Investigating Software Process in Practice: A Grounded Theory Perspective

Gerard Coleman

M.Sc.

Submitted for the degree of Doctor of Philosophy

School of Computing, Dublin City University

Supervisor: Dr. Rory V. O'Connor

February 2006

Declaration

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of PhD, is entirely my own work and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

Signed: Gerard C. O'Connell ID No: 94971480

Date: 31st August 2006

Abstract

This thesis is concerned with how software process and software process improvement is practiced within the indigenous Irish software product industry. Using the grounded theory methodology, the study utilises in-depth interviews to examine the attitude and perceptions of practitioners towards software process and software process improvement. The outcome of the work is a theory, grounded in the field data, that explains how software processes are formed and evolve, and when and why software process improvement is undertaken. The resultant grounded theory is based on two conceptual themes, Process Formation and Process Evolution, and one core theoretical category, Cost of Process.

The empirical investigation shows that software process improvement programmes are implemented by companies as a reaction to business events, and how many software managers reject software process improvement because of the associated costs. In addition, indigenous Irish software companies largely ignore commercial best practice software process improvement models, and the reasons for this are discussed.

The research also argues that software process improvement is not solely technology-centred but is also affected by wider human and organisational factors. As these 'socio-cultural' influences have been more widely addressed in the Information Systems discipline, than in Software Engineering, this work draws on the experiences and lessons from both disciplines and ultimately resides between these two academic fields.

The results of this work provide new light on the issues facing software process and process improvement in small software product companies and make a contribution towards bridging the gaps between research and practice, and theory and practice, in both Software Engineering and Information Systems.

Acknowledgements

Firstly, I would like to thank all of the practitioners who gave their time so willingly and generously, and for sharing their experiences of software development with me.

I wish to recognise Tom Collins, Director of Dundalk Institute of Technology, for his vision and support, for research and researchers, and whose initiatives created the space in which this work could be completed.

Thanks are also due to Norah Power and Joe McDonagh who kindly gave their time to discuss the finer points of the work.

I was fortunate to have two wise and committed supervisors. Tony Moynihan provided the initial inspiration and steered me in the direction of grounded theory as a solution to my methodological problems. Rory O'Connor supplied the drive and motivation I needed to translate the initial ideas into something real and concrete. I will always be grateful to Rory for his ever-professional approach and clear insight.

The lads at DkIT, Kevin McDaid, Frank Keenan and latterly Fergal McCaffery, provided endless support and encouragement, cleverly disguised as nagging and sarcasm, throughout the study period.

Last, but not least, to my wife Sandra whose love, kindness, patience, and understanding during the last few years transcend mere words. You know the difference you've made.

Contents

List of Tables

List of Figures

Glossary of Grounded Theory Terms

Axial Coding: The process of relating categories to their subcategories, termed "axial" because coding occurs around the axis of a category, linking categories at the level of properties and dimensions.

Categories: Concepts that stand for phenomena.

Coding: The analytic processes through which data are fractured, conceptualised, and integrated to form theory.

Concepts: Higher level codes which identify influencing factors on behaviour and describe the relationships between them. The conceptual code should have properties and dimensions and should be interpreted at a theoretical level.

Constant Comparison: The exploration of similarities and differences across incidents in the data. By comparing where the facts are similar or different the researcher can generate concepts based on recurring patterns of behaviour.

Core Category: A main theme which sums up a pattern of behaviour and explained in terms of its relevance to other core categories. It has theoretical significance and its development should be traceable back through the data.

Diagrams: Visual devices that depict the relationships among concepts.

Dimensions: The range along which general properties of a category vary, giving specification to a category and variation to the theory.

Grounded Theory: A method of conducting qualitative research that focuses on creating conceptual frameworks or theories through building inductive analysis from the data.

Memos: Written records of analysis that may vary in type or form.

Open Coding: The analytic process through which concepts are identified and their properties and dimensions are discovered in data.

Phenomena: Central ideas in the data represented as concepts.

Process: Sequences of action/interaction pertaining to a phenomenon as they evolve over time.

Properties: Characteristics of a category, the delineation of which defines and gives it meaning.

Selective Coding: The process of integrating and refining the theory.

Subcategories: Concepts that pertain to a category, giving it further clarification and specification.

Theoretical Sampling: Sampling on the basis of emerging concepts, with the aim of being able to explore the dimensional range or varied conditions along which the properties of concepts vary.

Theoretical Saturation: The point in category development at which no new properties, dimensions, or relationships emerge during analysis.

Theoretical Sensitivity: The ability of the researcher to think inductively and move from the particular (data) to the general or abstract, that is, to build theory from observations of specifics.

Theory: A set of well-developed concepts related through statements of relationship, which together constitute an integrated framework that can be used to explain or predict phenomena.

Part I Study Background

Part I – Overview

The first part of the thesis contains 4 Chapters. Chapter 1 outlines the reasons for the study and describes the research question and setting and the study objectives. Chapter 2 presents an overview of the Irish software industry and the development of the indigenous software sector. Chapter 3 contains a description of software process and software process improvement (SPI) and offers an analysis of the most common process and process improvement models. Chapter 4, on research methodology, describes the different types of methodologies available to quantitative and qualitative research studies and discusses in detail the reasons for the selection of grounded theory for use in this study.

Chapter 1 The Study and its Focus

1.1 Software Process in Practice

This thesis is concerned with software process, how it is used in practice, how and why changes are made to it and how it is improved. A software process essentially describes the way a company develops its software products and supporting services, such as documentation. Processes define what steps the development organisations should take at each stage of production. They provide assistance in making estimates, developing plans, and measuring quality. All companies follow a software process and a number of standard process models have been designed to help companies manage their software development activity.

There is a widely held belief that a better software process results in a better software product. This has led to a focus on software process improvement (SPI) to help companies realise this benefit. SPI models, developed to assist companies in this regard, purport to represent beacons of best practice. Contained within the scope of these models, according to their supporters, lies the road to budgetary and schedule adherence, better product quality and improved customer satisfaction. Translating these benefits into practice has, however, proved challenging. Studies illustrating how the major SPI models have been successfully implemented in companies represent a very small proportion of the software industry as a whole and tend to be confined to specific domain areas. Opponents believe that these process improvement models operate primarily at a theoretical level, are too prescriptive and bureaucratic to implement in practice, and require a subscribing company to adapt to the models rather than having the models easily adapt to them. Software companies, therefore, have a number of factors to consider before making a decision on how to improve their software development capability.

1.1.1 Software Process Improvement and Context

All software companies are not the same. They vary according to factors including size, market sector, time in business, management style, product range and geographical location. For example, a software company operating in India may have a completely different set of operational problems to contend with to a software company in Israel or Ireland. Even within a single geographical area, such as Ireland, the range of operational issues faced by a small Irish-owned firm can be radically different to those affecting a multinational subsidiary. The fact that all companies are not the same raises important matters for those who develop both software process and process improvement models. To be widely adopted by the software industry, any process or process improvement model should be capable of handling the differences in the operational contexts of the companies making up that industry. But process improvement models, though highly publicised and marketed, are far from being extensively deployed and their influence in the software industry therefore remains more at a theoretical than practical level.

1.2 The Research Agenda

1.2.1 The Research Question

The premise of this study is that, in practice, software companies are not following 'best practice' process improvement models. On this basis, the work initially set out to explore two primary questions:

- Why are software companies not using 'best practice' SPI models?
- What software processes are software companies using?

Preliminary investigation of the two research questions raised the following linked questions:

- How are software processes initially established in a software company?
- How do the software processes, that software companies are using, change?
- What causes these software processes to change?
- How do the operational and contextual factors, present in organisations, influence the content of software processes?

Because of the need to explore process in practice, and the fact that context or situation is important, it was necessary to impose some boundaries on the study and clarify the research setting.

1.2.2 The Research Setting

The research questions listed in the preceding section raised a number of fundamental issues which helped create boundaries around the study. It is clear that software process and SPI are fundamentally concerned with software production. However, there is a wide spectrum of organisations whose business, or part of, is developing software. These organisations span a broad range which includes pure software product companies, who are making generic 'shrink-wrapped' products for the mass market, to IT companies, whose business is not primarily software but who occasionally develop bespoke software systems for particular customers, to in-house software development departments of non-software companies. The diverse nature of these operations, and the potentially vast number of variables to be contended with, place such a study beyond the scope of the resources of a sole researcher.

To reduce the scope, it was decided that the investigation of a more homogeneous group would limit the number of variables to be considered. However, it was equally important to ensure diversity if a rich explanation of what was taking place in relation to software process in practice was to be arrived at. Software product companies provide such a group. This group's primary business is software development and as software development professionals they would be familiar with software process and the considerations involved in using both process and process improvement models. Given the resources available, and the geographical location of the researcher, it was decided to confine the study to Irish software product companies. An added advantage of restricting the study to within the same jurisdiction, would be that each company would have the same economic and regulatory regimes governing their operations.

However, the Irish software industry is populated by both indigenous and multinational software companies so, at this point, some of the other research questions helped narrow the scope of the study further. One of the research questions asked how process was established in an organisation. To investigate this would require information relating to the organisation's start-up phase and early days of operation. Another question related to how and why process changes in organisations. To help determine this would necessitate getting an historical perspective on company developments. In the case of non-Irish multinationals operating in the country, many of them have opened those offices a number of years after the parent company has been established. In many of these instances the software process used by the Irish offshoot has initially been developed by the parent company and then merely devolved to the Irish subsidiary. Therefore, determining how the process was initially established, and had subsequently evolved, would be practically impossible. As the causes of software process change are a central element in the research, a decision was made to concentrate on indigenous Irish software product companies who, as software developers both in a young industry and in a concentrated geographic space, could provide the historical information required to answer the questions on process foundation and evolution.

1.2.3 Research Objectives

In studies which report success in implementing SPI models, much content relates to very large companies or those seeking certification to operate in specific industry sectors. In addition, a number of implementations centred on in-house software development departments rather than software product companies. It is clear there is not a substantial corpus of work describing unsuccessful implementations of SPI models. More importantly there is a paucity of information on what is happening in the industry in general and, in particular, in companies who are not adopting these process improvement models. Critically, in the context of this study, there is limited published work regarding what is happening in Irish software companies. As a result, having clarified the research questions and determined the research agenda, the research objectives can be stated as:

- To provide a new perspective on software process as it is practiced in software development
- To explain the role of software process and SPI in software product companies
- To investigate the factors that influence software process evolution in software product companies
- To build theoretical concepts that are grounded in the voices and experience of Irish software development managers.
- To develop and incorporate the overall findings into a theoretical framework that has explanatory and descriptive power.

1.2.4 The Focus of the Study

This study is concerned with software process as it is practiced in indigenous Irish software companies. It examines the operating context in which software process is used and explores how software process is initially created and how it changes over time. It investigates what companies are doing in relation to SPI and equally importantly what they are not doing. The answers to the research questions are expressed in the form of a theory, derived from a study of practice, which attempts to explain software process in context. Because of this, the research concentrates initially on practice before progressing to theory.

1.2.5 Situating the Study

In relation to academic discipline this research study resides within the broad field of software development. Though the key aspects of the study relate to software process and process improvement, which have traditionally sat within the Software Engineering (SE) discipline, the work explores beyond the confines of SE into the field of Information Systems (IS). The study will argue that factors outside the technological framework which envelopes the common SPI models also affect SPI and must therefore be considered by any SPI programme. In addition, the qualitative nature of the work, and the research methodology used, require that lessons from studies in fields other than

software engineering are also incorporated. The issues relating to where this study is situated are further discussed in 4.10.

1.3 Structure of the Thesis

This thesis is divided into four parts. Part I, which contains this chapter, contains 3 further chapters. Chapter 2 reviews the role of the software industry in Ireland with specific concentration on indigenous software producers. Chapter 3 contains an overview of the software process literature and analyses software process and process improvement models. Chapter 4 contains a review of the research methods literature and presents the method chosen for this particular study and the reasons for its selection.

Part II incorporates four chapters and contains the main findings of the study. Chapter 5 discusses how the research was carried out, profiles the companies involved, explains how the research findings were drawn and introduces the theoretical framework which is presented as a network diagram. The diagram shows the relationships between the compendium of process variables elicited in the study, identifying the elements of software process in practice and its composition. Chapter 6 presents the conceptual theme, **Process Formation**, explaining how process is initially formed, the factors which influence this formation and how this links with operating context. Chapter 7 discusses the conceptual theme, **Process Evolution** describing why evolution takes place and the factors which cause process to change. Chapter 8 discusses the study's core theoretical category, **Cost of Process,** and relates the process models used in practice to the SPI models, CMM/CMMI, ISO 9000 and the development methodology XP.

Part III has three chapters. Chapter 9 presents support for the findings on **Process Formation** presented in Part II. Chapter 10 describes the support contained in the literature as confirmation on the findings on **Process Evolution** and Chapter 11 presents the published evidence to substantiate the theory generated for **Cost of Process**.

Part IV contains two chapters. Chapter 12 examines how the research results should be evaluated and discusses the issues of generalisability and verification. Chapter 13 concludes the thesis, reviewing the research questions and original research objectives in light of the results produced. It provides a summary of the work carried out and presents a set of conclusions, drawn from the findings. It contains an account of the contribution of the research and suggests what implications the results have, both for practice and for research. Finally a number of suggestions are made for future research based on the outcomes of this work.

Chapter 2 Indigenous Companies and the Irish Software Industry

2.1 Introduction

This chapter discusses the Irish software industry and the place and role of the indigenous software sector within it. It differentiates between foreign companies who have set up bases in Ireland, and native or indigenous companies who also thrive in the Irish software landscape. How the indigenous software industry can be classified according to size is examined and the chapter concludes by highlighting how important relative and absolute size, and its relationship to company development, is to this study.

2.2 Irish Software Industry

The software industry in Ireland is a key component of the national economy. For the purposes of this study McIver's (1998) definition of the Irish software industry will be used – "companies whose main business is in software or software-intensive products, that develop or modify software in Ireland". The PriceWaterhouseCoopers report (2005), "Doing Business and Investing in Ireland", highlights the success of the Information and Communications Technology (ICT) sector, and states that many of the world's leading software companies have established Centres of Excellence in the country. This has resulted from the impressive growth in the Irish software industry which, during the 1990s, grew at 2.5 times the rate of the economy generally (Arora *et al.*, 2001).

According to Enterprise Ireland (2005a) the growth of the Irish software industry can be traced to a decision by the government in the late 1970s to attract high-value industries, including software, to Ireland. During the 1980s however, the Irish software industry stagnated to a great extent because of its reliance on bespoke software services, limited exports and low profits (Enterprise Ireland, 2005a). Real growth in the software sector only became apparent in the 1990s when at the start of that period employment in the Irish software industry stood at just under 8,000 and subsequently grew, until the end of

the decade, at an annualised rate of 15% (Crone, 2002). At the end of 2004, it was estimated that the Irish software industry consisted of more than 900 companies, 140 of them foreign, employing 24,000 people and exporting over €16bn worth of products and services (Enterprise Ireland, 2005b). Exports from indigenous companies accounted for €1bn of this total. Average employment growth in the sector grew most markedly from 1996 to 2000, boosted by the greater availability of venture capital. Furthermore, the Industrial Development Authority's annual report (IDA Ireland, 2003) shows that of all the Foreign Direct Investment (FDI) into Europe, Ireland wins 41% of all software projects.

The reasons which attract the multinational sector to Ireland are many and varied. Flood *et al.* (2002) cite the attractive grants and tax incentives as a primary attraction and these coupled with the availability of a young, skilled workforce have helped create the Irish software success story. These findings support other sources (Enterprise Ireland, 2005a, Crone, 2002; Green *et al.*, 2001), and go on to state that the Multi-National Corporation (MNC) sector in Ireland contains over 300 international companies including five of the world's top ten. It is difficult to overstate the importance of the software industry to Ireland. In 1999 the IDA released figures that illustrated how software exports from Ireland represented 34% of the world's market, and over 40% of the packaged software and 60% of the business software market in Europe (IDA Ireland, 1999). By 2001, the software sector in Ireland was responsible for 8% of the country's GDP and 10% of its exports (HotOrigin, 2001).

In terms of revenue and exports in the Irish software sector, Enterprise Ireland reports stark contrasts between the MNCs and indigenous companies (Enterprise Ireland, 2005b). Their figures show that, in 2003, the MNC sector accounted for almost 92% of total revenue and 94% of the total exports generated by the industry. However, the numbers employed in the software sector are much more evenly distributed between the MNC and the indigenous sector. At the end of 2003, MNCs employed around 53% of a software workforce of c.24,000 whilst indigenous companies employed the remaining 47%.

2.3 Indigenous Irish Software Industry

As discussed in Section 1.2.4 the particular focus in this study is on indigenous companies. The New Oxford Dictionary of English (2001) defines 'indigenous' as, "originating or occurring naturally in a particular place; native". Therefore, for the purposes of this research, combining this with McIver's definition of the Irish software industry, indigenous Irish software companies can be defined as those:

> *founded in Ireland, and whose parent company resides there, and whose main business is in the development or modification of software or software-intensive products.*

In his analysis of the Irish Software Industry, Crone (2002) argues that the indigenous Irish software segment is comprised of a number of sub-sectors. He distinguishes between those who sell software products and those who sell software services. However, these categories are not always clear-cut. Those who sell software services are heavily engaged in bespoke software development, customer support and Internet services which also often involves a software development element. Though Arora *et al.* (2001) estimate that around 44% of indigenous software firms are involved purely in the development of software products, they suggest that many of these software products require service elements such as installation and training. Furthermore, generic software products are often customised in some way to meet individual customer requirements and this also brings in a service component. There is, therefore, a significant cross-fertilisation of activities within the indigenous software sector.

In the view of Enterprise Ireland (2005a), the success of the indigenous software product industry is based on a small number of common characteristics. Firstly, export markets are a priority. Secondly, the market targeted is typically a niche (or vertical) market where the competition does not include the leading worldwide software players. Thirdly, there is an emphasis on quality processes and products. And fourthly, major attention is paid to the management of the business.

Several commentators discuss how Ireland has gradually developed a range of firms which sell software products in international niche markets. Arora *et al.* (2001) believe this occurs because in niche markets there are low entry barriers. With regard to the software products themselves they tend to be quite technical and thus can avoid having to challenge directly the dominance of major US firms in consumer packaged software markets. Green *et al.* (2001) support this assertion, claiming that indigenous software producers tend to be more specialised in terms of both types of products and types of customers and at least a third of sales from indigenous companies have targeted niche markets both locally and globally. Indeed more than 70% of indigenous software firms are trading internationally (Flood *et al.*, 2002).

In the late 1990's the success of the indigenous software sector was reflected in the fact that seven of Ireland's leading indigenous software companies had stock market listings. However, since that time, through a difficult trading period and accompanying management buyouts, this number has been reduced and there are now only three publicly-quoted indigenous software companies, Iona Technologies and Trintech who are quoted on the Nasdaq market in the United States, and Datalex who are listed on the Irish stock exchange. With a combined annual turnover of 120 million Euro in 2003 (HotOrigin, 2004), these companies have remained relatively successful despite harsh trading conditions.

2.3.1 Size of Indigenous Irish Software Companies

In response to changing economic circumstances, and to acknowledge the hurdles facing smaller organisations, The European Commission (2005) introduced a new definition for the small to medium-sized enterprise (SME). The Commission estimate that within the European Union there are some 23 million SMEs which cover 99% of all enterprises. Within the EU, SMEs are now classified as "enterprises which employ fewer than 250 persons and which have an annual turnover not exceeding 50 million Euro, and/or an annual balance sheet total not exceeding 43 million Euro". They break this down further defining small enterprises as those "which employ fewer than 50

persons, and whose annual turnover or annual balance sheet total does not exceed 10 million Euro", and micro enterprises as those "which employ fewer than 10 persons and whose annual turnover or annual balance sheet total does not exceed 2 million Euro" (European Commission, 2005).

The great majority of indigenous Irish software firms are SMEs. Crone (2002) reports that in 1998 only 1.9% (10 companies), out of a total of 630 indigenous software companies, employed more than 100 people whilst 61% of the total employed 10 or fewer. This feature of very small Irish software firms goes back some time. O'Riain (1997) cites a report from 1987, from An Coras Trachtala, which showed that the indigenous software industry consisted primarily of micro firms which provided consultancy and services to businesses adopting IT systems. The author shows, that though a dynamic, indigenous software sector subsequently developed, two-thirds of companies employ fewer than 10 people. Arora *et al.* (2001) offer additional evidence for this asserting that the average size of indigenous Irish software firms is about 16 employees. But all organisations clearly go through different stages of growth. The venture capital group, HotOrigin produce an annual report on the state of the indigenous software industry. They estimate there is a total of 417 indigenous software product companies in Ireland (HotOrigin, 2004). They categorise indigenous software firms across three stages of company development, **'Start-up'** (1-25 employees), **'Build'** (26-75 employees), and **'Expansion'** (75+ employees). The 2004 report shows that almost three-quarters of indigenous software firms fall into the Start-up category, with about 9% in the Expansion category and the remainder in the Build category. The number of start-up product companies continues to grow and has now almost reached its historic high.

In their study of management teams and innovation, Flood *et al.* (2002) use similar size categories (Start-up 1-30; Build 30-75; Expansion 75+) to those of HotOrigin. They estimate that the average number of employees in Build and Expansion companies exceeds 90. However, their study was conducted between the economically-dynamic

years of 1998-2002 and as several reports show (HotOrigin, 2004, Enterprise Ireland 2005b), employment levels in the software sector have fallen since then.

The fact that authors have broken indigenous software companies down across three size categories, all within the European definition of an SME, suggests that companies experience significant change at even modest levels of employment and that different factors affect the different size brackets.

2.4 Focus of Current Research

As this study is examining how process evolves in practice the (HotOrigin, 2004 and Flood *et al.*, 2002) categories listed above are of particular importance as the relative size of companies can indicate how process evolves as the company grows and progresses from one size category to another. A process that is suitable for a company with 5 employees will likely not be suitable for one with 75 so the relative categories are important in studying evolution issues. As almost all of the indigenous software companies can be classified in absolute terms as SMEs, the relative size offers more potentially fruitful sources of information in studying process evolution and development.

Nonetheless, absolute size is also important. Many studies within the literature discuss whether best practice models, traditionally developed for large corporations, can be scaled down for use by smaller companies. But smaller companies, in the terms discussed within these studies, often straddle all three of the company size categories, start-up, build and expansion. The relative size categories can therefore assist in investigating how relevant best practice models are to software companies at various stages of growth and whether these models can be successfully scaled down for use by indigenous software firms.

2.5 Summary

This chapter presented a summary of the Irish Software Industry. This industry is composed of two major segments the MNC sector, who have established offices in Ireland, and a smaller, vibrant indigenous sector. The indigenous sector, though with a much smaller percentage of revenue, employs almost as many people as the multinational segment. The indigenous sector is populated in the majority by companies who employ fewer than 25 people and are heavily export-driven with products destined for niche markets. One of the ways in which the indigenous sector remains competitive is through improving the processes it uses to develop its software products. The role of software process and SPI is the focus of the next chapter.

Chapter 3 Software Process and Process Improvement

3.1 Introduction

This chapter examines the various facets of software process. It presents a number of software process models used in the development of software products and several SPI models which are aimed at assisting companies in improving the way they develop software. The chapter also discusses the concept of process maturity and assessment as ways of indicating the capability of an organisation to develop software. Some of the major improvement models are then analysed. The prime agile development methodology is also presented. Finally, there is a discussion on how the various models and methodologies interact in practice.

3.2 Software Process

Defined as "A set of activities, methods, practices and transformations that people use to develop and maintain software and the associated products (e.g. project plans, design documents, code, test cases and user manuals)" (Zahran, 1998), a software process is used by all organisations in the creation of software products. Kruchten (2000) states that a software process has four distinct roles:

1. To provide guidance as to the order of the activities to be undertaken
2. To specify the artefacts that should be developed and when
3. To direct the tasks of the development team
4. To offer ways of monitoring and measuring a project's progress and outputs.

The activities referred to in 1 above generally fall under four headings (Sommerville, 2004):

- Software specification - the functionality of the software is defined;
- Software design and implementation - the software product is created consistent with the specification;
- Software validation - the software is checked and tested to ensure that it complies with the specification and

- Software evolution - the software is modified and upgraded to meet changing customer demands.

The process and associated activities are often documented as sets of procedures to be followed during development. However, the documentation is not the process but should clearly represent the process as it is implemented within an organisation. To simplify understanding and to create a generic framework which can be adapted by organisations, software processes are represented in an abstract form as software process models. A number of different models exist as instantiations of how software development can be undertaken.

3.3 Software Process Models

Generic software process models are prescriptive in that they indicate how software should be developed. They fall under three general categories (Sommerville, 2004):

- Waterfall Development
- Evolutionary Development
- Component-based Development.

The Waterfall Development model (Royce, 1970) represents the fundamental development steps of specification, design, implementation, validation and evolution as separate process phases. Each of these phases must be complete before the next phase can commence.

Whilst the Evolutionary Development model replicates each of the fundamental development steps as used in Waterfall, these are iterated several times prior to system completion. A number of process models exist including (Boehm, 1988; McCracken and Jackson, 1992) which come under the evolutionary development banner. Evolutionary Development models attempt to address what they see as the deficiencies in the Waterfall model by enabling much greater customer interaction during development and not freezing requirements early in the process.

Component-based Development (Thomas, 1995) endeavours to take advantage of the ability to reuse previously developed and tested software. Components now exist for a range of software applications, and developers use this model to bolt systems, or parts of systems, together from them. The benefit of this approach is the increased speed of development and reduced testing time.

One of the more commonly used commercial process models is the Unified Process (UP), which is often better known under its original name, the Rational Unified Process (RUP) (Kruchten, 2000). The UP/RUP incorporates elements of all three generic models and has four distinct phases, Inception, Elaboration, Construction and Transition. The phases are driven primarily by business rather than technical concerns. Though the model contains discrete phases, each phase allows for a series of iterations and the system can then be built incrementally.

3.4 Software Process Improvement

SPI aims to understand the software process as it is used within an organisation and thus drive the implementation of changes to that process to achieve specific goals such as increasing development speed, achieving higher product quality or reducing costs.

The reason for this focus on SPI is encapsulated in the belief that there is an intrinsic link between the quality of the software process and the quality of the outputs emanating from that process and this belief is shared by a number of authors. Zahran (1998) claims "it is a widely accepted fact that the quality of a software product is largely determined by the quality of the process used to maintain and develop it". Humphrey (1995) states that "to improve your product, you must improve your process quality". Fitzgerald and O'Kane (1999) support the view that "increasing software process maturity is an obvious and logical step in addressing the software crisis". The SPIRE handbook (Sanders, 1998), which concentrates on small companies, also promotes improving software processes arguing that a software process is like any other business process and that process efficiency means business efficiency and better products. A number of

others support the link between process and outputs including (Ahern *et al.*, 2004; Florac and Carleton, 1999; Goldenson and Gibson, 2003; Grady, 1997).

However, there is division of opinion within the industry. Though Sommerville (2004) maintains that many organisations now believe a better software process will result in better software products, it is a position that he dissents from stating that technology and, in particular, people factors also affect the quality of the resultant software products. DeMarco and Lister (1999), after an extensive survey of failed software projects, concluded that sociological rather than technological factors were the main cause of failure and that people issues must be given primacy in software development. Madhavji (1991) supports this claiming that though process models help software development understanding, they can conceal important process details including human factors. Bach (1994) also comments on the importance of people suggesting "far too much is written about processes and methods for developing software and far too little about... the minds that actually write the software" and Perry *et al.* (1994) agree believing too much attention is paid by the software community to process and related technological factors and not enough to organisational and social factors.

3.5 SPI Models

In the 1970s and 1980s the work of Crosby (1979) and Juran (1988) demonstrated that, in the area of production management, product quality could be improved through a better production process. Motivated by these results, the Software Engineering Institute (SEI) commenced the examination of how the benefits could be translated to software development and this resulted in a focus on SPI. The output of the SEI work was the maturity framework for software development (Humphrey 1988). This framework soon after emerged as the Capability Maturity Model for software (SW-CMM) (Paulk *et al.*, 1991), hereafter in this study known as the CMM. Several amended versions of the CMM were subsequently released.

A number of other SPI models have been developed including BootStrap (Haase *et al.*, 1994) and Trillium (Zahran, 1998). The International Organisation for Standardisation

(ISO) also embarked on a programme to create a software process assessment standard. The SPICE (Software Process Improvement and Capability dEtermination) project (Dorling, 1993) culminated in the ISO/IEC 15504 standard. The standard includes guidelines for developing assessment instruments and conducting assessments, and provides a reference model and rating scheme (Zahran, 1998). As well as these corporate process improvement models several other improvement models were developed aimed at, managing software people (Curtis *et al.*, 1995), software teams (Humphrey, 2000), and the individual developer (Humphrey, 1995; Coleman and O'Connor, 2000). In addition, the ISO 9001 standard, used to accredit company quality systems, has also been implemented within software companies. It would be impractical in this study to review all of the existing models used by practitioners, so the researcher will concentrate on those most relevant to the work.

3.5.1 The Capability Maturity Model

The CMM is a five-level improvement model which specifies recommended practices in the particular areas that have been shown to enhance software development and maintenance capability (Paulk *et al.*, 1991). The levels (Table 3.1) represent a measure of process maturity within the software organisation.

Table 3.1 CMM Levels and Description

CMM Level	Description
1 - 'Initial'	Development process is chaotic and unstructured. Process is ad-hoc.
2 - 'Repeatable'	Objective is to ensure that successful techniques and approaches used previously can be assimilated and used on current and future developments. Process is disciplined.
3 - 'Defined'	Focus on ensuring process integration throughout the organisation. Process is standard and consistent.
4 - 'Managed'	Measurement introduced to assist in managing the quality of the process and the product. Process is predictable.
5 - 'Optimising'	Focus is on continuous quality improvement. Process is continuously improving.

The maturity levels lay the foundation for continuous process improvement and the model itself provides a road map for achieving this from Level 1, where development is

unstructured and many practices are operated in an ad-hoc fashion, to Level 5 where, with a clearly defined and managed software development process, the organisation can focus on continuous improvement.

Each of the maturity levels is comprised of a set of process goals to enable process improvement. These goals are encased within Key Process Areas (KPAs), which specify the issues that have to be addressed to achieve compliance at a particular maturity level. Each KPA has an associated set of practices which must be executed to satisfy the process goals of that KPA. Each of the KPAs, within a maturity level, must be satisfied before an organisation achieves certification at that particular maturity level.

3.5.2 The Capability Maturity Model Integration

The success of the CMM spawned a range of successors, and these converged successfully into the Capability Maturity Model Integration (CMMI) (Chrissis *et al.*, 2003). CMMI is a departure from the CMM in that it has two process representations, 'Staged' and 'Continuous'. The Staged representation retains the maturity level concept from the CMM with a slight change in nomenclature - Level 2 is now known as 'Managed' and Level 4, 'Quantitatively Managed'. The same concepts in relation to the satisfying of process areas and specific process goals apply to maturity levels in CMMI as applied with the CMM.

The Continuous representation denotes a major departure from the CMM. It offers greater flexibility in how process improvement can be achieved in that it allows the organisation to pursue process improvement in areas of particular process weakness. Improvement in these individual process areas is measured through Capability levels. There are 6 Capability levels ranging from Capability Level 0 'Incomplete', where the process is not performed or merely partially performed, through to Capability Level 5 'Optimising', where the process is constantly improved based on an understanding of the causes of variation within it.

3.6 Software Process Assessment

A software process assessment involves an appraisal of an organisation's software process. An assessment (referred to within CMM and CMMI as an 'appraisal' and as such used interchangeably in this document) involves a trained team of software professionals whose duty is to determine the state of an organisation's software process, ascertain the process-related issues facing the organisation and obtain the organisation's support for an SPI initiative (Paulk *et al.*, 1994).

3.6.1 Assessments Versus Audits

It is important to note the difference between an assessment and an audit. An assessment is essentially a review to 'advise [a company's] management and professionals on how they can improve their operation' (Humphrey, 1989). An audit, as defined by the IEEE (1991), is 'an independent examination of a work product or set of work products to assess compliance with specifications, standards, contractual agreements and other criteria'. Assessments are used in CMM/CMMI and audits are used in ISO 9000. A third-party assessment under CMM/CMMI may be used to award a maturity level rating to the organisation concerned. An audit conducted under ISO guidelines may result in the company concerned achieving certification against the relevant ISO standard.

3.6.2 Assessment Results for CMM

To determine how processes within the software community are maturing, the SEI publishes bi-annual reports which show the results of assessments carried out against the CMM and the CMMI. The 2002 report (Software Engineering Institute, 2002) provides details of assessments of 1,100+ organisations carried out between 1998 and 2002 whilst the mid-2005 report (Software Engineering Institute, 2005a) details 1,600+ organisational assessments carried out since mid-2001. A clear improvement in community maturity can be seen between 2002 and 2005. Of all the organisations assessed, 62.5% were rated below level 3 in 2002, as opposed to 48% in 2005 whilst a greater leap in maturity can be seen in the level 5 returns where only 6.8% of organisations achieved level 5 in 2002 compared to 9.8% of organisations in 2005.

Importantly, the figures vary somewhat when broken down between US-based organisations and non-US-based organisations. In 2002 the total number of US-based organisations assessed over the preceding four years was 645 with non-US-based organisations accounting for 479 assessments in the same period. However, during the period 2001-2005 the corresponding figures were 560 for US-based organisations and 1,053 for non-US-based. Significantly, higher CMM maturity is associated with non-US-based companies. The 2005 data shows that over 62% of US-based organisations were rated below CMM level 3 compared with just over 40% for non-US-based, whilst only 8% of US-based companies were rated above CMM level 3 compared with 23% of non-US-based companies.

Because of the reporting restrictions on the data it is difficult to be conclusive about why this major discrepancy is occurring. However, there is some evidence (Keeni, 2000; Cusumano *et al.*, 2003), to indicate that it is a result of 'offshoring', whereby US companies are locating some of their development work in lower cost countries such as India and China. In these instances high-CMM compliance may be being used as a way of attracting additional business perhaps even in some cases from the US parent.

3.6.3 Assessment Results for CMMI

The CMMI not only offers two representations, Staged and Continuous, but also different classes of assessments, known as A, B and C (Table 3.2), which are carried out under the ARC (Appraisal Requirements for CMMI) criteria. Class A Assessments in CMMI, which lead to a maturity rating, take place within the Standard CMMI Appraisal Method for Process Improvement (SCAMPI) framework (Ahern *et al.*, 2004). SCAMPI appraisals have been undertaken since 2002 and, as the SEI moves towards 'sunsetting' the CMM, now take on increased relevance.

Table 3.2 ARC Appraisal Classes

Appraisal Class	Description
A	Full comprehensive approach. Provides thorough analysis of all processes and provides the maturity level rating.
B	Less comprehensive than Class A. Initial or partial self-assessment undertaken. Does not provide a maturity level rating.
C	Brief examination. Least expensive and time-consuming approach. Does not provide a maturity level rating.

As Class A appraisals are the only type which generate maturity level ratings, it is these results that are reported by the SEI through bi-annual updates. The 2005 CMMI figures (Software Engineering Institute, 2005b) show that since 2002, 868 Class A appraisals have been carried out on 782 organisations. Of the maturity level ratings reported, 34% of organisations are rated at level 2 - 'Managed', 29% at level 3 - 'Defined', 4% at level 4 – 'Quantitatively Managed' and 19% at the highest 'Optimising' level. Though these figures show a higher maturity profile than those reported from CMM assessments, it is not possible without access to further information from the SEI to explain the differential. It is likely however, that many organisations, who have previously been assessed under the CMM standard, and who may then have instigated SPI initiatives, have moved directly onto the new CMMI standard. Also, it is worth noting that some of the highest proportions of SCAMPI assessments have taken place in the US-favoured offshore locations of India and China where the highest proportionate number of higher-level CMM companies reside.

3.7 Analysis of the CMM and CMMI

Because it has been in existence longer than CMMI, and more companies have had an opportunity to experiment with it, there is more literature available on the application of CMM within the industry. One of the earliest studies claimed the benefits of introducing CMM included returns of five times the expenditure on the SPI programme and annual savings of $2m (Humphrey *et al.*, 1991). Another study in 1993 showed that a five-year CMM-based improvement programme achieved better productivity and schedule adherence, reduced rework and a return in excess of 7:1 for each dollar invested (Dion,

1993). A multiple case study by Herbsleb *et al.* (1997) showed that CMM-based SPI resulted in improvements in cycle time, quality, and productivity. Additional CMM-based process improvement studies, including (Buchman, 1996; Daskalantonakis, 1994; Hollenbach *et al.*, 1997; Pitterman, 2000), also report positive results, particularly in the areas of project management and software quality.

Because of its recent arrival, fewer organisations have experimented with CMMI and there are correspondingly fewer reports in the literature of its application. One of the more comprehensive articles emanates from the SEI itself and the studies it has carried out with companies (Goldenson and Gibson, 2003). The results show that the organisations experienced, to varying degrees, improvements in schedule and quality, and reductions in cost. Miller *et al.* (2002), show how CMMI can work successfully with simulations (used, for example, to analyse the behaviour of systems, either real or imaginary, over time) whilst Heinz (2004) demonstrates how CMMI's Continuous representation can be used to improve process capability in small companies.

Despite its vocal body of supporters, the CMM also has a number of opponents. Criticism of the original version of the CMM is voiced by Bollinger and McGowan (1991). They express concern about the concept of maturity level grades arguing, "it is difficult to overstate the psychological and contractual implications of the numeric grade assigned as one result of an [assessment]". They fear that the grading system could be used as a crude measure to distinguish software organisations. Baker (1996), concurs with this, seeing the maturity level grading scheme as potentially useful when government contracts are at stake and suppliers are being selected but worries that organisations become obsessed with achieving a particular rating rather than actually examining the software process for potential improvement areas.

Bach (1994) also criticises the maturity level ratings but from a slightly different perspective. He highlights the number of very successful companies whose practices, he claims, would be classified at level 1 under the CMM's maturity grading scheme. He believes that the CMM has, "no formal theoretical basis [but] only vague empirical

support", and denounces its institutionalisation of process, and the fact that it ignores the people element which he feels is an integral part of software development. Fayad and Laitinen (1997) state that that the grading scheme is fundamentally flawed and argue that CMM assessments generate no benefit to young or start-up companies and that the practices contained within the model remain unproven. Finally, Card (2000), believes CMM focuses too heavily on the model assessors to the detriment of the model implementers and contends that CMM improvement programmes should be linked more closely to business performance.

Whatever the merits and demerits of the arguments above, the absolute number of assessments between 2001 and 2005, under 1700 for CMM and under 900 for CMMI, represents a very small proportion of the world's software companies and company in-house developers. With only 11 CMM, and fewer than 10 CMMI, assessments during the same period, from a population of more than 900 software companies, it is clear that the Irish software industry is largely ignoring the most highly-publicised SPI models.

3.8 ISO 9000

Developed by the International Organisation for Standardisation, ISO 9000 is a series of standards used to certify the quality systems used by an organisation (International Organisation for Standardisation, 1987). The ISO 9000 series essentially refers to a family of related standards which includes ISO 9001, 9002, 9003 and 9004. Derived from the British Standards Institute's initial work on quality for manufacturing processes, the ISO 9000 standard was released in 1987 and it has since become the most widely known and accepted standard in the ISO series (Rada, 1996). ISO 9000 provides organisations with guidance on managing quality systems. ISO 9001, as the certification standard, sets out the compliance requirements for companies involved in design, development, production, installation and servicing and is the one most relevant from a software perspective. ISO 9000-3 provides the necessary guidance for companies implementing ISO 9001 compliant processes. Within 10 years of its launch, over 200,000 companies worldwide had received ISO 9000 certification (Hysell, 1999). According to Rada (1996), "though the language and assumptions of ISO 9000 target

the manufacturer... (it) is being applied to quality systems in many organisations, whether or not they are manufacturers". Acknowledging this, in 1992 the ISO addressed the requirements of the software industry by releasing guidelines for the use of ISO 9001 in software development (International Organisation for Standardisation, 1992).

In seeking ISO 9000 certification companies must "prepare documentation that proves the [ISO] requirements are being met" and demonstrate that the documentation is "strictly controlled and that appropriate records of all quality-related activities are kept" (Schuler, 1995). Certification of ISO 9000 is carried out through third-party audits of the organisation's systems and processes and is awarded when the auditors are satisfied that the systems and processes are documented according to the quality standard, and that the staff involved clearly follow this documentation.

Like the CMM/CMMI, ISO 9001 purports to provide twin benefits. Firstly, it can assist organisations with improving quality. Second, it can be used as a marketing tool to assist companies in selling their product or service. Indeed in the 1990s, many European governments insisted that their suppliers were ISO 9000 certified. Unlike CMM/CMMI, ISO 9000 does not provide a road map for improvement beyond the adherence to quality management documents. Therefore, it is often classified as a 'binary' system for process improvement, in that companies are either ISO-certified or they are not. There are no graded levels of certification. An updated standard, ISO 9001:2000 was introduced in 2000 (International Organisation for Standardisation, 2000). The new ISO 9001:2000, is an integration of the existing three standards ISO 9001, ISO 9002 and ISO 9003. It differs from the original in that it places a particular emphasis on measuring customer satisfaction and stresses the importance of making process improvements. However, limited guidance is provided for this and no road map is supplied to support this work.

3.9 Analysis of ISO 9000

Schuler (1995) conducted a study examining ISO 9000 and the role of technical communicators in her organisation and concluded that adopting ISO resulted in higher-quality systems documentation and that even the process itself of pursuing ISO

certification helped technical organisations. Fitzgibbon (1996) believes the original ISO 9000 standards did not consider the specific requirements of software development and were therefore difficult to apply. He argues that, "the production process in software is a relatively insignificant part of the total development effort; definition of requirements, as well as design, implementation, maintenance, verification, and validation, account for a larger share of development activities", and that this fact led to the introduction of (International Organisation for Standardisation, 1992). He summarises by stating that ISO 9001 now provides software companies with an appropriate framework for improving software project management and quality and increasing customer satisfaction.

Coallier (1994) discusses the role of ISO 9001 in software development organisations. He states that ISO 9001 "assumes that products are purchased in a formal, contractual environment with detailed specifications that are correct... however, it is naïve to assume such conditions for complex products like those that incorporate software". He argues that because software products are constantly modified, expanded and upgraded, a total quality approach, incorporating continuous improvement is required, and this, he feels, is absent in ISO 9001. He is also of the opinion that many key process elements required in quality software production, such as, measurement, design, implementation, and maintenance and support are not properly covered and that ultimately an ISO 9001 certification tells little about an organisation's software development capability other than that some basic practices are present. He concludes that the CMM is a significantly more comprehensive mechanism for measuring software development capability. This view is supported by Grady (1997) who reports that "although over 100 Hewlett Packard organisations are ISO 9000 certified, few of them used such certification specifically to motivate software improvements".

ISO 9000's non-suitability for software development also appears in (Oskarsson and Glass, 1996) who state that ISO 9000 is not the best standard to impose on a software organisation and is most often applied in the software domain because of its market credibility. Support for this view comes from Demirors *et al.* (1998) who found that

both ISO 9000 and CMM were unsuitable for use in small organisations and could negate the advantages accruing to small companies, such as the capacity to innovate and respond quickly to business events.

Overall however, there is a limited number of published studies of software companies who have used ISO 9000, a fact commented on by El Emam and Briand (1997) who feel that the reported studies are of limited benefit as they did not involve software organisations and that "more research specifically with software organisations would help the community better understand the effects of [ISO] registration".

3.10 Agile Methodologies

The late 1990s saw something of a backlash against what was seen as the over-rigidity contained within the existing process and process improvement models and this culminated in the arrival of 'agile' methodologies or agile methods (Beck, 2000; Cockburn, 2002a). A methodology can be defined as "a collection of methods, procedures and standards, that defines an integrated synthesis of engineering approaches to the development of a product" (Zahran, 1998).

Though possessing different scope and objectives, agile methodologies are often compared directly with processes based on process improvement models (Boehm and Turner, 2004) and are frequently called 'agile processes' (Lycett *et al.*, 2003; Cohn and Ford, 2003). To clarify the differences between CMM/CMMI-based processes and agile 'processes', the label 'plan-driven' has been applied to processes based on the so-called 'disciplined' models, such as CMM/CMMI, to clearly distinguish them from the agile family (Boehm and Turner, 2004). As this study will show, this distinction is more blurred amongst industry practitioners who regard process models and agile methods as effectively the same thing. This is understandable as even some leading agile proponents, (Cockburn and Highsmith, 2001), use the phrases 'agile development', 'agile processes' and 'agile methodologies' interchangeably in their article. As this study examines software process in practice, and practitioners are not making clear

distinctions between process models and agile methods, both approaches are treated as equally valid alternatives to software development within this study.

Agile methodologies incorporate a family of related models including XP (Beck, 2000), Crystal (Cockburn, 2002a), Scrum (Schwaber and Beedle, 2002), Feature Driven Development (Palmer and Felsing, 2002) and Adaptive Software Development (Highsmith, 2000). In order to unify these methods around a common philosophy, the methodologies' founders, and others, dubbing themselves the Agile Alliance, produced the Agile Manifesto (Cockburn, 2002a). The Agile Manifesto contains four principles:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan.

As a unit, agile methods emphasise the role of people in software development. They favour continuously tested software, delivered through frequent releases, and place less emphasis on documentation. The methodologies also anticipate requirements change during the development cycle.

Since their inception, the use of agile methodologies has been accompanied by claims of higher customer satisfaction and improved delivery velocity (Beck, 2000; Cockburn, 2002a; Rising and Janoff, 2000). XP is the most popular and widely recognised methodology in the agile family and has by far the greatest coverage of any of the agile methodologies in the literature. It is also the agile methodology most widely deployed by the companies interviewed for this study and therefore will be reviewed in detail here.

3.11 Extreme Programming (XP)

Developed as a reaction against long development cycles and the failure of traditional software development approaches to meet customer requirements, XP purports to reduce delivery time and increase customer satisfaction. To achieve these objectives, it employs

iterative techniques, encourages active customer involvement during development, and allows for requirements changes throughout the production cycle. XP incorporates 12 key practices (Beck, 2000):

- *Planning game* – Used to determine the content and scope of system releases.
- *Small releases* – Release working versions of the system on short cycles.
- *Metaphor* – The collective vision of how the system works.
- *Simple design* – Produce the simplest design possible to satisfy requirements.
- *Test-first* – Tests must run successfully, prior to continuing code development.
- *Refactoring* – System restructuring to simplify, reduce duplication or aid communication.
- *Pair programming* – All code is written by two developers at the same machine.
- *Collective ownership* – Team ownership, all are empowered to make changes.
- *Continuous integration* – Build and integrate the system many times daily.
- *40-Hour week* – Limit overtime to reduce tiredness and potential mistakes.
- *On-site customer* – Ensure that a customer representative is available at all times.
- *Coding standards* – Agree conventions at the outset and ensure adherence.

The 12 practices focus on customer and developer interaction at all stages and the development team itself is empowered to make decisions on the product and the feature content of iterations. Functionality is delivered early to the customer and simple design and refactoring mean that requirements changes can be easily handled even late in production.

3.12 Analysis of XP

XP has made significant inroads into software development departments, and now even has its own international conference. Glass (2001) believes it has evolved into "a near religion". A number of authors have reported on using XP in a variety of environments including embedded systems (Grenning, 2001), web development (Grossman *et al.*, 2004; Murru *et al.*, 2003), event driven systems (Rasmusson, 2003), biotech systems

and project management (Sliger, 2004), and with legacy applications (Chapin, 2004; McAnallen and Coleman, 2005). Lippert *et al.* (2003) have used XP in conjunction with large, complex projects. They explain how they adapt the method to each development situation as needed and develop new process features for use with it as necessary. All of the studies listed above have used scaled-down versions of the methodology in their applications. Aveling (2004) analysed a number of XP case studies which also involved limited implementation of the method's practices. He reports that, "partial adoption of XP is more common than full adoption" and that the practices that were most difficult to implement within the studies were 'System Metaphor' and those requiring significant customer input, 'On-site customer', 'Planning Game' and 'Small Releases'. He concludes that it is possible to deviate from complete XP and still enjoy the benefits afforded the method. These results clearly show that full implementation of XP is not widespread and that companies are tailoring the XP method to suit their own particular environment. This is consistent with process models and process improvement generally in that certain contextual factors may influence which aspects of the process model are suitable and which are not.

3.13 Linking the Models

A number of studies in the literature detail how the various process and SPI models integrate or relate to each other. Paulk (1995) compares ISO 9001 to the CMM. He distinguishes between both models by stating that ISO 9001 identifies the minimal requirements for a quality system whilst the CMM underlines the need for continuous process improvement. On the key issue of equivalence between the two models, he suggests that an ISO 9001-certified organisation would satisfy most of the Level 2 CMM processes and many of the Level 3 areas also but could still ultimately be assessed at Level 1, as ISO 9001 doesn't address all the CMM practices. In terms of CMM mapping to ISO 9001, he feels that a Level 3 or even 2 organisation could be ISO 9001 compliant but would have to address several small issues contained in ISO 9001 to be assured of certification. Pitterman (2000) also discusses how ISO 9000 and CMM can successfully cohabit. He explains how his organisation pursued quality improvements in their software development activity initially through ISO 9000

accreditation and then through CMM certification. ISO registration was not only pursued for quality certification but also for international marketing purposes. To cement the benefits from the ISO certification, the CMM initiative was then undertaken because of its focus on continuous quality improvement. Coallier (1994) examines how CMM and ISO 9000 interrelate. He believes that CMM offers much more to the software developer than ISO as, when certified under the latter, the only challenge with it is to remain certified. By contrast, Demirors *et al.* (1998) illustrate how, in a software company, they used some of CMM's key process areas as the basis for an ISO 9000 quality system.

More recently, researchers have been looking at merging ISO 9000 demands with those of XP. Namioka and Bran (2004), propose that the key to merging the two is to treat both the developed functionality, as required by XP, and the written documents, as required by ISO, as deliverables at the end of an iteration. This allows XP to conform to the documentation requirements of ISO 9001. Melis *et al.* (2004) describe a tool that can support both XP and ISO 9001. Paulk (2001) examined how XP can be used to support CMM-based SPI. Though both have a focus on organisational culture, XP concentrates on technical activities whereas CMM's emphasis is on the management side. Paulk believes that XP lacks an institutionalisation element (an ability to ensure gains are spread company-wide) and does not scale successfully for use with larger projects. However, he maintains that, rather than being in conflict, the disciplines associated with the methodology make it complementary to CMM.

3.14 Summary

This chapter examined the role of software process and SPI in software development. A number of process and process improvement models were presented and appraised. Both ISO 9000 and the software-oriented CMM and CMMI were critiqued. The leading agile methodology, XP, was also analysed and how the various models link together was also investigated. The next chapter presents the research methodology chosen for the study and the reasons for its selection.

Chapter 4 Research Methods and Study Methodology

4.1 Introduction

This chapter presents an overview of the various 'schools' of research and the breakdown between quantitative and qualitative research methods. It presents a number of different qualitative research methodologies and describes their use in software development research. A detailed account of grounded theory, the methodology selected for use in this study, is provided and the chapter concludes with an outline of how grounded theory has been used in software development and IS research.

4.2 Levels of Research

Fitzgerald and Howcroft (1998), subdivide research activity over a number of different levels. At the key, epistemological, level, they detail two essentially different approaches, *Positivist* and *Interpretivist*. The *positivist* approach is denoted by cause and effect beliefs where there is a clear emphasis on objectivity, repeatability, and measurement. The competing *interpretivist* view is characterised by the belief that there are no universal truths, that theories are drawn from the researcher's own frame of reference and that awareness of context is a key criterion in illumination. Goulding (2002) believes that both positivism and interpretivism, "have their strengths and weaknesses, and their place in the research process, whether used alone or as complementary tools for generating valid and valuable knowledge".

The literature often uses the terms research *methods* and research *methodologies* interchangeably, so it is useful at this point to distinguish between them. Research methods primarily relate to the tools of data collection such as surveys and structured interviews whereas research methodologies refer to the overall paradigm that underpins the research (Blaxter *et al.*, 2001). In attempting to solve the research problem, the researcher has a number of different methodologies to choose from which are in essence linked to the positivist and interpretivist paradigms. Methodologies are classified under two general research headings, Quantitative and Qualitative, and the choice of which to

use will depend on the particular research problem, the objectives of the research, the strengths and weaknesses of the methodologies themselves, and contextual factors involved.

4.3 Quantitative Research

Scientific enquiry, which employs quantitative research methods, is used to establish general laws or principles (Burns, 2000). It is essentially concerned with four characteristics associated with the research (Burns, 2000):

- Control
- Operational Definition
- Replication
- Hypothesis Testing.

Control is used in experiments in an attempt to provide unambiguous, uncontestable answers to the research question. For example the influence of multiple variables within a particular experiment is controlled in order to isolate individual causes. *Operational Definition* describes how terms are defined by the steps used to measure them. *Replication* determines the reliability of the data emanating from an experiment such that if the study were repeated, either using the same approach, or carried out by someone else, or conducted at a different time, the same results would be obtained. *Hypothesis testing* refers to how the researcher systematically creates a hypothesis and then, to prove or disprove it, subjects it to an empirical test. This type of scientific approach is also known as hypothetico-deductive or logico-deductive, as it is based on hypotheses extracted from known theory which are then tested through the experimental method used. According to Jankowicz (1995), scientific or hypothetico-deductive methods incorporate the following components:

- A formally expressed general statement which has the potential to explain things: the theory
- A deduction that, if the theory is true, then you would expect to find a relationship between at least two variables, A and B: the hypothesis

- A careful definition of exactly what you need to measure to observe the variances in A and B: the operational definition
- The carrying out of the observations: the measurement
- The drawing of conclusions about the hypothesis: the testing
- The drawing of the implications back to the theory: the verification.

Researchers who believe that the knowledge accruing from this type of scientific enquiry is the only valid form of knowledge are often referred to as 'positivists'. The quantitative research approach, used within scientific enquiry, can be described as "empirical research, where data are in the form of numbers" (Punch, 1998) or, alternatively, the "collection and analysis of data in numeric form... (which) is often presented or perceived as being about the gathering of 'facts'" (Blaxter *et al.*, 2001). Quantitative research is seen to be value-free and to report reality objectively. To achieve this there are five main accepted methods used in quantitative research:

- Social survey – Random samples using measured variables
- Experiment – The use of an experimental stimulus on a 'study group'. A 'control group' is also involved which is not exposed to this stimulus.
- Official statistics – Analysis of previously collated data
- 'Structured' observation – Observations are recorded on a pre-determined 'schedule'.
- Content analysis – Pre-determined categories are used to count the content of mass-media products (Silverman, 2000).

Quantitative research's scientific approach has advantages in terms of precision and control and can provide answers which have a provable base. However, if one wants to study human behaviour and the social and cultural contexts in which it functions, then the limitations of quantitative research become apparent (Myers, 1997). Attitudes, beliefs and other facets, which make up a richer picture of studied phenomena, are essentially excluded using quantitative approaches. Therefore, to analyse human behaviour another approach is utilised, qualitative research.

4.4 Qualitative Research

Qualitative research is directed primarily at collecting and analysing non-numeric data with the aim of achieving information depth rather than breadth (Blaxter *et al.*, 2001). It is concerned with explaining social phenomena and exploring the world in which we live and broadly attempts to answer questions such as:

- Why do people behave as they do
- How are attitudes formed
- How are people affected by events happening around them
- What are the differences between social groups and how are these manifest?

Where quantitative research is concerned with questions such as, how much?, how many?, how often?, qualitative research is linked with questions such as why?, how?, and in what way? Also where quantitative research operates in a *deductive* way, qualitative research operates in an *inductive* way. A *deductive* process begins with existing theory, uses this to draw some hypotheses, and through testing these hypotheses tests the theory itself. By contrast, *inductive* research attempts to gather explanation and meaning through the collection and analysis of empirical data. Saunders *et al.* (1996) describe it thus, "Where you commence your research project from a deductive position, you will seek to use existing theory to shape the approach which you adopt to the qualitative research process and to aspects of data analysis. On the other hand, where you commence your research project from an inductive position, you will seek to build up a theory which is adequately grounded in a number of relevant cases". Similarly, inductive-based research can also play an important role in the generation of hypotheses (Fitzgerald, 1998).

There is, however, some disparity regarding the paradigms which are appropriate for use within qualitative research. For example, whilst Guba and Lincoln (1994), believe there are four paradigms in qualitative research, 'positivism', 'post-positivism', 'critical theory', and 'constructivism', others, such as Orlikowski and Baroudi (1991) contend that there are only three, with 'critical' accompanying the 'positivist' and 'interpretivist' categories. Though the three paradigm model has support from Myers (1997), he

cautions that there is much disagreement as to whether they can be used successfully within the same study or even whether they are necessarily opposed.

Within qualitative research, data collection can be a time-consuming process. This is because qualitative data are collected through encounters with individuals, often on a one-to-one basis, via group interviews or through extensive observation. There is no one particular type of qualitative analysis. Instead there are a variety of approaches "related to the different perspectives and purposes of the researchers" (Dey, 1993).

4.5 Study Methodologies used in Qualitative Research

Excluding surveys, which can be used in both quantitative and qualitative work, there are a number of basic study methodologies which are used within qualitative research, primarily:

- Phenomenology
- Ethnography
- Case Studies
- Action Research
- Grounded Theory.

4.5.1 Phenomenology

Phenomenology essentially means the study of phenomena and is a method for describing the world in which we live. Phenomena may include, events, experiences, situations and concepts. Phenomenology operates where there is a lack of understanding and will endeavour, if possible, to provide explanation or otherwise illuminate or clarify. Because the emphasis in this study is on theory building rather than illumination and clarity, phenomenology was not considered as a suitable approach.

4.5.2 Ethnography

Ethnography, given its anthropological antecedents, is a methodology used in studies of cultures and people. The studies themselves are predicated on the premise that the

people in question have something in common such as, geography, tribe, religion or shared experience. Ethnography involves extensive fieldwork but from its anthropological base has now moved into mainstream social science activities. Goulding (2002) details a number of examples of ethnography being used in management research. However, as ethnography primarily provides detailed description, as opposed to theory, this option was not pursued further.

4.5.3 Case Studies

Case Studies feature in both qualitative and quantitative research. Within qualitative research, case studies relate to the in-depth analysis of a single or small group of units such as individual persons, a department, organisation or group of companies. As with ethnography, case studies are used to generate 'rich' description rather than theory and so would not comply with the research objectives set out in 1.2.3. Because of this they were not deemed suitable for this work.

4.5.4 Action Research

Using action research the researcher attempts to generate new knowledge about a social system whilst at the same time trying to change it. It is essentially interventionist and is often used, within their own workplace, by practitioners who have an interest in analysing and improving current practices. This study is concerned with theory generation which is primarily based on detailed interviews with software practitioners. It is not interventionist and the researcher is not acting as a participant. For these reasons action research was rejected as a suitable methodology for this study.

4.6 Grounded Theory

Grounded Theory was first established by Glaser and Strauss (1967). The theoretical foundations of grounded theory stem from Symbolic Interactionism (SI), which sees humans as key participants and 'shapers' of the word they inhabit. Grounded theory was created from the 'constant comparative' method, developed by Glaser and Strauss, which alternated theory building with comparison of theory to the reality unveiled

through data collection and analysis. The emphasis in grounded theory is on new theory generation. A theory, according to Strauss and Corbin (1998), is "a set of well-developed categories (e.g. themes, concepts) that are systematically interrelated through statements of relationship to form a theoretical framework that explains some relevant social, psychological, educational, nursing or other phenomenon." This manifests itself in such a way that, rather than beginning with a pre-conceived theory in mind, the theory evolves during the research process itself and is a product of continuous interplay between data collection and analysis of that data (Goulding, 2002). Figure 4.1 shows how a typical grounded theory study may be conducted.

According to Strauss and Corbin (1998), the theory that is derived from the data is more likely to resemble what is actually going on than if it were assembled from putting together a series of concepts based on experience or through speculation. As the objective with the methodology is to uncover theory rather than have it pre-conceived, grounded theory incorporates a number of steps to ensure good theory development. The analytical process involves coding strategies: the process of breaking down interviews, observations, and other forms of appropriate data, into distinct units of meaning, which are labelled to generate concepts. These concepts are initially clustered into descriptive categories. The concepts are then re-evaluated for their interrelationships and, through a series of analytical steps, are gradually subsumed into higher-order categories, or one underlying core category, which suggests an emergent theory.

Since the initial launch of grounded theory, the Glaser and Strauss alliance gradually separated until each was developing a different version of the methodology. First in 1990 (Strauss and Corbin, 1990), and in a follow-up (Strauss and Corbin, 1998), Strauss, now in conjunction with Corbin, created an updated version of grounded theory with extended coding systems.

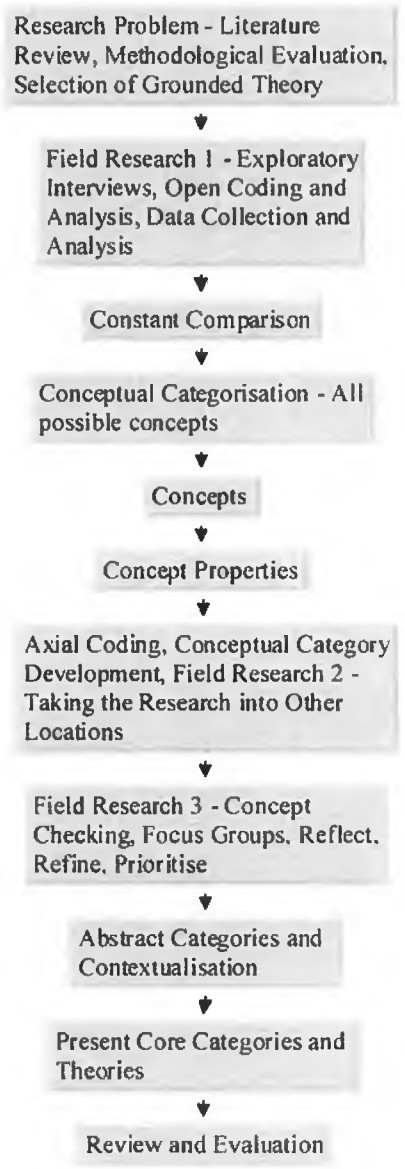

Figure 4.1 The Grounded Theory Research Process (Goulding, 2002)

This new implementation of the methodology drew criticism from Glaser (1992) for being formulaic and thereby forcing a theory from the data rather than letting the theory naturally emerge as suggested in the original incarnation. Some of Glaser's criticisms were acknowledged by Strauss and Corbin and were incorporated into (Strauss and Corbin, 1998). Glaser continues to argue in favour of being true to the original belief that the theory should "emerge" from the data and claims that Strauss and Corbin's

approach means, not a grounded theory but a "forced" description. Strauss and Corbin reject this saying the data "are not being forced; they are being allowed to speak".

Glaser, and Strauss and Corbin also differ on other fundamentals. Glaser believes that the research problem and question are only discovered when coding begins whilst Strauss and Corbin believe a question should be pre-set as it sets the boundaries around the study area. Similarly, Strauss and Corbin adopt a more pragmatic approach than Glaser by assuming the researcher enters the field with some knowledge of the phenomenon to be studied. Also the role of the literature separates the authors with Glaser believing that the literature should be largely avoided before study commencement for fear of creating prior assumptions whilst Strauss and Corbin recognise that there should be some pre-exposure to the literature which should be referred to as the need arises.

4.7 The Study Methodology and its Justification

As the objectives of the research, as specified in 1.2.3, are associated with generating theory, which is built on the 'voices' and 'experience' of software practitioners, a qualitative approach was chosen as the appropriate methodological vehicle for the study. Also, the study setting is indigenous Irish software companies and a particular strength of qualitative research is its ability to explain what is going on in organisations (Avison *et al.*, 1999). Of the qualitative methodologies available, grounded theory offered the best mechanism for achieving the research objectives. The reasons grounded theory was chosen are as follows:

- Given the lack of an integrated theory in the literature as to why software companies are avoiding SPI models an inductive approach, which allowed theory to emerge based on the experiential accounts of software development managers themselves, offered the greatest potential.
- It has a set of established guidelines for conducting inductive, theory-generating research.

- It is recognised for its application to human behaviour. Software development is a labour intensive activity and software process relies heavily on human compliance for its deployment.
- It is an established and credible methodology in sociological and health disciplines (e.g. nursing studies, psychology) and a burgeoning one in the IT and IS arena. This study provided an opportunity to apply a legitimate and suitable methodology to the software field.

The founders of grounded theory have not only been concerned with the processes associated with social psychology but also with the conditions that give rise to these processes. Furthermore, like others who have applied grounded theory (Baskerville and Pries-Heje, 1999a; Hansen and Kautz, 2005; Power 2002), this study attempts to understand a dimension of software development in practice. From a software process perspective the role of individual actors, and their environmental surroundings and conditions, weighs heavily on how the process is practiced. Facilitating the gathering and analysis of those human experiences and the associated interrelationships with other human actors, coupled with situational and contextual factors, are particular strengths of the methodology.

It is incumbent on every researcher using grounded theory to indicate which implementation of grounded theory they are using. Though acknowledging and recognising the spirit of Glaser's original version of the methodology, this study has essentially employed the Strauss and Corbin (1998) approach. Strauss and Corbin argue that the researcher's prior 'experiential data', basically their personal or professional experience, is supportive of theory building and contributes to 'theoretical sensitivity', the ability to understand the data's important elements and how they contribute to theory. The experience factor is also highlighted by Fitzgerald (1998), who describes the concept of the "cultural insider", as one who has prior expertise or practitioner knowledge of the domain. Having been a software process consultant and professional software engineer for a number of years, the researcher's 'insider knowledge' offered potential benefits to theory building, and strongly supported the use of Strauss and

Corbin's version of the methodology in this study. The researcher's professional experience also provided a familiarity with the literature surrounding the study area thus supporting theoretical sensitivity.

Section 1.2, which outlines the research agenda for this study, also encourages the use of the Strauss and Corbin version of grounded theory as they favour setting the research question in advance of commencing a grounded theory study, rather than it being allowed to 'emerge' at the coding phase as advocated by Glaser. Finally, Glaser states that grounded theory produces hypotheses, which do not require validation or verification, and that this is a task which should be carried out by others. Strauss and Corbin believe that, through continual data collection and analysis, provisionally generated hypotheses can be tested. This research adhered to the Strauss and Corbin approach by generating interim hypotheses in Stage 1 of the study which were then verified in Stage 2.

4.8 Using Grounded Theory

4.8.1 Theoretical Sampling

One of the first considerations in a grounded theory study is the concept of theoretical sampling. Theoretical sampling refers to the process of collecting, coding and analysing data whilst simultaneously generating theory. Interviews, both formal and informal, are at the core of the data collection process. Because the grounded theorist doesn't know in advance where the theory is going to lead him, only the initial sampling can be planned.

For interviewing purposes, during the data collection process, the researcher may wish to prepare an interview guide. Based on the emerging theory, the researcher may change the list of questions asked to reflect more closely the emergent categories. A category is a "phenomenon, that is, a problem, an issue, an event or a happening that is defined as being significant to the respondents" (Strauss and Corbin, 1998). Subcategories offer additional explanatory power to the categories. Based on category development, the researcher might then choose to interview certain types of individual or seek out other

sources of data. As the concepts and categories continue to emerge, theoretical sampling becomes an ever-changing process. The researcher engages in 'constant comparison' between the analysed data and the emerging theory. This process continues until 'theoretical saturation' has been reached (Glaser and Strauss, 1967). This describes the situation whereby any additional data that is being collected is providing no additional evidence or new knowledge about the theoretical categories. Central to the Strauss and Corbin version of grounded theory are three coding procedures Open Coding, Axial Coding and Selective Coding.

4.8.2 Open Coding and Analysis

From the interview transcripts or field notes, the researcher analyses the data line-by-line and allocates codes to the text. The analytical process involves coding strategies: the process of breaking down interviews, observations and other forms of appropriate data into distinct units of meaning which are labelled to generate concepts. The codes represent concepts that will later become part of the theory. The codes themselves provide meaning to the text and may be created by the researcher, or may be taken from the text itself. A code allocated in this way is known as an in-vivo code. In-vivo codes are especially important in that they come directly from the interviewees, do not require interpretation by the researcher, and provide additional ontological clarification or context-description. From the initial interviews, a list of codes emerges and this list is then used to code subsequent interviews. At the end of the sampling process a large number of codes should have emerged.

4.8.3 Axial Coding

Axial Coding is "the process of relating categories to their subcategories (and) termed axial because coding occurs around the axis of a category linking categories to subcategories at the level of properties and dimensions" (Strauss and Corbin, 1998) and involves:

- Documenting category properties and dimensions from the open coding phase

- Identifying the conditions, actions and interactions associated with a phenomenon
- Relating categories to subcategories
- Endeavouring to determine how the major categories relate.

4.8.4 Selective Coding

Selective Coding is "the process of integrating and refining the theory" (Strauss and Corbin, 1998). Because categories are merely descriptions of the data they must be further developed to form the theory. The first step is to identify the central, or 'core' category around which the theory will be built.

As the core category acts as the hub for all other identified categories, choosing the core category requires a series of steps to be followed:

- The category must be central in that all other categories must relate to it
- It must appear frequently in the data
- Data are not forced – the emerging theory is logical and consistent
- As the category is refined it grows in explanatory power
- The category is sufficiently strong to provide explanation for contradictory or competing cases (Strauss and Corbin, 1998).

4.8.5 Memoing

Memoing is "the ongoing process of making notes and ideas and questions that occur to the analyst during the process of data collection and analysis" (Schreiber, 2001). Typically ideas which are recorded during the coding process, memos assist in fleshing out the theory as it emerges and are written constantly during the grounded theory process. Memos may take the form of statements, hypotheses or questions. In the latter part of the study, following extensive coding and analysis, memos become increasingly theoretical and act as the building blocks for the final report.

4.9 Evaluating Qualitative Research

Quantitative research evaluates its findings by measuring, through scientific approaches, the internal and external validity, and reliability of the results produced. *Internal validity* is concerned with how the study itself was conducted and that the results actually represent what they claim to represent. *External validity* refers to the degree to which the findings can be generalised, beyond the study, to environments similar to that in which the study was undertaken. *Reliability* refers to the extent to which the findings can be replicated or reproduced by another researcher undertaking the same study.

Evaluating qualitative research provides greater challenges than for quantitative research. According to Dey (1993), validity in qualitative research is based on that "which can be defended as sound because it is well-grounded conceptually and empirically". He argues that by annotating, linking, and categorising data, a sound empirical base is provided for the identification of categories and concepts. A complementary approach to ensuring validity and reliability in qualitative research, proposed by Silverman (2000), is the constant comparative method which is the essence of grounded theory. Using this approach the researcher should always seek out additional cases including contradictory evidence to his provisional hypotheses. Similarly, the researcher should determine the frequency of occurrence of the selected examples to determine the scope of the concept (Dey, 1993).

4.9.1 Evaluating Grounded Theory

As with other divergences as listed in Section 4.6, Glaser and Strauss also separated on the approach to verifying or evaluating a grounded theory. As this study is using the Strauss and Corbin version of grounded theory, their method of verification has been adopted. Strauss and Corbin (1990) list three sets of criteria for judging grounded theory. The first set is aimed at judging the grounded theory produced by the study and is captured under four headings:

- Fit – The theory must fit the substantive area and correspond to the data.
- Understanding – The theory makes sense to practitioners in the study area.

- Generality – The theory must be sufficiently abstract to be a general guide without losing its relevance
- Control – The theory acts as a general guide and enables the person to fully understand the situation.

Strauss and Corbin's second set involves assessing the adequacy of the research process itself by determining how the initial samples were selected and how they varied during the course of the study; how the major categories emerged and how they influenced the sampling process; how and when the core category was selected; and how hypotheses were derived and verified or rejected.

The third set relates to how well the theory is empirically grounded including: what variation has been built into the theory; what significance do the theoretical findings have; how well are the concepts and categories related and to what extent are the broader conditions which affect the phenomena studied brought into the theory.

Chapter 12 will explain in detail how these three sets of criteria have been used to evaluate the study.

4.10 Qualitative Research in Software Development

The use of qualitative research in software development studies has been more widely embraced within IS than within SE. The focus in software engineering studies on technological issues, and the associated extensive use of quantitative methods, has been criticised by Bertelsen (1997) who argues for the use of qualitative research in software engineering. He contends that as software engineering is a "socio-culturally, not a technically, constituted phenomenon" any research conducted "cannot be based exclusively on natural science approaches but must include a way to understand psychological, social, and cultural phenomena". Many of the references previously cited in this thesis such as (Buchman, 1996; Daskalantonakis, 1994; Dion, 1993; Herbsleb *et al.*, 1997 and Humphrey *et al.*, 1991) do indeed have a primarily quantitative technological focus. This researcher agrees with Bertelsen in believing that, to get an

accurate picture of SPI in practice, one must investigate beyond purely technological factors. However, much of the published work, which uses qualitative research methods and which explores issues beyond technology, resides in the area of IS. Therefore, to see what lessons can be learned for qualitative studies in software development, which address social and cultural issues in addition to technological factors, it is necessary to draw on experiences from IS research.

Hevner and March (2003) believe the goal of IS research is to support the application of information technology for managerial and organisational purposes. They suggest that IS researchers generally follow one of two approaches, the behavioural science approach, which views IS as a social science, and the design science approach which treats IS as a technical science. They believe that, taken in isolation, each approach incorporates dangers with behavioural science favouring theory development and ignoring technologies, whilst design science often focuses exclusively on technologies and neglects well-rounded theory. This leads them to argue for a synergistic model which incorporates both approaches.

Lee and Liebenau (1997) believe that qualitative research is required in IS because, "while there has been great success in applying natural science and engineering models to research into computer technology, they have been inadequate and inappropriate in explaining the human, group, organisational and societal matters which surround the use of information systems". Myers (1997) notes that there has been a move away from technological to managerial and organizational issues, and this, he feels, is responsible for an increased interest in the use of qualitative research. With regard to the methodologies used in IS research, Avison *et al.*, (1999) and Oates and Fitzgerald (2001) propose the use of Action Research whilst Fitzgerald (1998) and Wixon (1995) have used a combination of approaches in their IS studies.

4.11 Grounded Theory in Software Development

Because of its interpretivist emphasis, and its ability to explain socio-cultural phenomena, grounded theory has been primarily used in the fields of sociology, nursing

and psychology from the time of its establishment in the late 1960s. Since then, however, it has widened its reach into the business sector and latterly into the software development and IS fields, where it has been used to explain intentions, actions, and opinions regarding management, change and professional interactions. Silva and Backhouse (1997) support its use arguing that, "qualitative research in information systems should be led by theories grounded in interpretive and phenomenological premises to make sense and to be consistent". Myers (1997) believes that grounded theory has gained growing acceptance in IS research because it is a very effective way of developing context-based, process-oriented explanations of the phenomena being studied. Probably the best example of the use of grounded theory in the IS field is (Orlikowski, 1993). Her study showed how grounded theory could be used to explain the impact on two organisations that implemented CASE (Computer-Aided Software Engineering) tools to support their software development activity. The use of grounded theory in Orlikowski's study enabled a focus on the contextual issues surrounding the introduction of CASE tools as well as the role of the key actors instigating, and at the receiving end of, their adoption.

Since Orlikowski's ground breaking work, a number of researchers have used grounded theory to look at a diverse range of socio-cultural activities in IS and software development. Baskerville and Pries-Heje (1999a) used a novel combination of action research and grounded theory to produce a grounded action research methodology for studying how IT is practiced. Others have used the methodology to examine, the use of 'systems thinking' practices (Goede and De Villiers, 2003), software inspections (Carver and Basili, 2003; Seaman and Basili, 1997), process modelling (Carvalho *et al.*, 2005), requirements documentation (Power, 2002) and virtual team development (Sarker *et al.*, 2001; Qureshi *et al.*, 2005). Hansen and Kautz (2005) used grounded theory to study the use of development practices in a Danish software company and concluded that it was a methodology well suited for use in software development. In addition, an article based on this work's use of grounded theory has been accepted for a forthcoming edition of an international peer-reviewed journal (Coleman and O'Connor, 2007).

4.12 Summary

This chapter presented a number of methodologies suitable for use in a research study and in particular qualitative research approaches. It examined different types of qualitative methodologies and illustrated how they have been deployed in software development research. A discussion on grounded theory, the methodology chosen for this study, then followed and some examples were presented of the use of grounded theory in software development and IS. The next section Part II shows how the research was carried out and introduces the theoretical framework produced by the empirical work.

Part II Findings

Part II – Overview

Part II of this thesis is divided into 4 chapters which present the main findings of the research. Chapter 5 shows how the theory was developed through the different stages of the study. It describes how the research questions were expanded following the initial interviews and how the grounded theory contained therein emerged from the coding, memoing and categorising activities central to the methodology. The emergent grounded theory is summarised and shown as a network diagram which identifies the relationships between the major themes, core category, linked categories, and associated attributes. Chapters 6 – 8 then break the theory down into its constituent parts and present these parts individually. The theory network presented in Chapter 5 is also further exploded in chapters 6 – 8 to show in detail the operators which link each of the attributes and categories together.

The theory is based on two conceptual themes, **Process Formation** and **Process Evolution**, originally referred to as part of the study's central focus in 1.2.4, and one core theoretical category, **Cost of Process**. Chapter 6 on **Process Formation** investigates how process is formed in indigenous Irish software product companies. Chapter 7, **Process Evolution**, examines how and why the process subsequently evolves from its original formation state. Chapter 8, **Cost of Process**, focuses on the study's core category and links the two conceptual themes. It also offers additional explanation for why software development managers are reluctant to engage in SPI.

Finally, a note on presentation and convention. Throughout Part II, the conceptual themes and the core category of the generated theory are highlighted in bold. The theoretical categories are denoted in bold and italics and direct quotations from the interview participants are italicised and indented.

Chapter 5 Investigation and Analysis

5.1 Introduction

This chapter outlines how grounded theory was used in the study. It describes in detail the various study phases, as illustrated in Figure 5.1, and the outcomes at each point.

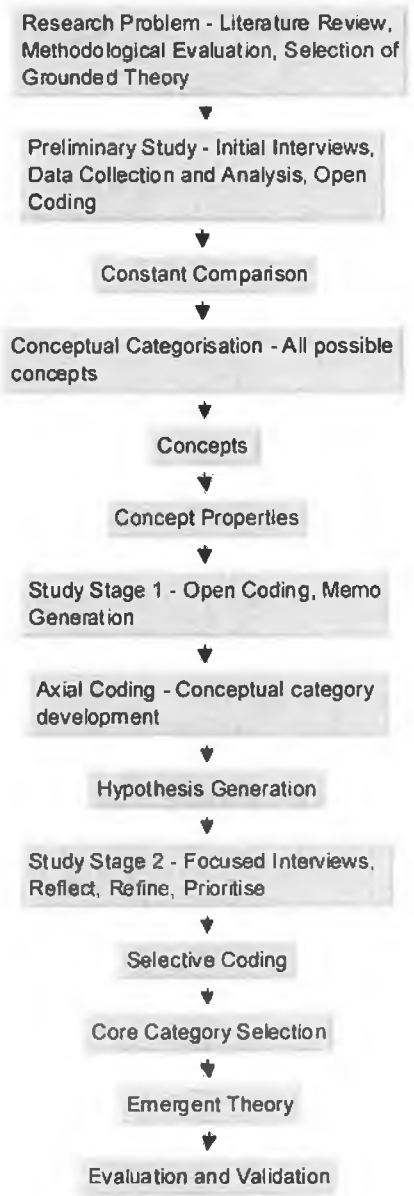

Figure 5.1 The Use of Grounded Theory in this Study

It presents the role of Atlas TI as the software tool used in data collection and analysis. The chapter illustrates how the Preliminary Study Stage and the subsequent study phases were used to derive hypotheses and theoretical categories. A detailed outline of the categories and subcategories developed during the data collection and analysis activity is presented and the chapter concludes by describing the theoretical framework generated by the grounded theory analysis.

5.2 Preliminary Study Stage

Despite the research questions from section 1.2.1 being clearly defined, the theoretical sampling approach of grounded theory means it is unclear in advance the number and types of practitioners that need to be interviewed to meet the research objectives. Because of this, a Preliminary Study Stage was embarked upon to generate more detailed information on how the sampling process should progress. Approaches were made to prospective participants, which resulted in four interviews with companies 1-3 as profiled in Table 5.1.

Company 1 was chosen as, within it, the researcher had several contacts including the CEO. The company is small (3 software developers) and has been in business for over ten years. This was a good company to commence with as their business history helped address a number of the research questions directly. The company has been in operation sufficiently long to have considered the issues around software process, has both expanded and contracted rapidly primarily due to the economic boom and subsequent downturn associated with the period 1997-2002, has been selected as a subcontractor by a major telecommunications multinational, and has been awarded ISO 9000 certification. The initial interview with the CEO lasted for over an hour. Because of the ease of access, and the diversity of his experience, a second person from company 1 was then interviewed. Interview 3 was also conducted in a company in which the researcher had a contact at senior level. This company has a larger number of software developers than Company 1 and has a presence outside of Ireland. Interview 4 was undertaken with a very small company where the CEO is personally known to the researcher.

Table 5.1 Company Breakdown by Category

Company	Market Sector	Interviewee	Total No. of Employees	No. in Software Development	Category ('S' – Start-up, 'B'- Build, 'E' – Expansion)
1	Telecommunications	CEO plus Dev. Manager	6	3	S
2	Company secretarial	Product Manager	50	20	B
3	Telecommunications	CEO	10	3	S
4	Telecommunications	CTO	70	30	B
5	Telecommunications	S/W Dev. Manager	12	6	S
6	Compliance Mgt.	Quality Manager	100	40	E
7	Enterprise	Product Manager	150	100	E
8	E-learning	S/W Dev. Manager	120	70	E
9	Information quality	S/W Dev. Manager	27	9	B
10	Telecommunications	S/W Dev. Manager	15	12	S
11	Telecommunications	CTO	160	110	E
12	Financial services	CTO	35	23	B
13	Financial services	Product Manager	130	90	E
14	Interactive TV	Project Manager	60	40	B
15	Public sector	Product Manager	150	90	E
16	Medical devices	CTO	19	9	S
17	Telecommunications	CTO	70	35	B
18	Public sector	CEO	3	3	S
19	HR solutions	General Manager	30	15	B
20	Games infrastructure	Product Manager	40	20	B
21	Personalisation	Technical Director	50	40	B

5.2.1 Using Grounded Theory in Practice

Section 1.2.3 indicated how the grounded theory created from this study would be based on the views and opinions of software practitioners and, throughout the study, semi-structured interviews were the method used to capture these. This approach is supported by Goulding (2002) who states, "with grounded theory the most common form of

interview is the face-to-face, semi-structured interview. This is favoured because it has the potential to generate rich and detailed accounts of the individual's experience. It should also be flexible enough to allow the discussion to lead into areas which may not have been considered prior to the interview but which may be potentially relevant to the study."

To support the semi-structured interviewing process a formal question set, Interview Guide 1 (Appendix A), was created for use with the first two interviews. The use of an interview guide within grounded theory studies is recommended by Schreiber (2001). The questions within the formal set were based on the researcher's experience as a 'cultural insider' and his prior familiarity with the literature. Within the formal set, there were 53 questions in total and these were divided over 4 categories: Company Background, Company Development, People Issues and Software Development Strategy. The motivation for this was to capture as much detail about the company as possible and to ensure no vital details were omitted.

The first interview was taped and then transcribed and printed. The interview was then coded, by hand, in accordance with the open coding procedure of grounded theory. Following this, a pair of scissors was used to cut the coded pieces of text into individual strips. Then, separate bundles of paper corresponding to identically coded sections were created. Memos were written as and when they occurred to the researcher during the coding. The second interview was coded in the same way as the first one, with the second being compared to the first and coded where possible according to the list of codes generated from the first interview. On completion there was an additional slew of memos but also an increasing amount of paper strips. As managing this increasing, and diverse, paper volume throughout the study was going to be impractical, a word processor was then used to manage the coding process, the links between the codes and quotations from the data, and any linking memos.

The first two interviews highlighted the fact that the formal question set, within the interview guide, had several drawbacks. Firstly, it was too long. Capturing all of the

information dictated by the questions took an inordinate amount of time and stretched the goodwill of the interviewees. Secondly, the software development questions, the part of the set which was of primary interest to the research, resided in the final section of the document. The length of time required to capture information related to the prior question segments meant that there was insufficient time, within the practitioner interview period, to explore the software development related material. Thirdly, in an effort to get through the full list of questions, some potential fruitful lines of enquiry had to remain unexplored. In this scenario, for example, the interviewee would say something which merited further examination but which could not be pursued because of the amount of material yet to be covered on the formal question set driving the interview.

Despite its limitations, Interview Guide 1 did provide some very valuable information which fed into the second question set, Interview Guide 2 (Appendix B). This second set of questions, of which there were 34 in total, this time across three categories, Company Background, People Issues and Software Development Strategy, was designed to be more fluid than the first set. Interview guide 2 also contained a list of memos, and guidance for questioning, which had been generated from analysis of interviews 1 and 2, and was in accordance with the grounded theory constant comparative approach. Interview Guide 2 was then used on interview 3 and was amended slightly for interview 4 in light of the analysis of interview 3. In each successive instance, the interviews and the line of questioning concentrated more on the memos and codes from the prior interview coding and analysis than on the formalised question set.

5.2.2 Preliminary Study Conclusions

The conclusion of interview 4 heralded the end of the Preliminary Study Stage, which was primarily used to drive the theoretical sampling process. The stage highlighted two issues in particular which would steer the immediately subsequent sampling activity. Firstly, the three companies interviewed operate in different target market sectors. However, even from this small sample the target market sector appeared to have an influence over the software processes the companies are using. This suggested that a

broad range of companies, operating in different markets, would need to be interviewed to determine the impact of this contextual factor. Secondly, it was obvious that a word processor was not going to be a practical way of managing grounded theory analysis; a specialist qualitative analysis tool, which supported coding and categorising, was essential.

5.3 Atlas TI

Having investigated the range of tools which are used for data management in qualitative research, Atlas TI (Muhr, 1997), a tool designed specifically for use with grounded theory was selected. Atlas allows for the linking, searching and sorting of data. It enables the researcher to keep track of interview transcripts, manage a list of codes and related memos, generate families of related codes and create graphical support for codes, concepts and categories. It also supports the axial and selective coding process as proposed by Strauss and Corbin (1998), which is used in this study.

Having installed the software, the interview transcripts from the Preliminary Study Stage were entered into the Atlas database. Having the ability to assign and allocate codes with quotations from multiple interviews speeded up the process dramatically and eased data management significantly. It also created an easier 'visual plane', which enabled clearer reflection and energised proposition development. A sample list of codes from this phase is contained in Table 5.2.

Table 5.2 Sample codes as assigned using Atlas TI

Absence of process	Automated documentation
Acceptance test process	Automated testing
Actual process Vs 'official' process	Background drives SPI
Admin heavy	Background of CEO
Adopt	Background of software development manager
Arduous	Baggage
Audit process	Beginnings of formality

5.4 Company Profile and Analysis

The research setting, as documented in 1.2.2, and relative organisation size as discussed in 2.4, outline the necessity to access a range of companies at varying stages of growth and age. Therefore, because of their greater software development history, and correspondingly greater experience of process evolution, it was decided that more significant emphasis would be placed on Build and Expansion companies rather than start-ups. Flood *et al.*'s (2002) leadership study of the indigenous software sector supports this focus on Build and Expansion companies, arguing that, "organisations with 30 employees or more have established management systems and structures". This researcher's view is that the "established management systems and structures" referred to by Flood, includes systems and structures to manage software development activity and are therefore of prime interest. On this basis, the target list of companies for the study was primarily composed of companies with more than 30 employees.

In addition, the finding from the Preliminary Study Stage that target product market also potentially had an influence on the software process used meant that the intended list of study companies should incorporate as many sectors as possible. A number of reference sources were used to compile the list including the Internet, trade magazines and yearbooks and professional/industry associations. In conjunction with this, the identity of the individual with responsibility for software process, within the identified companies, was sought. This resulted in a further 21 interviews across 18 companies and was conducted over two stages Stage 1 and Stage 2. Table 5.1 contains the breakdown, by category, of the 21 company subjects of the entire study. Of the 21 companies

interviewed, 6 (29%) are in the Start-up category, 9 (43%) are in the Build category whilst the remaining 6 (29%) are in the Expansion category.

5.5 Conducting the Full Study - Stages 1 and 2

5.5.1 Study Stage 1

In parallel with making contact with individuals known second-hand to the researcher, 'cold' e-mailing was used to set up the next series of interviews. The cold 'e-mailshot' proved surprisingly successful and generated a positive overall response rate of around 30%, which was much higher than anticipated. Study Stage 1 involved interviews with companies 4 to 14 (Table 5.1). Each interview lasted between one and one-and-a-half hours and the initial propositions emanating from the data analysis were used as general topics for investigation. Closely following the tenets of grounded theory meant that, following the initial open coding, the interviews were then re-analysed and coded axially across the higher-level categories that had emerged from earlier interviews. Any memos, or propositions, that emerged through the coding process were recorded for further analysis and inclusion as questions in subsequent interviews. A consequence of this was that the interview guide was constantly updated.

In conjunction with the theoretical sampling process, the constant comparative method was also used. This involved comparing interview-to-interview and searching for any themes or patterns in the data. Constant comparison assists in identifying concepts which go beyond description to explanations of the relationships within the data. By comparing the emerging facts for similarities or differences, broad categories, which possess multiple properties, emerge. Though a number of theoretical concepts emerged during the early fieldwork, the researcher decided to re-evaluate the study progress following the interview with Company 14. Despite the fact that similar occurrences were appearing within the data, straightforward analysis of the companies interviewed up to this point indicated a significant emphasis had been placed on the telecommunications sector. This was not a deliberate intention of the researcher but merely reflected the companies who agreed to be interviewed and the sequence in which

they occurred. Though there were no significant differences in the data emanating from the telecommunications software companies compared to other market sectors, in order to have real confidence in the emerging theory it was important to broaden the target company market. This approach is in accordance with both Strauss and Corbin (1998) and Goulding (2002), who advocate diversity in the data gathering and 'staying in the field' until no new evidence emerges. The researcher believed that to conclude the sampling process at this point would constitute premature closure, a mistake often associated with grounded theory (Glaser, 1978; 1992; Strauss 1987).

Because of the clear repetitions within the data, the memos and propositions created during the constant comparative process were further analysed by the researcher and a number of provisional hypotheses formulated (Table 5.3). These hypotheses had the potential to explain how the concepts and categories emerging from the study were linked. Strauss and Corbin (1998) highlight this possibility suggesting that, during the axial and selective coding phases, provisional hypotheses will naturally emerge as data are:

> *reassembled through statements about the nature of relationships among the various categories and their subcategories. These statements of relationships are commonly referred to as 'hypotheses' (and) the theoretical structure that ensues enables us to form new explanations about the nature of phenomena. These hypotheses must then be continually tested as more data becomes available.*

Occasionally, using grounded theory approaches, a set of hypotheses is often the main output of the study (Seaman and Basili, 1997). However, hypothesis testing can also be used within grounded theory to validate the theory that is emerging. The analysis of the results from 14 companies and the subsequent hypothesis creation, constituted the end of Stage 1. Study Stage 2 would be used to test these hypotheses and ensure the emergent theory was properly grounded.

Table 5.3 Study Stage 1 – Table of Provisional Stage 1 Hypotheses

H1	*The initial software development process used by Irish software product companies is based on the prior experience of the software development manager.*
H2	*The initial software development process used by Irish software product companies is tailored to suit the requirements of the target product market.*
H3	*Within Irish software product companies, SPI occurs as a result of positive and negative 'trigger' events*
H4	*The recruitment of external management expertise is used by Irish software product companies to solve positive and negative 'trigger' events*
H5	*The use of minimum process in Irish software product companies does not diminish the company's ability to satisfy its business objectives*
H6	*Within Irish software product companies, restrictions are imposed on team sizes to achieve minimum process requirements*
H7	*The use of XP practices satisfy an Irish software product company's minimum process requirement better than ISO 9000 or CMM/CMMI*
H8	*Development managers in Irish software product companies believe that by using XP practices they get more developer buy-in to process, than if using ISO 9000 or CMM/CMMI*
H9	*Non-ISO 9000/CMM/CMMI-certified Irish software product companies generate only minimum documentation*
H10	*Within Irish software product companies, adoption of ISO 9000 and CMM/CMMI is limited because of their emphasis on what development managers perceive as non-essential process elements*
H11	*XP is perceived by development managers in Irish software product companies to be more cost effective than ISO 9000 and CMM/CMMI*
H12	*The costs associated with achieving and adhering to ISO 9000 and CMM/CMMI prevent their adoption in Irish software product companies*

5.5.2 Study Stage 2

Strauss and Corbin (1998) strongly advocate the requirement to test hypotheses continually arguing that, "it is important that hypotheses be validated and further elaborated through continued comparisons of data incident to incident." The requirement to test these provisional hypotheses and the need to diversify the investigation into different market sectors drove the development of Stage 2 of the study. Taking the Strauss and Corbin (1998) approach, the constant comparative method was used to validate the hypotheses against the newly collected data. It is important to note that, the objective within this study was not to prove or disprove the provisional

hypotheses but, in common with other grounded theory studies (Orlikowski, 1993; Hansen and Kautz, 2005; Power, 2002), to use them to develop and saturate the core categories. These would then be used to produce a theory grounded in the experiences of the practitioners.

Study Stage 2 involved the participation of 7 new companies and comprised 10 further interviews. Three of the Stage 2 interviews involved re-interviewing Stage 1 participants. Companies 15 – 21 (Table 5.1) were the new subjects used for Stage 2 interviewing, whilst companies 7, 11 and 14 were the focus of re-interviews. Re-interviewing some of the original contributors is a technique available to grounded theory studies and is supported by Goulding (1999) who states that during theory development, "the interpretation should be presented to the original informants to ensure that it is an honest representation of participant accounts".

Building on the need for diversity within the data, the companies in Stage 2 came from different business sectors than those in Stage 1 and included application areas such as medical devices, public sector, computer games, and human resources. Only one of the companies interviewed in Stage 2 is involved in telecommunications software. Additionally, the researcher attempted to ensure that, consistent with Stage 1 and the research objectives, a company from each of the relative size categories, Start-up, Build and Expansion was included. Ensuring diversity such as this is consistent with grounded theory's validation techniques which include checking the development of ideas with additional observations, making further systematic comparisons, and taking the research beyond the confines of one topic or setting (Goulding, 1999).

During the Stage 2 fieldwork, the semi-structured interview questions were primarily derived from the Stage 1 hypotheses. Within the interview sessions, there was still some time devoted to capturing company demographic data but the prime emphasis was on testing the hypotheses. Because of this, the interviews had greater focus. Less time was spent exploring issues which did not directly relate to the hypotheses and greater effort was made to ensure the categories and subcategories were fully 'saturated'. Theoretical

saturation occurs when no new information about that category is revealed through further coding from additional interviews (Strauss and Corbin, 1998). When this point is reached is essentially a matter for the researcher as, if one looked sufficiently long and hard, one could potentially reveal more data. Goulding (2002) points out however, that:

> *there are no rules of thumb for when theoretical saturation is achieved, [but it] may involve searching and sampling groups that will stretch the diversity of data to ensure that saturation is based on the widest possible range of data. When similar incidences occur over again, the researcher may feel confident that the category is saturated.*

The combination of theoretical sampling and constant comparison ensures that an appropriately wide range of individuals and companies are interviewed which culminates in the core categories being saturated. Throughout the Stage 2 process, however, coding centred on the emerging categories and axial coding progressed to selective coding whereby the conceptual themes and the categories were developed further to the point of saturation. During Stage 2, full category saturation was reached after an additional 9 interviews as, in line with Goulding's (2002) assertion, similar incidences within the data were now occurring repeatedly.

5.6 The Emergent Categories

Where axial coding's role is to identify the categories into which the discovered codes and concepts can be placed, selective coding is used to explain the relationships between the categories to provide the overall theoretical picture. The objective of selective coding is to identify a key category or theme that can be used as the fulcrum of the study results (Strauss and Corbin, 1998). In this instance, the analysis showed that there was one central category to support and link the two theoretical themes. Furthermore, as the relationships were developed and populated, new categories emerged that were not explicitly covered by the hypotheses generated in Stage 1. The final list of themes, the core category and the main categories identified by the study are shown in Table 5.4.

Table 5.4 Themes, Core Category and Main Categories

Theme: Process Formation	**Category**
	• *Background of Software Development Manager* • *Background of Founder* • *Management Style* • *Process Tailoring* • *Market Requirements*
Theme: Process Evolution	**Category**
	• *Process Erosion* • *Minimum Process* • *Business Event* • *SPI Trigger* • *Employee Buy-in to Process* • *Hiring Expertise* • *Process Inertia*
Core Category: Cost of Process	**Category**
	• *Bureaucracy* • *Documentation* • *Communication* • *Tacit Knowledge* • *Creativity* • *Flexibility*

Each category and code can be linked to quotations within the interviews and these are used to provide support and rich explanation for the results. The 'saturated' categories and the various relationships were then combined to form the theoretical framework.

5.7 The Theoretical Framework

Within the Atlas TI software suite, the network function enables you to "express meaningful semantic relationships between elements" (Muhr, 1997). Using Atlas's network capability, therefore, allows the relationships between the categories to be visibly displayed. The use of diagrams with selective coding is recommended by Strauss and Corbin (1998) as they believe they "show the density and complexity of the theory" and that "in the end, it is important to have a clear and graphic version of the theory that synthesizes the major concepts and their connections". Following this approach, the theoretical framework generated by the study is shown graphically as a network in Figure 5.2.

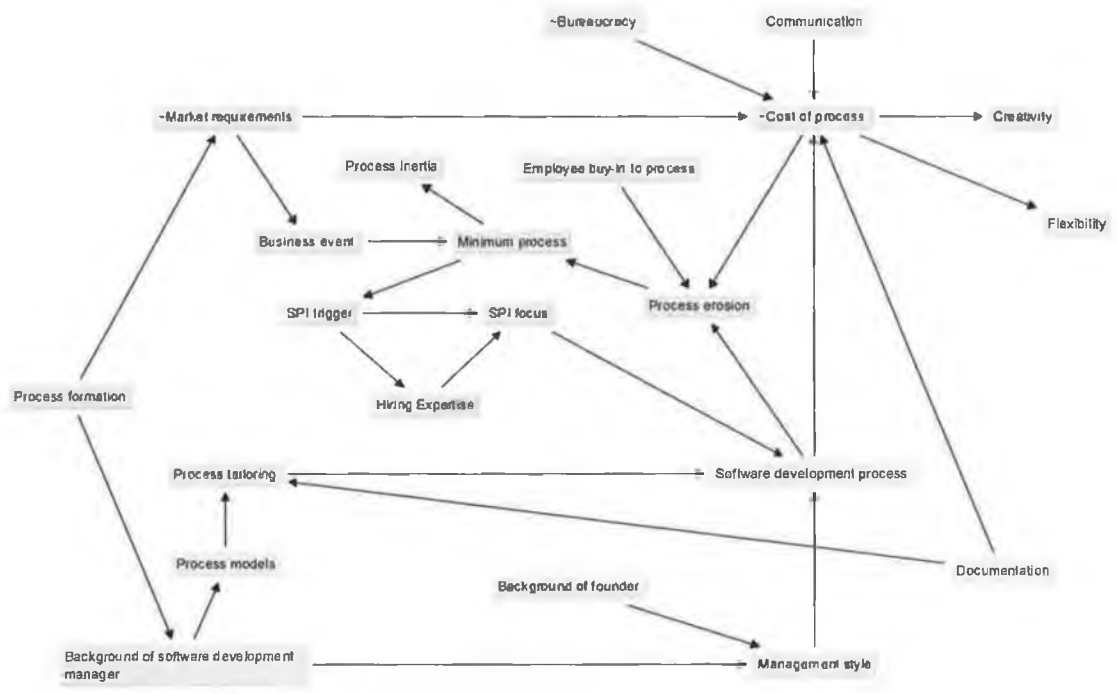

Figure 5.2 The Theoretical Framework

Within the theoretical framework, each node is linked by a precedence operator with the node attached to the arrowhead denoting the successor. No relationship types other than precedence are contained within the framework and the network is read from left to right. The tildes '~' represent codes in Atlas TI that were renamed or merged with other codes during the analysis process.

The root node of the framework, **Process Formation**, is a conceptual theme and is a predecessor of its two categories, ***Background of Software Development Manager*** and ***Market Requirements***. In the study companies, the title of the person with overall responsibility for software process, and ergo SPI, differed, from Software Development Manager to Chief Technology Officer (CTO), Director of Engineering, or Product Development Manager. For reasons of simplicity and clarity, the generic title Software Development Manager has been used in this study. The ***Background of Software Development Manager*** determines the ***Process Model*** used as the basis for the company's software development activity and this ***Process Model*** is then subject to

Process Tailoring. The ***Background of Software Development Manager*** coupled with the ***Background of Founder*** of the company creates an associated ***Management Style*** and this, in conjunction with the tailored process model, creates the company's initial ***Software Development Process***.

Software **Process Evolution** occurs as follows. Over time, the ***Software Development Process*** experiences ***Process Erosion***. The key causes of ***Process Erosion*** are the **Cost of Process** and ***Employee Buy-in to Process***. ***Process Erosion*** eventually leads to a ***Minimum Process***, which is the de facto operational ***Software Development Process*** until a ***Business Event*** renders it no longer sufficient. The ***Business Event*** causes an ***SPI Trigger*** and where the SPI activity is needed is the subject of ***SPI Focus***. Some companies seek experienced staff (***Hiring Expertise***) to solve ***SPI Trigger*** problems.

Following the SPI initiative, a new ***Software Development Process*** emerges. Soon after ***Process Erosion*** begins to recur and, as development activities begin to drift back to a ***Minimum Process***, some of the gains made during the SPI initiative are lost. The organisation then moves into a state of ***Process Inertia***, whereby it is apathetic towards any further process change. This continues until another ***Business Event*** causes the SPI cycle to repeat as described above.

Cost of Process is affected by a number of factors including ***Bureaucracy***, ***Documentation***, and ***Communication***. **Cost of Process** can itself then impact the organisation's ***Flexibility*** and ***Creativity***.

In creating the theoretical framework, several of the Atlas TI features were utilised. The 'Code Family' option allows codes, created from both the open and axial coding phases to be grouped together under a 'family' heading, for example, ***Bureaucracy***. This facility allowed the various interviews to be searched for passages where references to codes, which were classified as members of the ***Bureaucracy*** 'family', had been raised by the practitioners. Another feature of Atlas TI that was used in developing the framework was the 'Code Frequency Table'. This option shows how often codes

occurred within a particular interview, and across the entire suite of interviews, thus providing support for developing the more widespread categories. In addition to employing the 'code family' and 'frequency table' aids, Atlas's 'query tool' also provided major assistance with data analysis. The query tool contains Boolean and proximity operators which test for the co-occurrence of codes in the data. For example, a Boolean query can search for occurrences of Code A and/or Code B, whilst proximity can test the distance between, or 'closeness' of, code occurrences in the text. An example of a proximity query included examining the distance between practitioner references to CMMI and a subsequent reference to a code in the ***Bureaucracy*** category.

The next 3 Chapters will discuss each of the conceptual themes, the core category and associated sub-categories in detail and explain their position in the overall theory. Strauss and Corbin (1998) recommend the use of integrative diagrams which correspond to different parts of the theory. This approach has been used in the remainder of Part II as the theoretical framework is separated into each of its theoretical subsets. The diagrams for each subset are then exploded to provide more detail to each of the subcategories and associated relationships.

5.8 Summary

This chapter described how grounded theory was used during the course of data collection and analysis. Data collection and analysis was divided into three phases, a Preliminary Study Stage to determine how the theoretical sampling process should progress, Study Stage 1 whereby analysis of the data collected to date provided a series of hypotheses to be tested and Study Stage 2 where the hypotheses were tested to provide further detailed explanation to create the emerging theory. At the conclusion of Stage 2, the conceptual themes coupled with a number of theoretical categories and sub-categories were developed. The next part of the thesis will explore the nature of these, identify their properties, and describe the relationships between them.

Chapter 6 Process Formation

6.1 Introduction

This chapter takes the first conceptual theme of the study, **Process Formation**, and describes its categories and the relationships between them. It opens by discussing how software process is interpreted by software development managers and then shows how the ***Background of Software Development Manager*** and the ***Market Requirements*** are the key influencers on the initial software process in an organisation. Other factors such as the ***Background of Founder***, ***Management Style,*** and ***Process Tailoring***, which also help determine the initial process, are introduced.

6.2 What is Software Process in Practice?

In a software start-up, some form of process is at work in the organisation from day 1. In 3.2, a definition of software process was provided stating that it was a set of activities, methods, practices, and transformations used to develop and maintain software. However, 3.10 shows how there is confusion among researchers and practitioners as to their interpretation of a software process and the difference between a process and a methodology. This uncertainty was equally present amongst the industry participants in this study. When asked "what does software process mean to [your company]", answers varied.

> *It's some kind of putting a structure on developing code so that you end up with something that does the job.* [Company 3]

> *"Control". Controlling the project from start to finish ensuring that whatever you intend to build is what you actually build.* [Company 6]

> *It's really about the intersection points of the different functions. When am I going to hand you the baton?* [Company 8]

In every organisation, some process, albeit informal, is at work. In a number of the organisations the process itself wasn't visible or highly defined and could not be easily explained by the participants. Because of this the researcher focused more clearly on the "sequence of steps" or "sets of activities" described in definitions of process to get a clear picture of what process was followed within the organisation concerned. During the interviews, the software process being used within the companies represented a snapshot of what was happening in their software production at that particular time. However, software processes are also subject to change. Therefore, this research sought to address the two conceptual themes representing 'how did the process start out' (**Process Formation**) and 'what caused it to change and why' (**Process Evolution**).

6.3 Process Formation

In terms of how process is initially established within a software firm, the study found that there are two primary factors involved: ***Background of Software Development Manager,*** on which the company software process is based, and the ***Market Requirements*** which the software company must satisfy.

The ***Background of Founder*** of the company and the ***Background of Software Development Manager*** combine to create a ***Management Style,*** which also has an effect on the ***Software Development Process.*** These categories have a number of supporting attributes and the whole can be represented as a network, Figure 6.1, which is the expanded version of the **Process Formation** section of the Theoretical Framework denoted in Figure 5.2. The network links together the nodes through simple left to right precedence operators and culminates in the node ***Software Development Process***.

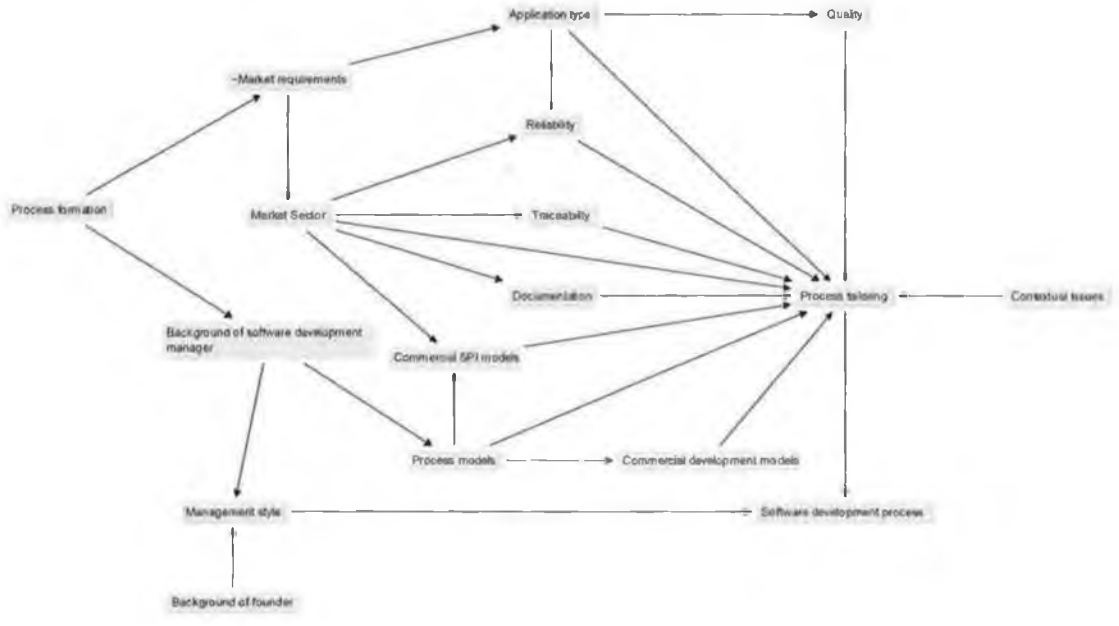

Figure 6.1 Process Formation Network

6.4 Background of the Software Development Manager

In some of the study software firms the founder has a software background and occasionally acts as software development manager. In other cases the founder has no software background with the result that someone who has the necessary expertise is hired to lead the software development effort. As might be expected, in many of the organisations interviewed, the original software development manager had left or moved on to a new position. In some instances, particularly in the smaller companies, it was possible to speak to the original software development manager. In other cases, it was necessary to speak to the person who hired or worked alongside the software development manager and who could provide the necessary process information. In the remaining firms there was a reliance on second-hand information from those close to the original process.

The majority of those interviewed had previously operated in a software development manager, or similar, role prior to joining their current company. From all of the interviews, it was clear that where the software development manager had worked

before, what their responsibilities were, what process and process improvement model was used, and the company culture, shaped the process that the software development manager used in their current company. The extract below, from company 8, is typical of the responses as to why a particular process model was used.

> *For software development we have used the RUP. The reason is that the guy we took in to head up our technology area brought that with him.*

If the managers had a prior positive experience with a particular process model and they understood it particularly well, then they opted for familiarity rather than something novel. This concept of bringing a particular model, or tool, with them was a common feature of the managers interviewed. The software development manager in company 11 also brought the RUP with him, the manager in company 12 brought XP to his current organisation whilst the manager in company 9 brought a commercial project management model.

6.4.1 Impact of Managerial Experience

In addition, all of the managers brought with them something less tangible, namely 'experience'. This is simply defined within this study as 'knowing what to do in a given situation'. One manager when asked about how he managed to grow the software development activity in his current organisation stated:

> *I guess a lot of it is our [previous company] experience because we understood what we needed to do when we got to a certain level.*

This factor was widespread across the interviews. The managers' knowledge, and the fact that they had encountered similar situations before, made them equipped to deal with the situations they found when joining their current employers. This experience included setting up a software process:

What the IT experience and the engineering experience gave me was the information as to what sort of processes I wanted to put in place and why I wanted them.

One company appointed a number of senior development staff simultaneously. They then used the backgrounds of all of these individuals to determine their initial process.

We sat down at the beginning and looked at what sort of environments have people worked in before, what sort of process did they have there and we tried to import them and tried to adapt them.

In a couple of the companies interviewed, during the start-up phase a senior developer was appointed rather than a software development manager. In very small organisations such as these, which have 1 or 2 person teams, the senior developer is effectively the software development manager. Subsequently, the practices used by the senior developer, created by their background experience, become the de facto initial process. This extract from Company 2 characterises it best:

In the early stages, when I was doing the development of the company system, I never had any functional specification. I didn't have any design documents. It was very much a one-man show. I would go out and talk to the clients. I might come up with a half-page document specifying any changes that they want. It wasn't reviewed by anybody else. I would just make the changes. And, many times it wasn't exactly what the client would want at all. It was just my idea of what they wanted and often we got it wrong.

But beyond the ***Background of Software Development Manager***, the impact of culture or more specifically ***Management Style*** also dictates how the process is implemented. This ***Management Style*** as it affects process, is either the style favoured by the software development manager or, as was often the case in the start-up companies, the style of the founder and the software development manager combined.

6.5 Management Style

6.5.1 Background of Founder

The company founders' backgrounds could be categorised as one of three different types, Information Technology (IT), Academia/IT, Non-IT (Table 6.1). It should be noted that those with an IT background were not all previously employed in the software sector. Also, those from the academia/IT background were essentially researchers within University IT departments who spun-off the company from research work. Those with non-IT backgrounds included a builder, engineer, teacher, geophysicist, TV executive, and HR executive. In a number of the companies (1, 4, 11, 12, 16, 17, and 21), the founder or co-founder was acting as Chief Technology Officer (CTO).

Table 6.1 Background of Founder

Background of Founder	Company
IT	1, 4, 5, 6, 10, 11, 12, 13, 15, 18
Academia (IT)	7, 16, 17, 20, 21
Non-IT	2, 3, 8, 9, 14, 19

6.5.2 Management Style and Process Formation

There was a sharp diversity between the ***Management Styles*** adopted within the different study companies. Some companies tend to be more enforcing of process allowing little deviation whilst others give the developers more latitude within it. During this study, whilst it was clear that ***Management Style*** helped the initial formation of the process, it also had an impact on how the process was implemented on an ongoing basis. From the extracts therefore, it was not possible to divorce completely ***Management Style*** issues at **Process Formation** from more recent management initiatives which influenced ongoing process adherence. From the study data, the key distinguishing factor in identifying the influence of ***Management Style*** on the formation of the process is company size, in that ***Management Style***, particularly that of the founder, was more clearly evident in Start-up companies. This occurs as, with fewer employees, the founder enjoys a narrower span of control and therefore has more day-to-day influence

over the process used. In addition, because of their maturity, Build and Expansion companies were in most cases further removed from the original ***Management Style*** of that of the founder and software development manager. Nonetheless, there was one excellent example from an Expansion company which showed how ***Management Style*** affected the initial software process and how it was managed.

> *A lot of that comes from the nature of the company. The company is based around its engineering team. Engineers have a lot of prestige and they get a lot of respect from [the CEO]. It's very difficult for management to come in and set the agenda. Because [the CEO] was the guy who originally wrote the code he never felt the need to put a strong engineering management team in place. He understood engineering, he understood software development, what's the problem?*

6.5.3 Management Approaches – 'Command and Control'

In three of the Start-up companies, the ***Management Style*** is very directive, which can be characterised for this study as a 'command and control' management approach. This type of 'command and control' style was illustrated by company managers who, closely supervised their staff, lacked trust in their staff's abilities, and made decisions without consultation. Some examples of how managers exercised 'command and control' follow.

Company 1 directed their staff on why they needed to follow SPI.

> *So we were telling people, "this [SPI] is for the growth of the company so it's for everybody's good to go along with it and embrace it".*

Company 3, one of the smallest interviewed, has a very 'hands-on' CEO who also adopts a 'command and control' ***Management Style.***

If a guy isn't delivering, we just don't want him in the company. You encourage him to leave or structure an exit for him.

However, this form of strict management was not confined to the smallest companies. Some of the larger organisations also had close management supervision of their developers. Company 9, which has reached the Build stage of growth, was typical.

If [process non-compliance] is happening constantly, then every week in the team meetings, it's highlighted that X didn't meet his objectives as he was fixing bugs in stuff he released last quarter. And to be honest it's a bit brutal but that's the way the process is and if you want to work here that's what you do.

Within the field data, there is clear evidence of a lack of trust in the developers by several company managers. The following represents some of the responses.

If you end up with process-type activity, which is purely known to the developers on the project, and is a language they speak among themselves, it becomes unhelpful, because it can be used as a defence for not getting things done.

There is a fear here of loss of control and power which is an element in the 'command and control' style. From the study, this fear is primarily confined to the smaller companies. It may be the case that as companies increase in size, more managerial staff are required to deal with the teams involved and the founder and software development manager realise they have little option but to allow others to take control thus challenging their fears with respect to delegating authority. But other managers also showed suspicion of developers within their teams as is evident in the following example.

And any process within the company shouldn't be designed to make software engineers' lives easier. If it does that as a by-product then that's fair enough but it should be designed to achieve business aims.

This posits the view that software engineers must conform to a business achieving its aims and therefore the team must be kept under strict control. In these 'command and control' cases the staff have very little latitude in how the ***Software Development Process*** is implemented. Limited process deviation is tolerated and adherence is closely monitored. From the interviews, more flexible and developer-centred development methods, such as XP, are held in suspicion by 'command and control' managers who wish to have project status visible and developers in some way accountable. One of the companies does this by making adherence to the process a factor in annual staff appraisal.

> *We have one person who has done a superb job, but the feedback from the Development manager is "I have to drag stuff out of him". So that will come up at a review meeting in that "you are doing a fantastic job but you are not helping your manager to do his job and clearly you understand there is an impact".*

Though ***Management Style*** has a major influence on **Process Formation**, there is no clear indication from the study whether adherence to process is greater in companies with this sort of directive style. Equally from analysis of the interview data, there is no evidence that these companies are more or less successful, in general business terms, than those with a more consensual management approach.

6.5.4 Management Approaches – 'Embrace and Empower'

In opposition to 'command and control' structures, many company managers, within the industry, operate what can be characterised for this study as an 'Embrace and Empower' regime. In this context, the opinions of subordinates are valued and included as part of software development policy. Also there is greater evidence of trust in development staff and their ability to carry out tasks with less direct supervision. Overall, there is greater delegation of responsibility, more participation by staff in decision-making, and, more generally, an environment where consensus prevails.

Company 6, one of the largest companies in the study, consults widely with its staff in relation to process usage. If a process change is considered necessary, each manager is consulted and they in turn solicit the opinions of their teams.

> *If our customers are recommending that we change code review, the manager goes away and sends an email out to all his department saying we are thinking of going this way, what do you think?*

Company 6 sells to the regulatory sector and requires very rigorous processes in its software development activity. From the outset it sought ISO 9000 certification status and a process to achieve this aim was put in place. Within the environment of regulation and certification, there is little room for process deviation, and all activities must be comprehensively documented and available for audit. The extract above shows that, even within a defined and rigorous process, the ***Management Style*** can encourage discussion and suggestions, which in turn allow the process to be improved or implemented differently. In this way the developers can have an influence over the process used and are more empowered than those working in 'command and control' companies.

XP, with its advocacy of self-empowered teams and shared ownership, is more associated with an 'embrace and empower' style of management. In this type of environment, managers trust their recruitment procedures and trust their employees. The style is much more 'hands-off' and suits XP. Senior engineers have more status in an organisation like this, as the extract from Company 12 shows:

> *If you have 1 guy working on a piece of consultancy with 15 years experience, he understands the principles of how we work. They know what they are doing. They don't need someone else pfaffing around. So at that point you may as well let them at it.*

This level of trust in the developers is in stark contrast to the 'command and control' approach taken by some of the other start-ups. ***Management Style*** is infused throughout an organisation. It affects the process either in 'command and control' fashion by relying on close monitoring to ensure that products are developed the way the senior managers want them to be or, alternatively, in 'embrace and empower' mode by entrusting the development team, and involving them, within broad limits, in how the products are to be developed. As companies grow, these ***Management Styles*** become less polarised, as those in charge early in the company's formation, especially the founder, have reduced influence. The evolution in ***Management Style*** is closely linked to one of the central themes of the study **Process Evolution** and the category ***Employee Buy-in to Process***. All of these linkages are covered in detail in the next chapter.

6.6 Market Requirements

The ***Market Requirements*** of the target market also have a fundamental effect on the establishment and use of the software process in an organisation. Software companies release products into specific ***Market Sectors***. Within this research, ***Market Sectors*** are treated as a subset of ***Market Requirements***. For example, almost all applications used by companies in regulated ***Market Sectors*** will have particular requirements and the nature of regulation means that the process used to create these applications must guarantee this. Other ***Market Sectors*** such as telecommunications also require applications which can meet high availability demands. However, ***Market Requirements***, such as a need for high ***Reliability***, or extensive ***Documentation***, or as is often the case, speed of delivery, can transcend multiple ***Market Sectors***.

6.6.1 Process and Regulation

Probably the best example in this study of *Market Sector* influencing the process occurred in the case of Company 6, whose products are bought by pharmaceutical companies operating in an industry which is regulated. Company 6 was formed specifically to sell to regulated industries and this meant that its processes had to cater for this from day 1.

The most important thing is the market we are producing to. We wouldn't sell without a good quality process.

Within the confines of this regulatory environment, Company 6 has very little latitude and ***Flexibility*** in the process they can use, as the companies to whom they sell must, under Food and Drug Administration (FDA) rules, audit their suppliers.

Because we produce for the pharmaceutical industry, every single client comes and does a detailed 2-day audit of us. They audit our software processes, the quality of our products etc. So when they in turn are audited, the FDA will say whom did you buy from? Did you do an audit of them and then they would have an audit report to show them.

This audit is conducted to satisfy regulatory compliance as the pharmaceutical companies themselves must show, not only are their own products compliant, but that how they are made also complies with the regulatory guidelines. It also means that the software producer in this sector must have appropriate ***Documentation*** for all its products for all stages of the development process. Any changes made during development must be recorded for ***Traceability*** and subsequent audit purposes. This imposes a rigour on the process which other companies may be able to avoid.

Company 16 operate in the medical space and business expansion plans will mean a move to a regulatory environment.

What we see is right now we developed the training product. And because it didn't require FDA approval, it allowed us to get the core software technology built and develop an early revenue stream before moving up the value chain into surgery where it did need FDA approval.

Company 16 have used XP as their development methodology up to now. However, they are aware of the fact that, as auditing may be a future fact of life for them, they are going to have to adjust their development process and methods within it if they are to satisfy the regulators. Had they been selling to the regulated market initially then their day 1 process would have had to take account of this, thus affecting the formation of the process.

6.6.2 Process and Application Type

Beyond regulated industries, the ***Application Type*** may require the system to be constantly available, thus placing a huge requirement on high ***Quality*** and ***Reliability***. Sectors such as telecommunications and banking can require such systems. Company 4 who develop systems for the mobile telecommunications domain best personify this.

> *Telecoms customers have different demands on quality and different demands on scalability. We had to deal with sustaining existing customers, penalty clauses on delivery dates and bug levels, and SLAs on services run on our product, and the sort of support requirements on that as well in terms of technical support.*

As the comment makes clear, this industry imposes penalties on late delivery and demands Service Level Agreements (SLAs). Failure to satisfy the delivery requirements can result in penalties and software failure means that SLAs are triggered. As a result any process, which produces products for this sector, must take account of this from its inception. This implies that careful estimating is done and that excellent ***Quality*** control and testing procedures are in place during development. The cost of poor ***Quality***, to a company in this sector, is extremely high. Another business area that has its own unique demands is the public sector. Several companies have experienced this with the following extract exemplifying things best.

> *Say with the system we are developing for the Garda. They have very strict documentation standards which we follow, and that involves a full functional spec, a full UML design, a very tightened development process, and a testing*

process. So in that case, they are putting certain demands on us, in terms not only of what we do, but the way we do it.

In summary, the above examples illustrate how market issues place an increased load on the software products. Both the actual code itself and the accompanying documents must be robust and fit for ***Quality***. Some attributes, such as ***Traceability***, are required across an entire sector such as pharmaceutical, which may require the process itself to be audited. Audits like this are used to satisfy the purchaser that the systems are professionally built. Companies operating in regulated sectors have moved towards ISO 9000 certification to provide a visible sign of this professionalism.

Elsewhere, for example, ***Reliability*** may be a key attribute where the product must perform 24 hours a day, for example in many telecommunications environments. In the case of the public sector it may be that a significant volume of ***Documentation***, such as, installation guides, operations guides, and user manuals, is produced to support the delivered products. In all of these instances, the process may require adjustment to provide the necessary services required by the market.

6.7 Process Tailoring

Though, in process terms, the software development manager brings with them a wealth of experience to their new organisation, some of that may have been gathered in organisations which were much different in nature, which means that some ***Process Tailoring***, to reflect their new environment, was necessary. Though that accumulated experience was used to benefit the new employer, it merited a change in the software process to reflect the new company's business environment. The process models used in the study companies were typically based on one of the standard industry development models, Waterfall or RUP, or the development methodology XP. The key word here is 'based' as none of the companies interviewed used the standard models in their entirety. All of the companies tailored the model, generally by dropping some of the practices contained within it and adding some new practices which reflected their own particular operating context. ***Process Tailoring*** such as this leads to a proprietary development

model which, although possibly based on a standard, is considered more suited to the company's business. Company 14 provides a good example:

> *We took the RUP and at this stage probably very little of our process resembles it. We have changed it around to suit our own needs. We are a small company as well. We didn't need all of the detail that was in that so eventually over time we have taken bits out and we have added in our own little bits as well.*

By contrast to the tailoring described above, very few companies are developing process models from a clean slate. All are influenced by the development manager's experience. Process elements are often then introduced piecemeal. One company 'experimented' with parts of the XP methodology. When they had determined the benefits, or otherwise, of the practices deployed, they moved on to use other elements of the approach. Companies are wary of adopting anything completely. There is a tendency to trial aspects or small subsets of the model. Some have done this, and finding the results not to be sufficiently positive, have rejected the entire model.

6.7.1 Process Tailoring – Influencing Factors

In every case however, ***Contextual Issues***, in addition to the ***Background of Software Development Manager*** and the ***Market Requirements***, were the main inputs to the tailoring process. It is important to recognise that when using process models, as part of process formation, most organisations scale down. Practices are routinely removed. Most of the process models concerned have been devised for generic use, but the software start-up has different demands to large, established organisations. Size and scale issues are to the fore, as start-ups will have small projects and therefore small teams, perhaps involving only 1 person. So following all of the steps within a model designed for large, complex projects, requiring multiple teams, is seen by these start-ups as wholly inappropriate. Company 12 put it most succinctly:

With most methodologies and approaches, very few stick to the letter of them and they are always adapted, so we adapted ours to the way we wanted it to work for us, for our own size and scale.

Despite its application to the initial software process a company uses, ***Process Tailoring*** is something that occurs throughout the lifetime of the organisation concerned. On every occasion that an improvement to the process is made, ***Contextual Issues*** act as inputs to the improvement process.

6.8 Summary

This chapter explained the role of the ***Background of Software Development Manager*** and the ***Market Requirements*** in **Process Formation**. These categories are theoretically linked to others including, ***Background of Founder***, ***Management Style*** and ***Process Tailoring*** to produce the initial software process an indigenous software company follows. A theoretical diagram was presented to support the analysis, and a range of opinions, from the managers interviewed, were documented as further theoretical support.

Chapter 7 Process Evolution

7.1 Introduction

This chapter explores how the software process changes in an organisation and discusses the factors which cause it to change. The chapter illustrates how existing processes erode over time leaving a working minimum. Furthermore, a reluctance on the part of managers to introduce process change leads to ***Process Inertia***. Process change is shown to be triggered by ***Business Events***, which can be either positive or negative. The chapter also details how the recruitment of external expertise is used to resolve these ***Trigger*** events.

7.2 Process Evolution – Overview

As Chapter 6 demonstrated process is formed from the ***Background of Software Development Manager***, and ***Management Style*** and tailored according to ***Contextual Issues*** and the ***Market Requirements***. The evolution of that process however, does not occur in a linear fashion. Examination of the study companies shows that **Process Evolution** is directly related to the events that the business experiences. These events contrive to highlight deficiencies, or insufficiencies, in the process which in turn drive process improvements. The theoretical network describing **Process Evolution** is contained in Figure 7.1. Reports from the managers within the study companies, show that over time, essentially for reasons of cost, there is a diminution in the application of the software processes which have been adapted by the organisations. When the initial process is established, or when some SPI initiative is undertaken, there is a real urgency on the part of their sponsors to make them work. But, after some time in use, elements of the initial or updated process, including both activities and ***Documentation***, are not followed in the same assiduous way and routinely get dropped. Significantly, this is often done with the complicity of the management who introduced the process or process changes in the first place. The way the application of the development process diminishes in this way can be described as ***Process Erosion***, and this is used to signify

atrophy, or corrosion, in the practice of some of the process elements, primarily through underuse or neglect.

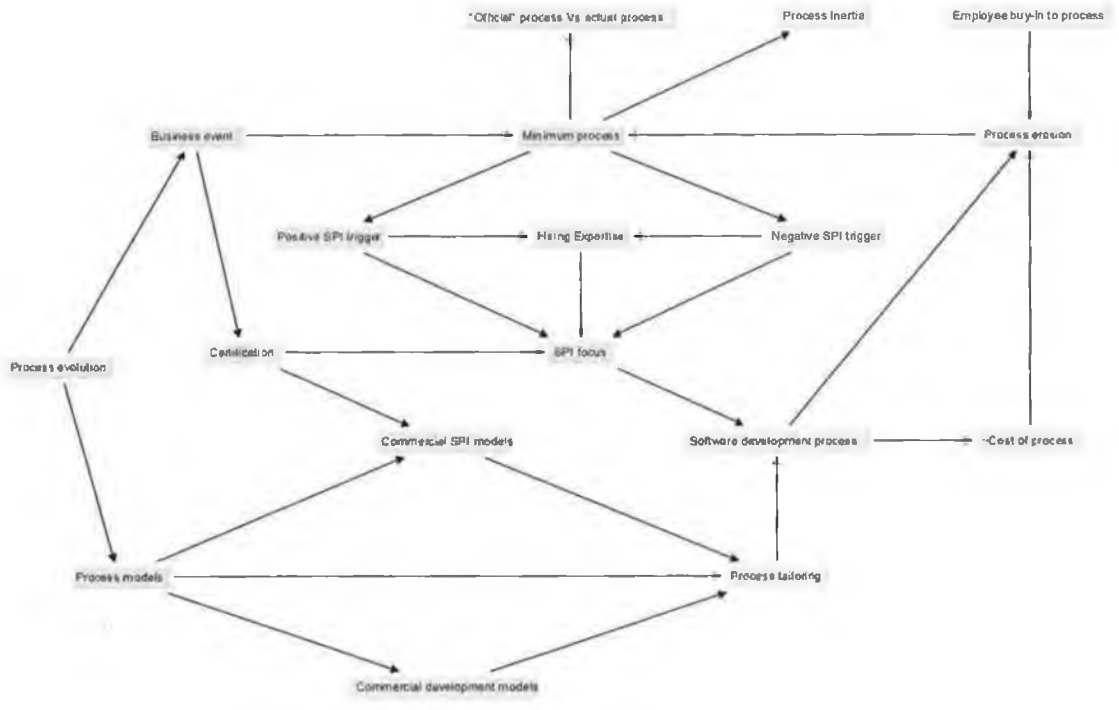

Figure 7.1 Process Evolution Network

7.3 Process Erosion

Analysis of the study data highlights a number of reasons why ***Process Erosion*** takes place. Process is initially established and tailored according to local requirements. When this process is improved, perhaps to cater for larger projects, a return to smaller projects often sees some process steps being omitted or set aside. Company 1 introduced ISO 9000 into its software development but had this experience after using it for a period of time:

> *For the first 2 years I'd say we followed it verbatim, and then we found out as you got through it some things aren't important that we initially thought were and now we've dropped off a little bit. Lately we probably haven't followed it as*

closely as we should, mostly because the projects we've had are small-scale projects.

In many cases, within the companies, size of project is the determining factor in relation to what process, or how much process, is used. Size of project has a number of corollary factors, for example, a smaller project generally requires fewer people to participate in it. Smaller projects also require less planning and management and, with smaller teams, will generally involve less ***Communication*** and ***Documentation***. Whilst all of these factors suggest a 'lighter' process can be used, what is happening in practice is that a subconscious decision rather than a conscious one is being made in relation to the process used. Furthermore, when practices get dropped they are often not reintroduced back into the process for subsequent projects. As Company 2 explains:

The test team don't write the test specification to the same degree that they would in the mid-'90s. We're not as strict on that now because we just don't have the time. There's so many different sub projects going on at the same time that in order to get the system testing done on all of them we have to cut some corners.

What is significant about this extract is that, not only is ***Process Erosion*** occurring, but it is also being done with complicity of management. There are other examples from the study where management complicity is at play. Company 21, when under delivery pressure, often ignore their own processes.

Unit tests are difficult because project managers usually want to get things out faster. Some of these things they see as being an administrative overhead. "Unit test documentation? The software is finished, let's just get it out there."

7.3.1 Process Erosion and Management Complicity – Tacit or Explicit?

The issue of management complicity with ***Process Erosion*** is a major one as it tacitly encourages development staff to ignore the practices that management have previously insisted upon. Examples from the study show that management complicity in this regard is both explicit and tacit, explicit in the sense that it is invoked, or at least encouraged, and tacit in the sense that management are aware that process steps are not being followed but do not take any remedial action. In the vast majority of cases where ***Erosion*** is deliberate, it is permitted for speed of delivery reasons. Company 2's experience here is typical.

> *We're still using the quality procedures, but we probably have slackened a little bit on it. In the earlier stages when we would do a design document, we would have all the team members giving their feedback on how that would impact on the system. Now we just don't have the luxury of having everybody around a table.*

But there are also examples of tacit acknowledgement of this form of ***Process Erosion***. The manager of company 11 tells it thus:

> *If we had left things where they were we would probably have got to the point of paring it down a bit. That would have been people deciding "we don't have to have meetings for every one of these things like it says in the quality doc."*

Similarly, the manager in company 21, whilst concerned about the level of ***Process Erosion*** in his team, is prepared to ignore it and leave it unenforced.

> *We don't have so much internal documentation. I think we are weak on that and I haven't hammered that home as hard as I should have. You pick your battles.*

The ***Process Erosion*** examples above demonstrate how, in the case of the study companies, following a process incorporates a cost, typically measured in time or effort,

and therefore in order to reduce cost, process itself is reduced. As a result of developer and management complicity in reducing cost in this way, the ultimate outcome is that development practices are reduced to an operational ***Minimum Process***. **Cost of Process** is discussed in greater detail in Chapter 8 but first the next section looks at how ***Minimum Process*** operates in practice.

7.4 Minimum Process

Taking the definitions for software process in 3.2, ***Minimum Process***, as applied to software development, is defined in this study as:

> *the least amount of activities, methods, practices and documentation required to develop and maintain software and its associated products that satisfies business objectives.*

It is important to note the difference between ***Process Tailoring*** and ***Minimum Process***. ***Process Tailoring*** is a conscious and deliberate effort to fashion a process, from a generic model, which takes account of local ***Contextual Issues***. As can be seen from the examples in 6.7, ***Process Tailoring*** typically involves removing elements from a larger process model. This tailoring down approach results in a process reduced in scope from the generic model. ***Minimum Process*** on the other hand results from ***Process Erosion*** and represents a further reduction of the already tailored, proprietary model. Some of the practices removed to make up ***Minimum Process*** are done so deliberately and with clear management consent, others less deliberately and with the tacit approval of management. Company 2 highlights how ***Minimum Process*** can occur in practice.

> *The configuration manager would be responsible for spot-checking the code to ensure that the variables conform to the naming convention. That's in place, though in recent days we have got a bit slack on that as well.*

Similarly, Company 17 uses a tailored version of XP. But even with XP, itself a 'light' approach, ***Minimum Process*** is followed.

It is only some of the elements of XP. We don't do pair programming for example. Also a lot of it is building the tests before you start the code, but they don't always do that.

Company 15 also provide an example of ***Minimum Process***, achieved by deploying more senior staff.

But definitely there is some other short-circuiting in terms of the top-down design, where we are applying a few experienced developers and short-cutting it. I would almost say there is a very ad hoc process in those cases.

Here, the company is deciding to use experienced staff to complete the task, to speed completion and is another example of where management are colluding in deliberately avoiding its own process raising the concept of an 'official' and an 'actual' process.

7.4.1 Official Vs Actual Process

Though there is no conclusive proof from this study, these examples, as with those relating to managerial trust in developers in 6.5, show that, in the companies concerned, process is something that is flexible and, depending on levels of experience, potentially ignorable by the developers concerned. It suggests that process is visible and clear, but elastic, and mainly for the use of more junior staff. However, it may also be the case that experienced staff in these organisations know which parts of the process can be eliminated and which are necessary to achieve the desired result, and this is the reason they are allocated to the tasks initially. From the extracts, the managers trust the developers to complete the task with or without the process. An additional aspect to this is that though the ***Minimum Process*** is being used, managers are content that a more detailed process is available if they wished to follow it.

[Code] check-in must have the reviewer's name against it as well as their check-in name. It's a fairly light process. We just use CVS, so we can go back and generate any reports that we want to, but we don't normally do that.

Company 11 concur admitting that there is the 'official' company process and the 'actual' company process, and which one is applied depends, to a large extent, on the individual developer.

If someone were to come in and say, "I am going to examine your processes because I am going to buy your software and I am going to check how you are doing that", I would say there is a compliance to it, and a veneer that says we are complying, and then there is the individual rigour and diligence.

What is especially interesting about ***Minimum Process*** is how it is acknowledged. All of the interviewees, very senior employees in their organisations, recognised that they were using the ***Minimum Process*** possible to achieve their objectives. They all had familiarity with the latest methodologies and at least some exposure to best practice SPI models. But as can be seen from the quotes above, pragmatic decisions on the processes employed were made to satisfy business objectives.

There is however, an additional linked issue which affects how 'minimum' the process is in practice and that is the level of ***Employee Buy-in to Process.***

7.5 Employee Buy-in to Process

Some of the study participants reported real difficulties getting the employees to follow the company's software process. Processes may be tailored correctly, and be appropriate for the company concerned, but if the employees ignore some of the process requirements then you have actual ***Process Erosion*** and a ***Minimum Process*** that is divorced from the 'official' process. In some instances it was a question of winning the employees over to a new way of doing things.

It took a lot of effort for a number of reasons. But getting everyone on board, getting everyone to understand the importance of it and be behind it, was quite difficult. And there's no point in having a system there if people aren't going to be 100% behind it and use it.

In other instances, the employees were hired as graduates. As graduates, these developers had no experience of working in other companies and therefore no experience of process other than the individual practices they followed.

Most of the employees we would have taken on didn't have any experience from any other companies. We had to do a lot of training and a lot of reviewing of how they were managing the system. It wasn't always easy.

In a number of cases the biggest issue rested with the senior staff. Having in most instances worked in other organisations, with their associated processes, and having built up what they saw as an expertise in software development, persuading them to follow a new process, or even adhere rigidly to the existing one, was difficult. In many instances they are often the staff most hostile to process and process improvement.

I have difficulty getting software developers to write their weekly status reports and send them to me. The better the developer, the less likely they are to send in the status report, if at all. The best ones are literally in mutiny.

Companies also experience a situation whereby engineers if they do not agree, or wish to conform to a process requirement, will engage in 'workarounds'. In other words they will bypass the process if they feel it is a hindrance or unsupportive of their objectives.

If you have to print off a document, and get it signed off, and bring it back, you are adding a lot of overhead. Typically engineers will see more elegant ways of doing something and if they are not allowed do it, you won't get the buy-in.

This is typical of what is experienced by companies who report that if engineers do not see the need for something, are unconvinced by the reasoning behind it, or just believe there is a better way to do it, then they either will do it a different way or not do it at all. Either way, the managers concerned will not get buy-in to the process.

All of the developer reluctance to follow process, and their recidivism in adhering to the required standards, results in additional ***Process Erosion*** with consequences for the companies concerned. In many instances, it is the senior staff who are the biggest process-avoidance culprits and who, by virtue of their position, suffer least censure. This can act as something of a catalyst for the less senior staff, who will follow suit if they believe others are avoiding process steps. Ultimately, the managers concerned retain the final sanction in this situation. However, this is clearly linked to the trust issues discussed in 6.5 whereby managers allocating senior staff to a task will ignore process transgressions as long as the task itself is completed satisfactorily.

On an ongoing basis, the ***Minimum Process*** used by an organisation will suffice as long as the operational conditions in which it is being used remain the same. However, what is happening in the study companies is that these operational conditions are not static and events occur which mean that the ***Minimum Process*** is no longer sufficient. These events generate an SPI 'trigger', wherein the resolution to the event necessitates an improvement to the software process.

7.6 SPI Triggers

As has been demonstrated in the preceding sections the ***Software Development Process*** erodes over time to leave a working minimum. However, company evolution and growth will give rise to a number of ***Business Events***. Some of these events the ***Minimum Process*** can deal with and some it cannot. Thus, the minimum software process used by a company is sufficient to achieve business objectives until a ***Business Event***, with which it cannot cope, renders it insufficient.

Business Events take a number of forms and can be both 'positive' and 'negative', meaning that positive events impact positively on the business and negative events, negatively. For example, a number of customers may complain to the software-producing organisation with regard to the ***Quality*** of a particular software release. Poor ***Quality*** is generally perceived as an outcome of process inadequacy. The poor ***Quality*** of the release, and the subsequent customer dissatisfaction, can therefore act as a ***Negative SPI Trigger***. But, ***Business Events*** can also act as positive ***Triggers***. One example of this would be where companies win large contracts whereby, existing ***Minimum Process***, which is used to dealing with small projects, cannot cater for major increases in desired functionality and complexity, with corresponding increases in team sizes and ***Communication*** requirements.

In any event the subsequent SPI initiative is the subject of an ***SPI Focus***. The nature of the ***SPI Focus***, which describes the SPI activity which will be undertaken to deal with the associated ***Business Event***, will be based on the positive or negative nature of the ***Trigger*** event. The study data contains numerous examples of both ***Negative*** and ***Positive SPI Triggers***.

7.6.1 Positive SPI Triggers

As marketing efforts generate new, larger customers, process changes are often essential to deal with the new business.

> *So, as you get progressively and significantly bigger deals, you also have to scale your development resources and group to be able to handle that. And a bigger team needs more process and then you have to kind of stop saying "look we all know what we are doing", and we now need a process to understand requirements, for doing QA etc.*

Though bigger and more lucrative projects may act as an ***SPI Trigger***, there may be corollary reasons why the process needs to be improved. Companies who act as subcontractors can be faced with ensuring their processes satisfy the organisation who

engages their services. In the example of a ***Positive SPI Trigger*** below, Company 1 was negotiating a contract with a Telecommunications MNC which wanted to see an auditable process in place. The establishment of such a process in Company 1, as recounted by the manager, was the key to securing the contract:

> *The [telecoms MNC] wanted us to have a quality system, which would be passed by their audit. At that stage we didn't have any formal documentation. We didn't really have any formal time sheets. We had a system that worked for us but it certainly wasn't what [the MNC] would have liked to see.*

7.6.2 Negative SPI Triggers

Despite the examples of ***Positive SPI Triggers*** above, the vast majority of process improvements reported by the study participants took place because of negative ***Business Events***. These ***Negative SPI Triggers*** took a number of different forms, but inadequate ***Quality*** was the most commonly reported issue. Company 2 explain:

> *Up to then we were selling to the Irish market and we realised people were coming back and they weren't happy even with the quality of the forms we generated from our software. There were typing mistakes and nothing was really tested as it should have been.*

Another example demonstrates how a poor initial understanding of the customer's requirements resulted in a substandard product.

> *What happened was that the developers were doing the design and the programming. So they were reading a requirement and interpreting it totally differently to the way the customer wanted. And you can't move on to the next project because you keep pulling resources back because issues are getting reported by customers and they want fixes, so you are in a longer support and maintenance.*

As the customer repeatedly reports problems with the software, the team gets sidelined away from new development to a cycle of 'fix and patch release'. Difficulties like these are common in Start-up and Build companies. As they attempt to generate increased revenue through new product development, the existing customer base will demand software upgrades. With limited resources available to engage simultaneously in both new product development and support provision to existing customers, the presence of faults, in its installed code base, creates a critical situation for the companies concerned. Over time this negative ***Business Event*** is unsustainable and process improvements are essential to remedy the problem.

Process proponents, like those referenced in 3.7, who favour CMM and CMMI, often argue that improved process results in better estimates and increased likelihood of delivering on time and to budget. Without process, they claim, development is chaotic and processes are ad hoc, which can result in all targets being missed. Company 8 provided a prime example for SPI advocates.

> *When I came in it was absolutely chaos. This particular organisation was very immature as a development organisation in terms of processes and procedures. They worked 24/7. Forever. They never made dates. It always went 5 times over budget. No one ever knew the status of the projects, it was just chaos.*

This example shows a case where there are multiple ***SPI Triggers*** in place. Any of these 'negative' factors could necessitate SPI and act as a ***Trigger*** in its own right. In a scenario such as this the company would have to decide whether its ***SPI Focus*** would be on tackling one, or a combination of these negatives, in an SPI initiative. As a way of solving both ***Positive***, and especially ***Negative SPI Trigger*** issues, one particular mechanism was repeatedly used. This was the hiring of suitable expertise from an external source.

7.7 Hiring Expertise

Hiring Expertise was a calculated decision on behalf of the companies to recruit someone externally who had expertise in the problem areas. In all cases where it occurred, the companies believed they did not have a suitable candidate internally and took the view that the crisis was either caused by a collective failure on behalf of all the current employees or could not be solved from within. The experience of one of the Expansion companies summarises it particularly well. The ***Business Event,*** which caused the ***SPI Trigger*** in this case, was a breakdown in the ability of the company to release software fixes to customers experiencing problems.

> *The company had expanded a lot and it was taking in a whole lot of new graduates who had no industry background and no real experience of professional engineering.*

The manager recruited, in a link to the ***Background of Software Development Manager*** category, introduced elements of the RUP-based software process he had used in his previous employment as part of the solution.

> *Initially [we focused on] getting the basic management processes right. We also implemented elements of the RUP – the configuration management and the source code control part we implemented lock, stock and barrel.*

Company 8, whose powerlessness in meeting budgets and schedules was documented in 7.6.2, also hired the practitioner interviewed for this study as part of the resolution.

> *They knew they had to take decisive action to the way they were doing development. They hired me deliberately. It was strategic. I had already done a start-up so I had gone through the evolution of that chaotic first phase.*

On taking up her position, she in turn, believed that the necessary expertise to resolve the prevalent problems did not exist in house as the staff were young and inexperienced.

To bring the staff to an appropriate level would, in her opinion, have taken too much time and resources, so she recruited additional experienced staff.

> *To achieve our goals we brought in people; somebody with 10 years experience of running QA; somebody with 10 years experience of running a department; we had no time to do that with such a young department.*

One manager reported how he had previously been headhunted to solve a ***Negative SPI Trigger*** problem.

> *I have been employed in one company where they had dreadful problems with the quality of their software and I was brought on board to improve it.*

The ***Hiring Expertise*** policy pursued by the companies in the study was undertaken as the companies observed that management experience is something that cannot necessarily be 'trained' but is rather acquired over time. Also recruits who had 'done it before' were likely to have encountered the typical pitfalls involved in the activity concerned and, by buying this experience, the recruiting companies could avoid these potential traps. A Build company engaged in the financial services sector typify the experience of all the companies who used the hiring approach.

> *And as we became a product company, we set out specifically to hire people who had worked in a product environment and who had that expertise which none of our existing people would necessarily have had; we tried to get key experience.*

The hiring of expertise in this way can have major implications for the process. Linked to the ***Background of Software Development Manager*** category, new arrivals such as this at senior level affect **Process Evolution**. In the same way as the ***Background of Software Development Manager*** helps form the original process, new senior hires, recruited for their experience of 'having done it before', bring process improvements with them. The way they deploy their expertise, for their new employers, generally

changes the way the development activity and associated process operates, resulting in a new ***Software Development Process***.

7.8 Process Inertia

There was additional theoretical support for the ***SPI Trigger*** category in an unexpected way from a number of the managers interviewed. Many of the managers expressed the view that, whilst their current processes were working, even if not perfectly, they had no desire to change them. One of the expansion companies, when asked if they would soon be pursuing further process improvements, described it thus:

> *I suppose, let sleeping dogs lie. We seem to be able to respond well to customer demands and there hasn't been too many complaints from them so I suppose that the process that we're using seems to work.*

There is a clear link here to the ***SPI Trigger*** category as the manager states that the process will not change as the 'customers have not complained'. The inference is that if the customers did complain, effectively a negative ***Business Event***, then this would have been the ***Trigger*** to change the process. The result of this type of unwillingness to countenance or promote change is ***Process Inertia***. This inertia basically describes an apathy, or indolence, towards the software process as it is used in the organisation. It represents a situation where, even though a company might recognise that there are inherent weaknesses in the process as it is used, these weaknesses are not sufficient to necessitate change or generate interest in SPI. The following passage from a company going through a difficult business period best describes managerial indifference and disinclination towards pursuing SPI measures.

> *There is little or no interest in other processes at a low level and the managers including myself have little or no interest in even learning about other processes at this time. Everyone is pretty happy with the way things work and why change it?*

It is also very rarely the case that the engineers themselves request or demand improvements to the company's process. In most cases the managers instigated SPI but where management were indifferent to the processes used, or reluctant to investigate new approaches, SPI was not undertaken. Process improvement is seen by many of the managers as a necessary evil and a wearying and distracting activity. Companies report that they neither have the time, nor the resources, to examine process best practices. They report that, from their perspective, process relates to issues of scale, cost and how quickly they can respond to market demands or opportunities. Process improvement, they contend, contains an element of risk which, unless the situation is critical, is not worth taking. The stated risk is linked to an associated 'fear', on the part of the managers, of what might happen if the SPI objective is not met. The managers are apprehensive that 'toying' with a slightly imperfect process may negatively interfere with the normal functioning of the business. As a financial services company express it:

> *What process we have doesn't work in some areas, but overall it does. We are not trying to overhaul it radically because if I radically overhaul it, there's a good chance that will a) stop things for a period of time and b) break something that I will have to fix. It's not that broken!*

One manager summed it best for all of the companies by stating that the processes his company use are "good enough until they are not good enough". No appetite exists to pursue best practice, for what is seen as its own sake, or to receive potential future benefits. In conclusion, as a Build company manager argues:

> *The process seems to work so I haven't gone out of my way to change it. Something that would be giving us benefits 3 years down the line or is just good practice but is not really measurable in any way, just wouldn't happen.*

7.9 Summary

This chapter examined how software process evolves in an organisation. It described how ***Process Erosion*** caused the software process to be reduced to a working minimum.

The level of ***Minimum Process*** was also shown to be affected by how well employees buy-into using it. Process change was demonstrated to be predicated on ***Business Events*** which could be positive or negative. It was shown that where these could not be resolved by the ***Minimum Process*** in use that this triggered SPI activity to effect a solution. One of the methods used by companies to solve this problem, ***Hiring Expertise***, was discussed and the chapter concluded by showing how most companies are satisfied with their existing process and this reluctance to change leads to a form of ***Process Inertia***. The next chapter will discuss **Cost of Process** issues as they are perceived by the software managers in the study.

Chapter 8 Cost of Process

8.1 Introduction

This chapter examines the core theoretical category of the study, **Cost of Process**. This category provides theoretical support for the **Process Evolution** theme presented in Chapter 7 and has a major impact on the SPI cycle. This chapter describes the categories, including ***Bureaucracy***, ***Documentation***, ***Communication,*** and ***Tacit Knowledge***, which affect the **Cost of Process**. The impact of **Cost of Process** on an organisation's ***Creativity*** and ***Flexibility*** is also discussed. The chapter continues by providing a summary of the practitioners' perceptions of the SPI models, ISO 9000 and CMM/CMMI and concludes by analysing the circumstances in which companies would consider implementing each of these models.

8.2 Cost of Process – Overview

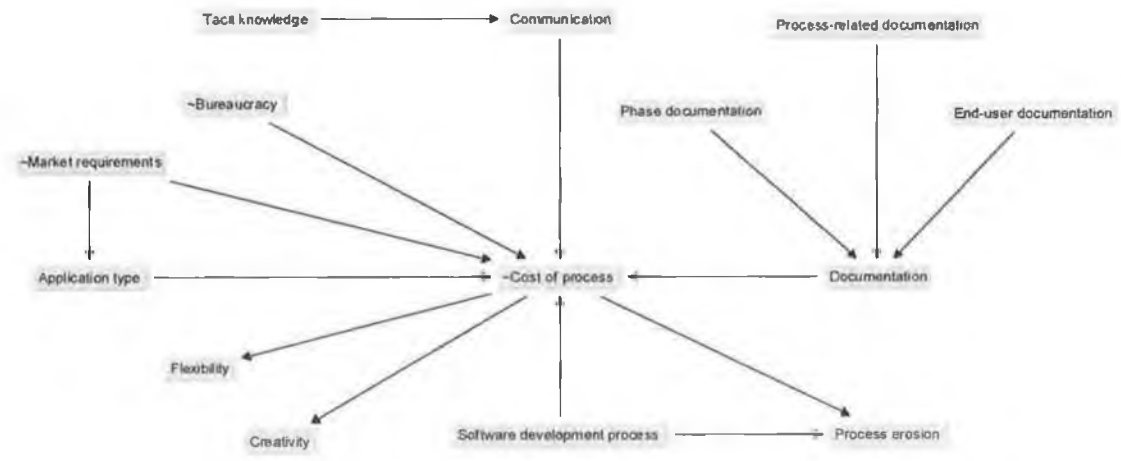

Figure 8.1 Cost of Process Network

In the course of the study interviews, few of the managers concerned expressed any enthusiasm about process or process improvement models. A far greater emphasis was placed on product, with process often believed to be a brake on product development.

The managers believed process to have a significant cost which, in their respective companies, they attempted to keep to a minimum. What the managers perceived as the **Cost of Process** centred on a number of factors and these are represented as a network in Figure 8.1. The key **Cost of Process** factors will now be discussed in detail.

8.3 Bureaucracy

The category ***Bureaucracy*** covers items including the time and resources which the managers in the study believe are required to administer and apply the software process used in their organisations. In essence, managers divided process into two separate categories, 'essential' and 'non-essential'. 'Essential' process was that which was most closely linked to the product; requirements gathering, testing and design. 'Non-essential' process, which in the view of managers could often be omitted, included process/quality-related documents and plans, software measurement, and even many management activities such as planning, estimating, and staging meetings. The interviews capture this in a number of different ways. Three separate managers described some process activities as a 'luxury' and not something essential to creating software products. The use of the word luxury is quite significant. Luxury is a synonym for extravagance, indulgence, or something inessential (New Oxford Thesaurus of English, 2001).

> *In the earlier stages when we would do a design document, we would have all the team members giving their feedback on how that would impact on the system. Now we have to bypass that because of time constraints. We just don't have the luxury of having everybody around a table.*

Another manager, this time using code reviews, had a similar view of process:

> *I've sat in development code reviews and seen a bunch of people discussing, not whether a particular block of code would work, but whether it was an example of good programming style and that discussion going on for 4 or 5 hours. It's wonderful to have the luxury to do it.*

In addition, Company 3 considered process definition not worth putting resources into.

> *The other key thing was resources. We didn't have the luxury of assigning someone, saying "look you go off and spend a couple of months designing and putting a process in place"... or even a day a week putting process in place.*

All three examples show that where managers believe they have limited resources they do not wish to allocate them to activities which, in their opinion, do not contribute directly to meeting product development deadlines.

Another belief of the practitioners is that following a process as it is defined is 'overkill'. A widespread opinion prevailed that there was an easier or less time-consuming way to achieve their objective and many were happy to ignore their own processes to do so.

> *If you're talking about 2- or 3-week projects which are sometimes what we're dealing with, it tends to be overkill to go through a full lifecycle and we had a number of ways of doing that, getting round that.*

The overkill described here is linked to the ***Management Style*** category described in 6.5, whereby managers are complicit in bypassing their own process believing that by eliminating some process steps, you also eliminate some of the costs.

The extracts demonstrate that many of the managers, far from being process converts, believe that many process activities are not essential and require too much time and resource. One of the process activities that managers consider can often delay, or hinder, product development, is ***Documentation***.

8.4 Documentation

Forward and Lethbridge (2002) define software documentation as "*Any artifact whose purpose is to communicate information about the software system to which it belongs, to individuals involved in the production of that software*". The managers interviewed for this study believe ***Documentation*** is one of the single biggest contributors to the cost of process. ***Documentation*** incurs a cost through the actual time taken to record the chosen information, but also through opportunity cost in that, whilst staff are engaged in ***Documentation***, they are not engaged in what management often see as more 'worthwhile' activities, such as coding. Reduced ***Documentation*** was associated with situations where managers had high levels of trust in their developers and their experience.

> *It comes down to experience, what are the key things to do. It's not about writing reams of documentation nor having huge heavyweight process.*

Across the interviewees, ***Process-related Documentation*** was seen as an overhead, which can delay development activity and whose merits, in many instances, can be difficult to convey to engineers.

> *So often, people were filling in time sheets and lists weeks after the project had finished in order that the quality process could be seen to pass its audit.*

Smaller companies, especially, feared having to allocate people, either to write the ***Documentation*** in the first place, or to manage it on an ongoing basis. Despite this there was an acceptance that, with growth, more formality in ***Documentation*** would be required. This was a matter of real concern as one manager explained:

> *With more people we would have to get involved in more administration, more recording and more documentation. And you could end up hiring administrators purely to document your processes and to ensure they are being followed.*

In accordance with the reticence to document, many managers linked improving the software process with the creation of additional ***Documentation***. This was a commonly held view and is discussed in greater detail in 8.9 and 8.10.

8.5 Communication

Because ***Documentation*** was seen by the managers as such a significant process cost, they believed that, if they could reduce their ***Documentation*** requirement, they could reduce the cost of their software process. Taking Forward and Lethbridge's (2002) definition of software ***Documentation*** from 8.4 - a way of communicating information about the software system to the individuals involved - many managers encourage verbal ***Communication*** as a way of sharing information and reducing the ***Documentation*** load. Within the study organisations, there is often conflict between explicit knowledge, represented by ***Documentation***, and ***Tacit Knowledge***, which is the undocumented, intuitive know-how of the individual or team. Recognising this, the companies in the study attempt to capitalise on exploiting ***Tacit Knowledge***, and verbal ***Communication***, and this is brought out in the practices they adopt. One company explain how they use simple ***Documentation*** and developer co-location to achieve knowledge sharing.

> *At that stage the product and project design was done on an A4 piece of paper and when something needed changing you could talk to the guy next to you because he knew what you were doing and you knew what he was doing.*

Informal ***Communication,*** and knowledge sharing, benefits from this form of team co-location. It can be achieved merely by having the entire team share a common office area. One manager described how his team profited from this arrangement:

> *In my team, we are sitting close together so I can just pull out the chair and start chatting and get interaction going. You've got people talking over and back. You just wouldn't have the same spontaneity in email.*

Even where the office layout doesn't support this sort of easy ***Communication***, it can often occur spontaneously, a factor brought out by one of the Start-up companies.

> *We're efficient in the way we apply whatever process we have and communication is fast and it doesn't require a big meeting, just a bunch of people talking in corridors until they get a problem solved and things move quickly.*

There is a conviction, firmly-held in the larger companies, that ***Documentation*** alone will not ensure that all team members have a shared understanding of a project's or business's requirements and that deficiencies here can be overcome through informal ***Communication***. By contrast, there is an acceptance in many of the smaller companies, that, though ***Tacit Knowledge*** and informal ***Communication*** is the norm, ***Documentation*** is necessary on occasion. This is exemplified by one of the companies who are engaging offshore third-parties to do some of their development work.

> *Before that the actual production component was done in house. So from a documentation point of view you weren't as tight because the person you wanted was next to you or down the corridor. So they could ask you "what exactly did you mean by that?" But now the quality of the documentation has to be spot on.*

Despite this, all of the larger companies, or those with a higher level of ***Documentation***, still report extensive informal ***Communication*** in their organisation.

> *For example, even the way we write the requirements spec, we still find ourselves in a lot of verbal discussions with people who are just trying to understand the background and the issues. And a lot of the outputs from those discussions don't get documented, they just get agreed.*

As a company grows, it often exploits ***Tacit Knowledge*** and informal ***Communication*** to reduce the ***Documentation*** overhead. Reliance on ***Tacit Knowledge*** is often implicitly

acknowledged through companies enabling and encouraging their experienced staff to operate without documenting their activities.

> *We have one senior techie who is looking after the 2 – 3 developers and has a really good focus on the architecture of the product. With three guys in a room, there is no need to do formal UML modelling, it can be done on a piece of paper.*

A support approach to ***Tacit Knowledge*** that companies often undertook, in an attempt to reduce ***Communication*** overhead, was to keep team sizes small. Small teams allow companies more potential to co-locate them, enable more informal ***Communication***, and obviate the need for greater ***Documentation***. The experience of one company shows how they aim to achieve this.

> *If you keep a team size small and the guys are all talking about what they are doing and describing it, discussing it, changing it around, there will be less need for them to refer to a document that they are all familiar with.*

Despite this, even the companies who use ***Tacit Knowledge*** extensively recognise that it has its limitations and may ultimately carry its own cost. This is especially true of those companies who are using XP and who worry about the emphasis on informal ***Communication*** at the expense of ***Documentation***. This is explored in greater detail in 8.7.3. In addition to carrying a ***Documentation*** load, process was also perceived by managers as having a negative impact on a development team's ***Creativity*** and ***Flexibility***, and this is discussed in the next section.

8.6 Creativity and Flexibility

Software companies, especially start-ups, need to be flexible, creative, dynamic and capable of delivering products quickly in order to survive. Therefore, any deployed software process must support ***Flexibility*** and ***Creativity***. From the interview data, though it is evident that Irish software companies value ***Creativity*** and ***Flexibility***, many believe that process can stifle these desirable attributes, and its use should therefore be

carefully considered. Some of the start-up companies see processes as primarily of benefit to established companies as Company 3 describe:

> *If you want to be more sure of the results, the processes will give you more likelihood of being sure, but it's probably a bit like playing it safe. I would think you won't get the same level of innovation or creativity.*

One company felt that they had too much process and felt that it impacted negatively on their ***Creativity*** and innovation.

> *I think that product development is about being inventive and creative and new ideas coming forward and being developed quickly into something mainstream. And when you don't see that happening I think that too much is being stifled.*

Product companies focus on product development and fear that increased process will detract from that focus and that the price of additional process is a decrease in ***Flexibility***.

> *When we set up we had more supervisory and managerial roles in that group than we have now and we had to scale that back which has made things a lot more flexible. I do think you have to be nimble, quick and capable of being responsive in our position. That works well and I don't want to lose it.*

Others also reduced the amount of process they used because of they impact they felt it was having on ***Flexibility***.

> *We started following a formal process for a while, but the guy who was driving that left and we abandoned it. What we have now is quite flexible, and not very formal.*

But fears of processes impacting negatively on ***Creativity*** and ***Flexibility*** are not the preserve of smaller software companies. As one Expansion company explains:

> *All of our competitors are several times our size, we are the smallest in the field and our only way of dealing with those guys is to make them look old and fat.*

All of the extracts above show how innovation, ***Creativity,*** and ***Flexibility*** are seen, by the managers concerned, as the lifeblood of their companies. Because SPI is sometimes seen as the enemy of ***Creativity*** and ***Flexibility*** it provides clear evidence why ***Business Events***, which cannot be resolved using the existing process, are the main drivers of SPI. Many managers believe the attributes of innovation, ***Creativity*** and ***Flexibility*** carry business advantages far in excess of the proposed benefits of repeatability, consistency, and ***Quality*** which are associated with process and process improvement models.

8.7 Cost of Process and Process Models

Not unlike the other aspects of process and process improvement, the choice of which model to use was also linked with its associated cost of adoption and implementation. As stated in 6.7, all of the companies interviewed are using a tailored ***Software Development Process***, which in most cases is based on a standard industry model.

8.7.1 RUP

In some of the companies, their initial software process had been tailored from the RUP. Almost all of those who used elements of it have since dispensed with it completely, blaming ***Documentation***:

> *RUP has a waterfall feel to it. There's quite a lot of documentation associated with it and that didn't work for the sort of lead times we were striving for.*

Others blamed the complexity and 'weight' of it.

When I joined here we had an approximation of the RUP. It was over-engineered, over the top process kind of stuff. RUP is unimplementably complex. Even people in the past who have been into heavily engineered process found it impractical.

Other companies, who had at one point considered using the RUP, subsequently rejected it because of the complexity of the associated support tool, Rational Rose.

What I would think is the challenge is to deliver a set of tools that are easy to use, and in as far as possible form part of the design cycle. UML/Rational Rose have been attempts in that direction but are very bulky and heavy.

Others merely felt that Rational Rose was too expensive.

At one stage we started going down the Rational route. It was going to cost us 300 grand and it was money we didn't have so we backed away from it.

Because of the costs, both resource-wise and in tool purchase, many companies moved away from processes based on Rational to ones based on XP.

8.7.2 XP

Whereas the RUP was seen to have merit but to be too expensive to deploy widely, XP, as a development methodology, attracted far greater support among the interview sample. Used by companies at all size levels, the tailored versions of XP, which the companies deployed, were seen to be very cost effective. One manager argues that XP provides the fastest time to market capability of all the models available.

There's now no way we could deliver faster with a different process than with this. XP gives you a lot of advantages in delivering quickly even on small projects.

Widespread gains were also reported from applying short iterations and test-first development.

> *I think a lot of the attributes of XP, around test-first design and iterations, and rapid feedback to developers are hugely valuable.*

Company 4 support this view.

> *First, we tried the planning stuff and then we tried the unit testing stuff. The unit testing stuff in particular gave us such a dramatic payback, that we then felt comfortable adopting more of the process.*

XP, where it replaced an existing process, was greatly assisted by ***Employee Buy-in***. In a number of cases, XP's predecessor had a much greater ***Documentation*** requirement. Given developers' misgivings concerning ***Documentation***, as described in 8.4, any successor with a reduced ***Documentation*** overhead had a real chance of succeeding. This clearly proved the case as higher levels of buy-in were reported in companies that implemented XP.

> *And what's more the developers actually like doing it because it gives them a chance to get clear in their head what the task is before they start to write it, because they have started to use it even though the feature hasn't existed yet.*

One of the organisations that introduced XP had previously used RUP and XP was essentially the engineers' choice.

> *I couldn't say that XP came from the management; XP came from the engineers. It was a bottom-up thing. RUP was a top down thing and didn't really wash.*

The ability to reduce the 'process' elements in development was a key factor in the success of XP. Companies reported developer benefits and how easily they embraced the methodology.

> *It's attractive to the coders because in theory it shortens their development cycle and has them doing less stuff that they dislike like documentation, test specs etc.*

When introducing XP, companies believed they got good value for money with the methodology. The ability to implement the practices piecemeal, and the use of iterations with regular feedback meant, particularly in the case of the smaller organisations, that for the first time they had control over development activity. It's best summed up in the following excerpt from Company 16.

> *XP was very cost effective because you didn't have to implement everything. You just had to implement those things that worked for you. And it did give us visibility into the software development process which was key.*

8.7.3 Limitations of XP

Though XP had clearly provided benefit for many organisations in the study group, some had reached what might be classified as the 'post-XP' stage, where, through using it, they had identified perceived limitations with the methodology. Ironically, despite many managers' reluctance to commit resources to ***Documentation*** tasks, as discussed in 8.4, by far the most common complaint about XP related to the insufficiency of ***Documentation*** produced by the method.

> *I'm not a believer in XP at all. I've seen it up close and it doesn't really work. We tried XP on the main core technology and we found that the documentation trail was extremely weak or even non-existent.*

XP's lack of ***Documentation*** meant it had restricted use in more rigorous environments.

It didn't work in here primarily because of the deliverables that are required, around following documentation and standards.

The limited ***Documentation*** output from XP becomes particularly noticeable as a company gets larger and expands the amount of activities it undertakes and customers it supports.

As the company got bigger, the marketing people spent more time on the road in sales functions, and it became more difficult. Also, the products and teams started to get bigger, and we needed written specifications really badly.

Fears were also expressed that the absence of ***Documentation*** would make it more difficult for new team members to understand the system thus also highlighting a limitation of ***Tacit Knowledge***.

If we were hiring and bringing in a bunch of new grads, would XP work given that there is no documentation for people to look at? That kind of methodology and knowledge is embedded in the teams and the way that they work rather than anything that people can look at.

There is an interesting mix of companies who are using XP and it ranges across all company size sectors. There is no doubt that XP has improved the companies' development capability but, despite its popularity, there were criticisms of it particularly around ***Documentation***. The evidence suggests that ***Documentation*** and its importance is more a feature of the Expansion companies. The companies who specifically bemoaned the lack of ***Documentation*** support in XP (companies 6, 11, 13, 16, 17) have reached this stage with the exception of Company 16. However, Company 16 is about to enter a regulated market with the associated emphasis on ***Documentation*** and ***Traceability***, which requires a suitably supportive process. Ultimately, the insufficiency of ***Documentation*** in these instances will act as an ***SPI Trigger*** and the companies

concerned will have to take remedial action. How XP is then incorporated into a new process, if at all, will be instructive.

8.8 Process Improvement Models

A key part of this research, as set out in 1.2.1, is to examine why software product companies do not appear to be following 'best practice' SPI models. There are a number of best practice models in existence but only the CMM (and its successor the CMMI) which are specifically geared for software, and ISO 9000 whose origins lie in manufacturing, resonated with the companies interviewed. Of the 21 study companies, 3 are ISO 9000 certified and one is embarking on the ISO 9000 certification process. None of the companies are using the CMM or the CMMI.

8.9 ISO 9000 and Cost of Process

As explained in 6.4, where a manager has previously used a process or process model that they felt had a beneficial impact on development, that process was generally imported into their new environment. By contrast, where managers had prior experience of a model they felt didn't work then they rejected its use within their new companies. This was most significantly felt in the case of ISO 9000.

8.9.1 ISO 9000 and Software Product Development

In some cases, best exemplified by Company 8, opposition to the introduction of ISO 9000 centred on its perceived emphasis on procedure rather than product ***Quality***.

> *I worked in companies who were so hung up on ISO 9001. And it just didn't work. They made crap products but by God they had ISO in.*

Another manager echoed this believing there was insufficient analysis of the actual process itself.

I was involved in implementing ISO before and the way I see it works is that you define what you are going to do and as long as you stick to what you do it's great. But what you define you are going to do could be rubbish.

The argument that ISO 9000 certification encouraged process observance rather than an examination or measurement of the process itself, or the products it produced, was taken a step further by another manager.

I have done ISO once before but in an organisational management department. It put me off trying to do ISO in here.

The manager in company 21, who also had first hand experience of using the standard, in the software localisation sector, supported the opinion that it was more suitable to manufacturing.

In the software localisation industry it semi-worked, as part of it is a manufacturing-type process. However, all of the parts which involved getting the translator to do something didn't work at all.

8.9.2 ISO 9000 – Bureaucracy and Documentation

From the managers' perspective, ISO 9000 can be seen to be closely associated with the ***Bureaucracy*** category described in 8.3. The participants variously describe ISO 9000 as "way over the top", carrying "a lot of baggage", being "heavyweight", and having significant "overhead". ***Bureaucracy*** is something feared by small and large companies alike. They believe prevention is better than cure and set out to resist or avoid bureaucratic activity for as long as possible. Despite the other arguments above in relation to ISO 9000, the major opposition to it is because of what managers believe is its overemphasis on ***Documentation***. The link between ISO and ***Documentation*** was best summarised by Company 5.

But in one way ISO doesn't focus on the important bits at all, it's still a very paper driven thing. You can get away with having an ISO system that doesn't actually do any source code control at all and still get your 9001 certification.

Other participants support the view that ISO 9000 and voluminous ***Documentation*** are strongly correlated.

People use it and get buried in paper and even I think the accrediting authority itself encourages people to head down the paper route, which they shouldn't do.

Small software companies and start-ups are especially wary of ISO and the amount of ***Documentation*** required by the standard. Company 16, who are preparing to enter a regulated market, attempted, unsuccessfully, to introduce ISO on start-up.

We started off with trying to follow a kind of ISO model, and that was just crushing us in paperwork and we abandoned it because we have a small number of engineers and we needed to be producing output.

Company 20 agreed, citing the ***Documentation*** overhead for small organisations.

ISO 9000 is fairly onerous in terms of documentation and I'd be very surprised if any company our size, certainly in an industry like ours, is seriously looking at it.

From the earlier interview extracts, and further analysis of the study data, there is a strong link between the reason for ISO 9000 rejection and the **Cost of Process** arguments discussed previously in this Chapter. With companies reluctant to engage in SPI because of its association with increased ***Documentation*** (Section 8.4), it is not surprising that they would be hostile to ISO if they perceive it as having a similar ***Documentation*** requirement.

8.9.3 ISO 9000 and Business Benefit

The three companies in the study who have ISO 9000 certification pursued it primarily for business reasons. Of the remaining 18 companies, only one company is actively considering it, and this is because they are entering a regulated sector. For Company 1, the introduction of ISO 9000, was undertaken to gain a contract from a telecommunications MNC.

> *[We sought certification] to get business. Also as a marketing tool, to say we have a process; we're a reputable company. We had been told that we had lost projects before because we didn't have it, so there was a motivation there to win business.*

Company 6, who sell to the pharmaceutical sector, also needed it for business purposes. In a highly-regulated market it is potentially a tool of competitive advantage and something that can provide assurance to a customer.

> *And a lot of the pharmaceutical companies expect certification. If they come in and you don't have certification, you have to justify more, show them more, and it's that little bit more difficult to get their confidence.*

Company 14 is the only other company in the study to have ISO 9000 certification. Market advantage was again the reason they pursued it.

> *It was felt by upper management that it would be very advantageous if we had it up front. It gives you more weight that you are serious about what you are doing. It gives you a good name and good reputation.*

Without certification, Company 16 are facing market barriers and need ISO 9000 to remove these.

The company is moving towards ISO 9000 because it's moving towards FDA approval. Certification is the "price of entry" into the market.

8.9.4 ISO 9000 and SPI Triggers

The fact that ISO 9000 certification is being sought for business reasons rather than process improvement reasons, lends credence to the theoretical perspective of ***SPI Triggers*** being driven by ***Business Events***. From the interviews, and detailed analysis of the managers' views, there is no evidence that in these companies certification was pursued in order to achieve improved ***Quality*** or development capability. Company 12's CTO, who best represents the views of a number of the study managers, describes it in the following terms:

If somebody said tomorrow, you won't sell into the financial services sector unless you are CMM or ISO 9000 compliant or whatever, we would very quickly get certification. It's commercial reality that if someday you are forced to do something, you will do it quickly.

Within the interview transcripts, there are quotes from 15 of the 21 companies studied which are critical of ISO 9000 from a ***Documentation***, ***Bureaucracy*** and administrative perspective. This leaves a situation where more than three quarters of the companies in this study firmly oppose the adoption of ISO 9000 in their software development.

8.10 CMM/CMMI and Cost of Process

In 3.7, the adoption of CMM and CMMI by the software industry was discussed. As the most widely publicised SPI models, it was important to this study to determine the attitudes of indigenous Irish software product companies towards them. Awareness of CMM and CMMI among the managers was far lower than was the case with ISO 9000. Though a number of the managers interviewed had experience of CMM from previous employment, none had incorporated it into their present positions. However, as with ISO 9000, it was where managers had previous experience of using CMM that greatest

hostility to its introduction arose. An example of a greater body of opinion is the manager in company 5 who, when asked what working with CMM was like in his previous company, responded thus:

> *It was dire. It just got in people's way. It was almost designed to get in people's way. It wasn't designed to enhance the development process. It wasn't for me.*

Support for the opinions of the manager of Company 5 came from Company 10's software development manager who previously worked in a large multinational which used CMM.

> *CMM is neither efficient nor would return huge benefits. Somebody with experience could go in and have much more effect in a lightweight way if they understood what they were doing.*

Company 11 rejected it feeling it would hinder their ability to deliver quickly.

> *If you look at CMM it was delivered for the likes of NASA. We might sell a piece of software that needs to be delivered in 3 months. So, the overhead of instigating a very rigorous CMM-style process is outweighed by the time it takes to deliver it.*

The opinions of one manager, who having investigated it and chosen not to introduce it, represents all companies who reached the same conclusion.

> *We felt CMM was overkill for the level of development that we were doing and so it wasn't really pursued.*

The belief that CMM and CMMI contain excessive levels of detail and require high levels of administration was expressed by a number of the participants. Notwithstanding the fact that they criticised ISO 9000 for not being suitable for software, CMM and

CMMI did not generate increased support even though they are software specific. The criticisms levelled against CMM and CMMI, by the managers, indicate it is "excessive", "over the top", "heavyweight", "onerous", "bureaucratic" and "too detailed". Managers were then asked under what circumstances might they use CMM or CMMI. The responses received were very similar to those relating to ISO 9000. Company 9 summarise it best:

> *It will depend on the companies with whom we will engage. Maybe where we get to the stage where we are dealing with government or defence and they are looking for certification, then we will go for it. That's because there is a business decision to tackle those customers and therefore the process has to evolve to get certified. You wouldn't do it the other way round. That would be crazy.*

8.11 Summary

This Chapter concludes Part II of the study and examined all of the issues relating to the costs associated with following process and process improvement. ***Bureaucracy*** was raised by the managers as a significant process cost and a reason for companies to be suspicious of SPI. ***Documentation*** was perceived as the greatest cost in process adherence and was something that participants wished to minimise. ***Communication***, though essential, could, in the opinions of the study companies, be better served through informal means, and the use of ***Tacit Knowledge***, rather than through ***Documentation***. Process and process improvement was also believed to have a negative impact on ***Creativity*** and ***Flexibility***. Adoption of ISO 9000 and CMM/CMMI was ruled out by managers for the same cost reasons as process and process improvement and would only be considered should there be a business imperative to do so. Part III of the study will now present the literature support for the theory described in detail in part II.

Part III Support for the Findings

Part III – Overview

On completion of a grounded theory study it is likely that the findings will be at variance with published studies in related areas (Strauss and Corbin, 1998). Therefore, the developed theory should either be integrated with the existing work, or act as a critique of it (Goulding, 2002). As the researcher progresses through the study, he/she will discover material that is of relevance to the work. How this material is used in the study is a key question. As explained in 4.7, this study employed the Strauss and Corbin (1998) version of grounded theory. The researcher, using his prior experience in the field, referred to the literature, as he felt necessary, whilst taking cognisance of Strauss and Corbin's (1998) advice that "familiarity with the literature can enhance sensitivity to the nuances in data, just as it can block creativity". On completion of the investigation and data analysis, and during the writing stage, the literature has a major role both in confirming findings and using the findings to highlight where the literature is incorrect or only partially explanatory (Strauss and Corbin, 1998). But an extensive trawl of the literature should only be done after the grounded theory has been formulated as, "running to the published literature to validate or negate everything one is finding hinders progress and stifles creativity. Used as a analytic tool it can foster conceptualisation" (Strauss and Corbin, 1998).

As stated in Section 1.2.3 there are a limited number of studies internationally, and more especially in the Irish case, describing, how process is initially established in software companies, what software companies are doing in practice, the reasoning behind why SPI programmes are undertaken, and the logic for software companies ignoring 'best practice' SPI and quality models. The grounded theory presented in Part II advocates a consideration of factors, other than merely technology, in SPI programmes. Therefore, this research, in conjunction with a review of the Irish and international software engineering literature, also examined and discovered support for the theory, in the IS, human, and social factors disciplines, and from academics, practitioners and other industry commentators.

Part III of the study is composed of three chapters. To simplify the linkages from literature back to the developed theory, the chapters are presented as a direct mapping onto each of the Research Themes and the Core Category described in Part II and covered in Chapters 6 – 8.

Chapter 9 Support for – Process Formation

9.1 Introduction

This chapter describes the published material that provides support for the research theme **Process Formation** as presented in Chapter 6. The relevant literature is compiled and displayed in the same order as in Part II, and focuses on published support for the major categories linked to the research theme.

9.2 Evidence For – Process Formation

Process Formation describes how the initial software process is designed and created in start-up software product organisations. In many software start-ups, the founders are experts in application domains other than software (Coleman Dangle *et al.*, 2005). Even where the founders have software experience, they often have very limited resources at their disposal and an absence of a business model (Voas, 1999). Factors such as deciding what type of software business you are going to be also arise (Bersoff, 1994). From a software process perspective, start-ups are ultimately concerned with survival rather than establishing procedures. Bach (1998) describes the typical start-up in which he worked as containing "a bunch of energetic and committed people without defined development processes". But overall, as Sutton (2000) states, "software start-ups represent a segment that has been mostly neglected in process studies". A trawl of the literature confirms Sutton's findings and reveals few accounts of how process is established in software start-ups. Consequently, the findings on **Process Formation** offer a fresh theoretical perspective on software process and the factors which affect its initiation.

9.2.1 Evidence For – Background of Software Development Manager

One of the theoretical categories raised by this research is that the initial development process used by a software start-up is based on the experiences of the software development manager (Section 6.4). In a Northern Irish context, McFall *et al.* (2003)

found that often companies are managed by entrepreneurs and directors who know the processes well and subsequently act as mentors to other members of staff.

But many managers just decide to apply what they know, as their experience tells them it is merely common sense (Nisse, 2000). In software companies, technical survival and success can depend most heavily on the managers and executives who have responsibility for technical strategies (Sutton, 2000). Baskerville and Pries-Heje (1999b), in detailing the first three years of business of a small software company, state that the Web and Internet knowledge used in system development by the employees, had been gained through personal interests, reading, experimentation, or exploration prior to them joining the company. Similarly, the knowledge of the business and target market was brought to the company by the founders.

Previous software process experience is often considered an indicator of success (Humphrey *et al.*, 1991). By contrast, previous negative experience of SPI can act as a de-motivator for practitioners towards implementing change. Baddoo and Hall (2003) consulted practitioners across three groups, developers, project managers and senior managers. Previous 'Negative/bad experience' was cited as an SPI de-motivator by 33% of senior managers as opposed to 5% of developers. These results are consistent with this research where interviews were conducted solely with senior managers.

Alternatively, where practitioners work, or have worked, in a non-process-driven environment, they need to be convinced of SPI's value. Armour (2001) describes the difficulties he encountered in trying to persuade some managers in a successful innovative products company, who did not use defined process models, of the benefits of SPI.

9.2.2 Evidence For – Management Style

Management Style describes the way a leader discharges their administrative functions, and motivates and communicates with their staff (Buchanan and Huczynski, 1985). Among the study practitioners interviewed, ***Management Style*** varied between

'command and control' approaches and 'embrace and empower' (Section 6.5). In software start-ups many managers encourage all employees to be involved in all aspects of development (Kelly and Culleton, 1999). Whilst numerous organisations retain this culture of involvement, many large companies delegate responsibility for software process to a dedicated process group. In smaller companies and start-ups senior management often allow their developers to have a significant influence over the way they work. By contrast some organisations enforce process on their employees. This 'command and control' ***Management Style*** has its opponents who believe that efforts to force developers to work according to procedures developed by those not immediately responsible for results have failed (Beck and Boehm, 2003). For example, XP proposes a strategy of decentralised decision making (Beck, 2000). As a result, agile development methodologies thrive in 'embrace and empower' environments, where staff are empowered and the organisation can be said to be thriving on chaos, whereas plan-driven approaches are more suited to a situation dominated by policy and procedure (Boehm and Turner, 2003). Nevertheless, some companies may struggle with adopting an agile development approach as many managers, particularly those at senior level, are reluctant to surrender the feeling of control that Gantt charts and other plan-driven process documents provide (Cohn and Ford, 2003).

Some organisations use a hybrid style of top-down instruction and bottom-up involvement. One large company only sought employee suggestions on SPI once they had accepted its basic merits (Keeni, 2000). Others used a more consensual approach informing and involving the team in SPI decision making (Kautz, 1998) and trusting them with the development effort even if some of the individuals were 'gifted hackers' (Nisse, 2000). Some argue that SPI will only work if the behaviour of managers and practitioners are properly aligned (Potter and Sakry, 2002). In this way managers keep in touch with SPI progress and explain to people how the changes are in line with organisational goals. Similarly a less disciplined and more flexible ***Management Style*** can also yield positive results (Royce, 2005).

But, on some occasions, ***Management Style*** and approach act as barriers to SPI. Many of the study practitioners believed there was a limitation to SPI effectiveness and, as a result, suppressed its use in their company. Describing a study he conducted with senior managers of a telecommunications company, Armour (2001) proclaims 'you could have tortured these people and they would not have admitted that process was a good idea'. In these instances the politics of the organisation and the desire of staff to protect their own area of work can have negative SPI impacts (Herbsleb and Goldenson, 1996). But organisations who support SPI often adopt different approaches.

To succeed in SPI, managers should be cognisant of the organisation's SPI history, culture, motivators, and ensure that a participative leadership style is used (Laporte and Trudel, 1998). Centralised management-driven SPI programmes make things too rigid and distant from practitioners' daily practice (Mathiassen *et al.*, 2001) and ultimately it is the attitude of senior management towards SPI that determines the organisation's culture and the prospects for SPI success (Kasse and McQuaid, 1998). The most effective ***Management Style*** is one whereby managers appear to relinquish power to their employees (Buchanan and Huczynski, 1985). DeMarco and Lister (1999) argue for an open, trusting style of management, which they term 'Open Kimono', as against a more defensive approach. Using 'Open Kimono' a manager takes no measures to defend themselves from those they have put in positions of trust, which is essentially everyone under their control.

In relation to software development, this concept of relinquishing power and placing trust in the ability of the employees is raised in a number of instances in the literature. Humphrey (2002) urges managers to trust their engineers claiming, "when you don't trust them they are not likely to trust you". This view is echoed by Yamamura (1999) who reports on the success of an SPI programme in the Boeing Corporation stating that employees were highly motivated, as between themselves and company management there was a deep well of mutual trust. Agile methodologies, if they are to work successfully, need management trust in the developers and their skills and ability to do the job (Lycett *et al.*, 2003) and there is evidence that empowering development

practitioners, and allowing them to take ownership of the processes they use, motivates SPI success (Baddoo and Hall, 2002).

9.2.3 Evidence For – Market Requirements

The task of a software company is to satisfy the needs of the customers and respond to changing market demands. This places great stress on the software development and maintenance processes (Kilpi, 1997). The CMM was born out of an initial request, to the Software Engineering Institute, from the US Department of Defense, and its original format and implementations reflect this (Humphrey, 1989). Also, in the development of advanced military applications, NASA used SPI approaches (Kuilboer and Ashrafi, 2000). Because of the application demands of the military sector, such as safety-critical systems, SPI activity was initiated to meet these objectives. This view is echoed by Lindvall and Rus (2000) who state that software for the space shuttle or a nuclear plant have different safety and ***Reliability*** constraints to that required for a word processor.

Jones (2003) comments on how the type of software being constructed influences the software development practices being used. He describes how the methods for building military software are very different from a basic end user application. ***Application Type***, he argues, significantly impacts the personnel and specialists required for development, with systems and military domains employing specialist expertise whilst MIS applications requires more 'generalists'. Jones also reports that companies building systems and military software are more than twice as likely to have QA departments than those building MIS software. This is a reflection of the demands of the market as systems and military application have a much higher ***Reliability*** and availability quotient than MIS applications. Similar ***Reliability*** demands are often made of telecommunications systems where a competitive marketplace, and the requirement for 'always available' services, can lead to the creation of special facilities to ensure rigorous testing (Fitzgerald and O'Kane, 1999). Evidence also shows how a different approach must be taken if the client is purchasing an application to enable the staging of conferences or events versus a system for national crisis management (Boehm and Turner, 2003). Software products can differ greatly by application, market and customer

requirements, making process choice more complex and context dependent (Cusumano, 2004). Rost (2005) maintains that the 'heavy processes', documented and proposed as best practice in the literature, are really best suited to large, one-of-a-kind projects such as air traffic control and modern weaponry, and many commercial systems don't fit this pattern.

Baskerville *et al.* (2001) present a case study of another business based on Internet services. Reflecting the demands of the market in which they were operating, the company's business model enabled them to be 'nimble, creative and extremely fast'. A subsequent article from the same authors argues that SPI approaches including ISO 9000 and the CMM are primarily effective in predictable, large-scale, long-term development projects whereas Internet-speed software development, used in unpredictable and changing markets, requires methodologies that balance discipline and ***Flexibility*** (Baskerville *et al.*, 2003). These factors can have a big influence on the early processes used by a software start-up. Sutton (2000) says that whilst a highly disciplined and systematic approach will be required for safety-critical software, it may be incompatible with highly dynamic application domains such as e-commerce. Sometimes the customer base can have a major input to the process a software company uses. One organisation describes how, prior to an SPI initiative, they were required to use a process supplied by their customers (Kelly and Culleton, 1999).

Agile methodologies, because of their claims to generate improved customer satisfaction, are very closely linked with ***Market Requirements***. Despite evidence to suggest that they are highly suitable for use in the Internet domain (Murru *et al.*, 2003), they face much greater usability challenges within global software development, heavily regulated environments, and through government restrictions that necessitate standardisation (Lycett *et al.*, 2003).

9.2.4 Evidence For – Process Tailoring

According to Hall *et al.* (2002), the need to tailor SPI to company requirements is a recurring theme in SPI literature. Jones (2003) states that, following an examination of

12,000 projects, no single development approach is universally deployed, and ***Process Tailoring*** is widespread. In software development, different projects need different processes (Lycett *et al.*, 2003; Deck, 2001) and tailoring should take into account the contingencies of each project (Fitzgerald and O'Kane, 1999) and the local environment (Casey and Richardson, 2004). Every project has a different combination of people and product (Phillips, 1999) and small projects can be executed with less formality than larger projects (Jalote, 2002). "One size does not fit all" (Kasse and McQuaid, 1998) and every process should be selected, tailored, and adapted to the individuals that are working on a particular project team (Cockburn and Highsmith, 2001). This means that software teams adopt a flexible approach to development processes so that each individual team can apply what's best or appropriate (Rising and Janoff, 2000). Choosing the right process model and tailoring the process accordingly is particularly important for start-ups as they do not have the collective experience of using a particular process. As such, the nature of software process for a creative group producing something for the first time should be different than for an experienced group producing the fifth in a series of system upgrades (Armour, 2001).

But it is ***Contextual Issues*** that are the key divider. Russ and McGregor (2000) state that if the environmental factors (size, complexity, ***Quality***, people interactions), within which a project will be developed, are understood, then a process can be defined and tailored accordingly. Tailoring is key, as process success will be elusive if the process does not suit the organisational culture and business (Moitra, 1998; Hardgrave and Armstrong, 2005) and the correct process is one which suits the company's way of working, the degree of formality demanded, and the level of safety-criticality required (Henderson-Sellers, 2002). Section 9.2.3, already discussed how the demands of individual customers and markets affects the process used. Tailoring a software process involves taking account of the context in which the process will operate, as a process that is generic enough to deal with any situation is generally too high-level to be practical (Lycett *et al.*, 2003). Software development groups and projects within the same organisation need to define different kinds of processes for different kinds of products, markets and customer requirements (Cusumano, 2004), and it is possible to

use the level of risk within the project as the basis on which the process is tailored (Boehm and Turner, 2003).

It is argued that processes can be tailored to balance both agile and plan-driven approaches to development (Boehm and Turner, 2003). Lippert *et al.* (2003) report on how they adapted XP for each project and then developed extensions to the method to cover specific aspects of their business. Proponents of plan-driven approaches also support tailoring. Although standard processes provide a foundation, each project has unique needs and processes need to be tailored accordingly (Paulk, 1998).

9.3 Summary

This chapter examined the theoretical support and evidence for the research theme **Process Formation**. The literature support was presented under each of the sub-category headings. The next chapter investigates the level of published evidence to support the theoretical argument for the research theme **Process Evolution**.

Chapter 10 Support for – Process Evolution

10.1 Introduction

This chapter explores the published material that provides support for the research theme **Process Evolution** as presented in Chapter 7. The relevant literature is compiled and displayed in the same order as in Part II, and focuses on published support for the major categories linked to the research theme.

10.2 Evidence For – Process Evolution

As software companies grow the development process must adapt to meet the demands brought on by the changes affecting the company. As software organisations change, a certain amount of resistance is to be expected and management practices in terms of planning policies, incentives and culture must also change correspondingly (Mathiassen *et al.*, 2005). Any new process is likely to appeal to some developers who want to be the first to try it but equally others may oppose it for the changes it brings (Cohn and Ford, 2003).

Many of the reports in the literature, including (Andres *et al.*, 1997; Coleman Dangle *et al.*, 2005; Debou and Kuntzman-Combelles, 2000; Ibrahim and Pyster, 2004; Kasse and McQuaid, 1998; Moitra, 1998; Stelzer and Mellis, 1998) are 'how-tos' of software process listing the critical success factors for SPI, whilst other SPI success stories are company specific (Daskalantonakis, 1994; Dion, 1993; Fitzgerald and O'Kane, 1999; Humphrey *et al.*, 1991; Jalote, 2002; Laporte and Trudel, 1998).

10.2.1 Evidence For – SPI in Small Software Companies

Undertaking SPI in small companies is a particular challenge and according to Horvat *et al.* (2000) must take into account the following factors:

- High dependency on individuals
- Small number of employees and the necessity for individuals to have multiple roles

- Proportionately greater impact of human factors
- Dependence on a small number of projects
- Importance of communication with customers
- Difficulties in finding resources for SPI.

As documented in 2.3.1, the overwhelming majority of the Irish software industry, and the participant companies in this study, are Small- to Medium-sized Enterprises (SMEs). However, the vast majority of SPI success reports are from large software organisations and there is correspondingly limited literature coverage of SPI in small settings. Therefore, this chapter, in support of the **Process Evolution** theory, refers to much of what does exist on SPI in small companies and, where appropriate, draws on the key related findings from larger company studies. As a result, like **Process Formation**, the findings related to **Process Evolution** present a new theoretical viewpoint on software process and the factors which cause it to change.

10.3 Evidence For – Process Erosion

Section 7.3 described how, after a period following an SPI initiative, the processes used in the study companies interviewed began to erode and elements were ignored or omitted. An example of this from the literature occurs in (Hayes and Zubrow, 1995), whereby, in an SPI analysis of CMM-assessed companies, one organisation, following a level 3 assessment rating, subsequently regressed to level 1. However, for confidential reasons, no further details of the company or an explanation for this ***Process Erosion*** is provided. Nonetheless, the vast majority of published case studies can conceal the true picture, as companies, who have not succeeded with SPI, or have regressed following an improvement initiative, are unlikely to publicise their results (El Emam and Briand, 1997). This means that in many cases in industry, where ***Process Erosion*** occurs, it will go unreported.

However, with some investigation, several examples of the concept of ***Process Erosion*** can be found in the literature. Probably the best example is a study that was carried out using the IDEAL (Initiating, Diagnosing, Establishing, Acting, and Leveraging) model

in a small Danish software company (Kautz *et al.*, 2000). IDEAL was developed by the SEI to support SPI. Following the implementation of the model, the company carried out an evaluation of its performance. This revealed that many of the improvements gained during the study had not been institutionalised and had subsequently been eroded. For example, at the outset of the study, all of the developers followed the code ***Documentation*** guidelines but after a period had stopped using them. The Danish project leaders reported that the guidelines fell into disuse due to time pressures and lack of control. In addition, managers also wanted things to be as non-bureaucratic as possible and so ignored the ***Process Erosion*** effects.

Middleton *et al.* (2004) report on how, after an SPI initiative in a US company, the employees found that the process for submitting further innovation suggestions was cumbersome and slow and therefore went unused. Leung and Yuen (2001) explain how a company developed a 'standard' software process framework but the reluctance of the employees to follow it, because of the overhead involved, meant that a new process, tailored for small projects, was developed. Even where companies have been certified, such as through achieving ISO 9000 accreditation, there can be an ***Erosion*** of the ***Quality*** system after the certificate is granted (Biro *et al.*, 2000).

10.3.1 Evidence For – Minimum Process

Many companies interviewed for this thesis have an 'official' company software process and an 'actual' ***Minimum Process***, which is what the development teams use to develop the software products. The evidence for the use of ***Minimum Process*** by companies is also closely linked with ***Process Erosion***. Probably the best example of this is contained in the following extract:

> *Given a choice between following a process and doing whatever we feel like, few humans follow the process. The grim little secret of many projects is that they continually have to spend time going back to create deliverables for sign-off purposes and none of these deliverables have added any value to the overall development of the system. One company has even gone so far as to create a*

process for working "off-process", and it is not possible to find a single project that follows their official methodology (McBreen, 2000).

Other published work also reveals the link with ***Process Erosion*** and ***Minimum Process***. A study by Baddoo and Hall (2003) found that time pressures on development staff meant that SPI changes were completely ignored at times of greatest stress. Chiang and Mookerjee (2004) report how, faced with limited resources or a tight schedule, managers may forego system design activities for more productive coding tasks. In some instances the decision on the volume of process to use, for example the formality of design documents or reviews, is left to the individual engineer (Cusumano and Yoffie, 1999). ***Minimum Process*** suggests that things need not be perfect, that systems work 'well enough', and that many practices associated with traditional development models can be truncated (Dicks, 2000). A study reporting the results of a pilot CMMI Class 'C' appraisal programme, within the indigenous Northern Irish software industry, similarly concluded that, as the organisations seeking appraisal "have been in business, developing software, for several years" it can be assumed "that from an engineering and management standpoint, these companies are all doing at least "enough" of the right activities to survive" (Wilkie *et al.*, 2005).

Minimum Process is typically used in software companies to reduce the cost and effort of following a defined process, especially on small projects (Leung and Yuen, 2001). Baskerville *et al.* (2003) argue that, in Internet software development, intense demands for speed of delivery meant that 'many companies used just enough process to be effective' and the tendency was for them to skip phases or tasks (something also recommended by Leung and Yuen (2001) for small companies) that might impede ability to deliver the software on time. To a great extent this is the modus operandi of companies who adopt agile methods whereby development effort is concentrated only on that which is judged to be essential but sufficient for a particular situation, whether through coding and management support (Lycett *et al.*, 2003), or through minimising the ***Documentation*** produced (Grenning, 2001; Kutschera and Schafer, 2002). Phillips (1999) contends that, in the 21st century, people will resist process improvement until

they feel they need it and that companies, driven by time to market demands, will become experts at a specific, minimal process that optimises time to market. Finally, the difficulties from which ***Minimum Process*** stems are best summed up by Jalote (2002) who states that a process may have some extra steps but you will not always know in advance which ones are not required.

10.3.2 Evidence For – Employee Buy-in to Process

The issue of ***Employee Buy-in to Process*** emerged as a key factor in this research. As reported in 6.5 and 7.5, the ***Management Style*** used by the organisations studied, and the level of developer support, dictated how well the respective processes were followed. Developer support may be contingent on how supportive they believe process is to their ***Creativity***, and the fact that it often stifles this stands as a legitimate criticism of SPI (Armour, 2001). Kasse and McQuaid (1998) argue that "a successful process improvement initiative must have the support of the practitioners". Boeing reported additional employee satisfaction by directly involving staff in their process improvement initiative (Yamamura, 1999). Similarly Fitzgerald and O'Kane (1999), in their study of Motorola's Cellular Infrastructure Group, suggest that giving the employees more ownership of the process generates more ***Employee Buy-in*** to SPI. Laporte and Trudel (1998) concur, as they found that getting ideas from those closest to the process resulted in greater buy-in. An inclusive approach is vital, as a process will only be successful if the engineers using it like it and feel it is of benefit to them in their daily work (Henderson-Sellers, 2002). Findings elsewhere suggest that SPI success in small companies depends to a greater extent on employee participation than is the case in large companies (Dyba, 2003) and one small Chilean company succeeded in quickly institutionalising improvements to its software process by involving all of its personnel in the work (Guerrero and Eterovic, 2004).

Nonetheless, it is not always easy to get the appropriate level of developer buy-in to SPI. In the case of the introduction of agile processes, there are published instances where developers strongly resisted the change (Cohn and Ford, 2003; Schuh, 2001). By contrast, Grossman *et al.* (2004) experienced fewer problems with introducing XP, as

there was significant buy-in from the developers and other stakeholders from the outset. Working towards engineer buy-in, however, may be worth the effort, as one survey showed that agile methodologies score much higher than rigorous methodologies in terms of employee morale (Cockburn and Highsmith, 2001).

Nonetheless, some company management often attempt to impose process improvement on their staff. A developer's comments, reported in Beecham *et al.* (2003), shows how a CMM level 4 company makes SPI adherence part of the staff member's annual performance goals. Equally, Lethbridge *et al.* (2003) found that where ***Documentation*** had fallen into disuse, managers had attempted to impose more discipline on their development staff, forcing them to execute updates rather than trying to get them to buy-into a consensual solution.

10.3.3 Evidence For – SPI Triggers

It is claimed that process change will only occur when staff and management are sufficiently dissatisfied with the status quo and wish to do things differently (Paulk, 1998). Sutton (2000) describes ***SPI Trigger*** factors which can affect start-ups including, the fact that the firm might be responding to influences from cooperating or competing organisations, or might be under pressure to take on one-time only projects for highly-valued customers. Some practitioners look for existing 'painful outcomes' and then determine how they can be remedied (Bach, 1998). Others state that companies should examine what business consequences have resulted from weak or ineffective processes (Kasse and McQuaid, 1998). Baskerville and Pries-Heje (1999b), in a longitudinal study of a Danish software company, demonstrate how customer objections to the absence of a basic development methodology and project management practices led company management to introduce SPI.

SPI may be triggered by negative customer feedback on products or services, such as an error-ridden premature software release (Demirors *et al.*, 1998) or just general ***Quality*** problems (Pitterman, 2000). However one study reports that SPI was triggered as the company concerned had leadership, technical, and customer satisfaction problems

(Batista and de Figueiredo, 2000). Another study shows how a Danish company received significant funding from the European Commission and were then contractually obliged to establish a QA group (Kautz *et al.*, 2000).

Paulk (1998), in advocating CMM usage, offers support for the finding in this thesis which links SPI with ***Business Events*** by maintaining that SPI should only be carried out where it is of benefit to the business. Market conditions and customer requirements can act as an ***SPI Trigger*** for the introduction of CMM (Johnson and Brodman, 2000), as can the desire for global recognition (Keeni, 2000). Alternatively, a customer-driven software capability evaluation may drive CMM-based activity (Hollenbach *et al.*, 1997).

Other studies detail how XP was introduced as a development methodology into organisations to deal with legacy applications which were difficult to maintain, complex to extend and unresponsive to fast turnaround (Namioka and Bran, 2004; McAnallen and Coleman, 2005). XP has also been introduced as an SPI remedy triggered by unrealistic deadlines, surprises late in a project, late delivery, excessive process overhead, and poor ***Quality*** (Grenning, 2001).

10.3.4 Evidence For – Hiring Expertise

In order to deal with ***SPI Triggers***, some companies look to their own resources for a resolution. However, in many instances, the nature of the ***Trigger*** event, and the type of SPI solution required, led many organisations to look externally for the 'answer'. Hiring the right person, however, is not a straightforward task as future success depends on the quality of the persons hired making it one of the most important things managers do (Hass, 1997). Also, as early stage companies go through different 'stages' of development, top management, who have the experience of being through these stages, are invaluable to such a company (Flood *et al.*, 2002).

Some companies recruit expertise because they don't feel they have the time to wait until their own staff are sufficiently trained and experienced. One company, embarking on a CMM-based improvement programme, who had a very clear understanding of what

they wished to achieve, decided to "buy expertise" in order to get the project under way as quickly as possible (Kelly and Culleton, 1999). This is also the case for small software companies who, faced with finding the resources to train people in SPI, tend to recruit senior people who are well educated and already trained for the task (Brodman and Johnson, 1994; Batista and de Figueiredo, 2000). One such software company, wishing to make significant modifications to its corporate culture, hired a software change agent to provide development and project management vision (Coleman Dangle *et al.*, 2005). Jarvis and Hayes (1999), in a collection of case studies demonstrating the benefits of SPI, show how expertise was hired by several different companies to:

- Introduce a comprehensive requirements management programme
- Establish a project support office or
- Implement a software reliability engineering programme.

Cusumano (2004) presents a case study of a successful start up software company. Following initial success, the company reached a critical size where improvements to the organisation were needed. At that point they hired some "top notch" engineers and experienced managers in engineering, marketing and sales. In addition the founders hired a new CEO to steer the company forward. One senior SPI commentator reasons that he was hired to his position because he "knows the solutions to certain problems" (Bach, 1998).

10.3.5 Evidence For – Process Inertia

Much of the ***Process Inertia,*** described in the literature, relates to employee resistance to SPI or to an organisation's inability to institutionalise a project-based SPI improvement initiative. Probably the best example in relation to this research is in Baddoo and Hall (2003) where the authors state:

> *One of the biggest obstacles to introducing any new practice is the unwillingness of practitioners to take them up. This problem often arises when practitioners perceive no incentive for giving up practices with which they are accustomed and feel comfortable with. It reflects the old adage 'why fix what's not broken'.*

Condon (2002) states that many software companies who achieve success relax and "rest on their laurels" and this, he believes, induces inertia, which can make future SPI more difficult. Dyba (2003) argues that large companies are less likely to change than small companies and that even when stimuli change, will continue to follow the same processes rather than change and risk failure. Telcordia Technologies also experienced ***Process Inertia*** problems when embarking on a ***Quality*** journey in the 1990s. Though every attempt at improving ***Quality*** provided some success the result was an overly-bureaucratic process that the software developers would not adopt (Pitterman, 2000). This can only be overcome through 'unfreezing' the factors maintaining current behaviour (Hardgrave and Armstrong, 2005). ***Process Inertia*** also occurs when changes, created through SPI are not institutionalised. One company report how, as part of an SPI initiative, they assessed two pilot projects, defined corporate processes, and created supporting manuals, yet no further institutionalisation took place after the projects completed (Hollenbach *et al.*, 1997). However, self-assessments and process audits have been successfully used by companies to highlight barriers to process institutionalisation (Laporte and Trudel, 1998). Acknowledging the risk of non-institutionalisation, Borjesson and Mathiassen (2004) believe that SPI iteration is the key to change acceptance arguing that SPI initiatives which execute only a single iteration never properly expose the new process to practice, whereas SPI initiatives, which go through several iterations, are more likely to overcome initial practitioner resistance and move into everyday practice. One approach to overcome employee resistance, is the use of 'software circle' discussion forums where issues can be highlighted and resolutions reached in a non-threatening environment (Biro *et al.*, 2000).

10.4 Summary

This chapter examined the support and evidence for the research theme **Process Evolution**. The literature support was presented under each of the sub-category headings. The next chapter investigates the level of published evidence to support the theoretical argument for the core category **Cost of Process**.

Chapter 11 Support for – Cost of Process

11.1 Introduction

This chapter explores the published material that provides support for the core category **Cost of Process** as presented in Chapter 8. The relevant literature is compiled and displayed in the same order as in Part II, and focuses on published support for the major categories linked to the core category.

11.2 Evidence For – Cost of Process (Bureaucracy)

Within this particular study ***Bureaucracy***, as described by the practitioners, refers to the administrative or 'non-productive' time and resources required to manage software process activity. As Fayad (1997) states:

> *Processes are commonly seen as extra bureaucracy only serving to make a project less effective. In far too many cases this is correct and process adoption is resisted.*

Whenever **Process Evolution** is treated as a project, engineers often complain and engage in "passive sabotage" as they want to do "real work" (Bach, 1998). But, even project managers resent processes that seem to be unnecessarily bureaucratic and do not actively support their work (Jalote, 2002). Bollinger (1997) lauds the Lockheed 'Skunkworks' approach to design and development where "bureaucracy is kept to a minimum and communication between technical peers is excellent". Whilst small organisations are designed to be innovative, large organisations become more formalised and develop bureaucracies which emphasise order and control and include role specialisation and division of labour (Dyba, 2003). To enable this, company personnel resources must be allocated to the task. Some argue that a dedicated budget should be allocated to SPI effort (Debou and Kuntzmann-Combelles, 2000). However, there is a minimum cost that must be borne for SPI work, irrespective of company size, and this is proportionately greater in small companies than large (Brodman and Johnson,

1994). Many small software developers are reluctant to commence an SPI programme as they think they cannot afford the investment (Kautz, 1998). Small software companies typically lack both resources and the ability to plan and execute an SPI programme (Kilpi, 1997) and therefore SPI can require an excessive capital outlay (Saiedian and Carr, 1997). As a result, small software companies recognise that these resources have competing demands leaving SPI programmes as a much lower priority.

In the case of a software start-up, the organisation generally dislikes ***Bureaucracy*** and has to compete with lean budgets in fast-paced markets, as well as continually making changes to their products during the development process (Cusumano, 2004). Speedy delivery may mean company survival (Anacleto *et al.*, 2004), but is frequently at the cost of formal process (Guerrero and Eterovic, 2004). Nonetheless, gaining product acceptance in the market place is a key objective for start-ups, and it is therefore vital to get even a low-functionality version of the product into customers' hands at the earliest opportunity (MacCormack, 2001).

11.3 Evidence For – Cost of Process (Documentation)

Forward and Lethbridge (2002) observed that small- to medium-scale software projects had little or no software ***Documentation*** and that, within these projects, budgetary and schedule constraints, coupled with time to market demands, left limited resources available for ***Documentation*** work. The emphasis in small companies is on product development. This takes precedence over document development and time spent documenting can be classified as time not spent developing product features (Ambler, 2005b). Creating ***Documentation*** carries a cost and unless this cost can be justified, such as, for example, the user has requested certain documents and is willing to pay for them, then not creating them can be more cost effective (Ambler, 2005b). One company who introduced XP into a more formalised environment attempted to build a product with 'sufficient' ***Documentation*** to enable effective maintenance (Grenning, 2001). This supports an experiment conducted in Hong Kong where, in order to reduce the **Cost of Process** for small projects, the amount of ***Documentation*** was minimised (Leung and Yuen, 2001).

Most PC and Internet companies put a premium on code and will therefore tolerate incomplete ***Documentation*** (Cusumano and Yoffie, 1999). Constantine (2001a) reports on how product also has primacy in Microsoft where the only document that has any real value is the source code and "the developer's job is to write code, not high-level documents". As Highsmith (2004) puts it, "to use key engineering staff effectively, project managers should offload nearly all compliance ***Documentation*** to administrative staff". This supports evidence from Woodward (1999) who states that in many companies staff are under pressure to focus on product, and to minimise unpaid ***Documentation*** or project recording work. The key to success is producing the right document at the right time and documents should only be produced if they have value for the project participants and stakeholders (Turk *et al.* 2002).

11.4 Evidence For – Cost of Process (Communication)

Many organisations are avoiding the requirement to create project ***Documentation*** by co-locating their developers and taking advantage of informal ***Communication*** mechanisms and ***Tacit Knowledge***. Constantine (2001b) argues that inadequate requirements are less costly to resolve in a project if you are co-located with your development colleagues. It is claimed that co-location and informal ***Communication*** offers increased development speed and decreased time to market (Baskerville *et al.*, 2001). Developer co-location has also been linked with increased team focus (Sliger, 2004), reduced defect detection times (Ebert and De Neve, 2001), increased morale and productivity (Javed *et al.*, 2004), and improved project scheduling (Teasley *et al.*, 2000).

Ebert and De Neve (2001) propose that engineers working on the same project should sit, not only in the same building, but also in the same room. This is supported by Eischen (2002) who believes that the more social the development process is, the better. Small companies have an advantage over large organisations in this regard, as often the managers in small organisations sit alongside the engineers, and this arrangement allows for other supporting services to be co-located (Brodman and Johnson, 1994). As a result, small companies can use face-to-face ***Communication*** more effectively than large

organisations (Dyba, 2000), and this form of interaction reduces the need for external ***Documentation*** (Cockburn, 2002b).

Kraut and Streeter (1995) caution against the use of more formal methods of ***Communication*** during software development. Their data suggests that these methods will only be successful if supported by interpersonal, informal ***Communication***. Others agree, arguing that focusing too much on traditional 'Blueprint SPI' can result in a disregard for ***Tacit Knowledge*** (Aaen, 2003).

11.5 Evidence For – Cost of Process (Creativity and Flexibility)

According to Chisnall (1987) ***Flexibility*** is an attribute that should be highly prized by small entrepreneurs in their own management behaviour, in staffing arrangements and in responses to their customers. This is particularly true for software companies who operate in dynamic and ever-changing, markets. The creative team, essentially present in early life software product companies, may even require a lack of process, as well defined process supports known activities but often restricts unknown activities (Armour, 2001). For start-up software companies, the key to success is for developers and managers to "create enough structure to keep projects under control but not so much that the process stifles creativity and flexibility" (Cusumano, 2004). Kelly and Culleton (1999) discuss the small software organisation thus:

> *The culture of smaller organisations can often be characterised as creative, dynamic and innovative. The success of these organisations is often due, in no small part, to the creativity and innovation of their employees. SPI is frequently viewed as the antithesis of these qualities, leading to bureaucracy that restricts the freedom of individuals. [Any] SPI initiative should not stifle creativity.*

Existing process assessment and improvement models fail to take account of the fact that small companies are more flexible (Horvat *et al.*, 2000), and react more quickly than large companies (Nunes and Cunha, 2000). Whilst some level of structure is essential, the challenge facing software companies is how much structure is appropriate.

It is claimed that too much structure can suppress ***Creativity*** (Highsmith, 2004). Too much structure can also have a detrimental effect on a software start-up, as ***Flexibility*** is needed to accommodate changes in personnel, infrastructure, and product requests (Sutton, 2000). ***Flexibility*** has a human dimension, and it is a necessary attribute in a start-up company's developers, as they will be constantly taking on new tasks, filling new roles and using their experience in new and innovative ways. Software SMEs are known to thrive in unstable environments and are better equipped to adapt flexibly to changes in technology and competition than their large-scale counterparts (Baskerville and Pries-Heje, 1999b). Fayad (1997) summed it up thus: "software is a creative process, not an assembly line".

11.6 Evidence For – Cost of Process (Process Models)

11.6.1 Evidence For – Cost of Process (XP /Agile Methods)

In this research study, managers gave accounts of very significant cost savings through the use of XP. Though in the majority of instances this was not quantified, they claimed to have improved their delivery capability, and reduced the general process overhead they equated with ***Documentation*** and ***Communication***. They also, again without quantification, believed they retained ***Creativity*** and ***Flexibility*** through the 'lighter' process that is XP.

Within the literature, much of the cost savings reported from using XP is through reduced ***Documentation***. Ambler (2005a) states that though there is often a requirement for ***Documentation*** both externally and internally, within XP it should only be written as it becomes necessary. Highsmith (2002) argues that the use of 'barely sufficient methodologies', like XP, minimise the ***Documentation*** requirement, and therefore the cost, in a software project, and this has found support elsewhere (Murru *et al.*, 2003). Law and Charron (2005) confirm this, as in a software development project, they successfully used minimal ***Documentation***.

Though XP is often criticised from a maintenance perspective, because of its lack of a document base, maintenance engineers can benefit from XP's test-first development as the tests themselves act as ***Documentation***. Tests written in this way are of benefit to many developers who, when learning something new, prefer to start at the source code level (Grenning, 2001). Thus, extensively documenting the source code is highly recommended at all times (Kutschera and Schafer, 2002) and can be a source of cost reduction. In addition XP's extensive use of prototyping means users get an opportunity to use working software rather than having to sign-off the system based purely on paper documents (Simons, 2002). In addition, benefits, in the form of low staff turnover and significantly reduced overtime, have been reported with the use of the Scrum methodology (Schatz and Abdelshafi, 2005) and a tailored XP deployment resulted in improved productivity (Drobka *et al.*, 2004). A reduction in ***Communication*** overhead, through using XP, has also been documented, (Law and Charron, 2005; McAnallen and Coleman, 2005).

However, there do appear to be limits to XP's ability to reduce ***Communication*** overhead. Where XP teams are distributed, there is a need for greater ***Communication*** formality and this creates additional developer overhead and decreases agility (Lindvall *et al.*, 2004). Also global software development, where co-location is not possible because of the use of distributed teams means that good ***Documentation*** of requirements and design is essential (Turk *et al.*, 2002) and even the 'western orientation' of agile methods has been documented as potentially having managerial cost implications for their usage in other cultures (MacGregor *et al.*, 2005).

11.7 Evidence For – Cost of Process (Process Improvement Models)

11.7.1 Evidence For – Cost of Process (ISO 9000)

Like much of the literature, accounts of the use of ISO 9000 are typically success stories with few published commentaries on related implementation difficulties, a fact borne out by Stelzer and Mellis (1998). The theory presented in this thesis therefore challenges this and bolsters anecdotal evidence of the overhead reportedly associated with the ISO

certification process. An exception to the published norm is Kasse and McQuaid, (1998) who describe the ISO 9000 series as, though having been intended to allow countries to trade with each other and maintain ***Quality***, instead acted as an imposition where, "the pressure to be ISO certified pushed organisations to develop pages of documented processes that were rarely known or used throughout the organisation". Fitzgibbon (1996) outlines how, in Canada, by 1996, "less than two dozen of the 10,000 companies that design and develop software have registered their QMS to ISO 9001", thus, by inference, suggesting that the published studies are not a true reflection of the practice reality.

Some companies use ISO 9000 as the basis for CMM-oriented process improvement (Laporte and Trudel, 1998; Jalote, 2002). This suggests that when ISO certification is achieved the effort required for further SPI is reduced and perhaps allows the organisation to move from a general standard to a more software specific-standard. But others have experienced difficulty in making the ISO-CMM transition (Mathiassen *et al.*, 2001). Nevertheless there is clear evidence from the literature, as discovered in this research, that many software companies overwhelming pursue ISO 9000 certification for marketing reasons (Biro *et al.*, 2000; Horvat *et al.*, 2000; Thomson and Mayhew, 1997). Andres *et al.* (1997) summarise it best stating "companies pursue ISO 9000 certification for several reasons: customers ask for it, requirements for bids include it and, even sometimes, companies are truly interested in improving the way they do things". A study of Northern Ireland software organisations by McFall *et al.* (2004) found that many indigenous companies found adhering to ISO 9000 standards difficult, and is viewed by the companies as a "badge for marketing" rather than a model for continuous improvement.

There are some examples of negativity towards ISO 9000 in the literature. Bach (1998) criticises the standard giving an account of how his organisation preferred a problem-based SPI approach to using ISO 9000 or CMM, "which mandate that certain processes and institutions be put in place, regardless of the actual problems faced by the companies and projects". Also ISO standards can be costly in terms of resources as it

has high ***Documentation*** demands (Woodward, 1999), one published example illustrating how an employee took longer preparing the necessary ISO compliance sheets than the actual three-page document that was being certified (Dicks, 2000).

11.7.2 Evidence For – Cost of Process (CMM and CMMI)

Much of the discussion in the literature regarding SPI relates to implementing CMM and CMMI. But the cost of implementing CMM/CMMI-based improvement can be very high, from in excess of 100,000 dollars (Saiedian and Carr, 1997), 180 person-days on process redefinition, 70 person-days on training and 20 person-days on evaluations on a programme to secure CMM Level 2 accreditation (Kelly and Culleton, 1999), to 45,000 dollars for the initial assessment activities and 400,000 dollars to move from level 2 to level 3 (Humphrey *et al.*, 1991). Herbsleb *et al.* (1997) conducted a multiple case study of companies who had experienced success with CMM. However, the majority of those companies felt that implementing CMM had cost more than expected, leading the SEI-employed authors to concede that CMM is neither a cheap nor quick fix. In addition, CMM can negatively impact a small software company's competitive potential (Bach, 1994). Brodman and Johnson, (1994) report a number of resource-related difficulties that small companies have in attempting to implement the CMM, a fact conceded by the CMM's own proponents (Paulk, 1998).

But much of the perceived excessive CMM-related cost is ***Documentation***. Humphrey and Curtis (1991) accentuate the ***Documentation*** need stating that mature companies with mature processes have "well-run software projects" which "leave a clear documented trail" including approval documents, change-review procedures, and minutes of control board meetings. Brodman and Johnson's (1994) study reflects this, detailing how one of the main issues cited by small software companies in relation to using CMM is excessive ***Documentation*** and that to follow CMM's requirements in small projects would be, in the companies' opinion, "counter-productive". Beecham *et al.* (2003) also comment on how project managers believe that CMM has too much paperwork. For this reason, companies often move to alternatives, such as XP,

particularly when there is a pressing demand to get a product onto the market (Reifer, 2003).

11.8 Summary

This chapter examined the theoretical support and evidence for the core category **Cost of Process**. The literature support was presented under each of the sub-category headings. The final part of the thesis presents a wider discussion of the grounded theory developed in this study, explores its implications, sets it in context, discusses its limitations, and presents conclusions. How the theory should be evaluated and assessed is also considered.

Part IV Discussion

Part IV – Overview

The final part of this thesis contains two chapters. Chapter 12 examines the methodology used in the study, provides a full evaluation of it, discusses how the findings can be verified, and how the results of the study should be judged.

Chapter 13, the concluding chapter, initially refers back to the research question and the research objectives and considers how these initial goals have been met by the study. It then provides a detailed summary of the work and draws a set of conclusions. The chapter then presents the research contribution and proceeds to consider the implications of the findings for both practice and research. Finally it discusses the limitations of the study and makes a number of proposals for further research.

Chapter 12 Evaluation

12.1 Introduction

This chapter will explore how the grounded theory produced by the study can be evaluated. How the theory can be verified is discussed in detail. Then the criteria to be used for judging the work are presented and the study is assessed according to these criteria.

12.2 Evaluating the Study

How a grounded theory is presented offers a number of challenges to the researcher in terms of structure, level of detail included, and how the data are portrayed to display evidence for the emergent categories. It has been suggested that the author should write the theory in such a way that it clearly demonstrates how the concepts emerged and were developed from the data, how the researcher reached the point of abstraction, and how the core categories were generated (Goulding, 2002). In terms of what should be included in a grounded theory thesis, Strauss and Corbin (1998) offer the following advice:

> *It all goes back to answering the questions 'what was this research all about?' and 'what were the main issues and problems with which these informants were grappling?' Then there should be sufficient conceptual detail and descriptive quotations to give readers a comprehensive understanding of these.*

As this study used the Strauss and Corbin version of grounded theory, it complies with their recommendations above. Significant effort has been made in Part II of the thesis to show how grounded theory was used, how the research process evolved, and how the categories emerged. The research explains the issues that concerned the participants, how problems arose, and how they set about solving them. Goulding (2002) argues that data, such as quotations, should only be used to provide credence for theory and theoretical development, rather than to supply low levels of description. This approach

to using quotations is consistent with Strauss and Corbin (1998) who suggest that data should only be presented with sufficient analytical comment, and state that propositions should be interweaved, using the results of coding and memos, with carefully selected words or phrases combined with theoretical points. This researcher incorporated the advice of Strauss and Corbin in the work by making extensive use of explanatory quotations from the managers to describe what was going on in the practice surroundings. Any quotations included to explain the theoretical setting were framed with full analytical commentary.

12.3 Verification of the Theory

The issue of verification of a grounded theory study is one which distinguishes the positions of Glaser and Strauss, the founders of the methodology (MacDonald, 2001). Glaser (1992) rejects the notion stating that a grounded theory is not verified. He argues that the theory is consistently modified by constantly integrating new data into it. To Glaser, grounded theory merely produces hypotheses and nothing more and these need not be verified or validated because that is the responsibility of verificational studies which are carried out using a different methodology. Therefore, Glaser's approach leans towards discovery rather than verification.

Strauss and Corbin's position is somewhat different. They argue that theories are conceived, elaborated on, and checked out, in that order and this is facilitated through the concurrent processes of induction, deduction, and verification (Strauss and Corbin, 1998). Table 12.1 shows the differences between Glaser, and Strauss and Corbin on the issue of verification. As the Strauss and Corbin version of grounded theory was used in this study, their original approach to verification was employed (Strauss and Corbin, 1990).

Table 12.1 Verification - Glaser Vs Strauss and Corbin (MacDonald, 2001)

Glaser (1992)	Strauss and Corbin (1990)
A GT (Grounded Theory) is not verified. Rather, it is modified to accommodate new data by integrating them into the theory.	A GT is discovered, developed and provisionally verified through systematic data collection and analysis.
Hypotheses need not be verified or validated, because these are the properties of verificational studies which require a different methodology. These two types of methodology should be seen in sequential relation to each other, with hypothesis discovery methodology coming first, then the most relevant hypotheses being tested with a different type of methodology.	Alternating between collecting and analysing data allows emerging concepts to direct sampling and also allows verification of provisional hypotheses.

12.3.1 Generalisation

On the issue of theory generalisability, differences arise between the two founders of grounded theory. These are illustrated in Table 12.2.

Table 12.2 Generalisability - Glaser Vs Strauss and Corbin (MacDonald, 2001)

Glaser (1992)	Strauss and Corbin (1990)
Generalisability is related to verificational studies and not to GT.	The purpose of GT is to specify the conditions that give rise to specific sets of action/interaction pertaining to a phenomenon. Thus, a GT is generalisable to those specific situations only.
What applies to GT is its generalisability from a substantive theory of limited scope to a process of larger scope with parsimony, based on its ability to fit, work and be relevant.	All theories are temporally limited and always provisional. Thus, there can be no time and context-free generalisations of grounded theory. To the extent that situations and conditions in the new context are similar to the context in which the theory was developed, then a GT may be generalisable.

Strauss and Corbin (1998) contend that the use of a theory-building methodology is to build theory and, therefore, in grounded theory studies, the researcher is talking more about explanatory power than generalisability. For Strauss and Corbin, context is always

relevant to any grounded theory study, whereas generalisability describes a situation that is essentially context-free. Within this study, the research covered concepts and their relationships and explored the conditions under which events, happenings, actions and interactions could occur and the ensuing consequences. The study also examined dimensional variation and provided explanation. Therefore, it can be stated that if the developed concepts are sufficiently abstract, they are likely to occur in similar or slightly different form in other software product companies. Yin (1994) describes this approach as "analytic generalisation" where the generalisation is of theoretical concepts and patterns. This is distinguished from the more typical "statistical generalisation" whereby an inference is made about a population based on data collected from a sample. In this research the outcome of the "analytic generalisation" process has resulted in a general conceptualisation of the technological, human and organisational factors linked with implementing software process and process improvement programmes. This outcome has implications for both practice and research and contributes to our knowledge of SPI. The implications for practitioners and researchers from this study and the contribution of the research are discussed in greater detail in Chapter 13.

12.4 Assessing a Grounded Theory Study

Section 4.9.1 outlined Strauss and Corbin's (1990) three sets of criteria for judging a grounded theory, namely assessing the theory itself, assessing the adequacy of the research process, and determining if the theory is sufficiently well grounded. The next sections will look at each of these in turn.

12.4.1 Judging the Theory

The following four factors are suggested by Strauss and Corbin (1990) to judge a grounded theory:

- Fit – The theory must fit the substantive area and correspond to the data.
- Understanding – The theory makes sense to practitioners in the study area.
- Generality – The theory must be sufficiently abstract to be a general guide without losing its relevance

- Control – The theory acts as a general guide and enables the person to fully understand the situation.

In terms of *fitness*, there is always a danger that researchers develop a theory, of the studied phenomena, that embodies their own ideals and perceptions as well as popular views and common myths. When these theories subsequently do not fit the developed categories very well, the consequences are often a forcing of the data to do so, and rejection of the data that do not fit or cannot be forced to do so. Therefore it is imperative that, for a grounded theory to fit, it is induced from the diverse set of collected data. In this way it is closely related to the actual realities of the substantive areas and applicable to dealing with them. The theory developed in this study has faithfully adhered to the inductive methods contained in the grounded theory methodology. Though the researcher is a "cultural insider", his professional expertise was used merely to assist theoretical sensitivity rather than drive the theoretical conclusions. The constant comparative method, overturning of some early categories as new data came to light, generation and testing of interim hypotheses, and constant re-evaluation of the interview transcripts ensured that researcher bias was minimised and theoretical fit maintained.

A theory that closely represents the realities of an area will make sense and be *understandable* to practitioners in that area. This understanding is important in that it encourages the theory's usage, increases awareness of the issues faced, and provides a mechanism for instigating change. In developing the grounded theory in this study, the concepts and categories were carefully developed to support understanding by software development personnel. Where appropriate, in-vivo codes were used. In-vivo codes have an important role to play as they are the actual words or phrases used by the practitioners and thus reflect their reality as they perceive it. Using in-vivo codes ensured that the developed theory closely corresponded to the realities of software process in practice. Also, in Study Stage 2, some of the Stage 1 participants were re-interviewed in light of the Stage 1 findings and the resultant hypotheses. The developing theory was presented to the re-interviewed participants and the new interviewees who

had not been included in Stage 1. To prevent potential response bias, the theory was only presented to the interviewees after the interviews had been conducted. The reactions of the interviewees to the presented theory was very positive and one which they believed represented their reality as they perceived it.

From a *generality* perspective the researcher must ensure that the categories contained within the theory should not be so abstract as to lose their sensitising characteristics, but yet should be sufficiently abstract to make the theory a general guide to constantly-changing situations. The issue of generalisation in relation to this study has already been discussed in detail in 12.3.1.

Glaser and Strauss (1967) argue that, "a theory with '*controllable*' concepts of sufficient generality, that fits and is understandable, gives anyone who wishes to apply these concepts to bring about change a *controllable theoretical foothold* in diverse situations". In summary, the theory should ensure the person who uses it has enough control in the situations they encounter to make the application of the theory worth considering. The theory should allow the person to be able to understand and analyse situations, be able to predict change and its consequences, and be capable of revising his actions, or the theory itself, if appropriate. To enable this, the theory must provide a sufficient number of categories and concepts and explain the relationships between them. The theory in this study has achieved this by providing a comprehensive set of categories with detailed interrelationships to explain how process is formed and the reasons for change. Using both the methodological tools, and those provided by the supporting software, each category and the strength of relationships between them has been fully explored and tested. Hypotheses, derived from, and related to, the controllable situations which face software practitioners, have also been tested. Deviant cases have been sought to ensure theory robustness and applicability. Through these approaches, and the investigation of the SPI literature, a comprehensive theory was developed which can be considered by practitioners faced with situations demanding an SPI solution.

12.4.2 Adequacy of the Research Process

In judging the quality of any research study designed to generate theory, the reviewer should be able to make judgements about the research process (Strauss and Corbin, 1998). As readers are not actually present during the research activity, they must be provided with information to allow them to assess its adequacy. This information can be presented in the form of questions:

1. How was the original sample selected?
2. What major categories emerged?
3. What were the events/incidents/actions that pointed to the categories?
4. On the basis of what categories did sampling proceed?
5. What were some of the hypotheses pertaining to conceptual relations among categories and on what basis were they formulated and validated?
6. Were there instances where hypotheses did not explain what was happening in the data? Were hypotheses modified?
7. How and why was the core category selected? Was this sudden or gradual? (Strauss and Corbin, 1998).

The first question on how the original sample was selected was answered in detail in 5.2. The second question relates to the major categories and these have been presented in Part II. Category development continued throughout the thesis and Part II contains both detailed descriptions of each category and diagrammatic networks showing category attributes and interrelationships.

The incidents and actions, that pointed to the categories, and referred to in question 3, emerged during the interview analysis, and are discussed in detail in conjunction with each category's presentation in Part II of this thesis. Strauss and Corbin suggest that to support the identification of categories the researcher should look for phrases such as "because" or "since". Then, to find the consequences, the researcher should follow up

on such terms as "as a result of" and "because of". There are numerous examples of these phrases in the field data. For example, the following set drove the development of the ***Business Event*** and ***SPI Trigger*** categories:

> *But because we're now on a world scale we have to introduce other process-support type software such as Lotus Notes and SupportForce. [Company 2]*

> *You can't move on to the next project because you keep pulling resources back because issues are getting reported by customers and they want fixes. [Company 6]*

> *The process was established because we never shipped on time, we were always over budget. [Company 8]*

The consequences (that Strauss and Corbin referred to) for each of these actions, was some form of ***SPI Focus*** to resolve the problem, which in a number of instances also involved ***Hiring Expertise***.

Question 4 is concerned with the categories that initiated subsequent theoretical sampling. For example, it was clear from the very early interviews, particularly in the start-up companies, that ***Background of Software Development Manager*** was central to the initial process that a software company used. This drove an early line of questioning as the manager's background was clearly being used to set-up the initial development process. Later on in the study however, when larger companies were interviewed, it became clear that ***Hiring Expertise*** was a key solution to process difficulties. Almost all of the initial interviews in the study were with CEOs and CTOs, essentially people who were involved at the company's outset. Later interviewees had been recruited some time after the company's establishment and are exemplars of the ***Hiring Expertise*** category. Most were recruited directly to senior positions to resolve a development issue the organisation was experiencing, or to take advantage of a business opportunity and lead the company in a new direction. The way this category emerged influenced subsequent

interviewing, and each successive new interviewee was asked why they were recruited and what were their reasons for joining their current organisations.

Questions 5 and 6 relate to how hypotheses are formulated and validated in a grounded theory study and whether they explained in full what was happening in the data. How the hypotheses were formulated and validated in this research is described in 5.5.1. Whilst all of the hypotheses were 'tested' and verified in Stage 2 of the study, one hypothesis (H6) – *Within Irish software product companies, restrictions are imposed on team sizes to achieve minimum process requirements* – failed to develop further during that Stage. The following sample Stage 2 responses explain why.

> *The XXXX product, because it is so big, could pull in pretty much everyone if it wanted to. But the limiting factor there is a budget rather than a conscious decision. With XXXX and YYYY, in an ideal world you would put more bodies onto them, but the limiting factor at the moment is money.*

> *The team is getting bigger. That's not a conscious decision. We just need more functionality.*

> *In R&D, we haven't got to that scale yet. We have been in a situation where there should nearly be a* ***minimum*** *team size. There was a case recently where our second product wasn't being worked on because there was a rush to build product 3 and no one was working on it. So I would say we had the other problem.*

Though not fully supporting hypothesis H6, the findings in Stage 2 did support the remaining hypotheses and these in turn were incorporated into the theoretical categories and attributes. However, a number of categories emerged in Stage 2 which were not directly included in the Hypotheses list in Stage 1. These include ***Process Erosion***, ***Process Inertia***, ***Communication***, ***Tacit Knowledge***, ***Creativity***, and ***Flexibility.*** The

field data from the diversity of companies used for Stage 2 helped these categories to emerge.

Question 7 raises the issue of how quickly the core category was selected. Once the core category does emerge, the researcher must return to constant comparison with the data to see if it has the power to explain what is going on within the data (Schreiber, 2001). The selection of the core category, **Cost of Process**, was made during Stage 2 of the study, though attributes of it had been apparent in Stage 1. In selecting the core category, the researcher closely followed the steps recommended by Strauss and Corbin (1998), outlined in 4.8.4. Analysis of the Stage 1 data showed that companies were concerned with ***Documentation***, issues around the 'weight' of process and ***Bureaucracy***, the desire to retain ***Creativity/Flexibility,*** and the basic lack of resources to implement SPI. Strauss and Corbin suggest that considerable manipulation of the data is required before a core category emerges. Whilst each of the Stage 1 categories "told part of the story", none "captured it completely" (Strauss and Corbin, 1998). In such an instance a "more abstract term or phrase is needed, a conceptual idea under which all of the other categories can be subsumed" (Strauss and Corbin, 1998). It was the additional analysis from Stage 2 that demonstrated that process was being avoided, or compromised, for reasons of 'Cost', and that the companies' concerns were properties, or dimensions, of that 'Cost', that crystallised the more conceptual **Cost of Process** as the core category. That this did not occur until Stage 2 of the study provided confidence to the researcher that the correct core category had been identified. This was partly because of the ever-present danger in a grounded theory of "premature selection" (Glaser, 1978) of a core category, but more importantly because Stage 2 specifically aimed at broadening the study, incorporating different ***Market Sectors*** and seeking out deviant cases, whilst simultaneously testing the hypotheses derived at the end of Stage 1 and engaging in constant comparison with the prior interview data.

12.4.3 Grounding the Findings

Strauss and Corbin also provide a list of criteria to assist in determining how well the findings are grounded. These are:

1. Are concepts generated?
2. Are the concepts systematically related?
3. Are there many conceptual linkages and are the categories well developed? Do categories have conceptual density?
4. Is variation built into the theory?
5. Are the conditions under which variation can be found built into the study and explained?
6. Has [the theory generation] process been taken into account?
7. Do the theoretical findings seem significant?
8. Does the theory stand the test of time? (Strauss and Corbin, 1998).

In relation to question 1, the foundations of any theory are a set of concepts grounded in the data (Strauss and Corbin, 1998). The concepts, and how they emerged, were discussed in Part II. Table 5.2 shows an example of some of the codes produced from the open, axial, and selective coding processes. A full list of the codes used is provided in Appendix C. The codes include both in-vivo codes, terms used by the practitioners, and conceptual codes assigned by the researcher. Many of the researcher-assigned codes denote concepts generated from the analysis of the data, described in Part II. Questions 2 and 3 examine the linkages and relationships between concepts. Chapters 6-8 show through the use of network diagrams how the concepts and categories are related, what categories act as predecessors and successors within the theory, and how the categories link to the core category and research themes. Four diagrams were presented, one to illustrate the theoretical network and others showing decomposition to research theme and core category level. Further theoretical sub categories and properties exist, but for reasons of clarity these are not shown. Nonetheless, the network diagrams, and the analytical supporting commentary contained in Chapters 6-8, show the conceptual density of the categories and give the theory its explanatory power.

For questions 4 and 5, Strauss and Corbin suggest that variation is important because it signifies that a concept has been examined under a range of different conditions and dimensions. Though this research is concerned with indigenous Irish software product

companies, the researcher has endeavoured to incorporate the views of as wide a range of practitioners as possible. It can be seen from the list of companies in Table 5.1, that the respondents came from a cross section of company types and ***Market Sectors***. In addition, the researcher attempted to interview companies across all size ranges from some of the largest indigenous companies to new start-ups. At the end of Study Stage 1, the researcher was concerned that the interviews to that point did not contain enough diversity and so focused on increasing variation in Stage 2. This culminated in interviews with companies operating in the following markets: Public Sector; Medical Devices; Telecommunications; HR Solutions; Games Infrastructure; Personalisation Systems; and in re-interviews with companies in the Enterprise Systems and Interactive TV segments. By widening the interview base and increasing the range of field data, the prospects of phenomena relating only to specific market domains, or company size sectors, was dramatically reduced.

Question 6, which relates to the grounded theory process of data collection and analysis, is important because it enables theory users to explain action under changing conditions (Strauss and Corbin, 1998). Whilst the interviews represent a snapshot of the period in time in which they were conducted, much of the focus of questioning related to the conditions prevalent in the company, how these changed, and what circumstances or events gave rise to these changes. Much was made of how things used to be in the organisation concerned, how things were at the time of the interview, and the evolutionary path that was followed to arrive at that juncture.

Questions 7 and 8 are related and raise the issues of how important the theoretical findings are. Strauss and Corbin argue that a researcher could merely go through the motions and arrive at findings which are mundane and insignificant. Three published papers (Coleman, 2002; 2004; 2005), based on this study, support this researcher's belief that the research findings are significant and add to the literature on SPI. The significance of the findings, their implications for practice and research, and the contribution of this work are discussed in the final chapter.

12.5 Summary

This chapter described how the theoretical framework presented in Part II of this study should be evaluated. A discussion then followed on verification of the research and the criteria for judging the work. Three criteria were discussed how the study was assessed, according to these criteria, was then demonstrated.

Chapter 13 Summary and Conclusions

13.1 Introduction

This is the final chapter in this thesis and provides the summary and conclusions of the study. The original research questions and study objectives are revisited and compared with the actual outcomes of the thesis. The research contribution is presented and the implications of the study findings are explored and discussed. In addition, the study's limitations are examined from several perspectives and finally some future research options are presented.

13.2 Revisiting the Research Question

This research set out to explore two specific research questions and a number of related questions (Section 1.2.1):

- *What software processes are software companies using?*
- *How are software processes initially established in a software company?*
- *How do the software processes, that software companies are using, change?*
- *What causes these software processes to change?*
- *How do the operational and contextual factors, present in organisations, influence the content of software processes?*
- *Why are software companies not using 'best practice' SPI models?*

To enable these issues to be properly explored, boundaries were placed on the research setting, limiting the study to an investigation of indigenous Irish software product companies. This allowed for the analysis of a discrete group, and facilitated the exploration of key questions relating to process establishment in the organisations concerned, and the potential to document the changing nature of the development processes. A narrowing of the scope and the selection of the study group produced the following set of research objectives:

- To provide a new perspective on software process as it is practiced in software development
- To explain the role of software process and SPI in software product companies
- To investigate the factors that influence software process evolution in software product companies
- To build theoretical concepts that are grounded in the voices and experience of Irish software development managers.
- To develop and incorporate the overall findings into a theoretical framework that has explanatory and descriptive power.

This researcher would contend that all of the objectives outlined above have been met in the study. The research process has produced a theory which has been grounded and verified in accordance with the principles espoused in Strauss and Corbin (1990; 1998). Data collection took place between January 2003 and January 2005 and all interviews were fully transcribed. The managers were unfailingly courteous and generous with their time, and the opportunity for the researcher to hear 'war stories' from practice was especially interesting. All 25 interviews, across 21 companies, were open coded and underwent the process of constant comparison. Axial coding, where concepts are identified, began mid-way during Study Stage 1. Selective coding, whereby coding takes place around core categories commenced towards the end of Stage 1 and throughout Stage 2. Using the coding approaches, and the constant comparison techniques, ensured that a key research objective was met, namely that the theoretical concepts generated were grounded in the voices and experiences of the respondents. These theoretical concepts, as Part II demonstrates, were then developed as a detailed theoretical framework.

However, a note of caution should be recorded on the use of grounded theory in research work in SPI. Though this author would argue that the methodology has much to offer such research, the nature of this type of study means that the prior experience of the researcher can significantly influence methodological success. Bringing grounded theory's 'unconventional' approach to the area of SPI has the potential to provide major

challenges to the novice researcher. This author believes that, to succeed, a grounded theory researcher should be both experienced in conducting detailed research studies and a "cultural insider" as described in 4.7. The absence of these credentials could prove fatal for novice researchers, who may wish to use grounded theory in software process studies, and suggests that an alternative methodology should be considered.

13.3 Summary of the Findings

The research has answered the list of research questions identified at the outset of the study. Firstly, on the issue of *what software processes are software companies using*, the study has found that no company is using an 'out of the box' process model but rather all are using some kind of proprietary software process, which to a greater or lesser extent is based on standard models. All of the companies concerned engage in ***Process Tailoring*** and have adapted the software process to their own particular operating context. This operating context reflects the size of the company, the market in which they are operating, the types of projects in which they are engaged, such as new development versus modification/enhancement, and other individual project factors.

One of the key theoretical themes addressed by the research was **Process Formation** which related to *how process is formed or created* within a software product company. The findings show that this depends on several factors. The main one relates to the ***Background of the Software Development Manager***. This describes how the expertise accumulated by the person tasked with managing the initial software development effort dictates what the start-up software process will be. The final shape the process model takes will be influenced by additional factors including the demands of the market in which the company operates, their own and the founder's ***Management Style***, and the culture of the organisation. Different market segments have been shown to have different requirements. A pharmaceutical or medical device market may have to satisfy external approval bodies, such as the FDA, and therefore any software process must allow for ***Traceability*** and auditability. By contrast, the Internet domain, a fast-moving environment with ongoing change, places a premium on having products delivered quickly. Companies adjust their processes to take these factors into account by tailoring

the process they have decided to use to accommodate these demands. In addition whether the ***Management Style***, used within the organisation, is controlling and directive, or consensual and involving, will further influence how closely developers adhere to the firm's defined working methods.

The second key theoretical theme of the study, **Process Evolution**, addresses another of the research questions, that of *how and why development processes change* within software product companies. There is no evidence from the study data to suggest that practitioners are proactive in making changes to their development processes. Most respondents reported themselves satisfied with their current processes and, whilst these processes worked, they were not going to adjust them for fear of 'breaking something'. This means that process improvement is, in essence, *reactive*. Managers instigate SPI as a reaction to ***Business Events*** with which the current process cannot cope. Managers, therefore, must change the process in response to these ***SPI Triggers***. The field data shows that many of the companies feel they don't have the capability to deal with the change from within their own resources and, therefore, hire an individual externally who has the necessary expertise to deal with the ***Business Event***.

However, the effects of SPI are typically limited. The findings from the field data suggest that, following SPI implementation, over time ***Process Erosion*** occurs and leads to a point where a ***Minimum Process*** is operational. The ***Minimum Process*** is a working process which is 'barely sufficient' to satisfy the organisation's business objectives. Different projects place different demands on the process, and in this way the *operational and contextual factors, present in organisations*, referred to in the research question, do *influence the content of software processes*. The factors which affect the type of process established, (***Market Requirements***, ***Management Style***, ***Process Tailoring***), also act as inputs here. For example, a patch release, to remedy a small product fault, may follow a 'lighter' process than new product development. In addition, within the study companies, management complicity with developers often leads to process 'workarounds', or process short-circuiting. The periods between SPI initiatives witness ***Process Inertia***, wherein the existing process is capable of satisfying all of the

business demands that arise. Whilst this situation prevails SPI remains inactive. The cycle restarts again when the appropriate ***Business Event*** triggers the necessity for change.

The final research question addressed in the study, w*hy are software companies not using 'best practice' SPI models* produced the study's core category **Cost of Process**. Implementing and maintaining any SPI initiative incurs significant cost, and the financial and time implications of introducing some of the commercial SPI and quality models was discussed in 11.7. Significantly, the resources required to implement SPI are proportionately much greater in smaller companies, and those smaller companies intent on, firstly, survival and then stability, have many competing and higher priorities than SPI. As all of the study companies, at time of interview, fell into the EU-defined SME category (Section 2.3.1), it is therefore perhaps not surprising that they would reflect greater hostility to SPI models that required them to divert resources from what they would perceive as more deserving activities. For many of the interviewees, SPI creates an additional burden or weight to their development efforts resulting in increased ***Documentation*** and ***Bureaucracy.*** Companies, to reduce their process overhead, substituted verbal ***Communication*** for ***Documentation***. Development teams were co-located to ensure ease of verbal exchange and reduce the need for the written word. Even larger companies attempted to reduce ***Documentation*** cost by decomposing teams into smaller, more manageable, units. A benefit of doing this was an increase in ***Tacit Knowledge*** exchange, whereby the knowledge present in each team member was more easily shared. SPI was also resisted by the smaller companies who believed it would negatively impact their ***Creativity*** and ***Flexibility***.

From the commercial SPI perspective, the study was dominated by two particular models, CMM/CMMI and ISO 9000, and the development methodology XP. Respondents did not differentiate between processes and methodologies. As a result, XP, albeit tailored to various degrees, was by far the most popular commercial 'process' model used by the organisations across all of the Start-up, Build and Expansion size sectors. XP was perceived to have the least associated **Cost of Process** and its low level

of ***Documentation*** and ***Bureaucracy*** was deemed to be attractive. None of the study companies are using CMM or CMMI but several of the managers had experience of CMM prior to joining their current employers. All of those who had used CMM previously were against introducing it to their new organisations arguing that, whilst it may have a role in a very large multinational, it had no role in a small software product company.

ISO 9000 also received major criticism from the majority of the study companies many of whose managers, again, had used it previously. However, three companies in the study are ISO 9000 certified. All of those sought certification for business reasons.

Overall, respondents felt that the resources required to implement the commercial models far exceeded the benefits that may accrue. In some cases however, managers saw no benefit at all to the commercial models and believed they would hamper business prospects.

13.4 Research Contribution

This research makes several key contributions. By careful and comprehensive comparison, analysis, and abstraction of interviews with 21 software product companies, the research provides a grounded understanding of the practice of software process and software process improvement, explains the factors that influence the way process is established and evolves in software companies, and describes the reasoning behind why software companies largely ignore commercial best practice software process and process improvement models. The resulting grounded theory makes a major contribution to the discipline of software engineering as it explores and describes factors outside the typical technology-centred study. It moves beyond much of the current theoretical literature in two ways. Firstly, by employing an inductive approach it challenges the current mores and truisms in software development theory which have typically been derived using deductive methods to prove 'accepted wisdom'. By contrast this research has given voice to practitioners, most with multiple years professional expertise, thereby enabling 'practice to inform theory' and importantly provide a

challenge to that 'accepted wisdom'. Secondly, it has deployed a qualitative methodology, more associated with the social sciences, in a primarily scientific field. The use of grounded theory in this way has culminated in empirically-valid theory and has the capacity to provide encouragement to other researchers to bring alternative methodologies to bear on aspects of software development.

As stated in 9.2, there is an absence of published material describing how process is initially formed in software product companies. This research provides a new contribution to the body of work in this area. Using evidence from practice, a theory has been generated which explains the factors which influence the first software process a company will use. The research also contributes to knowledge and understanding of the domain of process change and process improvement. Unlike much of the literature, which discusses how to implement SPI, this study demonstrates why SPI is undertaken. Understanding the reasons for SPI, and the interrelationships between the key associated variables, provides vital knowledge and information to the field. Similarly, from a practice perspective, this research illustrates why commercial process models are being tailored and why best practice SPI models are being ignored. Within the software community there is much discussion of the gap between research and practice and theory and practice. This research makes a significant contribution towards bridging that gap.

In a further departure from standard practice, the research explains how SPI is not solely technology-centred but rather is affected by wider human and organisational factors. This suggests that SPI studies which concentrate purely on procedural and bureaucratic adherence, and thus neglect the human and organisational dimension, are flawed by failing to take account of key pieces in the SPI jigsaw. Consequently, this research offers support to the authors quoted in 3.4 who argue that people issues, amongst other factors, must be considered in SPI initiatives.

In a challenge to the mainstream SPI literature, this work moves beyond the 'single case study' success story which is the dominant model in software process publications. The

majority of these studies concern large multi-national corporations and their lessons have extremely limited resonance in a micro to small software product company. This research contributes a 'warts and all' view of software process in practice, untainted by a desire for company self-promotion. What is therefore provided is a reality, which is singularly different from the typical success-laden report. By providing this, this study has a resonance for software SMEs who can identify with what is being stated and with the described prevailing conditions of limited resources, personnel and time. Without 'me too' examples, such as provided here, being contained in the literature, there is a danger that small companies may reject all of the studies, and ergo the heavily-promoted best practice SPI models, as being out of touch with, or irrelevant to, their everyday challenges. Therefore, by describing their experiences, and explaining the actions and interactions of the variables concerned, this study does a major service to software SMEs and SPI in the small.

At the conclusion of this research, there is now additional clarity and understanding of the issues facing software process and process improvement in small software product companies and in particular the indigenous Irish software sector. This work, by focusing on what is currently happening in software development practice, sheds new light on the challenges to SPI in small settings. It explored and revealed the factors that influence process establishment, the role of trigger events and the key human aspect to SPI success, and the debilitating cost, in terms of administrative overhead, that is perceived to be associated with SPI models. Knowledge of the discipline of software engineering is now enhanced in that a much-neglected area, SPI in small product companies, has been investigated in detail, and a conceptually-dense theory generated to explain the issues faced by practitioners on a day-to-day basis. The new information uncovered provides a strong basis for further research in the small software company arena.

13.5 Implications for the Field

13.5.1 Implications for Practitioners

The findings of this research contain useful lessons for software entrepreneurs who need to make decisions about process and process change within their organisations as they grow. The theory presented here represents a form of 'experience map' illustrating some of the potential pitfalls an Irish software product company could face and how others have avoided or resolved them. The lessons from practice, uncovered in this study, indicate that the first process used by a software company is based, in the main, on the prior experience of the person appointed as Software Development Manager. This has clear implications for the hiring policy of the software start-up who will require an appropriate software process to meet the demands of the ***Market Sector*** they are entering. In effect, the findings here imply that, where a company needs a formalised process to support a regulated market, or a light, flexible process to support a dynamically-changing market, the person appointed as Software Development Manager is pivotal to future success. Similarly, the key role of people in buying-into SPI, and following process, has additional implications for an organisation's hiring policy. If strict adherence to process is fundamental to an organisation's software development success, then that organisation's recruitment procedures should focus on hiring staff who can comfortably fit within that particular culture.

That SPI, in small companies, results from trigger events also carries implications for professional software developers. The option here is for companies to attempt to foresee some of these triggers and then make appropriate provision to deal with them as they arise. Companies also have the option to manage SPI activity on an ongoing basis, thus operating a prevention policy rather than a reactive one. But, as practitioners report, the resources are typically not available for, or committed to, such a pro-active policy. Therefore, companies are left with a choice to make as to whether they will plan for SPI, in an attempt to ensure a smooth transition between stages of growth, or commit

resources elsewhere and hope that when events do require an SPI solution that that solution can be enacted with the time and resources available at that point.

The study has uncovered evidence that many companies are benefiting from informal ***Communication***, particularly verbal ***Communication***, and ***Tacit Knowledge*** at the expense of detailed ***Documentation***. Any organisation that follows this route needs to be aware of the advantages and disadvantages associated with this approach. Companies who have gained from sharing ***Tacit Knowledge*** have generally had a workspace and supporting environment conducive to informal information exchange between employees. These workspaces were generally open-plan, with the relevant project team members co-located. Other provisions such as central whiteboards, informal meeting spaces, local seating/refreshment areas, and even common and games rooms facilitated information flow. Organisations who have a more rigid office and workspace infrastructure will have to consider measures to overcome this if they are to implement a policy supporting informal ***Communication***. Notwithstanding this, the study also showed how company expansion brings with it a requirement for greater explicit knowledge, particularly in the form of ***Documentation***. Companies need to be aware of the necessity for increased formality as they expand.

13.5.2 Implications for Researchers

The studies highlighted in 10.2 are either single company case studies or 'how tos' of SPI. The underlying inference contained within these SPI studies is that if other companies can incorporate the lessons from them into their own environments then they too can experience similar success. However, what this research indicates is that SPI adoption and success is not merely a matter of knowledge and training. The reasons that companies avoid SPI, this research contends, is not because they don't know what to do or how to approach it, but that they don't feel any necessity to do it until events overtake them and, because of the cost involved, even then they will do the minimum required. This poses questions for many SPI researchers whose approach is to prove the benefits of SPI through case studies and reports of the benefits accruing to companies who implement SPI. If the companies in this study are broadly representative of the small

software product community then clearly that message is either not getting through, or being ignored. This suggests that more research should be conducted on small software company dynamics and the role of process and process improvement in start-ups and 'build' organisations in order to understand more fully the relationship between software company growth and the need for SPI.

Software start-ups and small companies, in the first instance, focus exclusively on survival. All resources are channelled in this direction and SPI is not seen as an enabling factor for that survival. This, in part, explains the success of agile methodologies whose 'light', non-bureaucratic techniques support companies in survival mode attempting to establish good, base software development practices. Though CMM/CMMI is firmly anchored in the belief that better processes mean better products, many small Irish software product companies are merely concerned about getting a product released to the market as quickly as possible. Development models, such as those within the agile family, rather than CMM/CMMI or ISO 9000, are perceived as supporting this objective. This clearly poses questions for CMM/CMMI and ISO 9000 researchers. Despite the fact that researchers may classify methodologies as only one element within a software process, practitioners, as shown in this study, clearly do not make such distinctions between methods and process. SPI researchers must reflect on the fact that, as this study shows, start-ups and small companies are significantly more interested in methods than process, and methods such as XP are far more attractive to practitioners in these situations than processes such as CMM/CMMI or ISO 9000. Clearly, practitioners can be educated and trained to understand the differences between methods and process and the necessity to go beyond mere methodological adoption in pursuit of SPI. However, if they are to be more widely deployed by early stage companies, existing SPI models may have to be broadened to take account of the necessity for these companies to meet their development targets and 'walk before they can run'.

The question of how CMM/CMMI can produce positive results in small settings has been explored by a number of researchers including (Brodman, and Johnson, 1994; Coleman Dangle *et al.*, 2005; Horvat *et al.*, 2005; Saiedian and Carr, 1997) and those

associated with the SEI (Heinz, 2004; Paulk, 1998). However, the argument put forward within this research is that small software companies grudgingly commit resources to SPI only when absolutely necessary and even then operate off a minimum process. As a result, 'one-size fits all' models such as CMM/CMMI, originally designed for large US defence contractors, and subsequently adopted by large MNCs, are always going to find it difficult to penetrate small software organisations. The implications therefore are that, though significant research time has been spent on endeavouring to prove that CMM/CMMI can work in small settings, perhaps too little time has been spent investigating why software SMEs are not prepared to adopt or even experiment with these models. Thus, examining the reasons for the rejection of CMM/CMMI by small software companies is something that could be usefully addressed in future studies.

For the minority of companies in this study who have experience of CMM/CMMI through the background of their managers, or who have pursued ISO 9000 accreditation for business reasons, the lessons from the practitioners are that the models can be useful, but only in certain well-defined situations and where the resources allow. Also, despite what the commercial SPI proponents argue, company size is a major factor in whether or not a model will be adopted. Though not the sole factor, the field data shows that there is a correlation between size and enthusiasm for a commercial standard, and that as the company gets larger, enthusiasm for the use of a model, or at least acceptance that it may have benefit, increases. But size, as indicated in the analysis, may contain contradictory messages and can disguise other factors such as amount of resources available, number of projects currently underway, ***Market Sector*** demands etc. Thus, a small software company creating embedded solutions for medical devices will likely have to conform to external standards such as those defined by the FDA, potentially making ISO 9000 certification a necessity. However, as Baskerville *et al.* (2001), in their study of Internet companies, conclude, ***Quality*** is not the most important thing in this fast-changing environment, rather time to market and innovation are key. Such contextual realities must be considered by SPI researchers. To cater for these differences in a company's operating context, and the fact that all companies may, therefore, not

have the same business objectives, the provision of more flexible SPI models should be investigated.

Most of the SPI models are based on standardisation and ensuring replication and consistency. However, unless heavily policed, which will often require resources which small companies do not have available, the human element can conspire to ensure that process adherence is reduced and the working process eroded. It is a simple fact of life that not all developers like the boundaries which following processes can impose upon them. Many companies laud, celebrate, and promote their best developers who, by virtue of their experience and talent, are allowed eschew the process. Companies in innovative and dynamic markets are not willing to corral these 'maverick geniuses' within process confines. But this has a ripple effect amongst other developers who are themselves unwilling to operate within process restrictions when others are exempted. The human element and the psychological aspects which feed into the creative field that is software development cannot be ignored by both process modellers and SPI champions.

The findings from this research indicate that human and social factors have a major role to play in SPI. However, this is an angle that has largely been ignored in SPI studies within software engineering. This is not true of the IS discipline. Studies there, many cited within this thesis, do attempt to take the human and social dimension into account when examining methodological and process issues. There is evidence that there is something of a recognition of this fact in that the most recent International Conference on Software Engineering (ICSE), arguably the world's largest and most prestigious software engineering conference, incorporated a workshop on the Human and Social Factors in Software Engineering (HSSE). This human and social element should be explored further to get a full picture of its role in SPI.

13.6 Conclusions

This research has addressed the two key aspects of software process usage in software product companies; how the process is initially formed and how and why it subsequently changes. **Process Formation** is primarily of relevance to software start-

ups. The study has revealed that software process in a start-up situation is a nebulous concept in that organisations will use whatever works to support their immediate business objective. Typically this business objective is survival. Getting a product to market as quickly as possible may mean the difference between survival and decline. But any small software company suffers from having limited resources and is focused merely on 'doing things' rather than 'doing things right'. The resources are simply not available to explore the best way to develop software, for that organisation, at that time. As a result start-ups depend largely on the experience of the person acting as Software Development Manager whose expertise and know-how can help them meet their deadlines and reach the next stage of development. For companies like this who, by necessity, typically have a skeleton process in place, any attempt to interest them in SPI will be somewhat redundant. However, agile methods, as have been shown in this study, do have a lot to offer such organisations. Start-ups are product-driven and, with very small development teams, often developer-led. Agile methods too are product-driven and developer-led. Because of the confluence between these two factors, there is more value in offering start-up companies 'software practice improvement' rather than software process improvement. Then when survival has been achieved, and development approaches have somewhat stabilised, should the issue of SPI be examined.

It is important for managers to understand how and why software processes change in their organisations. Process change has been shown to be reactive rather than planned for and controlled. In many instances where SPI has been undertaken it has not been a complete success. Companies instigate SPI but do not ensure that the gains made from an SPI initiative are maintained. As a result many SPI initiatives are not institutionalised. But, perhaps surprisingly, managers are as culpable as developers in not ensuring that process is followed. Management complicity with developers, in avoiding pre-defined process elements when the need arises, highlights the fact that companies do not have an always-used standard development process. Rather there is an 'official' standard process which is what all agree is the company process and the 'actual' process used on projects which, though based on the 'official' version, rarely

adheres to it. Evidence from the managers suggests that 'official' processes are there for customers, and other interested outsiders, or where appropriate, auditors and assessors. The 'actual' process is what the developers use on projects.

From the **Process Evolution** perspective, the grounded theory model provides concise information on how and why SPI occurs in software product companies. The model informs managers that if they make process improvements then these improvements may not carry through to subsequent projects and the initiative will suffer a reduction in application. Because, as the study suggests, process change is reactive and only occurs as a result of ***Business Events***, then planning and implementing SPI activity outside of event occurrences may be difficult. Whilst the ideal situation is to be able to anticipate ***Business Events*** and make advance process provision for them, they are not always predictable. For example, an approach by a very large potential customer, or the need to develop or modify systems as a result of legislative change, often cannot be seen very far in advance.

The results of this research show how XP has made significant inroads into small Irish software companies. XP offers start-up and build companies a way to improve their development activity and at minimal cost. Whereas XP is not designed with small companies in mind, some of the companies in the study grew to a point where XP no longer satisfied their demands. This is a key SPI point as companies are saying that methods are no longer sufficient by themselves and a more all-encompassing SPI effort is needed. This led companies to examine the introduction of greater formality and ***Documentation*** in their development process. Crucially, even with the limitations of XP exposed, these companies still did not consider CMM/CMMI as a solution thus raising issues of the value of these models outside the large, multi-site company arena.

Though it is not new to claim that SPI has an associated cost, many companies are deterred from investigating SPI models because of a *perceived* cost. Managers' perceptions are that SPI means increased ***Documentation*** and ***Bureaucracy***. Such a perception is widespread and is seen as a 'feature' of CMM/CMMI. Whether or not this

is true is a moot point. The fact that managers associate CMM/CMMI with increased overhead results in most small company instances in the model not being considered as a solution or even worthy of investigation.

Supporters of CMM/CMMI claim that use of the models can lead to greater predictability and repeatability (Boehm and Turner, 2004). Paradoxically, this works against CMM/CMMI from the perception of small, early-stage, software firms. Many small software companies, some of who may have only a single product in their marketing suite, would argue that each project and situation is new to them and that ***Creativity*** and ***Flexibility*** are far higher on their list of desired capabilities than predictability and repeatability. The companies in this study have shown that they see agile methodologies as supporting ***Creativity*** and ***Flexibility***. Accordingly, it is easy to see how XP has achieved much higher usage in indigenous Irish software companies than CMM/CMMI.

Given the volume of material in the literature, it is perhaps surprising that there was no reference whatsoever, by any of the study respondents, to the ISO/IEC 15504 ('SPICE') software process assessment standard. Despite its relatively long existence, ISO/IEC 15504 has failed to pierce the consciousness of Irish software product managers and was not listed as a process option by them, this despite the fact that it is an ISO standard designed specifically for SPI. The literature available on ISO/IEC 15504 suggests that it can be scaled for use by small and very small companies much more easily than CMM/CMMI. However, the complete absence of knowledge about the standard should give cause for concern amongst its founders and advocates.

13.7 Limitations of the Study

As qualitative research studies, using semi-structured interviews, grounded theory investigations centre on respondents' opinions. The findings, and the resultant theory, depend on the data gathered in the field, that is directly from the participant interviews. Unlike quantitative studies, where independent laboratory conditions may prevail, grounded theory relies on opinion. However, this opinion is the respondent's view or

perception of what is taking place, which of course may be at odds with reality. In many instances there may be no supporting evidence to verify the opinion expressed. In addition, it is possible, that the participants may report what they believe the researcher wishes to hear. This may be particularly true of smaller companies who are reluctant to admit that they are not following received best practice, as this is not something they wish to make public. Like companies who may not wish to publish negative results, for fear that it presents the organisation in a bad light, participants may be tempted to do likewise in qualitative interview-based studies, in order to be seen in a favourable light by the interviewer, or to boost the status of the company. However, it is not the role of researchers to second-guess their interviewees. As such, researchers must accept the veracity of what respondents say during the study interviews (Hansen and Kautz, 2005).

Notwithstanding the issues surrounding semi-structured interviews, the opinions of the participants are vital. In this research, even though the reality of the situation could be potentially different to that described, it is the managers' perception of what is happening, and it is on this perception that they base their decisions. A simple example, from this study, would be where a manager believes that his/her organisation does not have sufficient expertise to establish a configuration management department and, as a result, decides to recruit externally. It is these actions and interactions, arising from the participants opinions, beliefs, and perceptions, which are essential to a grounded theory study (Strauss and Corbin, 1998).

Another potential limitation of the research is the fact that interviews were only sought, and conducted, with senior managers. Whilst extensive efforts were made to ensure proper diversity in the field data, and that reports were gathered from different sized companies in different sectors, the interview pool consisted solely of a very senior person in each organisation. In most cases the managers interviewed are one or more steps removed from those who are carrying out many of the process steps promoted or defined by them or the organisation. For example, only one or two of those managers interviewed actually engaged in any coding work. A similarly small number regularly get involved in product design or testing. Therefore, the researcher is presented with the

manager's interpretation of what the engineers and testers do rather than hearing first hand from the engineers and testers themselves how they carry out the work and what process they follow.

However, whilst a study gathering data purely from the engineers' perspective might generate a different outcome, it would lack the crucial, over-arching 'big picture' view that senior managers can provide. Similarly, it is generally the senior managers who have decision-making responsibility for such as, process model adopted, hiring, product road maps and target market. Also, in larger organisations, multiple projects are being undertaken at any one time. As the reports from the companies in this thesis have shown, a number of the organisations use different processes on different projects, and within different sectors of the business (e.g. development Vs support). This knowledge of corporate events would typically be far beyond what engineers could provide from their lower position in the organisational hierarchy and, therefore, a study of process in practice which focused exclusively on engineers would be seriously deficient in depth and breadth of organisational approaches.

13.8 Further Research

One of the major contributions of this work is a grounded theory explaining how software process is initially established in a software start-up. As stated in 9.2 and 13.4, the literature lacks a comprehensive investigation of software process initiation and usage in beginning software product companies. The opportunity arises therefore for other researchers to explore this area to determine support for, or a challenge to, the generated theory.

This research is concerned with how software process is practiced in indigenous Irish software product companies. A study which concentrated on the practices used by indigenous software product companies in other countries in Europe and beyond, would provide further validity for this research and indicate if the findings can be replicated elsewhere or if they are peculiar to the Irish context. However, much software is developed outside the software product company domain. As stated in 1.2.2, there is a

wide spectrum of organisations whose business ranges from bespoke software solutions to the in-house software departments of non-software companies. These developers also use software processes and a study of how these are formed, evolve and improve, in this non-software product company environment, could be counter-balanced against this work.

As discussed in 13.7, another research focus could involve capturing the opinions and experiences of the engineers themselves. This would add to the data and analysis on ***Management Style*** and cultural issues as they exist in organisations, and would also develop the category of ***Employee Buy-in to Process*** which emerged in this study. Further development in such a work would include the ***Minimum Process***, ***Process Erosion*** and ***Process Inertia*** categories as they are significantly affected by engineer attitudes. Another theme, which emerged in one of the study companies, was the idea of offering rewards or incentives to employees to follow the organisation's software process. If companies could be found who were prepared to participate in trialling such schemes, then some useful results could emerge.

One issue alluded to in this research is the fact that at certain stages in a company's development, more formality is required. In this study, this was particularly pertinent in the case of companies who used XP. Several of those that had implemented XP had discontinued or scaled-down its use because of the fact that they required more formality in their process. This issue of process scaling merits further research. Exploration could centre on when certain processes/process models stop being effective and why. In relation to XP it would be especially useful to ascertain at what point its use become more negative than positive and the factors/set of occurrences that lead to a decision to desist from using it.

Also, as with XP, many of the companies indirectly acknowledged that at certain points in their development they needed to increase ***Documentation*** levels and to have some form of written history of their products and their development. Again, it would be beneficial for the research and practice community to see the factors that give rise to the

requirement for increased ***Documentation*** and at what points in a software company growth cycle this takes precedence. This could be incorporated into a study on ***Communication*** issues, which determine where the limits of verbal ***Communication*** lie, and would also include, ***Tacit Knowledge***, co-location, and office layout factors in its investigation scope.

References

Aaen, I., 2003, 'Software Process Improvement: Blueprint versus Recipes', in *IEEE Software*, September/October, pp. 86-93.

Ahern, D.M., Clouse, A. & Turner, R., 2004, *CMMI Distilled: A Practical Introduction to Integrated Process Improvement*, 2nd Ed, Addison Wesley.

Ambler, S.W., 2005a, 'Agile Modeling and Extreme Programming', Available at www.agilemodeling.com/essays/agileModelingXP.htm [Viewed on 13.01.06]

Ambler, S.W., 2005b, 'Agile Documentation: Strategies for Agile Software Development', Available at www.agilemodeling.com/essays/agileDocumentation.htm [Viewed on 13.01.06]

Anacleto, A., Gresse von Wangenheim, C., Salviano, C.F. and Savi, R., 2004, 'A Method for Process Assessment in Small Software Companies', in *Proceedings of 4th International SPICE Conference on Process Assessment and Improvement*, Portugal, pp. 69-76.

Andres, A., Ferrer, P., Gutierrez, P.A. and Satriani, G., 1997, 'ISO9000 Certification as a Business Driver: The SPICE Road', in *Proceedings of Quality Week Europe*, November, Brussels, Belgium.

Armour, P.G., 2001, 'Matching Process to Types of Teams', in *Communications of the ACM*, Vol. 44, No. 7, pp. 21-23.

Arora, A., Gambardella, A. and Torrisi, S., 2001, 'In the Footsteps of Silicon Valley? Indian and Irish Software' in *International Division of Labour*, Stanford Institute for Economic Policy Research (SIEPR) Discussion Paper, No. 00-41, Stanford University, California, USA.

Aveling, B., 2004, 'XP Lite Considered Harmful?', in *Proceedings of the 5th International Conference of Extreme Programming and Agile Processes in Software Engineering*, Springer, LNCS 3092, pp. 94-103.

Avison, D, Lau, F., Myers, M. and Nielsen, P., 1999, 'Action Research', in *Communications of the ACM*, January, Vol. 42, No. 1, pp. 94-97.

Bach, J., 1994, 'The Immaturity of CMM', in *American Programmer*, September, Vol. 7, No. 9, pp. 13-18.

Bach, J., 1998, 'Microdynamics of Process Evolution', in *IEEE Computer*, February, pp. 111-113.

Baddoo, N. and Hall, T., 2002, 'Motivators of Software Process Improvement: An Analysis of Practitioners' Views', in *The Journal of Systems and Software*, Vol. 62, No. 2, pp. 85-96.

Baddoo, N. and Hall, T., 2003, 'De-Motivators for Software Process Improvement: An Analysis of Practitioners' Views', in *The Journal of Systems and Software*, Vol. 66, No. 1, pp. 23-33.

Baker, R., 1996, 'The Corporate Politics of CMM Ratings', in *Communications of the ACM*, Vol. 39, No. 9, pp. 105-106.

Baskerville, R. and Pries-Heje, J., 1999a, 'Grounded Action Research: A Method for Understanding IT in Practice', in *Accounting, Management and Information Technologies*, Vol. 9, No. 1, pp. 1-23.

Baskerville, R. and Pries-Heje, J., 1999b, 'Knowledge Capability and Maturity in Software Management', in *The Data Base for Advances in Information Systems*, Vol. 30, No. 2, 26-43.

Baskerville, R., Levine, L., Pries-Heje, J. and Slaughter, S., 2001, 'How Internet Software Companies Negotiate Quality', in *IEEE Computer*, May, pp. 51-57.

Baskerville, R., Ramesh, B., Levine, L., Pries-Heje, J. and Slaughter, S., 2003, 'Is Internet-Speed Software Development Different?', in *IEEE Software*, November/December, pp. 70-77.

Batista, J. and de Figueiredo, A.D., 2000, 'SPI in a Very Small Team: A Case with CMM', in *Software Process Improvement and Practice*, Vol. 5, No. 4, pp. 243-250.

Beck, K, 2000, *Extreme Programming Explained: Embrace Change*, Addison Wesley.

Beck K. and Boehm B., 2003, 'Agility Through Discipline: A Debate', in *IEEE Computer*, June, pp. 44-46.

Beecham, S., Hall, T. and Rainer, A., 2003, 'Software Process Improvement Problems in Twelve Software Companies: An Empirical Analysis', in *Empirical Software Engineering*, Vol. 8, No. 1, pp. 7-42.

Bersoff, E, 1994, 'Anatomy of a Software Start-up', in *IEEE Software*, January, pp. 92-100.

Bertelsen, O.W., 1997, 'Towards a Unified Field of SE Research and Practice', in *IEEE Software*, November/December, pp. 87-88.

Biro, M., Ivanyos, J. and Messnarz, R., 2000, 'Pioneering Process Improvement Experiment in Hungary', in *Software Process Improvement and Practice*, Vol. 5, No. 4, pp. 213-229.

Blaxter, L., Hughes, C. and Tight, M., 2001, *How to Research*, 2nd Edition, Open University Press.

Boehm, B.W., 1988, 'A Spiral Model of Software Development and Enhancement', in *IEEE Computer*, May, pp. 61-72.

Boehm, B., and Turner, R., 2003, 'Using Risk to Balance Agile and Plan-Driven Methods', in *IEEE Computer*, June, pp. 57-66.

Boehm, B., and Turner, R., 2004, *Balancing Agility and Discipline*, Addison Wesley.

Bollinger, T.B., 1997, 'The Interplay of Art and Science in Software', in *IEEE Computer*, October, pp. 125-128.

Bollinger, T.B. & McGowan, C., 1991, 'A Critical Look at Software Capability Evaluations', in *IEEE Software*, July, pp. 25-41.

Borjesson, A. and Mathiassen, L., 2004, 'Successful Process Implementation', in *IEEE Software*, July/August, pp. 36-44.

Brodman, J.G. and Johnson D.L., 1994, 'What Small Businesses and Small Organisations say about the CMM', in *Proceedings of the 16th International Conference on Software Engineering*, pp. 331-340.

Buchanan, D.A. and Huczynski, A.A., 1985, *Organisational Behaviour: An Introductory Text*, Prentice-Hall International (UK) Ltd., London.

Buchman, C, 1996, 'Software Process Improvement at AlliedSignal Aerospace', in *Proceedings of the 29th Annual Hawaiian International Conference on System Sciences*, Vol.1, Software Technology and Architecture, pp. 673-680.

Burns, R. B., 2000, *Introduction to Research Methods*, 4th Edition, Sage Publications.

Card, D., 2000, 'Sorting out Six Sigma and the CMM', in *IEEE Software*, May/June, pp. 11-13.

Carvalho, L., Scott, L. and Jeffery, R., 2005, 'An Exploratory Study into the Use of Qualitative Research Methods in Descriptive Process Modelling', in *Information and Software Technology*, Vol. 47, No. 2, pp. 113-127.

Carver, J. and Basili, V., 2003, 'Identifying Implicit Process Variables to Support Future Empirical Work', in *Proceedings of 17th Brazilian Symposium on Software Engineering (SBES 2003),* October, pp. 5-18.

Casey, V. and Richardson, I., 2004 'A Practical Application of the IDEAL Model' in *Software Process Improvement and Practice*, Vol. 9, No.3, pp. 123-132.

Chapin, N., 2004, 'Agile Methods Contributions in Software Evolution', in *Proceedings of 20th International Conference on Software Maintenance*, IEEE Computer Society, pp. 522.

Chiang, I.R. and Mookerjee, V.S., 2004, 'Improving Software Team Productivity', in *Communications of the ACM*, Vol. 47, No. 5, pp. 89-93.

Chisnall, P. M., 1987, *Small Firms in Action: Case Histories in Entrepreneurship*, McGraw-Hill.

Chrissis, M.B., Konrad, M. & Shrum, S., 2003, *CMMI: Guidelines for Process Integration and Product Improvement*, Addison Wesley, Boston, MA.

Coallier, F, 1994, 'How ISO 9001 Fits into the Software World', in *IEEE Software*, January, pp. 98-100.

Cockburn, A, 2002a, *Agile Software Development*, Addison Wesley.

Cockburn, A, 2002b, 'Agile Software Development Joins the "Would-Be" Crowd', in *Cutter IT Journal*, January, pp. 6-12.

Cockburn, A. and Highsmith, J., 2001, 'Agile Software Development: The People Factor', in *IEEE Computer*, November, pp. 131-133.

Cohn, M. and Ford D., 2003, 'Introducing an Agile Process to an Organisation', in *IEEE Computer*, June, pp. 74-78.

Coleman, G., 2002, 'Practice Not Process – Improving the Capability of Software Start-ups', in *Proceedings of Third International Conference on Extreme Programming and Flexible Processes in Software Engineering*, Italy, May, pp. 229-230.

Coleman, G., 2004, 'eXtreme Programming (XP) as a Minimum Software Process: A Grounded Theory', in *Proceedings of Computer Software and Applications Conference (COMPSAC)* - Workshops and Fast Abstracts, Hong Kong, September 2004, pp. 30-31.

Coleman, G., 2005, 'An Empirical Study of Software Process in Practice', in *Proceedings of the 38th Annual Hawaiian International Conference on System Sciences*, - Track 9, Big Island, HI, p. 315c.

Coleman G. & O'Connor R., 2000, 'Power to the Programmer: Using Measurement to Optimise the Software Process at the Individual Level', in *Proceedings of ESCOM-SCOPE*, Munich, April.

Coleman G. & O'Connor R., 2007, 'Using Grounded Theory to Understand Software Process Improvement: A Case Study of Irish Software Product Companies', in *Information and Software Technology*, Forthcoming.

Coleman Dangle, K, Larsen, P, Shaw, M. and Zelkowitz, M.V., 2005, 'Software Process Improvement in Small Organisations: A Case Study', in *IEEE Software*, November/December, pp. 68-75.

Condon, D., 2002, *Software Product Management: Managing Software Development from Idea to Product to Marketing to Sales*, Aspatore Books, USA.

Constantine, L., 2001a, 'Job Qualifications: On Hiring the Best', in *Beyond Chaos: The Expert Edge in Managing Software Development*, ACM Press, Addison Wesley, pp. 33-38.

Constantine, L., 2001b, 'Cutting Corners: Shortcuts in Model-Driven Web Development', in *Beyond Chaos: The Expert Edge in Managing Software Development*, ACM Press, Addison Wesley, pp. 177-184.

Crone, M., 2002, *A Profile of the Irish Software Industry*, Northern Ireland Economic Research Centre (NIERC), Belfast NI.

Crosby, P.B., 1979, *Quality is Free: The Art of Making Quality Certain*, McGraw-Hill, USA.

Curtis, B., Hefley, W.E. and Miller, S., 1995, 'People Capability Maturity Model', *Technical Report CMU/SEI-95-MM-02*, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA.

Cusumano, M. A., 2004, *The Business of Software: What Every Manager, Programmer and Entrepreneur Must Know to Thrive and Survive in Good Times and Bad*, Free Press, NY.

Cusumano, M. A. and Yoffie, D.B., 1999, 'Software Development on Internet Time', in *IEEE Computer*, October, pp. 60-69.

Cusumano M.A., MacCormack, A., Kemerer, C.F. and Crandall, W., 2003, 'Software Development Worldwide: The State of the Practice', in *IEEE Software*, November/December, pp. 28-34.

Daskalantonakis, M.K., 1994, 'Achieving Higher SEI Levels', in *IEEE Software*, July, pp. 17-24.

Debou, C. and Kuntzmann-Combelles, A., 2000, 'Linking Software Process Improvement to Business Strategies: Experiences from Industry', in *Software Process Improvement and Practice*, Vol. 5, No. 1, pp. 55-64.

Deck, M., 2001, 'Managing Process Diversity While Improving your Practices', in *IEEE Software*, May/June, pp. 21-27.

DeMarco, T. and Lister, T., 1999, *Peopleware: Productive Projects and Teams*, 2nd Edition, Dorset House, New York.

Demirors, E, Demirors, O, Dikenelli, O and Keskin, B., 1998, 'Process Improvement Towards ISO 9001 Certification in a Small Software Organisation', in *Proceedings of the 20th International Conference on Software Engineering*, pp. 435-438.

Dey, I., 1993, *Qualitative Data Analysis: A User-friendly Guide for Social Scientists*, Routledge.

Dicks, R.S., 2000, 'The Paradox of Information: Control Versus Chaos in Managing Documentation Projects with Multiple Audiences', in *Proceedings of the 18th Annual ACM International Conference on Computer Documentation: Technology & Teamwork*, Cambridge, Massachusetts, USA, pp. 253-259.

Dion, R., 1993, 'Process Improvement and the Corporate Balance Sheet', in *IEEE Software*, July, pp. 28-35.

Dorling, A., 1993, 'SPICE: Software Process Improvement and Capability dEtermination', in *Information and Software Technology*, Vol. 36, No. 6/7, pp. 404-406.

Drobka, J., Noftz, D. and Raghu, R., 2004, 'Piloting XP on Four Mission-Critical Projects', in *IEEE Software*, November/December, 2004, pp. 70-75.

Dyba, T., 2000, 'Improvisation in Small Software Organisations', in *IEEE Software*, September/October, pp. 82-87.

Dyba, T., 2003, 'Factors of Software Process Improvement Success in Small and Large Organisations', in *Proceedings of European Software Engineering Conference/ Foundations of Software Engineering*, September 1-5, Finland, pp. 148-157.

Ebert, C. and De Neve, P., 2001, 'Surviving Global Software Development', in *IEEE Software*, March/April, pp. 62-69.

Eischen, K., 2002, 'Software Development: An Outsider's View', in *IEEE Computer*, May, pp. 36-44.

El Emam, K. and Briand, L., 1997, 'Costs and Benefits of Software Process Improvement', *Technical Report ISERN 97-12*, Fraunhofer Institute for Experimental Software Engineering, Germany.

Enterprise Ireland, 2005a, *Background to the Irish Software Industry*, available at http://www.nsd.ie/htm/ssii/back.htm. [Viewed on 02.02.06]

Enterprise Ireland, 2005b, *Software Industry Statistics 1991-2004*, available at http://www.nsd.ie/htm/ssii/stat.htm. [Viewed on 02.02.06]

European Commission, 2005, *The New SME Definition: User Guide and Model Declaration*, available at: http://europa.eu.int/comm/enterprise/enterprise_policy/sme_definition/sme_user_guide.pdf [Viewed on 15.08.05]

Fayad, M., 1997, 'Software Development Process: A Necessary Evil', in *Communications of the ACM*, Vol. 40, No. 9, pp. 101-103.

Fayad, M and Laitinen, M, 1997, 'Process Assessment Considered Wasteful', in *Communications of the ACM*, Vol. 40, No. 11, pp. 125-128.

Fitzgerald, B., 1998, 'An Empirical Investigation into the Adoption of Systems Development Methodologies', in *Information and Management*, Vol. 34, No. 6, pp. 317-328.

Fitzgerald, B. and Howcroft, D., 1998, 'Towards Dissolution of the IS Research Debate: From Polarisation to Polarity', in *Journal of Information Technology*, Vol. 13, No. 4, pp. 313-326.

Fitzgerald, B. and O'Kane, T., 1999, 'A Longitudinal Study of Software Process Improvement', in *IEEE Software*, May/June, pp. 37-45.

Fitzgibbon, C., 1996, 'ISO 9001 Registration: Lessons Learned by Canadian Software Companies', in *Proceedings of the Fifth International Conference on Management of Technology*, Miami, Florida, pp. 193-201.

Flood, P., Heffernan, M., Farrell, J., MacCurtain, S., O'Hara, T., O'Regan, P. and Carroll, C., Dromgoole, T. and Mangan, J., 2002, *Managing Knowledge Based Organisations: Top Management Teams and Innovation in the Indigenous Software Industry*, Blackhall Publishing.

Florac, W.A. and Carleton, A.D., 1999, *Measuring the Software Process: Statistical Process Control for Software Process Improvement*, Addison Wesley, Boston, MA.

Forward, A. and Lethbridge T. C., 2002, 'The Relevance of Software Documentation, Tools and Technologies: A Survey', in *Proceedings of the 2002 ACM Symposium on Document Engineering*, McLean, Virginia, USA, pp. 26-33.

Glaser B., 1978, *Theoretical Sensitivity*, Mill Valley, CA, Sociology Press.

Glaser, B., 1992, *Basics of Grounded Theory Analysis: Emergence Vs Forcing*, Mill Valley, CA, Sociology Press.

Glaser, B. and Strauss, A., 1967, *The Discovery of Grounded Theory: Strategies for Qualitative Research*, Chicago, Aldine.

Glass, R. 2001, 'Extreme Programming: The Good, the Bad and the Bottom Line', in *IEEE Software*, November/December, pp. 111-112.

Goede, R. and De Villiers, C., 2003, 'The Applicability of Grounded Theory as Research Methodology in Studies on the Use of Methodologies in IS Practices', in *Proceedings of the Conference of the South African Institute of Computer Scientists and Information Technologists*, Gauteng, South Africa, pp. 208-217.

Goldenson, D. and Gibson, D., 2003, 'Demonstrating the Impact and Benefits of CMMI: An Update and Preliminary Results', *Technical Report CMU/SEI-2003-SR-009*, Software Engineering Institute, Pittsburgh, PA.

Goulding, C., 1999, 'Grounded Theory: Some Reflections on Paradigm, Procedures and Misconceptions', Technical Working Paper, University of Wolverhampton, UK.

Goulding, C., 2002, *Grounded Theory: A Practical Guide for Management, Business and Market Researchers*, Sage Publications.

Grady, R.B., 1997, *Successful Software Process Improvement*, Prentice Hall, NJ.

Green, R., Cunningham, J., Duggan, I., Giblin M., Moroney, M., Smyth, L., 2001, 'Boundaryless Cluster: Information and Communications Technology in Ireland', in *Proceedings of The Future of Innovation Studies*, Eindhoven Centre for Innovation Studies, Eindhoven University of Technology, The Netherlands, 20-23 September.

Grenning, J., 2001, 'Launching Extreme Programming at a Process-Intensive Company', in *IEEE Software*, November/December, pp. 27-33.

Grossman, F., Bergin, J., Leip, D, Merritt, S. and Gotel, O., 2004, 'One XP Experience: Introducing Agile (XP) Software Development into a Culture that is Willing but not Ready', in *Proceedings of the 2004 Conference of the Centre for Advanced Studies on Collaborative Research*, Markham, Canada, pp. 242-254.

Guba E. and Lincoln Y., 1994, 'Competing Paradigms in Qualitative Research', in *The Handbook of Qualitative Research*, eds. N. Denzin and Y. Lincoln, Sage Publications, pp. 105-117.

Guerrero, F. and Eterovic, Y., 2004, 'Adopting the CMM in a Small IT Organisation', in *IEEE Software*, July/August, pp. 29-35.

Haase, V., Messnarz, R, Koch, G, Kugler, H. J. & Decrinis, P., 1994, 'Bootstrap: Fine-tuning Process Assessment', in *IEEE Software*, July, pp. 25-35.

Hall, T., Rainer, A. and Baddoo, N., 2002, 'Implementing Software Process Improvement: An Empirical Study', in *Software Process Improvement and Practice*, Vol. 7, No. 1, pp. 3-15.

Hansen, B. and Kautz, K, 2005, 'Grounded Theory Applied – Studying Information Systems Development Methodologies in Practice', in *Proceedings of 38th Annual Hawaiian International Conference on Systems Sciences*, Big Island, HI.

Hardgrave, B. C. and Armstrong, D.J., 2005, 'Software Process Improvement: It's a Journey, Not a Destination', in *Communications of the ACM*, Vol. 48, No. 11, pp. 93-96.

Hass, I.M., 1997, 'Hiring the Best', in *IEEE Computer*, May, pp. 100-101.

Hayes, W. and Zubrow, D., 1995, 'Moving on Up: Data and Experience Doing CMM-based Process Improvement', *Technical Report CMU/SEI-95-TR-008*, Software Engineering Institute, Pittsburgh, PA.

Heinz, L., 2004, 'CMMI for Small Businesses: Initial Results of the Pilot Study', http://www.sei.cmu.edu/news-at-sei/features/2004/3/pdf/feature-1-2004-3.pdf, Software Engineering Institute, Pittsburgh, PA. [Viewed on 13.01.06]

Henderson-Sellers, B., 2002, 'Agile or Rigorous OO Methodologies: Getting the Best of Both Worlds', in *Cutter IT Journal*, Vol. 15, No. 1, January, pp. 25-33.

Herbsleb, J. and Goldenson, D., 1996, 'A Systematic Survey of CMM Experience and Results', in *Proceedings of the 18th International Conference on Software Engineering*, Berlin, Germany, pp. 323-330.

Herbsleb, J., Zubrow, D., Goldenson, D., Hayes, W. and Paulk M., 1997, 'Software Quality and the Capability Maturity Model', in *Communications of the ACM*, Vol. 40, No. 6, pp. 30-40.

Hevner, A. and March, S., 2003, 'The Information Systems Research Cycle', in *IEEE Computer*, November, pp. 111-113.

Highsmith, J., 2000, *Adaptive Software Development: A Collaborative Approach to Managing Complex Systems*, Dorset House.

Highsmith, J., 2002, 'Opening Statement', in *Cutter IT Journal*, Vol. 15, No. 1, January, pp. 2-5

Highsmith, J., 2004, *Agile Project Management: Creating Innovative Products*, Addison Wesley.

Hollenbach, C, Young, R., Pflugrad, A. and Smith, D., 1997, 'Combining Quality and Software Improvement', in *Communications of the ACM*, Vol. 40, No. 6, pp. 41-45.

Horvat, R.V., Rozman, I. and Gyorkos, J., 2000, 'Managing the Complexity of SPI in Small Companies', in *Software Process Improvement and Practice*, Vol. 5, No. 1, pp. 45-54.

HotOrigin, 2001, *Ireland's Emerging Software Cluster: a Hothouse of Future Stars*, HotOrigin Ltd., Dublin, Ireland.

HotOrigin, 2004, *Ireland's Software Cluster: Preparing for Consolidation*, HotOrigin Ltd., Dublin, Ireland.

Humphrey, W.S., 1988, 'Characterising the Software Process: A Maturity Framework', in *IEEE Software*, March, pp. 73-79.

Humphrey, W.S., 1989, *Managing the Software Process*, Addison Wesley, Boston, MA.

Humphrey, W.S., 1995, *A Discipline for Software Engineering*, Addison Wesley, Boston, MA.

Humphrey, W.S., 2000, *Introduction to the Team Software Process*, Addison Wesley, Boston, MA.

Humphrey, W.S., 2002, *Winning With Software: An Executive Strategy*, Addison Wesley, Boston, MA.

Humphrey, W.S. and Curtis, B., 1991, 'Comments on "A Critical Look"', in *IEEE Software*, July, pp. 42-46.

Humphrey, W.S., Snyder, T. and Willis, R., 1991, 'Software Process Improvement at Hughes Aircraft', in *IEEE Software*, July, pp. 11-23.

Hysell, D., 1999, 'ISO 9001: Traditions Before and After', in *Proceedings of the 17th Annual International Conference on Computer Documentation*, New Orleans, Louisiana, USA, pp. 99-104.

Ibrahim, L., and Pyster, A., 2004, 'A Single Model for Process Improvement: Lessons Learned at the US Federal Aviation Administration', in *IT Professional*, May/June, pp. pp. 43-49

IDA Ireland, 1999, *Achieve Competitive Advantage in Software*, IDA, Dublin 2, Ireland.

IDA Ireland, 2003, *IDA Annual Report 2003*, IDA, Dublin 2, Ireland.

IEEE, 1991, *Standard Glossary of Software Engineering Terminology*. ANSI/IEEE Std. IEEE-STD-610-1990, IEEE Computer Society.

International Organisation for Standardisation, 1987, *Quality Systems: Model for Quality Assurance in Design/Development, Production, Installation and Servicing, ISO 9001*, Geneva, Switzerland.

International Organisation for Standardisation, 1992, *Quality Management and Quality Assurance Standards, Part 3: Guidelines for the Application of ISO 9001 to the Development, Supply and Maintenance of Software*, Geneva, Switzerland.

International Organisation for Standardisation, 2000, *ISO 9001:2000 Quality Management Systems*, Geneva, Switzerland.

Jalote, P., 2002, 'Managing Software Projects: The Infosys Model', *Addison Wesley Professional Articles*, http://www.awprofessional.com/articles/article.asp?p=26317, [Viewed on 22.12.05]

Jankowicz, A.D., 1995, *Business Research Projects*, 2nd Edition, Thomson Business Press.

Jarvis, A. and Hayes, L., 1999, 'Project Support Office', in *Dare to be Excellent*, Prentice Hall, Upper Saddle River, NJ, pp. 127-146.

Javed, T, e-Maqsood, M. and Durrani, Q.S., 2004, 'A Survey to Examine the Effect of Team Communication on Job Satisfaction in Software Industry', in *Software Engineering Notes*, Vol. 29, No. 2.

Johnson, D. L. and Brodman J. G., 2000, 'Applying CMM Project Planning Practices to Diverse Environments', in *IEEE Software*, July/August, pp. 40-47.

Jones, C., 2003, 'Variations in Software Development Practices', in *IEEE Software*, November/December, pp. 22-27.

Juran, J.M., 1988, *Juran on Planning for Quality*, The Free Press, New York.

Kasse, T. and McQuaid, P.A., 1998, 'Entry Strategies into the Process Improvement Initiative', in *Software Process Improvement and Practice*, Vol. 4, No. 2, pp. 73-88.

Kautz, K., 1998, 'Software Process Improvement in Very Small Enterprises: Does it Pay Off?', in *Software Process Improvement and Practice*, Vol. 4, No. 4, pp. 209-226.

Kautz, K., Hansen, H.W. and Thaysen, K., 2000, 'Applying and Adjusting a Software Process Improvement Model in Practice: The Use of the IDEAL Model in a Small Software Enterprise', in *Proceedings of the International Conference on Software Engineering,* Limerick, Ireland, pp. 626-633.

Keeni, G, 2000, 'The Evolution of Quality Processes at Tata Consultancy Services', in *IEEE Software*, July/August 2000, pp. 79-88.

Kelly, D.P. and Culleton, B., 1999, 'Process Improvement for Small Organisations', in *IEEE Computer*, October, pp. 41-47.

Kilpi, T., 1997, 'Product Management Challenge to Software Change Process: Preliminary Results from Three SMEs Experiment', in *Software Process Improvement and Practice*, Vol. 3, No. 3, pp. 165-175.

Kraut, R.E. and Streeter, L.A., 1995, 'Coordination in Software Development', in *Communications of the ACM*, Vol. 38, No. 3, pp. 69-81.

Kruchten, P, 2000, *The Rational Unified Process*, Addison Wesley, Reading, MA.

Kuilboer, J.P. and Ashrafi, N., 2000, 'Software Process and Product Improvement: An Empirical Assessment', in *Information and Software Technology*, Vol. 42, No. 1, pp. 27-34.

Kutschera, P. and Schafer, S, 2002, 'Applying Agile Methods in Rapidly Changing Environments', available at:
http://www.agilealliance.com/articles/kutscherapeterschafer, [Viewed on 22.12.05]

Laporte, C.Y. and Trudel, S., 1998, 'Addressing the People Issues of Process Improvement at Oerlikon Aerospace', in *Software Process Improvement and Practice*, Vol. 4, No. 4, pp. 187-198.

Law A. and Charron, R., 2005, 'Effects of Agile Practices on Social Factors', in *Proceedings of Workshop on Human and Social Factors of Software Engineering (HSSE)*, May, St. Louis, USA.

Lee, A.S. and Liebenau, J., 1997, 'Information Systems and Qualitative Research', in *Proceedings of Information Systems and Qualitative Research*, eds. A. Lee, J Liebenau and J.I. DeGross, Kluwer Academic, Boston, MA.

Lethbridge T. C., Singer J. and Forward A., 2003, 'How Software Engineers Use Documentation: The State of the Practice', in *IEEE Software*, November/December, pp. 35-39.

Leung, H.K.N. and Yuen, T.C.F., 2001, 'A Process Framework for Small Projects', in *Software Process Improvement and Practice*, Vol. 6, No. 2, pp. 67-83.

Lindvall, M. and Rus, I, 2000, 'Process Diversity in Software Development', in *IEEE Software*, July/August, pp. 14-18.

Lindvall, M., Muthig, D., Dagnino, A., Wallin, C., Stupperich, M., Kiefer, D., May, J., Kahkonen, T., 2004, 'Agile Software Development in Large Organisations', in *IEEE Computer*, December, pp. 26-34.

Lippert, M., Becker-Pechau, P, Breitling, H., Koch, J., Kornstadt, A., Roock, S., Schmolitzky, A., Wolf, H. and Zullighoven, H., 2003, 'Developing Complex Projects Using XP with Extensions', in *IEEE Computer*, June, pp. 67-73.

Lycett, M., Macredie, R.D., Patel, C. and Paul, R.J., 2003, 'Migrating Agile Methods to Standardised Development Practices', in *IEEE Computer*, June, pp. 79-85.

MacCormack, A., 2001, 'Product-Development Practices That Work: How Internet Companies Build Software', in *MIT Sloan Management Review*, Vol. 42, No. 2, pp. 75-84.

MacDonald, M, 2001, 'Finding a Critical Perspective in Grounded Theory', in *Using Grounded Theory in Nursing*, Eds. Schreiber, R.S. and Noerager Stern, P., Springer Publishing Company, Broadway, NY.

MacGregor, E., Hsieh, Y. and Kruchten, P., 2005, 'Cultural Patterns in Software Process Mishaps: Incidents in Global Projects', in *Proceedings of Workshop on Human and Social Factors of Software Engineering (HSSE)*, May, St. Louis, USA.

McAnallen, M. and Coleman, G., 2005, 'Tailoring Extreme Programming for Legacy Systems: Lessons Learned', in *Experience Session Proceedings of European Software Process Improvement (EuroSPI) Conference*, Budapest, Hungary, pp. 1.21-1.29.

McBreen, P., 2000, 'Applying the Lessons of eXtreme Programming', in *Proceedings of Tools-34,* Santa Barbara, CA.

McCracken, D.D. and Jackson, M.A., 1992, 'Life-Cycle Concepts Considered Harmful', in *ACM Software Engineering Notes*, April, pp. 29-32.

McFall D., Wilkie F.G., Mc Caffery F., Lester N. and Sterritt R., 2003, 'Software Processes and Process Improvement in Northern Ireland', in *Proceedings of 16th*

International Conference on Software and Systems Engineering and their Applications (ICSSEA), CNAM — Paris, France, December.

McFall D., McCaffery, F., Wilkie, F.G., 2004, 'The Software Development Culture of Northern Ireland', in *Industrial Session Proceedings of European Software Process Improvement (EuroSPI) Conference*, November, Trondheim, Norway. pp. 1.D.13-1-D.18

McIver Consulting, 1998, *Manpower, Education and Training Study of the Irish Software Sector*, A report submitted to the Software Training Advisory Committee and FAS, Dublin, Ireland.

Madhavji, Nazim H., 1991, 'The Process Cycle', in *Software Engineering Journal*, Vol.6, No. 5, September, pp. 234-242.

Mathiassen, L., Axel Nielsen, P. and Pries-Heje, J., 2001, 'Learning SPI in Practice', *Addison Wesley Professional Articles*, http://www.awprofessional.com/articles/article.asp?p=167927, [viewed on 22.12.05]

Mathiassen, L., Ngwenyama, O.K. and Aaen, I., 2005, 'Managing Change in Software Process Improvement', in *IEEE Software*, November/December, pp. 84-91.

Melis, M., Ambu, W., Pinna, S. and Mannaro, K., 2004, 'Requirements of an ISO Compliant XP Tool', *Proceedings of the 5th International Conference of Extreme Programming and Agile Processes in Software Engineering*, Springer, LNCS 3092, pp. 266-269.

Middleton, P., Woo Lee, H. and Irani, S.A., 2004, 'Why Culling Software Colleagues is Popular', in *IEEE Software*, September/October, pp. 28-32.

Miller, M., Pulgar-Vidal, F. and Ferrin, D., 2002, 'Achieving Higher Levels of CMMI Maturity using Simulation', in *Proceedings of the Winter Simulation Conference*, December, San Diego, CA., pp. 1473-1478.

Moitra, D., 1998, Managing Change for Software Process Improvement Initiatives: A Practical Experience-based Approach', in *Software Process Improvement and Practice*, Vol. 4, No. 4, pp. 199-207.

Muhr, T., 1997, *Atlas TI User's Manual*, Scientific Software Development, Berlin.

Murru, O., Deias, R., and Mugheddu, G., 2003, 'Assessing XP at a European Internet Company', in *IEEE Software*, May/June, pp. 37-43.

Myers, M.D., 1997, 'Qualitative Research in Information Systems', in *Management Information Systems Quarterly*, Vol. 21, No. 2, June, pp. 241-242.

Namioka, A. and Bran. C, 2004, 'eXtreme ISO ?!?', in *Proceedings of OOPSLA*, Vancouver, B.C., Canada, pp. 260-263.

New Oxford Dictionary of English, 2001, Oxford University Press.

New Oxford Thesaurus of English, 2001, Oxford University Press.

Nisse, D., 2000, 'Leadership, Army Style', in *IEEE Software*, March/April, pp. 92-94.

Nunes, N.J. and Cunha, J.F., 2000, 'Wisdom: A Software Engineering Method for Small Software Development Companies', in *IEEE Software*, September/October, pp. 113-119.

O'Riain, S., 1997, 'An Offshore Silicon Valley: The Emerging Irish Software Industry', in *Competition and Change: The Journal of Global Business and Political Economy*, Vol. 2, 175-212.

Oates, B. and Fitzgerald, B., 2001, 'Action Research: Putting Theory into Practice, Organisations and Society' in *Information Systems Workshop*, New Orleans, USA.

Orlikowski, W., 1993, 'CASE Tools as Organizational Change: Investigating Incremental and Radical Changes in Systems Development', in *Management Information Systems Quarterly*, Vol. 17, No. 3, pp. 309-340.

Orlikowski, W. and Baroudi, J., 1991, 'Studying Information Technology in Organisations: Research Approaches and Assumptions', in *Information Systems Research*, Vol. 2, No. 1, pp. 1-28

Oskarsson, O. and Glass, R.L., 1996, *An ISO 9000 Approach to Building Quality Software*, Prentice Hall, NJ.

Palmer, S., and Felsing, J., 2002, *A Practical Guide to Feature-Driven Development*, Prentice Hall, NJ.

Paulk, M., 1995, 'How ISO 9001 Compares with the CMM', in *IEEE Software*, January, pp. 74-83.

Paulk, M., 1998, 'Using the Software CMM in Small Organisations', in *Proceedings of 16th Pacific Northwest Software Quality Conference*, pp. 350-361.

Paulk, M., 2001, 'Extreme Programming from a CMM Perspective', in *IEEE Software*, November/December, pp. 19-26.

Paulk, M., Curtis, B. and Chrissis, M.B., 1991, 'The Capability Maturity Model for Software, *Technical Report SEI-93-TR-24*, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA.

Paulk, M., Weber, C., Curtis, B. and Chrissis, M.B., 1994, *The Capability Maturity Model: Guidelines for Improving the Software Process*, Addison Wesley, Boston, MA.

Perry, D.E., Staudenmayer, N.A., and Votta, L.G., 1994, 'People, Organisations and Process Improvement', in *IEEE Software*, July, pp. 36-45.

Phillips, D., 1999, 'Show Me How to Do That: "Just Enough" Software Process for the 21st Century', in *Cutter IT Journal*, Vol. 12, No. 9, pp. 31-35.

Pitterman, B., 2000, 'Telcordia Technologies: The Journey to High Maturity', in *IEEE Software*, July/August, pp. 89-96.

Potter, N.S. and Sakry, M.E., 2002, *Making Process Improvement Work: A Concise Action Guide for Software Managers and Practitioners*, Addison Wesley, Boston, MA.

Power, N., 2002, 'A Grounded Theory of Requirements Documentation in the Practice of Software Development', *PhD Thesis*, Dublin City University, Ireland.

PriceWaterhouseCoopers, 2005, *Doing Business and Investing in Ireland*, Wilton Place, Dublin 2.

Punch, M., 1998, *Introduction to Social Research: Quantitative and Qualitative Approaches*, Sage Publications.

Qureshi, S., Liu, M. and Vogel, D., 2005, 'A Grounded Theory Analysis of e-Collaboration Effects for Distributed Project Management', *in Proceedings of 38th Annual Hawaiian International Conference on Systems Sciences*, Big Island, HI.

Rada, R., 1996, 'ISO 9000 Reflects the Best in Standards', in *Communications of the ACM*, Vol. 39, No. 3, pp. 17-20.

Rasmusson, J., 2003, 'Introducing XP into Greenfield Projects: Lessons Learned', in *IEEE Software*, May/June, pp. 21-28.

Reifer, D. J., 2003, 'XP and the CMM', in *IEEE Software*, May/June, pp. 14-15.

Rising, L. and Janoff, N., 2000, 'The Scrum Software Development Process for Small Teams', in *IEEE Software*, July/August, pp. 26-32.

Rost, J., 2005, 'Software Engineering Theory in Practice', in *IEEE Software*, March/April, pp. 94-96.

Royce, W. W., 1970, 'Managing the Development of Large Software Systems: Concepts and Techniques', in *Proceedings IEEE, Wescon 1970*/also in *Proc. 9th. International Conference on Software Engineering, 1987*, IEEE Computer Society Press, pp. 328-338.

Royce, W., 2005, 'Successful Software Management Style: Steering and Balance', in *IEEE Software*, September/October, pp. 40-47.

Russ, M.L. and McGregor, J.D., 2000, 'A Software Development Process for Small Projects', in *IEEE Software*, September/October, pp. 96-101.

Saiedian, H., and Carr, N., 1997, 'Characterising a Software Process Maturity Model for Small Organisations', in *ACM SIGICE Bulletin*, Vol. 23, No. 1, July, pp. 2-11.

Sanders, M., 1998, *The Spire Handbook: Better, Faster, Cheaper Software Development in Small Organisations*, Centre for Software Engineering, Dublin, Ireland.

Sarker, S., Lau, F. and Sahay, S., 2001, 'Using an Adapted Grounded Theory Approach for Inductive Theory Building About Virtual Team Development', in *The Data Base for Advances in Information Systems*, Vol. 32, No. 1, pp. 38-56.

Saunders, M.N.K., Lewis P., and Thornhill A., 1996, *Research Methods for Business Students*, Pitman, London.

Schatz, B. and Abdelshafi, I., 2005, 'Primavera Gets Agile: A Successful Transition to Agile Development', in *IEEE Software*, May/June, pp. 36-42.

Schreiber, R.S., 2001, 'The 'How To' of Grounded Theory: Avoiding the Pitfalls', in *Using Grounded Theory in Nursing*, Eds. Schreiber, R.S. and Noerager Stern, P., Springer Publishing Company, Broadway, NY.

Schuh, P, 2001, 'Recovery, Redemption and Extreme Programming', in *IEEE Software*, November/December, pp. 34-41.

Schuler, K., 1995, 'Preparing for ISO 9000 Registration: The Role of the Technical Communicator', in *Proceedings of the 13th Annual International Conference on Systems Documentation*, Savannah, GA, pp. 148-154.

Schwaber, K. and Beedle, M., 2002, *Agile Software Development with Scrum*, Prentice Hall.

Seaman, C. and Basili, V., 1997, 'An Empirical Study of Communication in Code Inspections', in *Proceedings of the 19th International Conference on Software Engineering*, May, Boston, MA. pp. 17-23

Silva, L. and Backhouse, J., 1997, 'Becoming Part of the Furniture: The Institutionalisation of Information Systems', in *Proceedings of Information Systems and Qualitative Research*, Philadelphia, PA, Chapman and Hall, London.

Silverman, D., 2000, *Doing Qualitative Research: A Practical Handbook*, Sage Publications.

Simons, M., 2002, 'Big and Agile?', in *Cutter IT Journal*, Vol. 15, No.1, pp. 34-39.

Sliger, M., 2004, 'Fooling Around with XP: Why I Lost Interest in PMI and Took Up With Something More Extreme', in *Better Software*, May/June, pp. 16-18.

Software Engineering Institute, 2002, *Process Maturity Profile: Software CMM 2002 Mid-year Update*, Available at: http://www.sei.cmu.edu/appraisal-program/profile/pdf/SW-CMM/2002aug.pdf [Viewed on 13.02.05]

Software Engineering Institute, 2005a, *Process Maturity Profile: Software CMM 2005 Mid-year Update*, Available at: http://www.sei.cmu.edu/appraisal-program/profile/pdf/SW-CMM/2005sepSwCMM.pdf [Viewed on 01.02.06]

Software Engineering Institute, 2005b, *Process Maturity Profile: CMMI V1.1 SCAMPI V1.1 Class A Appraisal Results 2005 Mid-year Update*, Available at: http://www.sei.cmu.edu/appraisal-program/profile/pdf/CMMI/2005sepCMMI.pdf [Viewed on 01.02.06]

Sommerville, I., 2004, *Software Engineering*, 7th Edition, Addison Wesley, Reading MA.

Stelzer, D. and Mellis, W., 1998, 'Success Factors of Organisational Change in Software Process Improvement', in *Software Process Improvement and Practice*, Vol. 4, No. 4, pp. 227-250.

Strauss, A., 1987, *Qualitative Analysis for Social Scientists,* Cambridge University Press, New York.

Strauss, A. and Corbin, J.M., 1990, *Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory*, 1st Edition, Sage Publications.

Strauss, A. and Corbin, J.M., 1998, *Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory*, 2nd Edition, Sage Publications.

Sutton, S.M., 2000, 'The Role of Process in a Software Start-up', in *IEEE Software*, July/August, pp. 33-39.

Teasley S., Covi, L., Krishnan, M.S. and Olson, J.S., 2000, 'How Does Radical Co-location Help a Team Succeed?', in *Proceedings of the ACM Conference on Computer Supported Cooperative Work*, Philadelphia, USA, pp. 339-346

Thomas, D., 1995, 'Component-Based Software Construction: Making the Transition from Craft to Engineering', Object Management Group, New York.

Thomson, H.E. and Mayhew, P., 1997, 'Approaches to Software Process Improvement', in *Software Process Improvement and Practice*, Vol. 3, No. 1, pp. 3-17.

Turk, D., France, R. and Rumpe, B., 2002, 'Limitations of Agile Software Processes', in *Proceedings of Third International Conference on Extreme Programming and Flexible Processes in Software Engineering*, Italy, pp. 43-46.

Voas, J., 1999, 'Advice for Those Bitten by the Startup Bug', in *IT Professional*, May-June, pp. 38-45.

Wilkie, F.G., McFall D., and Mc Caffery F., 2005, 'An Evaluation of CMMI Process Areas for Small to Medium Sized Software Development Organisations', in *Software Process Improvement and Practice*, Vol. 10, No.2, June, pp. 189-201.

Wixon, D., 1995, 'Qualitative Research Methods in Design and Development', in *Interactions*, Vol. 2, No. 4, October, pp. 19-26.

Woodward, S., 1999, 'Evolutionary Project Management', in *IEEE Computer*, October, October, pp. 49-57.

Yamamura, G., 1999, 'Process Improvement Satisfies Employees', in *IEEE Software*, September/October, pp. 83-85.

Yin, R. K., 1994, *Case Study Research: Design and Methods*, 2nd Edition, Sage Publications.

Zahran, S., 1998, *Software Process Improvement: Practical Guidelines for Business Success*, Addison Wesley, Boston, MA.

List of Appendices

Appendix A

Interview Guide 1

Section 1. *Company Background*	
Question	**Comments**
Demographic (name, location, year established etc.)	
What is your business?	
How many are employed in total / In software development?	

Section 2. *Company Development*	
Question	**Comments**
Who founded the organisation / what is their background?	
Are all founders still with company?	
What expertise did the founders bring from their previous employment (technical, managerial, general confidence etc.)?	
How has the company developed since its foundation?	
What were the high points / low points in that development?	
What were the major events in the company development / what were the turning points?	
What went well during company's development / what defeats did you experience?	
What mistakes did you make along the way / what would you do differently next time?	
What were the greatest challenges you faced in starting up (recruiting, marketing, capital etc.)?	
What are the greatest challenges you now face?	
What is the single biggest issue that you/your company face in the next 12 months?	
What are your organisation's greatest strengths?	
What are your organisation's greatest weaknesses?	
Please tell me about your company's product history (no. of products, release dates, upgrades etc.)	
Currently what / where are your major markets?	
How have these developed over time?	

Section 3. ***People Issues***	
Question	**Comments**
What are the current roles within the organisation (***look for organisation chart***)?	
How have these roles changed over time?	
What are the current reporting structures?	
How have these changed over time?	
What are the major skills which you recruited?	
When were these people recruited (***look for recruitment adverts***)?	
Why did you pursue these skills in particular?	
What has staff turnover been like since the company was established?	
Please outline your recruitment procedures.	
How reliant on key employees do you feel you are?	
Please outline your employee training plans.	
Delegation and Training – What management training have you had since setting up the business?	
When did this training take place?	
On what sorts of training course have employees embarked since the company establishment?	
How often do you have company/team meetings?	

Section 4. ***Software Development Strategy***	
Question	**Comments**
Please outline your software development process.	
Is this process defined and documented?	
When was this process introduced?	
How has the process developed over time?	
What technical issues have emerged as the company has grown?	
How has the maintenance and enhancement strategy for software products evolved?	
What is the process for managing software projects and how has this evolved since the company started?	
What is the process for managing software quality and how has this evolved since the company started?	
What is the process for managing risk and how has this evolved since the company started?	
What is the process for developing and maintaining software documents and how has this evolved since the company started?	

How are requirements captured and defined?	
How has this changed over time?	
Have you introduced any software development initiatives over the years (e.g. quality, process definition, configuration mgt)?	
At what points were these introduced and why?	
What software development methodologies have you used over the years?	
What development platforms have you used?	
Have you sought external accreditation since your company's creation?	
At what point was this sought and why?	
Which of the following statements best reflect your thinking: - We have spent too little time documenting our policies and procedures - We have spent the right amount of time developing our policies and procedures - We have spent too much time documenting our policies and procedures.	

Appendix B

Interview Guide 2

Potential direction for questioning

How does way company start out affect process. Consultancy first? Straight into product development?

What impact do business realities have on the definition of process or certification? How important is customer demand?

Examine the reasons for, and the ways in which, formality is established

Assess impact of market downturn on use and application of process

What impact does background of senior development staff have on the definition, formality and use of process?

Examine how the absence of process manifests itself.

Is "frequency of upgrade" any reflection on the quality of a software process?

Assess the impact of key employees on the process. This could be important. If they support it and use it then it will be widely used. If they don't use it or are seen to work around it then it may be ignored by other employees. Check the role of "opinion formers". Also check the role management have in ensuring the process is enacted. How is this role dispensed and is it overt or covert? Is there an understanding that you can use less process on some projects?

Implication here that process training also demanded management training. Can historical management capability fit into ISO or other quality system? Process training here seems to be pure mercenary to achieve a specific aim, so that everyone "sings from the same hymn sheet". No evidence of training having any intrinsic value. What is the value of process from a company's perspective? What is the cost? Why did it take so long to introduce process?

Process decision made after requirements capture? Are both processes, and how they are to be used, defined and documented?

Has the process had any role in fixed pricing? Has the fact that ISO is now in place improved the accuracy of fixed price contracting?

What sort of judgement calls are we discussing here? Does it mean the use of the MoSCoW rules? What impact does it have on the customer? You may have to change process mid-stream; for example if you are using Waterfall and you find that you can't deliver everything on time then do you revert to a RAD process and deliver limited functionality.

Requires the use or collation of metrics. How much have projects been out in the past? In this case, the fact that projects may have lost money or varied greatly from original price charged, has not influenced or driven the development of process. Again the key question – what drives the development/creation of formal process within an organisation?

Despite the fact that metrics are in place, "rules of thumb" are still used.

No evidence from remainder of interview that ISO was done for marketing reasons.

If company had lost business (MS/FC/19/F) through not having ISO, then surely this was a driver to introduce it, rather than the arguments suggested here.

What resource have companies used to implement process? How much does it cost in time/effort and financially? Perhaps cost is not as great as imagined.

In this case it appears to have a direct relationship with the Motorola work, i.e., they got the Motorola work in house and could therefore charge a premium, which created the payback. What is payback time normally? ***Do companies only introduce process when they can see a direct payback? Is process introduction driven by direct financial benefits or some intangible, potential future reward?***

Seem to contradict the arguments in MS/FC/20/A. ***Also – key question. Is process introduced more quickly where key people/opinion formers have previous experience of using quality systems?***

In a small company, the CEO/founder has a major influence on all contracts and ergo on the process itself. Project management in this instance is essentially technical management and not account management. Is this typical in other small companies?

Again, limited role for project manager. CEO "would insist" that all metrics and reports were gathered and would then work out the schedule for the following week.

"When we could afford independent testing..." – it appears that the process has been reduced and is dependent on resources. ***Is this an example of "just enough" process? Is process perceived as overhead, a nice thing to do when you have the time and can afford it?***

Why do company make assumptions that only software engineers can write acceptance tests? Is this because it's the founder's culture ["you need people who know what they're selling..."]? What is the process for acceptance testing elsewhere? However, does customer have to write the acceptance tests? Buying a car, you allow the mechanic to do the checks for you. Buying a house you let a surveyor do the checks for you, so why not allow the software engineers to do the acceptance testing for you. However, there is an issue about the company who develop the software testing it for you.

Again process followed is resource dependent. Appears to be no standard process. ***Whilst one exists it is applied with varying degrees of formality to different projects.***

Again process followed is resource dependent. More tailoring of process as in MS/FC/23/D. Example of short-changed quality, perhaps.

There is a suggestion here and which I picked up from the interview that all employees were not supportive and that a certain amount of coercion or imposition was involved. ***How do you win employee approval/support for process?***

Does ability to record documentation online ensure more widespread use of process?

This suggests that testing is not taken very seriously. ***Testing is still perceived as the least important/most malleable part of the lifecycle.***

MS/RB/8/B – ***User acceptance tests are not designed by the customer.***

How flexible are other quality systems? Are firms aware of the flexibility issue or do they think things are essentially binary e.g. either a quality system in which **everything** must be documented and followed or no system which though ad-hoc, provides flexibility?

Small companies will not follow process "religiously" unless they have to. In other words where external customers/bodies dictate they will follow the process otherwise they will do "just enough" and cut out what they see is unnecessary. Why is this the case? Is it an issue of discipline? Cost? Time? What are the costs of following process fully? What are the costs of not doing so? Is it a human factor?

In small companies, is there an "official" software process (ISO 9001, proprietary) and an unofficial one (the one that's more commonly used!)?

The presence of ISO does not seem to have been of any benefit when foreign workers were recruited, yet it should have been. The process seems very light here ("we didn't produce reams and reams of documentation") and informal. From this and other comments it seems that the company were not following ISO very rigidly. It's the **official versus the actual** policy as seen in "9B". ***Do certified software companies follow the process or is it there to satisfy the certifying authorities (and potential customers!)?***

ISO 9001 certification not seen as a strength.

Process described is informal, yet the company has ISO 9001. Seems similar to the point made in 15B above.

Key Ideas

- Actual Vs "Official" process
- What impact does "champion" have? If senior figures have had prior exposure to process does that make a difference? Role of "opinion formers".
- What is the value/cost of process? What is the cost of not following a process?
- What drives the creation of process?

- Do companies only introduce process when they can see a direct payback? Is process introduction driven by direct financial benefits or some intangible, potential future reward?
- Is process perceived as overhead, a nice thing to do when you have the time and can afford it?
- Is process/amount of process used resource dependent?
- How do you win employee approval/support for process?
- Does ability to record documentation online ensure more widespread use of process?
- How flexible are other quality systems? Are firms aware of the flexibility issue or do they think things are essentially binary e.g. either a quality system in which ***everything*** must be documented and followed or no system which though ad-hoc, provides flexibility?

List of Questions

Section 1. *Company Background*	
Question	**Comments**
Tell me a little about the company, when it started, when you joined how many are employed in software development etc.	
How many are employed in total / In software development?	
Who founded the organisation / what is their background?	
Are all founders still with company?	
What expertise did the founders bring from their previous employment (technical, managerial, general confidence etc.)?	
What were the high points / low points in that development?	
Please tell me about your company's product history (no. of products, release dates, upgrades etc.)	

Section 2. *People Issues*	
Question	**Comments**
What are the current roles within the organisation?	
How have these roles changed over time?	
What are the major skills, which you recruited?	
What has staff turnover been like since the company was established?	
How reliant on key employees do you feel you are?	
Please outline your employee training plans.	

Section 3. *Software Development Strategy*	
Question	**Comments**
Please outline your software development process.	
Is this process defined and documented?	
When and how was this process introduced?	
Why did you introduce process at that time? What drove the development/creation of formal process?	
What did it cost to introduce process? In financial/effort terms?	

What has the payback been? How long did it take to recover investment?	
How did things work before the process was introduced?	
How did you get employee buy-in for process establishment?	
How does process impact on software development (***All stages from bid to delivery – check in particular the testing phases***)?	
How has the process developed over time?	
Has market downturn affected the process in any way?	
Is the same process used for all projects or does it vary from project to project? (“***Official Vs Actual***”)	
Do new employees get formal training in the company software process?	
What is the process for managing software projects and how has this evolved since the company started?	
What is the process for managing software quality and how has this evolved since the company started?	
What is the process for managing risk and how has this evolved since the company started?	
What is the process for developing and maintaining software documents and how has this evolved since the company started?	
Have you introduced any software development initiatives over the years (e.g. quality, configuration mgt)?	
At what points were these introduced and why?	
Have you sought external accreditation since your company’s creation?	
At what point was this sought and why?	

Appendix C

Full List of Codes from Atlas

Absence of documentation management
Absence of process
Absence of quality system
Acceptance test process
Actual SDLC Vs "Official" SDLC
Actual Vs Estimates
Admin heavy
Administration
Adopt
Analysis and design
Application type
Arduous
Attitude towards process - Fear/antagonism
Attitude towards process - Positive/embracing
Audit process
Automated documentation
Automated process
Automated testing
Background drives SPI
Background of CEO
Background of founder
Background of founder - academia
Background of founder - IT
Background of founder - non-IT
Background of Interviewee
Background of software development manager
Baggage
Beginnings of formality
Benchmarking
Benefits of CMM
Benefits of co-location
Benefits of distributed development
Benefits of documentation
Benefits of early integration
Benefits of experienced staff
Benefits of flexibility
Benefits of having a quality system
Benefits of process
Benefits of rigorous processes
Benefits of RUP
Benefits of small teams/companies
Benefits of XP
Benefits of XP to developers
Beta testing
Bigger team needs more process
Black-box testing
Bogged down
Boring
Bottom-up SPI
Build manager role
Bulky
Bureaucracy
Buried in paper
Business critical
Business decision
Business development
Business event
Business evolution
Business focus
Business HR culture
Business mistake
Business mix
Business model
Business models
Business objective
Business realities drive quality initiative
Business refocus
Business turning point
Business vision
Casual
Certification
Challenges to introducing process
Change request management
CMM
Co-location
Code reviews
Code wins
Coding standards
Collaborative development
Commercial development models
Commercial SPI models

Common sense
Communication
Company Business Model
Company size
Component-based development
Concurrent project development
Configuration management
Consultancy services
Contextual Issues
Control
Cost based on coding effort
Cost of implementing ISO
Cost of poor quality
Cost of process
Cost of support
Creative activity
Creativity
Cross-functional team
Crushing us in paperwork
Current business challenge
Current business position
Current product suite
Current Staff Levels
Customer - developer relations
Customer base
Customer driven development
Customer expects quality
Customer feedback
Customer involvement in development
Customer support
Cut some corners
Cut some stuff out of it.
Defect analysis
Defect recording
Defined
Deliverables
Delivery date is crucial
Delivery rate
Depend on the personalities
Dependence on key staff
Dependence on one customer
Design
Design specification
Developer estimates
Developer productivity
Developer psychology
Developer responsibility
Developer support for SPI
Developers and process
Developing process plans
Development - bespoke
Development - customised
Development - product
Development challenges
Development department structure
Development models
Development partners
Development team size
Development tools
Discipline
Distributed development
Distributed XP
Document management system
Documentation
Documentation and bureaucracy +
Documentation support for training
Downsides of inexperienced staff
Downsides of RUP
Downsides of XP
Drag
Effort
Employee buy-in to process
End-user documentation
Enforcement
Engineer-driven
Engineering esteem
Engineering velocity
Entrepreneurial
Establishing business credibility
Establishing Process
Estimating project cost
Estimating testing time
Exception reporting
Expanding customer base
External audit
Feature-driven development
Fill in all this paperwork
Filling in forms
Find the time
Fixed price development
Flexibility
Flexible quality system/process

Freedom
Frequency of upgrade
Functional manager role
Gqm
Heavy
Heavyweight
Hire profile
Hiring experienced staff
Hiring Expertise
Hiring focus (background and applications)
Hiring focus (behavioural/personality)
Hiring focus (technologies)
Hiring procedures
Historical data
Horrible BSI audits
Impeded
Implicit requirements
Importance of estimates
Importance of following process
Importance of project management
Importance of quality
Importance of structure
Importance of understanding target domain
Imposition of quality system by customer
Incremental process improvement
Individual responsibility
Individualistic
Influence of key staff
Informal process activities
Informal process used for estimating
Innovation
Internal audit process
Inventive
ISO 9000
ISO not suitable for software
ISO Vs Non-branded process
Iterative development
Just for the sake of filling out paper
Key staff remuneration
Large overhead and administration
Licencing details
Lifecycle models
Limitations of existing process
Luxury
Maintenance
Maintenance contract details
Management style
Management style and staff buy-in
Market forces
Market niche
Market requirements
Market Sector
Mature
Mentality
Messy
Metrics-driven improvement
Metrics - attributes
Metrics - complexity
Metrics - effort
Metrics - methods
Metrics - quality
Metrics - schedule and estimation
Metrics - testing effort
Metrics collected
Mindset
Minimum cost
Minimum Documentation
Minimum process
Modelling
Modular development
Motivation
Moving from small to large
Moving from waterfall to an iterative process model
Multi-skilling
Multimedia development
Multiple platforms
Negative business developments
Negative SPI trigger
New CEO
No return to pre-process
Non-software process
Non-specialised development team
Nonsense detail
Off-site development
Operational factors
Organisation structure
Our non-existent process
Outsourcing

Outsourcing problems
Over-engineering
Over the top
Overdo it
Overhead
Overkill
Pair programming
Paper mountain
Paper trail
Pedantic
Peer pressure
Peer review
Penalty clauses
People factors
Phase documentation
Platform support
Pool of Engineers
Poor estimating
Poor management
Poor quality
Poor testing
Positive business developments
Positive SPI trigger
Pre-release testing process
Previous process
Pride
Prioritising requirements
Proactive
Proactive problem solving
Process-related documentation
Process - development
Process - first steps
Process - services
Process - turning points
Process activities
Process based on system size
Process based on team size
Process based on third party software
Process benefits from hiring
Process challenges
Process definition
Process depends on staff
Process diversity
Process erosion
Process evolution
Process formality
Process formation
Process formation and evolution
Process heavy
Process improvement funded through revenue increase
Process incentives
Process Inertia
Process influencers
Process Influences
Process limitations
Process management
Process measurement
Process models
Process negatives
Process negatives & Documentation
Process opportunity
Process outcomes
Process ownership
Process prototyping
Process reflects what company did pre-process
Process review
Process scaling
Process scope
Process startup
Process support
Process tailoring
Process Vs Product
Process weaknesses
Process/ISO training
Product development
Product is prime
Product line development
Product management
Product model
Product Price
Product Revenue
Product suite expansion
Product type
Product/services model
Product/support model
Project documents
Project estimation
Project kick-off
Project management
Project manager role

Project meetings
Project planning
Project pricing decision
Project size
Project tracking and control
Project/team size
Proprietary development model
Prototyping
QA activities
QA role
QA write user acceptance tests
Quality
Quality control process
Quality control techniques
Quality depends on staff
Quality focus
Quality level
Quality management
Quality management review process
Quality manager
Quality manager role
Quality reviews
Quality software, not quality documents
Quality standards in practice
Quality system
Quality system improvements
Quality system Vs process
Quality team
Quality team role
Rational Rose
Re-engineering
Real jobs
Reams and reams of paper
Reams of documentation
Reason for joining company
Reasons for delay in introducing process
Reasons for documentation
Reasons for introduction of CMM
Reasons for introduction of ISO
Reasons for not doing integration testing
Reasons for not following process
Reasons for not implementing XP
Reasons for not introducing CMM
Reasons for not introducing ISO
Reasons for not pursuing certification
Refactoring
Reference customer
Reference models
Reflections/recommendations on introducing ISO
Regulated market
Rejection of quality system by customer
Release process
Reliability
Requirements capture
Requirements change
Requirements gold-plating
Requirements management
Research and development
Resources
Restrictive
Results-driven
Reuse
Reuse criteria
Revenue more important than process
Review procedures
Reviews and Inspections
Rework
Rigid
Rigorous
Risk management strategy
Road map development
Role consolidation
Role of Interviewee
Role specialisation
Rote
RUP
Scalability
Scale
Second phase process
Service level
Services model
Set-up and administration
Simple Design
Simple documentation
Situation pre-process
Six sigma
Slow everything down
Small company/team issues
Small team overhead

Small team productivity
Software application area
Software development activities
Software development model decision
Software development process
Software release process
Software tools
Source code documentation
Speed
SPI focus
SPI trigger
Staff appraisals/reviews
Staff motivation
Staff Numbers Expansion
Staff skill composition
Staff training
Staff turnover
Stand-up meeting
Standards
Start of coding
Structured
Support for employee recruitment
Support from customers
Support team
System acceptance criteria
System and acceptance test
System architecture
System customisation
System/unit testing process
Systemisation
Tacit knowledge
Team Leader responsibility
Team size
Technical/technology challenges
Technology initiatives
Template
Test-first development
Test Scripts
Test team
Tester acting as user
Testing checklist
The need to tailor a quality system
Theory X management
Theory Y management
Third phase process
Thrashing around
Throwaway code
TickIT
Time to market
Too detailed
Top down SPI
Top management support for process
Traceability
Triggers & evolution
UML
Under-engineering
Unnecessary evils
Usability
Use cases
Use of 'V' model
Use of checklists
Use of RAD model
User acceptance testing
User profiling
User stories
Vague requirements
Value
Verbose
Version management
Wasn't relevant
Waste of everyone's time
Waterfall model
Way they work
Ways to introduce process
Weight
What is software process?
Work breakdown structure
Wouldn't have the patience
XP