	0	0	0	0	1	0	0	0	1	1	0	1	0	1
A5	1	1	0	0	0	0	1	0	1	0	0	0	0	0	1
A6	1	0	0	0	0	0	1	0	0	0	1	1	0	0	1
A7	1	1	0	0	0	0	1	0	0	1	1	0	0	0	1
A8	1	0	0	1	0	0	0	0	0	1	0	1	0	0	0
A9	1	0	0	0	0	1	0	0	0	0	1	0	0	0	1
A10	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0
A12	1	0	1	1	0	0	0	1	0	1	1	0	0	0	1

Data matrix of senior managers' de-motivators

Appendix L: A list of bi-polar constructs, elicited from developers, describing the relationship of three staff groups to SPI.

	Rating = 1	Element			Rating = 0
No.	Constructs	D	P	S	Constructs
1	More practical	1	0	0	Less practical
2	Delegate work	0	1	1	Do not delegate work
3	More pragmatic expectations of SPI	1	0	0	Idealistic expectations of SPI
4	Project focus.: Deal with day to day project activities. Are task orientated. Share common project goals.	1	1	0	Lack of project focus: Do not deal with project activities. Not task orientated. Do not share common pjt. goals.
5	Define processes. Suggest processes	0	1	1	Do not define processes. Just follow the processes
6	Use processes	1	0	0	Scarcely use processes.
7	Focus on more technical aspects of SPI	1	0	0	Focus on cost and time
8	Have a wider perspective of SPI	0	1	1	Have a narrower perspective of SPI
9	Closer to senior management. Aware of overall company goals	0	0	1	Far from senior management. Unaware of overall company goals
10	Closer to actual production	1	1	0	Not close to actual processes
11	More interested in nature of problems	1	1	0	More interested in the effect of problems than the nature
12	Very enthusiastic about SPI (CMM)	1	0	1	Less enthusiastic. Most likely to "kick it into touch".
13	Technical background, Technically skilled Worried about detailed technical issues	1	1	0	Non technical background. Not technically skilled. Not worried about technical issues
14	Subject to day to day interruptions	1	1	0	Not subject to day to day interruptions
15	Communicate via reports and presentations	0	1	1	Do not communicate via reports and documentation
16	Posses a long term view of SPI	1	0	1	Posses a project term view
17	Posses overall view of projects	0	1	1	Possess a limited view
18	Very close to work environment and possess great knowledge of it	1	1	0	Not very close to the work environment
19	Good knowledge of software	1	1	0	Poor knowledge of software
20	Understand deadlines	0	1	0	Do not understand deadlines
21	A lot of customer interaction. Interface with customers	0	1	1	Little customer interaction. Isolated from customers.
22	Aware of business and strategic issues. Access to long term info	0	1	1	Less aware of business and strategic issues. No access to long term info
23	Higher accountability	0	1	1	Lower accountability
24	Higher salaries	0	1	1	Lower salaries
25	Knowledge of cross fertilisation	0	1	1	Do not hear about other projects
26	Closer domain knowledge	1	1	0	Knowledge of domain not close
27	Variable workload	1	1	0	Consistent workload
28	Function as part of a team. Visibly in the same team	1	1	0	Function in isolation. Not in any visible team with others
29	Realistic about problems	1	0	1	Not realistic about problems
30	Apply pressure to project managers	1	0	1	Do not apply project to PM's
31	Have authority	0	1	1	Do not have authority
32	Can see bigger picture	0	1	1	Do not see bigger picture
33	Provide driving/guiding force	0	1	1	Do not provide/guiding force
34	Short term objectives	1	1	0	Do not work with short term objectives
35	Grasp on reality	1	1	0	Lack a grasp on reality
36	Have long term roles (positions)	1	0	1	Have short term roles/positions
37	Consult on decisions affecting developers	0	1	1	Are not consulted on developer decisions
38	Do 'real' work	1	0	0	Do not do 'real' work
39	Blame developers for problems	0	1	1	Do not blame developers for problems
40	People management skills	0	0	1	Do not have people management skills

Appendix M: A list of bi-polar constructs, elicited from project managers, describing the relationship of three staff groups to SPI

	Rating = 1	Element			Rating = 0
No.	Constructs	D	P	S	Constructs
1	Want to use metrics to monitor developers	0	1	1	Do not want to use metrics to monitor developers
2	Want to measure improvement	0	1	1	Want to see improvement
3	Good understanding of day to day development issues	1	1	0	Poor understanding of day to day development issues
4	More in touch with processes	1	1	0	Less in touch with processes
5	Focus on big picture	0	1	1	Focus on specific deliverables
6	Complain to project managers	1	0	1	Do not complain to project managers
7	Short to medium term view / goals of SPI	1	1	0	Long term (strategic) view / goal of SPI
8	Perceive SPI in terms of product and project	1	1	0	Perceive SPI in an organisational context
9	Abdicate responsibility for SPI changes	1	0	1	Do not abdicate responsibility for SPI changes
10	Design, plan and model SPI programmes	0	1	1	Carry out the actual SPI work
11	Lead and enable processes	0	1	1	Follow and do processes
12	Share personal, long term aspirations for SPI	1	0	1	Do not have personal, nor long term aspirations for SPI
13	Posses a technical understanding of processes	1	0	0	Non-technical understanding of SPI
14	Possess a project aim of SPI	1	1	0	Possess a strategic aim of SPI
15	Have a feeling of owning the development processes	1	0	1	Do not feel they own the development processes
16	Made more accountable for software development	0	1	0	Are not made more accountable for SW development than PM's
17	Sell SPI	0	1	1	Buy SPI
18	Are more focused on key milestones	1	0	1	Less focused on key milestones, more project milestones.
19	Possess a better view of the whole project	0	1	0	Possess a poor view of overall project. More likely to focus on single issues
20	Work to schedules: to life cycles beginning to end	1	1	0	Do not work to schedules
21	More technically minded and technical background	1	1	0	Less technically minded, less technical background
22	Do appreciate financial imperatives of projects	0	1	1	Do not appreciate financial imperatives of projects
23	Motivations are mostly product orientated	1	0	0	Motivations are not solely product orientated
24	Posses more of a customer focus, hence commercial minded	0	1	1	Posses less of a customer focus, less commercial minded
25	Have more responsibilities	0	1	1	Have less responsibilities
26	Have people management responsibilities	0	1	1	Do not have people management responsibilities
27	More responsible for project failures	0	1	0	Less responsible for project failures
28	Empowered to make changes	0	0	1	Not empowered to make changes
29	Perceive that manager groups have a hidden agenda	1	0	0	Do not perceive hidden agenda
30	Deliver objectives	1	1	0	Set objectives
31	Clear role definition	1	0	1	Role not defined
32	Main interest is financial	0	0	1	Main interest is not financial

Appendix N: A list of bi-polar constructs, elicited from senior managers, describing the relationship of three staff groups to SPI.

	Rating = 1	Element			Rating = 0
No.	Constructs	D	P	S	Constructs
1	More focused on deliverables. Manage commitment to deliverables	0	1	1	Less focused on deliverables. Do work that leads to deliverables
2	Good appreciation of customer's perspective	0	1	1	Poor appreciation of customer's perspective
3	More internally focused	1	0	0	More externally focused
4	More aware of the financial costs of SPI	0	1	1	Less aware of the financial costs of SPI
5	Implement SPI	1	1	0	Sponsor SPI
6	Possess good visibility of projects	1	1	0	Possess poor visibility of projects
7	More aware of people issues (e.g. training development etc)	1	0	1	Less aware of people issues
8	Have responsibility for building environment for SPI	0	1	1	Do not have responsibility for building SPI environment
9	Responsible for promoting SPI to others in company	0	1	1	Are not responsible for promoting SPI
10	Generate ideas for SPI	1	0	0	Create environment for pushing forward generated ideas
11	Think it is up to Senior managers to make SPI work	1	1	0	Do not think that responsibility for SPI lies solely on SMs
12	Face pressure from both technical and management processes	0	1	0	Do not face pressure from both technical and management processes
13	Have access to resources	0	1	1	Do not have access to resources
14	Have power	0	1	1	Do not have the power
15	Technical background. Tend to do hands-on technical work	1	1	0	Do not necessarily come from technical backgrounds. Do not do hands-on technical work
16	Share same technology	1	1	0	Do not share same technology with other practitioner groups
17	Share common interest in project performance	1	1	0	Do not share similar interests in project performance
18	Have responsibility for the bottom line, for the margin	0	1	1	Do not have responsibility for the bottom line, margin
19	Have diverse set of responsibilities	0	1	1	Have limited set of responsibilities
20	Tend to manage a portfolio of things: have many objectives	0	0	1	Tend to have fewer objectives
21	'Hate' project managers	1	0	1	Do not 'hate' project managers
22	Have long term projects. Long term vision	1	0	1	Have short term projects. Short term vision
23	Have people management responsibilities	0	1	1	Do not have people management responsibilities
24	Possess business acumen and business responsibilities	0	1	1	Do not have business responsibilities
25	Look at the bigger picture	0	1	1	Do not look at the bigger picture
26	Less able to effect changes. Work within a framework	1	1	0	More able to effect changes. Do not work within framework
27	More accepting of the status quo	1	1	0	Less accepting of the status quo
28	More reluctant to change things	1	1	0	Less reluctant to change things
29	Possess a business view of company	0	1	1	Do not posses business view of company
30	Share common project goals	1	1	0	Do not share common project goals

Appendix O: This author's publication references

Conference papers:

Baddoo N, Hall T and Wilson D (2000). Implementing A People Focused SPI Programme. *Proceedings Of The 11th European Software Control And Metrics Conference and The Third SCOPE Conference On Software Product Quality*. Munich. 373-381. April 18-20

Hall T, Baddoo N and Wilson D (2000). Measurement in software process improvement programmes: an empirical study. *10th International Workshop, IWSM 2000*. Springer-Verlag, Berlin, Germany. 4th-6th October.

Hall T, Baddoo N, Wilson D and Rainer A (2000). Optimising Software Measurement Programmes Using Practitioner Input. *Australian Conference On Software Measurement*. Sydney, Australia. 1-3 November

Hall T, Wilson D and Baddoo N (2000). Towards Implementing Successful Software Inspections. *Proceedings International Conference On Software Methods And Tools*. SMT 2000. IEEE Computer Society, Los Alamitos, CA, USA; 279 pp. p127-36

Wilson D, Hall T and Baddoo N (2000) The Software Process Improvement Paradox. *Approaches To Quality Management: Software Quality Management VIII*:97-101.

Baddoo N and Hall T (1999). The Impact Of Software Practitioners On Software Quality. *Proceedings Of The 12th International Software Quality Week Conference*. San Jose, California. 1098-1118. May 24-28

Baddoo N and Hall T (1999). An Empirical Study Of Software Quality. *Proceedings Of Empirical Studies Of Software Development And Evolution Workshop*. ICSE Workshop. Los Angeles, California. 83-87. 18th May

Journal papers:

Baddoo N and Hall T (2002) Motivators Of Software Process Improvement: An Analysis Of Practitioners' Views. *Journal Of Systems & Software*. Accepted awaiting publication.

Baddoo N and Hall T (2002) Software Process Improvement Motivators: An Analysis Using Multidimensional Scaling. *Empirical Software Engineering*. Accepted awaiting publication.

Baddoo N and Hall T (2002) De-motivators Of Software Process Improvement: An Analysis Of Practitioners' Views. *Journal Of Systems & Software*. Accepted awaiting publication.

Wilson D, Hall T and Baddoo N (2001) A Framework For Evaluation And Prediction Of Software Process Improvement Success. *Journal of Systems and Software*, **59 (2)**: 135-142

Baddoo N and Hall T (in review) Using Experiences And Opinions To Measure Software Process Improvement. *Information And Software Technology*.

Baddoo N and Hall T (in review) Practitioner Roles In Software Process Improvement: An Analysis Using Grid Technique. *Software Process - Improvement and Practice*.

Harrison R, Baddoo N, Barry E, Biffl S, Parra A, Winter B and Wuest J (1999) Directions and Methodologies for Empirical Software Engineering Research. *Empirical Software Engineering* **4 (4)**:405-410.