Available online at www.sciencedirect.com

ScienceDirect

The Journal of Systems and Software 81 (2008) 772–784

The Journal of Systems and Software

www.elsevier.com/locate/jss

Investigating software process in practice: A grounded theory perspective

Gerry Coleman [a], Rory O'Connor [b,*]

[a] Department of Computing, Dundalk Institute of Technology, Dundalk, Co. Louth, Ireland
[b] School of Computing, Dublin City University, Glasnevin, Dublin 9, Ireland

Received 26 January 2007; received in revised form 9 July 2007; accepted 15 July 2007
Available online 3 August 2007

Abstract

This paper presents the results of a study of how software process and software process improvement (SPI) is applied in actual practice in the software industry using the indigenous Irish software product industry as a test-bed. The study used the grounded theory methodology to produce a theory, grounded in the field data, that explains how software processes are formed and evolve and when and why SPI is undertaken. Our research found that SPI programmes are implemented reactively and many software managers are reluctant to implement SPI best practice models because of the associated costs.
© 2007 Elsevier Inc. All rights reserved.

Keywords: Software process improvement; Grounded theory; CMMI; ISO 9000; XP

1. Introduction

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 achieving higher product quality or reducing costs. SPI models have been developed to assist companies in this regard and purport to represent beacons of 'best practice'. Some large software organisations have used 'best practice' process improvement models such as the Capability Maturity Model/Capability Maturity Model Integrated (CMM/CMMI) (Ahern et al., 2004) and the International Organisation for Standardization (ISO) 9000 series (ISO, 1992). More recently, agile methodologies such as Extreme Programming (XP) (Beck, 2000) have been used in SPI programmes and have been widely embraced by software organisations. Although commercial SPI models have been highly publicised and marketed, they are not being widely adopted and their influence in the software industry therefore remains more at a theoretical than practical level.

In the case of CMMI, evidence for this lack of adoption can be seen by examining the SEI CMMI appraisal data for the years 2002–2006 (SEI, 2006), in which time just 1581 CMMI appraisals were reported to the SEI. Whilst we acknowledge the SEI data includes only appraisals that have been both reported to the SEI and authorized for public release and there are appraisals that are not reported or authorized for public release, it is clear that the published figures represents a very small proportion of the world's software companies and company in-house developers. In addition, there is evidence that the majority of small software organisations are not adopting standards such as CMMI. For example, an Australian study (Staples et al., 2007) found that small organisations considered that CMMI "*would be infeasible*". Further investigation of the SEI CMMI appraisal data reveals that in the case of Ireland – a country whose indigenous software industry is primarily made of small to medium sized organisations (SMEs) – fewer than 10 CMMI appraisals were conducted during 2002–2006 from a population of more than 900 software companies (Enterprise Ireland, 2005). Therefore, it is

* Corresponding author. Tel.: +353 1 700 5643; fax: +353 1 700 5442.
E-mail addresses: gerry.coleman@dkit.ie (G. Coleman), roconnor@computing.dcu.ie (R. O'Connor).

0164-1212/$ - see front matter © 2007 Elsevier Inc. All rights reserved.
doi:10.1016/j.jss.2007.07.027

also clear that the Irish software industry is largely ignoring the most highly-publicised SPI models. In the case of CMMI (and its predecessor CMM), Staples and Niazi (2006) discovered, after systematically reviewing 600 papers, that there has been little published evidence about those organisations who have decided not to adopt CMM(I).

Accordingly, the motivation for our research originates in the premise that in practice software companies are not following 'best practice' process improvement models. On this basis, we set out to answer two related research questions: *Why are software companies not using 'best practice' SPI models?* and *What software processes are software companies using?* – with a view to creating a rich, explanatory theory of software process in practice. Preliminary investigation of the two research questions raised the following linked questions:

RQ1 *How are software processes initially established in a software company?*
RQ2 *How do these software processes change?*
RQ3 *What causes these software processes to change?*
RQ4 *How do the operational and contextual factors present in organisations influence the content of and adherence to software processes?*

This paper is organized as follows: The remainder of Section 1 describes the context in which this study was undertaken. Section 2 looks at SPI from an industry perspective. Section 3 examines the chosen research methodology and Section 4 describes how the methodology was used in the study. Section 5 looks in detail at the research results and Section 6 contains a discussion on the findings and looks at the implications the findings have for practitioners and researchers. Finally, Section 7 presents some concluding remarks and presents a future research agenda.

1.1. Study setting

To ensure the participation of software development professionals who would be familiar with the considerations involved in using both software process and process improvement models, it was decided to limit the scope to software product companies whose primary business is software development. In addition, given the geographical location of the researchers, it was decided to confine the study to Irish software product companies which has the added advantage of restricting the study to within the same economic and regulatory regime. Furthermore, restricting the study to indigenous Irish software product companies significantly increased the prospects of obtaining the historical information required to understand process foundation and evolution which would not be the case with non-Irish multinationals operating in the country, as their process would likely have been initially developed and used within the parent company prior to being devolved to the Irish subsidiary.

Indigenous Irish software companies have played a key role in the Irish software economy. However, 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. Of those 630 indigenous software companies, the venture capital group, HotOrigin (2004) estimate there is a total of 417 indigenous software product companies and classify them according to three stages of development: 'Start-up' (1–25 employees), 'Build' (26–75 employees), and 'Expansion' (75+ employees). The most recent figures available show 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. Thus, the indigenous software product sector offers a potentially fruitful area for research enquiry.

2. Industry adoption of software process improvement

In addition to the plan-driven approaches such as CMMI and ISO 9000, agile methodologies have been used in SPI programmes. Two other ISO standards are directly related to SPI: ISO/IEC 15504 ('SPICE') which is a framework for the assessment of software process and ISO 12207 which aims to be 'the' standard that defines all the tasks required for developing and maintaining software. However, from the commercial SPI perspective, this study was dominated by two particular models CMM(I) and ISO 9000 and the development methodology XP. Accordingly, the following sub-sections which provide a brief discussion on industrial perspectives of SPI will be restricted to CMM(I), ISO 9000 and XP.

2.1. Industry perspectives on CMM(I)

Numerous studies including (Humphrey et al., 1991; Herbsleb et al., 1997; Pitterman, 2000), report significant success with using CMM. Because of its relatively recent arrival, fewer organisations have adopted CMMI and there are correspondingly fewer reports in the literature of its application. However, Goldenson and Gibson (2003), Miller et al. (2002) and Hansen (2004) all claim benefit from its deployment.

The CMMI and the approaches associated with it also has a number of opponents. Bollinger and McGowan (1991), Baker (1996) and Fayad and Laitinen (1997) express reservations about the maturity level grading scheme whilst Bach (1994) highlights the number of very successful companies whose practices, he believes, would be classified at CMM level 1. A frequently expressed reservation about CMM(I) is its suitability for small organisations (Brodman and Johnson, 1994). Case studies document the difficulties (Batista and Dias de Figueiredo, 2000) that small organisations encounter using CMM(I) and the mismatch between the 'perceived benefit and the actual benefit' and as 'perceived value and actual value' (Wilkie et al., 2005). Staples et al. (2007) suggest that '*small organisations should not be seen as being at fault for not adopting CMMI, instead the SPI approaches, sales and marketing should be improved*'.

2.2. Industry perspectives on ISO 9000

ISO 9000 is a series of standards used to certify the quality of systems used by an organisation (ISO, 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). However, unlike CMM(I), ISO 9000 does not provide a road map for improvement beyond the adherence to quality management documents. There are few published studies which directly report on ISO 9000's application in a software development environment (El Emam and Briand, 1997).

A common criticism of ISO 9000 is the amount of money, time and paperwork required for registration (Clifford, 2005). According to Barnes (2000), "*Opponents claim that it is only for documentation. Proponents believe that if a company has documented its quality systems, then most of the paperwork has already been completed*". Much of what is written is critical of the fact that ISO 9000 is a general standard and not specifically geared for software production. This is documented by Fitzgibbon (1996) who believes ISO 9000 is difficult to apply in a software environment and by Coallier (1994) who feels the standard is insufficient and that a total quality approach, incorporating continuous improvement, is needed. Further support for this view comes from Oskarsson and Glass (1996) who believe that ISO 9000 is primarily applied in the software domain because of its market credibility and from Demirors et al. (1998) who believe it negates the advantages accruing to small software firms.

2.3. Industry perspectives on agile methodologies

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(I)-based processes and agile 'processes', the label 'plan-driven' has been applied to processes based on the so-called 'disciplined' models, such as CMM(I), 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.

XP is the most popular and widely recognised methodology (Bowers et al., 2007) in the agile family and has by far the greatest coverage of any of the agile methodologies in the literature. A number of authors have reported on using XP in a variety of environments including embedded systems (Grenning, 2001), web development (Murru et al., 2003), event driven systems (Rasmusson, 2003), biotech systems and project management (Sliger, 2004) and with legacy applications (Coleman and McAnallen, 2006). Interestingly, all of these studies have used scaled-down versions of the methodology in their applications. Aveling's (2004) research supports this concluding that "*partial adoption of XP is more common than full adoption*". Further support in found in Bowers et al. (2007) who state that "*XP adopters would be best served striving to apply the practices in spirit*".

3. Methodology

The methodology chosen for the study was grounded theory (Glaser and Strauss, 1967). The emphasis in grounded theory is on new theory generation. 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). 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. 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. Grounded theory was chosen as the method of enquiry for the following reasons:

- 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 practitioners, offered the greatest potential.
- It has established guidelines for conducting inductive, theory-generating research.
- It is renowned for its application to human behaviour (Martin and Turner, 1986). Software development is labour-intensive and software process relies heavily on human compliance.
- It is an established and credible methodology in sociological and health disciplines (Sheldon, 1998), and a burgeoning one in the IT arena.

Furthermore, like others who have applied grounded theory (Baskerville and Pries-Heje, 1999; Hansen and Kautz, 2005), 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. For a fuller discussion on grounded theory, the rationale behind its selection and how it was implemented in this study please refer to Coleman and O'Connor (2007).

4. Conducting the grounded theory study

The study was divided into three phases, a Preliminary phase (P) to help frame the study and test the interview guide and approach, a more detailed phase (Stage 1) which developed the initial concepts and categories and enabled evaluation of the theoretical sampling process and the final phase (Stage 2) which further developed the categories and concepts to produce the grounded theory. In total the study involved 25 interviews across the 21 companies profiled in Table 1. The participants in Stage P were chosen from personal contacts of the researchers. For Stages 1 and 2, in parallel with making contact with individuals known second-hand to the researchers, 'cold' e-mailing was used to set up the next series of interviews.

4.1. Preliminary study phase

To generate more detailed information on how the sampling process should progress, a preliminary study phase involving four interviews across companies 1–3 was undertaken. To support the semi-structured interviewing process, an interview guide based on the researchers' experience as 'cultural insiders' and their prior familiarity with the literature, was created for use with the first two interviews. There were 53 questions divided over four categories: Company Background, Company Development, People Issues and Software Development Strategy. The interviews were taped, transcribed and then coded by hand in accordance with the open coding procedure of grounded theory. The initial interviews highlighted several drawbacks with the interview guide which drove the development of a second interview guide containing 34 questions across three categories: Company Background, People Issues and Software Development Strategy. This interview guide was then used on interview 3 and 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. In addition, Stage 2 analysis of the software companies' target market indicated that the intended list of companies, in the full study, should incorporate as many sectors as possible.

4.2. Study Stage 1

The next phase of the study (Stage 1) involved interviews with an additional 11 companies. 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, after 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 anal-

Table 1
Company profile by category

Company	Market sector	Category	Total number of employees	Number in software development	Study stage
1	Telecommunications	Start-up	6	3	P
2	Corporate secretarial	Build	50	20	P
3	Telecommunications	Start-up	10	3	P
4	Telecommunications	Build	70	30	1
5	Telecommunications	Start-up	12	6	1
6	Compliance management	Expansion	100	40	1
7	Enterprise	Expansion	150	100	1 & 2
8	E-learning	Expansion	120	70	1
9	Information quality	Build	27	9	1
10	Telecommunications	Start-up	15	12	1
11	Telecommunications	Expansion	160	110	1 & 2
12	Financial services	Build	35	23	1
13	Financial services	Expansion	130	90	1
14	Interactive TV	Build	60	40	1 & 2
15	Public sector	Expansion	150	90	2
16	Medical devices	Start-up	19	9	2
17	Telecommunications	Build	70	35	2
18	Public sector	Start-up	3	3	2
19	HR solutions	Build	30	15	2
20	Games infrastructure	Build	40	20	2
21	Personalisation	Build	50	40	2

Table 2
Study Stage 1 provisional hypotheses

Number	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	In 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(I)
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(I)
H9	Non-ISO 9000/CMM(I)-certified Irish software product companies generate only minimum documentation
H10	Within Irish software product companies, adoption of ISO 9000 and CMM(I) 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(I)
H12	The costs associated with achieving and adhering to ISO 9000 and CMM(I) prevent their adoption in Irish software product companies

ysis and inclusion as questions in subsequent interviews. A consequence of this was that the interview guide was constantly updated.

Because of the clear repetitions within the data, the memos and propositions created during the constant comparative process were further analysed and a number of provisional hypotheses formulated (Table 2), which had the potential to explain how the concepts and categories emerging from the study were linked. 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.

4.3. Study phase 2

The requirement to test these provisional hypotheses drove the development of Stage 2 which involved the participation of seven new companies and comprised 10 further interviews. Three of these interviews involved re-interviewing earlier participants, a technique available to grounded theory studies and supported by Goulding (2002) as it allows for a comprehensive checking and verification process of the data already analysed. The seven new companies (companies 15–21) were specifically selected as their business sectors helped extend the scope of the study and ensured that theoretical categories were not being established on an excessively narrow basis. During the Stage 2 fieldwork the semi-structured interview questions were primarily derived from the Stage 1 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). Full category 'saturation' was reached on the conclusion of interview 25 as, in line with Goulding's (2002) assertion, similar incidences within the data were now occurring repeatedly and proceeding would be unlikely to generate any further contrary data.

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), to use them to develop and saturate the core categories. Whilst all of the hypotheses were 'tested' and verified in Stage 2 of the study, one hypothesis (H6) failed to develop further during that Stage 2. 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, which are presented in the next section.

5. Research results

The emphasis in grounded theory is on theory generation, where a theory is 'a set of well-developed categories (e.g. themes, concepts) that are systematically interrelated through statements of relationship to form a theoretical framework' (Strauss and Corbin, 1998). The analysis in this study showed that there was one central category to support and link two theoretical themes. The final list of themes, the core category and the main categories identified by the study are shown in Table 3. The categories and the various relationships were then combined to form the theoretical framework as shown in Fig. 1. Within the theoretical framework each node is linked by a precedence

Table 3
Themes, core categories and categories

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

[a] From hereon, the themes, categories and core category are denoted in italics.

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 that were renamed or merged with other codes during the analysis process.

The reasoning behind the processes companies are using is contained within the explanations of the study's two key theoretical themes, *Process Formation* and *Process Evolution* and its core theoretical category, *Cost of Process*. The following subsections will present these findings in more detail. In keeping with the fundamental tenets of grounded theory, extracts of the interview transcripts will also be presented in support of the findings.

5.1. Process formation

One of the key theoretical themes is *Process Formation*. In the study companies the title of the person with overall responsibility for software process differed. For clarity the generic title 'Software Development Manager' has been used in this study. The findings show that how process is formed depends on several factors: *Background of the Software Development Manager*, essentially the expertise that manager has accumulated over their working and educational lives; the founder's, and the Software Development Manager's *Management Style*; the *Market Requirements* or demands of the market in which the company operates.

Where the software development manager had worked before and what process/process improvement model they used shaped the process that the software development manager used in their current company. The following extract from company 7 is typical of the company 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*". The CTO of company 9 also provides a representative comment on the influence of the *Background of the Software Development Manager*: "*In terms of technology, I'm the CTO, I was hired [in week 2*

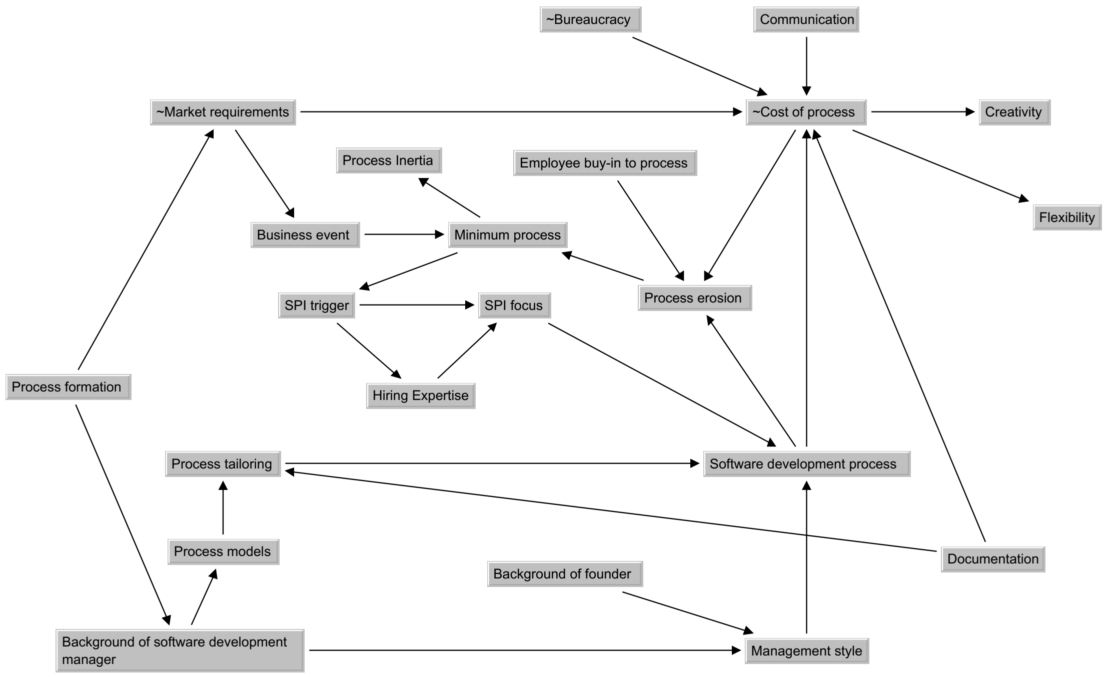

Fig. 1. The theoretical framework.

of company's existence] to build the team, build the vision and build the products... I've been involved in SPI wherever I have gone and here I make sure that the processes from day 1 are reasonable if not great".

The category *Management Style* describes the way a leader discharges their administrative functions and motivates and communicates with their staff (Buchanan and Huczynski, 1985). 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 which we categorised as 'Command and Control' with strong similarities to McGregor (1985) 'Theory X' style. Examples of this *Management Style* can be seen in company 1 who 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*". In opposition to 'command and control' structures, many company managers operate what can be characterised for this study as an 'Embrace and Empower' regime, similar to McGregor (1985) 'Theory Y' style. In this context, the opinions of subordinates are valued and included as part of software development policy and there is greater evidence of trust in development staff and their ability to carry out tasks with less direct supervision. Agile methods such as XP, with its advocacy of self-empowered teams and shared ownership, is more associated with this style of management and was more widely deployed in companies exhibiting this style of management, as exemplified by company 12: "*If you have 1 guy working on a piece of consultancy with 15 years experience he understands the principles of how we work. He knows what he's doing and doesn't need me interfering*".

The *Market Requirements* of the target market are fundamental influencers of the process adopted by a software organisation. A good example of this is a company who propose to target the medical sector but who initially had a short timeline to develop prototype training products for demonstration at a medical trade show: "*We developed the training product using XP and this allowed us to get the core software technology built and develop an early revenue stream. When we move up the value chain into surgery where it will need FDA approval we will have to change the process*".

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. 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. 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 for our own size and scale*".

5.2. Process evolution

The theoretical network describing *Process Evolution* is contained in Fig. 2. The study shows that *Process Evolution* does not occur in a linear fashion and is directly related to the events that the business experiences. Software *Process Evolution* occurs as follows. Over time, the company's existing *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.

Process Erosion takes place for a number of reasons. 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: "*Lately we haven't followed it [ISO 9000] as closely as we should because the projects we've had are small-scale*". In many cases within the companies, size of project is the determining factor in relation to what process, or how much process, is used. However, 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-1990s as we just don't have the time. There are so many different sub projects going on simultaneously that in order to get system testing done 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 management compliance.

The outcome of *Process Erosion* is an operational *Minimum Process*, which we define 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. *Minimum Process* results from *Process Erosion* and represents a further reduction of the already tailored process. The experience of one company illustrates this in relation to configuration management:

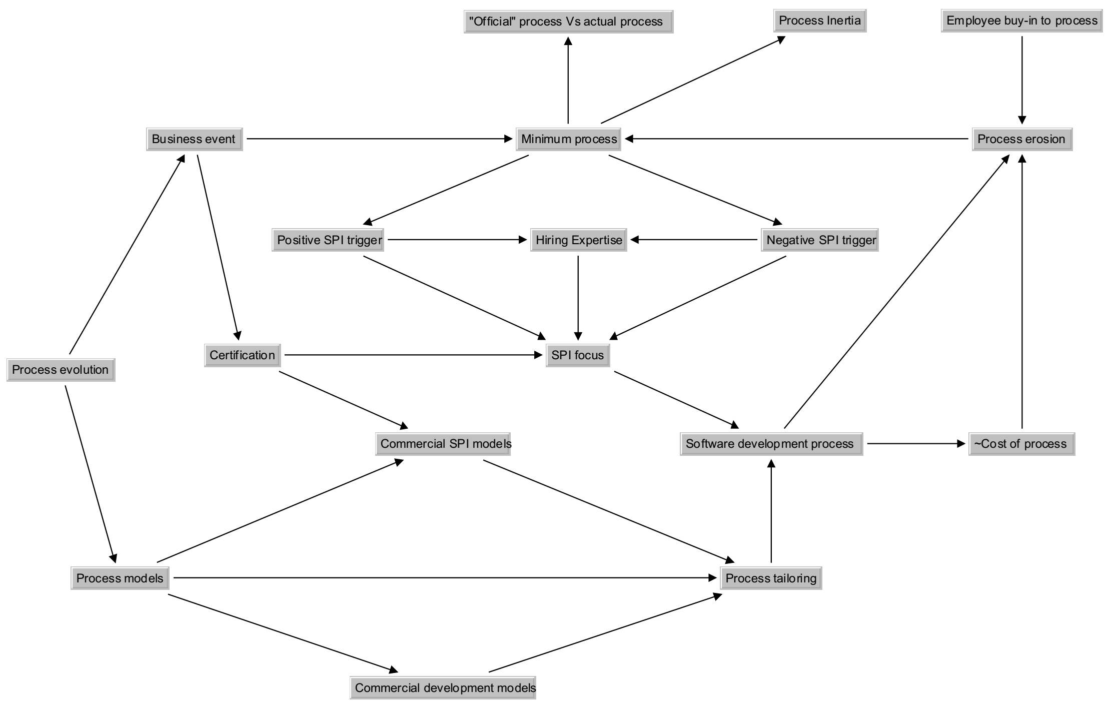

Fig. 2. Process evolution network.

"*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*".

A linked issue which affects how 'minimum' the process is in practice is the level of *Employee Buy-in to Process*. In a number of cases the biggest issue rested with the senior staff as exemplified by this interview extract: "*I have difficulty getting developers to write their weekly status reports. The better the developer, the less likely they are to send them. The best developers 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'.

On an ongoing basis the *Minimum Process* will suffice as long as the operational conditions in which it is being used remain the same. However changing business conditions generate an *SPI Trigger*, which can be 'positive' or 'negative', and necessitates process change. For example, as marketing efforts generate new, larger customers, process changes are often essential, thus creating a *Positive SPI Trigger*, as explained here: "*So, as you get progressively bigger deals you also have to scale your development resources and group to be able to handle that. A bigger team needs more process*". Nonetheless, the vast majority of process improvements reported by the study participants took place because of *Negative SPI Triggers*. These took a number of different forms, including inadequate *Quality* and poor project management and is best captured by this interview extract: "*Up to then we were selling to the Irish market and we realised people were coming back and they weren't even happy with the quality of the forms we generated from our software. There were numerous typing mistakes and nothing was really tested as it should have been*".

In many cases *Hiring Expertise* was used to deal with a *Trigger*, as companies took the view that the business event was either caused by a collective failure on behalf of all the current employees or could not be solved from within. Company 8 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*".

Process Inertia is an apathy 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 interview extract best describes managerial indifference: "*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?*".

5.3. Cost of process

In the course of the study interviews, managers expressed the belief that process has a significant cost which they attempted to keep to a minimum. What the managers perceived as the *Cost of Process* centred on a number of factors, illustrated in the network diagram Fig. 3.

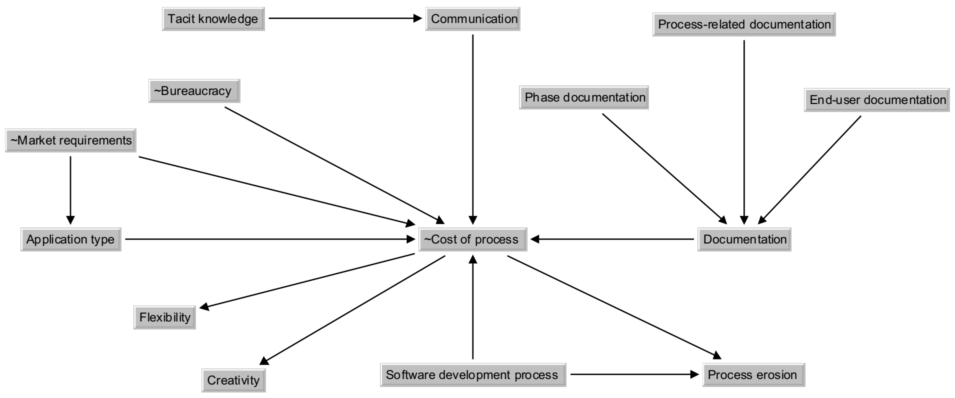

Fig. 3. Cost of process network.

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. Managers divided process into two separate categories, 'essential' and 'non-essential'. Essential process was that which was most closely linked to the product such as requirements gathering, testing and design. Non-essential process, which in the view of managers could often be omitted included process/quality-related documents, software measurement and many management activities such as planning, estimating and staging meetings. Several 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, as it is a synonym for 'extravagance', 'indulgence', or 'something inessential' (New Oxford Thesaurus of English, 2001). The following comment from a company 2 illustrates this point: "*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*".

The managers interviewed for this study believe *Documentation* is one of the single biggest contributors to the *Cost of Process*. 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*. Reduced *Documentation* was associated with situations where managers had high levels of trust in their developers and their experience as explained in this interviewee comment: "*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*".

Because *Documentation* was seen by the managers as such a significant process cost they encouraged 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. One company explained 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*". 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 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. 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*.

Process was also perceived by managers as having a negative impact on a development team's *Creativity* and *Flexibility*. Software companies, especially start-ups, need to be flexible, creative, dynamic and capable of delivering products quickly in order to survive. Several of the start-up companies in this study saw processes as primarily of benefit to established companies, as described here: "*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 think you won't get the same level of innovation or creativity*". 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*, as illustrated by this interviewee comment: "*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*".

5.4. Process improvement models

Of the 21 study companies, three are currently ISO 9000 certified and one is embarking on the ISO 9000 certification process. None of the companies are using CMMI. Significantly, though many of the companies classified their usage of XP as a process improvement initiative and an example of best practice. Resistance to all of these models stemmed from their perceived cost of adoption.

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*". 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 such as with this company: "*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.*".

There was strong opposition to the use of ISO 9000 by a number of software managers in the study. Some of this centred on its perceived emphasis on procedure and *Documentation* rather than product *Quality*. This 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*". 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: "*Initially we tried to follow an ISO mode. But that just crushed us in paperwork. So we abandoned it as we have a small number of engineers and we needed to be producing output*".

Awareness of 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(I) from previous employment, none had incorporated it into their present positions, deeming it unsuitable for a small software product company: "*If you look at CMMI 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 CMMI-style process is outweighed by the time it takes to deliver it*". The opinions of one manager, 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 we were doing, so it wasn't pursued*".

6. Discussion

6.1. Research contribution

This 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. This study moves beyond much of the mainstream 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 thereby enabling 'practice to inform theory' and importantly provide a challenge to '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.

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 multinational corporations and their lessons have extremely limited resonance in a small software product company. Software SMEs can identify with what is being stated in this study and with the described prevailing conditions of limited resources, personnel and time. 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.

6.2. Implications for practitioners

The findings of this research contain useful lessons for software entrepreneurs who need to make decisions about process and process change as their organisations grow. The theory presented here represents a form of 'experience map' illustrating some of the pitfalls a software product company could face and how others have avoided or resolved them. One of the lessons from practice 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, as this role is pivotal to future success. Similarly, 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.

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*. Companies who have gained from sharing *Tacit Knowledge* have generally had a workspace and supporting environment conducive to informal information exchange between employees. Organisations who have a more closed and rigid workspace will have to consider measures to overcome this if they are to implement a policy supporting informal *Communication*.

6.3. Implications for researchers

This research indicates that SPI adoption and success is not merely a matter of knowledge and training. The reasons that companies avoid SPI is not because they do not know what to do or how to do it but that they do not feel any necessity to do it until events dictate otherwise and even then 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.

The question of how CMMI can produce positive results in small settings has been explored by a number of researchers including (Coleman Dangle et al., 2005; Horvat et al., 2000; Hansen, 2004). However, the argument put forward within our study 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 CMMI are always going to find it difficult to penetrate small software organisations. Such contextual realities must be considered by SPI researchers. The implication being that perhaps too little time has been spent investigating why software SMEs are not prepared to adopt or even experiment with these models.

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 (Dorling, 1993). 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 CMMI. However, the absence of knowledge about the standard amongst practitioners should give cause for concern amongst its founders and advocates.

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' perception is that SPI means increased *Documentation* and *Bureaucracy*. Such a perception is widespread and is seen as a 'feature' of plan-driven approachs such as CMMI. Whether or not this is true is a moot point. The fact that managers associate CMMI with increased overheads results, in most small company instances, in the model not being considered as a solution or worthy of investigation. Supporters of CMMI claim that use of the models can lead to greater predictability and repeatability (Boehm and Turner, 2004). Paradoxically this works against CMMI from the perception of software start-ups. Many small software companies would argue that each project and situation is new to them and that *Creativity* and *Flexibility* are higher on their list of desired capabilities than predictability and repeatability. The companies in this study believe agile methodologies support *Creativity* and *Flexibility*. Accordingly, it is easy to see how XP has achieved much higher usage in indigenous Irish software companies than plan-driven approaches such as CMMI.

6.4. Limitation of the study

Grounded theory as a qualitative research method, using semi-structured interviews, centre on respondents' opinions. 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. However, 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 study, 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. It is these actions and interactions, arising from the participants opinions, beliefs and perceptions, which are essential to a grounded theory study.

7. Conclusion

7.1. Summary and conclusions

This research set out to explore two specific research questions and a number of related questions: *Why are software companies not using 'best practice' SPI models*? and *What software processes are software companies using*? 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. 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 and the types of projects in which they are engaged.

The research question *why 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 have been presented in this paper. For many of the interviewees SPI creates an additional burden to their development efforts resulting in increased *Documentation* and *Bureaucracy*. In the case of the smaller companies SPI was resisted as they believed it would negatively impact their *Creativity* and *Flexibility*.

7.2. Future research

One of the contributions of this work is a grounded theory explaining how software process is initially established in a software start-up. As 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 study concentrated in one geographical location. A study which examines practices in other countries 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. As much software is developed outside the software product company domain, a study including a wider range of software development from bespoke software solutions to the in-house software departments of non-software companies could be counter-balanced against this work. 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.

References

Ahern, D.M., Clouse, A., Turner, R., 2004. CMMI Distilled: A Practical Introduction to Integrated Process Improvement, second ed. Addison-Wesley, Boston.

Aveling, B., 2004. XP lite considered harmful? In: Proceedings of the Fifth International Conference of Extreme Programming and Agile Processes in Software Engineering, LNCS 3092, Springer, pp. 94–103.

Bach, J., 1994. The immaturity of CMM. In: American Programmer, vol. 7(9), pp. 13–18.

Baker, R., 1996. The corporate politics of CMM ratings. In: Communications of the ACM, vol. 39(9), pp. 105–106.

Barnes, F., 2000. Good business sense is the key to confronting ISO 9000, Review of Business.

Baskerville, R., Pries-Heje, J., 1999. Grounded action research: a method for understanding IT. In: Accounting, Management and Information Technologies (9), pp. 1–23.

Batista, J., Dias de Figueiredo, A., 2000. SPI in a very small team: a case with CMM. Software Process Improvement and Practice 5, 243–250.

Beck, K., 2000. Extreme Programming Explained: Embrace Change. Addison-esley.

Boehm, B., Turner, R., 2004. Balancing Agility and Discipline. Addison-esley.

Bollinger, T.B., McGowan, C., 1991. A critical look at software capability evaluations. In: IEEE Software, July, pp. 25–41.

Bowers, A., Sangwan, R., Neill, C., 2007. Adoption of XP practices in the industry – a survey. Software Process: Improvement and Practice 12 (3), 283–294.

Brodman, J.G., 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., Huczynski, A.A., 1985. Organisational Behaviour: An Introductory Text. Prentice-Hall International (UK) Ltd., London.

Clifford, S., 2005. So many standards to follow, so little payoff, Inc. Magazine, May.

Coallier, F., 1994. How ISO 9001 fits into the software world. In: IEEE Software, January, pp. 98–100.

Cohn, M., Ford D., 2003. Introducing an agile process to an organisation. In: IEEE Computer, June, pp. 74–78.

Coleman, G., McAnallen, M., 2006. Managing the challenges of legacy systems using extreme programming. In: Software Process Improvement and Practice, No. 11, pp. 269–275.

Coleman, G., O'Connor, R., 2007. Using grounded theory to understand software process improvement: a case study of Irish software product companies. Information and Software Technology 49 (6), 654–667.

Coleman Dangle, K., Larsen, P., Shaw, M., Zelkowitz, M.V., 2005. Software process improvement in small organisations: a case study. In: IEEE Software, November/December, pp. 68–75.

Crone, M., 2002. A Profile of the Irish Software Industry. Northern Ireland Economic Research Centre.

Demirors, E., Demirors, O., Dikenelli, O., 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.

Dorling, A., 1993. SPICE: Software Process Improvement and Capability dEtermination. In: Information and Software Technology, vol. 36(6/7), pp. 404-406.

El Emam, K., Briand, L., 1997. Costs and benefits of software process improvement, Technical Report ISERN 97-12, Fraunhofer Institute for Experimental Software Engineering, Germany.

Enterprise Ireland, 2005. Software Industry Statistics 1991–2004, available at http://www.nsd.ie/htm/ssii/stat.htm.

Fayad, M., Laitinen, M., 1997. Process assessment considered wasteful. In: Communications of the ACM, vol. 40(11), pp. 125–128.

Fitzgibbon, C., 1996. ISO 9001 registration: lessons learned by Canadian Software Companies. In: Proceedings of the Fifth International Conference on Management of Technology, Florida, pp. 193–201.

Glaser, B., Strauss, A., 1967. The Discovery of Grounded Theory: Strategies for Qualitative Research, Aldine.

Goldenson, D., 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., 2002. Grounded Theory: A Practical Guide for Management. In: Business and Market Researchers. Sage Publications.

Grenning, J., 2001. Launching extreme programming at a process-intensive company. In: IEEE Software, November/December, pp. 27–33.

Heinz, L., 2004. CMMI for small businesses: initial results of the pilot study, Software Engineering Institute, Pittsburgh, PA, http://www.sei.cmu.edu/news-at-sei/features/2004/3/pdf/feature-1-2004-3.pdf.

Hansen, B., 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.

Herbsleb, J., Zubrow, D., Goldenson, D., Hayes, W., Paulk, M., 1997. Software quality and the capability maturity model. In: Communications of the ACM, vol. 40(6), pp. 30–40.

Horvat, R.V., Rozman, I., Gyorkos, J., 2000. Managing the complexity of SPI in small companies. In: Software Process Improvement and Practice, vol. 5(1), pp. 45–54.

HotOrigin, 2004. Ireland's Software Cluster: Preparing for Consolidation. HotOrigin Ltd., Dublin, Ireland.

Humphrey, W.S., Snyder, T., Willis, R., 1991. Software process improvement at Hughes Aircraft. In: IEEE Software, July, pp. 11–23.

ISO, 1992. Quality management and quality assurance standards, Part 3: Guidelines for the application of ISO 9001 to the development, supply and maintenance of software. International Organisation for Standardisation, Geneva, Switzerland.

Lycett, M., Macredie, R.D., Patel, C., Paul, R.J., 2003. Migrating agile methods to standardised development practices. In: IEEE Computer, June, pp. 79–85.

Martin, P.Y., Turner, B.A., 1986. Grounded theory and organizational research. The Journal of Applied Behavioural Science 22 (2), 141–157.

McGregor, D., 1985. The Human Side of Enterprise: 25th Anniversary Printing. McGraw-Hill/Irwin.

Miller, M., Pulgar-Vidal, F., Ferrin, D., 2002. Achieving higher levels of CMMI maturity using simulation. In: Proceedings of the Winter Simulation Conference, San Diego, December, pp. 1473–1478.

Murru, O., Deias, R., Mugheddu, G., 2003. Assessing XP at a European internet company. In: IEEE Software, May/June, pp. 37–43.

New Oxford Thesaurus of English, 2001. Oxford University Press.

Orlikowski, W., 1993. CASE tools as organizational change: investigating incremental and radical changes in systems development. In: Management Information Systems Quarterly, vol. 17(3), pp. 309–340.

Oskarsson, O., Glass, R.L., 1996. An ISO 9000 Approach to Building Quality Software. Prentice Hall, NJ.

Pitterman, B., 2000. Telcordia technologies: journey to high maturity. In: IEEE Software, July/August, pp. 89–96.

Rasmusson, J., 2003. Introducing XP into Greenfield Projects. In: IEEE Software, May/June, pp. 21–28.

Schuler, K., 1995. Preparing for ISO 9000 registration: the role of the technical communicator. In: Proceedings of the 13th International Conference on Systems Documentation, Savannah, GA, pp. 148–154.

Seaman, C., 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.

SEI, 2006. Process maturity profile – CMMI SCAMPI Class A appraisal results mid-year update 2006, Software Engineering Institute viewed May 2007, <www.sei.cmu.edu>.

Sheldon, L., 1998. Grounded theory: issues for research in nursing. Nursing Standard 12 (52), 47–50.

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.

Staples, M., Niazi, M., 2006. Systematic review of organizational motivations for adopting CMM-based SPI, Technical Report PA005957, National ICT Australia.

Staples, M., Niazi, M., Jeffery, R., Abrahams, A., Byatt, P., Murphy, R., 2007. An exploratory study of why organizations do not adopt CMMI. The Journal of Systems and Software 80 (6), 883–895.

Strauss, A., Corbin, J.M., 1998. Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory, second ed. Sage Publications.

Wilkie, F., McFall, D., McCaffery, F., 2005. An evaluation of CMMI process areas for small- to medium-sized software development organisations. Software Process Improvement and Practice 10 (2), 189–201.