IGNORING 'BEST PRACTICE': WHY IRISH SOFTWARE SMES ARE REJECTING CMMI AND ISO 9000

Rory V. O'Connor

School of Computing
Dublin City University, Ireland
roconnor@computing.dcu.ie

Gerry Coleman

Department of Computing
Dundalk Institute of Technology, Ireland
gerry.coleman@dkit.ie

ABSTRACT

Software Process Improvement (SPI) 'best practice' models such as ISO 9000 and the Capability Maturity Model Integrated (CMMI) have been developed to assist software development organisations by harnessing their experience and providing them with support so that they can produce software products on time, within budget and to a high level of quality. However there is increasing evidence that these models are not being adopted by Small to Medium sized Enterprises (SME) and primarily remain the remit of large organisations. This paper presents the results of a Grounded Theory study into why Irish SME software product companies are not using these SPI models. The key inhibiting factor found was the issue of cost. We discuss the findings in relation to cost of process and the factors affecting it, including bureaucracy, documentation, communication, tacit knowledge and organisational creativity and flexibility, and the associated impact on the adoption of SPI best practice models.

Keywords Software Process, Software Process Improvement, Best Practice Models, CMMI, ISO 9000, XP, Small and Medium Enterprises, SME

Earlier version A shorter, earlier version of this paper appeared in the 18th Australasian Conference on Information Systems, Toowoomba, Australia, 5-7 December 2007, pp. 780-789.

INTRODUCTION

For many SME software companies, implementing controls and structures to properly manage their software development activity is a major challenge. Administering software development in this

way is usually achieved through the introduction of a software process. A software process essentially describes the way an organisation develops its software products and supporting services, such as documentation. Processes define what steps the development organisations should take at each stage of production, and also provide assistance in making estimates, developing plans and measuring quality. 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.

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. There is a widely held belief that a better software process results in a better software product, with authors such as Humphrey (1995) claiming that to improve your product, you must improve your process quality. In support of this Zahran (1998) considers *"... 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"*. These ideas have led to a focus on SPI, which can be traced back to the 1970s and 1980s and the work of Crosby (1979) and Juran (1988), who demonstrated that, in the area of production management, product quality could be improved through a better production process.

SPI models have been developed to assist companies in this regard and 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. Some large software organisations have used SPI 'best practice' models, such as the Capability Maturity Model Integrated (CMMI) (Ahern et al., 2004) and the ISO 9000 series (ISO, 1992). More recently, agile methodologies such as Extreme Programming (XP) (Beck, 2000) have been used in SPI programmes as a way of improving delivery time and increasing customer satisfaction, and these agile approaches 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 (Software Engineering Institute) CMMI data for the years 2002 to 2007 (SEI, 2007), which shows that worldwide during that period only 2,140 organisations were appraised. It is clear that this 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 adopting 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 (SME) - fewer than 10 CMMI appraisals were conducted during the 2002 – 2006 period, from a population of more than 900 software companies. Therefore it is 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 CMMI.

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 (Coleman & O'Connor, 2007). 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 (Coleman & O'Connor, 2007). 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.

There is evidence (Laporte et al., 2008) that the majority of small software organisations are not adopting existing standards because they perceive the standards as being orientated towards large organisations. Studies have shown that small firms' negative perceptions of process model standards are primarily driven by negative views of cost, documentation and bureaucracy. In addition, it has been reported that SMEs find it difficult to relate standards to their business needs and to justify the application of the international standards in their operations. Most SMEs cannot afford the resources for, or see a net benefit in, establishing software processes as defined by current standards (e.g. ISO/IEC 12207) and maturity models (e.g CMMI).

However to date, the majority of evidence for non-adoption (such as that presented above) originates from high level surveys (Crowder, 2007; Laporte et al., 2008) of software organisations and not from in-depth exploratory studies into SME understanding and experiences of software processes. Accordingly, the motivation for our research originates in the premise that software companies are not following 'best practice' process improvement models. On these bases, through the utilisation of a more in-depth and qualitative research approach, we set out to answer the question: *Why are SME software companies not using 'best practice' SPI models?*

THE STUDY CONTEXT

The setting for this study is that of indigenous Irish software product companies. The software industry in Ireland is a key component of the national economy and its growth 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. 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 (the most recent statistics available) 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. 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 into Europe, Ireland wins 41% of all software projects.

Within the European Union (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 and they have played a key role in the Irish software economy. 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) shows that the indigenous software industry consisted primarily of micro firms providing consultancy and services to businesses adopting IT systems. O'Riain shows that, although 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 when asserting that the average size of indigenous Irish software firms is about 16 employees.

However, 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. In 2004, they estimated a total of 417 indigenous software product companies in Ireland (HotOrigin, 2004) and categorised indigenous software firms across three stages of company development: 1) 'Start-up' (1-25 employees); 2) 'Build' (26-75 employees); and 3) 'Expansion' (75+ employees). These most recent figures available (HotOrigin, 2004) 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. 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 might experience significant change at even modest levels of employment and that different factors might affect the different size brackets.

As this study is examining how software processes are used in actual practice, the categories listed above are of particular importance because 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 (Laporte et al., 2008). Accordingly, the relative categories are important in studying process issues because a small problem can have significant repercussions in a smaller organisation, whereas the effect in a larger company may not have any impact.

Since almost all of the indigenous software companies in Ireland 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 (Crowder, 2007; Laporte et al., 2008). 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.

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 because their process would likely have been initially developed and used within the parent company prior to being devolved to the Irish subsidiary.

RESEARCH METHOD

Research Method

The investigation of software process in practice relies heavily on eliciting and understanding the experience of those who use the software processes in situ and on the interpretation of these experiences and the reality of the situation under study. The study therefore naturally lends itself to the application of qualitative research methods because it is orientated towards how individuals and groups view and understand the world and construct meaning out of their experiences.

The use of qualitative research in software development studies has been more widely embraced within Information Systems (IS) than within Software Engineering (SE). The focus on technological issues in SE studies (such as Buchman, 1996; Daskalantonakis, 1994; Dion, 1993; Herbsleb et al., 1997) and the associated extensive use of quantitative methods has been criticised by Bertelsen (1997), who argues for the use of qualitative research in SE. He contends that as SE is a socio-culturally, not a technically, constituted phenomenon and any research conducted cannot be based exclusively on natural science approaches but must include a way to understand psychological, social, and cultural phenomena. We agree 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 learn from 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.

The research method chosen for the study was Grounded Theory. Grounded Theory was first established by Glaser and Strauss (1967). The theoretical foundations of grounded theory stem from Symbolic Interactionism, which sees humans as key participants and 'shapers' of the world they inhabit. Grounded theory was created from the 'constant comparative' method, developed by Glaser and Strauss, which alternated theory building with the 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" (Strauss & Corbin 1998, p 22). 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 and is a product of the continuous interplay between data collection and analysis of that data (Goulding, 2002). According to Strauss and Corbin, the theory which is derived from the data is more likely to resemble what is actually going on when compared to a theory assembled from putting together a series of concepts based on experience or through speculation (Strauss & Corbin, 1998).

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

Since the initial launch of grounded theory, the Glaser and Strauss alliance gradually separated until each was developing a different version of the methodology. As a result of these divergences, it is incumbent on every researcher using grounded theory to indicate which implementation of the methodology they are using. Though acknowledging and recognising the spirit of Glaser's original version, this study employed the Strauss and Corbin (1998) approach. In particular, 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' (that is the ability to understand the data's important elements and how they contribute to theory). The experience factor is also highlighted in Fitzgerald (1998), who describes the concept of the 'cultural insider' as someone who has prior expertise or practitioner knowledge of the domain. Having operated as software process consultants and professional software engineers for a number of years, our '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. Our professional experience also provided a familiarity with the literature surrounding the study area, thus supporting theoretical sensitivity. The motivation for our research, as described in the introduction to this paper, also encourages the use of the Strauss and Corbin version of grounded theory because their methodology favours 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.

Because of grounded theory's interpretivist emphasis and its ability to explain socio-cultural phenomena it 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 later into the IS field, 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's (1993) study which showed how it could be used to explain the impact on two organisations which implemented CASE 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.

A number of researchers have used grounded theory to look at a diverse range of socio-cultural activities in IS. Baskerville and Pries-Heje (1999) 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 research method to examine the use of 'systems thinking' practices (Goede & De Villiers, 2003), software inspections (Carver & Basili, 2003), process modelling (Carvalho et al., 2005), requirements documentation (Power, 2002) and virtual team development. Qureshi et al. (2005) and 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 method well suited for use in the IS sector. The strengths of the method include facilitating the gathering and analysis of those human experiences and the associated interrelationships with other human actors, coupled with situational and contextual factors. For a more detailed 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).

Research Study Structure

This study was divided into three major phases and involved the participation of 21 indigenous Irish software product companies. The grounded theory was produced using three phases: a preliminary phase (P) to help frame the study and test the interview guide and approach; a more detailed phase (Phase 1) which developed the initial concepts and categories, and enabled evaluation of the theoretical sampling process; and the final phase (Phase 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. Participants were chosen from our personal contacts, individuals we knew second-hand, and responses to 'cold' e-mailing.

Company	**Market Sector**	**Number of Employees in Software development**	**Study Phase**
1	Telecommunications	3	P
2	Corporate secretarial	20	P
3	Telecommunications	3	P
4	Telecommunications	30	1
5	Telecommunications	6	1
6	Compliance Mgt.	40	1
7	Enterprise	100	1 & 2
8	E-learning	70	1
9	Information quality	9	1
10	Telecommunications	12	1
11	Telecommunications	110	1 & 2
12	Financial services	23	1
13	Financial services	90	1

14	Interactive TV	40	1 & 2
15	Public sector	90	2
16	Medical devices	9	2
17	Telecommunications	35	2
18	Public sector	3	2
19	HR solutions	15	2
20	Games infrastructure	20	2
21	Personalisation	40	2

Table 1: Participant Company Profile

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, 2 and 3 was undertaken. To support the semi-structured interviewing process an interview guide (based on our experience as 'cultural insiders' and our prior familiarity with the literature) was created for use with the first two interviews. There were 53 questions divided into 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 two interviews highlighted several drawbacks. The first centred around the length of the interviews because capturing all of the information took an inordinate amount of time and stretched the goodwill of the interviewees. This had the secondary effect of leaving some potential fruitful lines of enquiry unexplored. We therefore redeveloped the interview guide so that it contained 34 questions across three categories: Company Background, People Issues and Software Development Strategy. This interview guide was then used during the remaining third and fourth interviews comprising the preliminary phase. The interviews and the line of questioning during the third and fourth interviews concentrated more on the memos (the ongoing process of making notes and ideas and questions that occur to the analyst during the process of data collection and analysis) and codes (which are allocated to passages in the text) arising from the coding and analysis of the first two interviews, rather than on the formal set of questions.

Study Phase 1

The next phase of the study involved interviews with an additional 11 companies using the interview guide developed during the preliminary phase. 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 the initial open coding, the interviews were then re-analysed and coded axially across the higher-level categories which had emerged from earlier interviews. Any memos or propositions which 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.

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. This approach had the potential to explain how the concepts and categories emerging from the study were linked. Study Phase 2 was used to ensure the emergent theory was properly grounded.

Study Phase 2

Phase 2 involved the participation of seven new companies and comprised ten further interviews. Three of these interviews involved re-interviewing earlier participants. This technique is supported in grounded theory studies by Goulding (2002), because it allows for a comprehensive checking and verification process of the data already analysed. The seven new companies (Companies 15-21) were specifically selected because their business sectors helped extend the scope of the study and ensured that theoretical categories were not being established on an excessively narrow sectoral basis. During the Phase 2 fieldwork the semi-structured interview questions were primarily derived from the Phase 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 & Corbin, 1998). Full category 'saturation' was reached on the conclusion of interview 25 because, in line with Goulding's (2002) assertion, similar incidences within the data were now occurring repeatedly and continuing would have been unlikely to generate any further contrary data.

RESULTS

Grounded theory provides mechanisms (Strauss & Corbin, 1998) to: identify the categories into which the concepts discovered in the data can be placed; to explain the relationships between these categories; to provide the overall theoretical picture; and to identify a key category or theme which can be used as the fulcrum of the study results.

In this instance, the analysis showed that there was one central category *Cost of Process*[1] to support and link the two theoretical themes *Process Formation* and *Process Evolution*. The final list of themes, the core category and the main categories identified by the study are shown in Table 2. Each category and code can be linked to quotations within the interviews and these are used to provide support and rich explanation of the results. The 'saturated' categories and the various relationships were then combined to form the theoretical framework. The network feature, contained within Atlas TI (which is a grounded theory software support system), allows the researchers to present their findings in graphical or pictorial fashion which, for this study, helped us to create a clear image of how the research themes, categories and subcategories were interrelated.

Theme	**Category**
Process Formation	*Background of Software Development Manager*
	Background of Founder
	Management Style
	Process Tailoring
	Market Requirements
Theme	**Category**
Process Evolution	*Process Erosion*
	Minimum Process

[1] From hereon, the themes, categories and core category produced by the study are denoted in italics

	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

Table 2: Themes, Core Category and Main Categories

The study found that all of the companies were tailoring standard software processes to their own particular operating context such as the size of the company, the target market, and project and system type. In addition there was evidence from the data suggesting that managers instigate SPI as a reaction to business occurrences for which the current process did not adequately cater.

The research question addressed in the study "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. Participant companies perceive *Documentation* as the greatest process-related cost-inducing element. There was also a clear link between the amount of *Documentation* carried out and the size and growth stage of the company; the smaller the company the greater the hostility towards *Documentation*. However, even in the larger organisations, *Documentation* was regarded as a 'necessary evil'. Many companies substituted verbal *Communication* for *Documentation* and co-located their development teams in an effort to reduce process cost. A benefit of co-location was an increase in the sharing of *Tacit Knowledge*.

From the commercial SPI perspective, the study was dominated by two particular models (CMM(I) and ISO 9000) and the development methodology XP (Extreme Programming). It is interesting to note that the respondents did not differentiate between processes and methodologies and categorised XP as a process. As a result, XP (albeit tailored to various degrees) was by far the most popular commercial 'process' model used by organisations across all size sectors. XP was perceived to have the least associated *Cost of Process* and its low level of *Documentation* was deemed to be attractive. Where managers were familiar with CMM(I) they 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 were ISO 9000 certified, all of those sought certification for business reasons and not process/quality reasons. Overall, respondents felt that the resources required to implement the commercial models far exceeded the benefits which may accrue.

In the course of the 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 diagram in Figure 1.

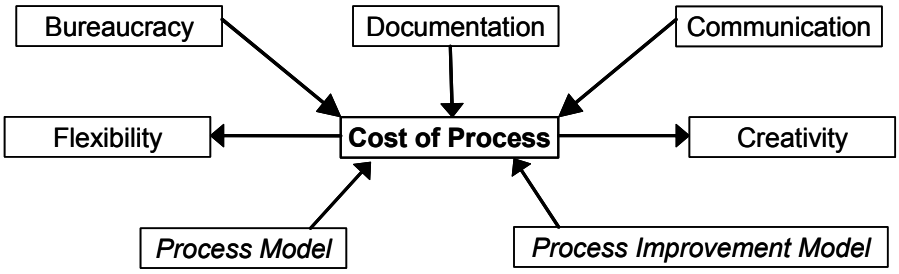

Figure 1: Cost of Process Network Diagram

The key *Cost of Process* factors *Bureaucracy*, *Documentation*, *Creativity* and *Flexibility* are presented in the following subsections, along with a discussion of the associated impact on both the *Process Model* and *Process Improvement Model* used. In keeping with the fundamental tenets of grounded theory, extracts of the interview transcripts will also be presented in support of the findings.

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 - that is, 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 because it is a synonym for 'extravagance', 'indulgence' or 'something inessential' (New Oxford Thesaurus of English, 2001).

The following comment from 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*". 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'. The prevailing opinion was 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, as exemplified by this

interviewee: "*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 have a number of ways of getting round that*". The overkill described shows that managers are complicit in bypassing their own process and believe that by eliminating some process steps they also eliminate some of the costs.

The interview extracts above 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 resources. One of the process activities which managers believe can often delay, or hinder, product development is *Documentation*.

Documentation

Forward and Lethbridge (2002) define software documentation as "*Any artefact 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* results in a cost due to the actual time taken to record the chosen information, and also due to the 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 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*".

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, as described by this comment: "*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 later in this paper.

Communications

Because *Documentation* was seen by the managers as such a significant process cost, they believed that if they could reduce *Documentation* they could reduce the cost of their software process. Taking Forward and Lethbridge's (2002) definition of software *Documentation* (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 was 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 attempted to capitalise on exploiting *Tacit Knowledge* and verbal *Communication*, and this was brought out in the practices they adopted. 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*".

Informal *Communication* and knowledge sharing benefits from this form of team co-location. Co-location 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 does not 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 was 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: "*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, such as in this case: "*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* which 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, enable more informal *Communication*, and obviate the need for greater *Documentation*. The experience of one company shows how they aimed 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 were using XP and who worried about the emphasis on informal *Communication* at the expense of *Documentation*. 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*, as discussed in the next section.

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 should therefore be considered carefully. Some of the start-up companies see process as primarily of benefit to established companies as Company 3 describes: "*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*".

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 this 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*".

Others also reduced the amount of process they used because of the 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 process impacting negatively on *Creativity* and *Flexibility* are not the preserve of smaller software companies, as one 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*, the above discussion provides evidence of why business events, which cannot be resolved using the existing process, appear to be the main drivers of process change / 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.

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 above, all of the companies interviewed were using a tailored Software Development Process, which in most cases was based on a standard industry model. The following subsections will discuss the *Cost of Process* impacts on the two most popular process models to emerge from the interviews: the RUP (Rational Unified Process) and XP.

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 had since dispensed with it completely, blaming *Documentation,* as exemplified by this company: "*RUP has 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 started 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*".

Some companies, who had at one point considered using the RUP, subsequently rejected it because of the complexity of the associated support tool called 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 in terms of resources and of purchasing associated tools, many companies moved away from processes based on RUP to ones based on XP.

Extreme Programming (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 deployed by the companies were seen to be very cost effective. One manager argued that XP provided 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, as explained by this interviewee comment: "*I think a lot of the attributes of XP, around test-first design and iterations, and rapid feedback to developers are hugely valuable*".

In cases where XP replaced an existing process, the predecessor had a much greater *Documentation* requirement. Therefore any successor with a reduced *Documentation* overhead had a real chance of succeeding. This clearly proved to be the case when higher levels of employee buy-in to XP were reported: "*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*".

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. When introducing XP, companies believed they got good value for money from 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 is 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*".

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, by far the most common complaint about XP related to the insufficiency of *Documentation* produced by the method: "*We tried XP and we found that the documentation trail was extremely weak or even non-existent*". Fears were also expressed that the absence of *Documentation* would make it more difficult for new team members to understand the system, thus 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?*"

There is an interesting mix of companies who were using XP and it ranged 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) had reached this stage with the exception of Company 16. However, Company 16 was about to enter a regulated market with the associated emphasis on *Documentation* and *Traceability*, which required 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.

Process Improvement Models

A key part of this research was 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 CMMI (and its predecessor the CMM) which are specifically geared for software, and ISO 9000 whose origins lie in manufacturing, resonated with the companies interviewed. Of the 21 study companies, three were ISO 9000 certified and one was embarking on the ISO 9000 certification process. None of the companies were using the CMMI.

CMMI

As the most widely publicised SPI models, it was important to this study to determine the attitudes of indigenous Irish software product companies towards CMMI and ISO 9000. 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. However, as with ISO 9000, it was the managers' previous experience of using CMM(I) which resulted in the greatest hostility to its introduction. 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: "*It [CMM] 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 CMM because of the belief it would hinder the company's 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 had investigated CMMI and chosen not to introduce it, represents all companies who reached the same conclusion: "*We felt CMMI was overkill for the level of development that we were doing and so it wasn't really pursued*".

The belief that CMMI contains excessive levels of detail and requires 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, CMMI did not generate increased support even though it is software specific. The criticisms levelled against it include that it is 'excessive', 'over the top', 'heavyweight', 'onerous', 'bureaucratic' and 'too detailed'. Managers were then asked under what circumstances they might use it, with the manager from Company 9 comment being representative: "*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*".

ISO 9000

Where a manager has previously used a process or process model which 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 did not work they rejected its use within their new companies. This was most significantly felt in the case of ISO 9000. 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 9000. And it just didn't work. They made crap products but by God they had ISO in*".

ISO 9000 was seen to be closely associated with *Bureaucracy* because the participants variously described ISO 9000 as 'way over the top', carrying 'a lot of baggage', being 'heavyweight', and having significant 'overhead'. This was mainly because managers believed ISO 9000 has an overemphasis on *Documentation*, which 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 ISO certification*". Small software companies and start-ups were especially wary of ISO and the amount of *Documentation* required by the standard. Company 16, who was 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*".

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, because companies were reluctant to engage in SPI due to its association with increased *Documentation*. It is not surprising that they would be hostile to ISO 9000 if they perceive it as having a similar *Documentation* requirement.

The three companies in the study which had ISO 9000 certification pursued it primarily for business reasons. Of the remaining 18 companies, only one company was actively considering it, and this was because they were entering a regulated sector. For Company 1, the introduction of ISO 9000 was undertaken to gain a contract from a telecommunications multi-national, whilst Company 6

(which was in the pharmaceutical sector) was facing market barriers (FDA approval) and needed ISO 9000 certification to remove these barriers.

Company 14 was the only other company in the study to have ISO 9000 certification and cited 'market advantage' as 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*".

The fact that ISO 9000 certification was being sought for business reasons rather than process improvement reasons lends credence to our view that process change is reactive (to business events) and not proactive (for quality / process improvement). From the interviews, and detailed analysis of the managers' views, there was no evidence that in these companies certification was pursued in order to achieve improved quality or development capability. Company 12's Chief Technology Officer, who best represents the views of a number of the study managers, describes it in the following way: "*If somebody said tomorrow, you won't sell into the financial services sector unless you are CMMI 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 interviews, there were quotes from 15 of the 21 companies which were critical of ISO 9000 from a *Documentation*, *Bureaucracy* and administrative perspective. This leads to a situation where more than three quarters of the companies in this study firmly opposed the adoption of ISO 9000 in their software development.

DISCUSSION

The research question addressed in the study "Why are software companies not using 'best practice' SPI models?" produced the study's core concept *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 were presented above. 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 European Union defined SME category, it is therefore perhaps not surprising that they would have greater hostility to SPI models which required them to divert resources from what they perceived as more deserving activities. For many of the interviewees, SPI creates an additional burden to their development efforts which results in increased *Documentation* and *Bureaucracy*. To reduce their process overhead, companies substituted verbal *Communication* for *Documentation*. Development teams were co-located to ensure ease of verbal exchange and to reduce the need for *Documentation*. 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 were 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, with many managers in these companies having used it previously. However, three companies in the study were 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 which may accrue. In some cases, however, managers saw no benefit at all to the commercial models and believed they would hamper business prospects.

Research Contribution

This research provides a grounded understanding of the practice of software process and software process improvement 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 to enable 'practice to inform theory' and, more importantly, to provide a challenge to 'accepted wisdom'. Secondly, it has deployed a qualitative method 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 methods 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 multi-national 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.

Implications for Small Business

The findings of this research contains useful lessons for software entrepreneurs who need to make decisions about process and process change within their organisations as they grow. 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 which follows this route needs to be aware of the advantages and disadvantages associated with this approach. Companies which 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 which 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.

Implications for Researchers

Small software companies, in the first instance, focus exclusively on survival. This, in part, explains the success of agile methodologies whose 'light', non-bureaucratic techniques support companies in survival mode attempting to establish good, fundamental software development practices. Though 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 CMMI or ISO 9000, are perceived as supporting this objective. This clearly poses questions for 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, small companies are significantly more interested in methods than process and that methods such as XP are far more attractive to practitioners in these situations rather than processes such as CMMI or ISO 9000. Clearly, practitioners can be educated and trained to understand the differences between methods and process and to appreciate the necessity to go beyond mere methodological adoption in pursuit of SPI. However, if SPI models are to be more widely deployed by early stage companies, existing 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 CMMI can produce positive results in small settings has been explored by a number of researchers, however the argument put forward in our paper 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, it will be difficult for 'one-size fits all' models such as CMMI are always going to penetrate small software organisations. Such contextual realities must be considered by SPI researchers.

The wider implications of this study are that, although significant research time has been spent on endeavouring to prove that 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 CMMI by small software companies is something which could be usefully addressed in future studies.

The findings from this research indicate that human and social factors have a major role to play in SPI. However, this is an angle which has largely been ignored in SPI studies within Software Engineering. This is not true of the Information Systems discipline. Studies within IS do attempt to take the human and social dimension into account when examining methodological and process issues. However there is evidence of some recognition of this fact in the Software Engineering community in recent times.

Limitations

Grounded theory has some definite limitations (Norman, 2007). As with all qualitative methods 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 directly from participant interviews. However, opinions are the respondents' views or perceptions regarding what is taking place, which of course may be at odds with reality. 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 & 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 which 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 (Strauss & 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 to ensure that reports were gathered from different sized companies in different sectors, the interview pool consisted solely of a very senior person in each organisation. A study purely from the engineers' perspective might have generated a different outcome, but it would lack the crucial 'big picture' view which senior managers can provide. Similarly, it is generally the senior managers who have decision-making responsibility for the process model adopted. A study focusing exclusively on engineers may be deficient in depth and breadth of organisational approaches.

CONCLUSION

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 CMMI. Whether or not this is true is a moot point. The fact that managers associate CMMI with increased overhead means that most small companies do not see the model as being a viable solution or even worthy of investigation.

Supporters of CMMI claim that use of the models can lead to greater predictability and repeatability. Paradoxically, this works against CMMI from the perception of small, early-stage software firms. Many small software companies, some of whom may only have 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 see agile methodologies as supporting *Creativity* and *Flexibility*, which is why XP has achieved substantially higher usage in indigenous Irish software companies than CMMI.

Given the volume of material in the literature about the ISO/IEC 15504 ('SPICE') software process assessment standard, it is perhaps surprising that none of the study respondents mentioned it. 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, 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 CMMI. However, the complete absence of knowledge about the standard should give cause for concern amongst its founders and advocates.

This research was concerned with how software process was practiced in indigenous Irish software product companies. There are many opportunities for future research arising from this paper. A study which concentrates on the practices used by indigenous software product companies in other countries in Europe and beyond would provide further validity for this research and determine if the findings can be replicated elsewhere. In addition, much software is developed outside the software product company domain because 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 improved in this non-software product company environment could be compared with this work. Another research focus could involve capturing the opinions and experiences of the software engineers themselves, not just the owner-managers of software firms. This would add to the data.

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 practitioner communities to identify the factors which give rise to the requirement for increased *Documentation* and to determine 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* lay, and would also include *Tacit Knowledge*, co-location and office layout factors in its investigation scope.

REFERENCES

Ahern, D.M., Clouse, A. & Turner, R. (2004) CMMI Distilled: A Practical Introduction to Integrated Process Improvement, 2nd edition, Addison Wesley, Boston, Massachusetts.

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

Beck, K. (2000) Extreme Programming Explained: Embrace Change, Addison Wesley, Boston, Massachusetts.

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

Buchman, C. (1996) "Software Process Improvement at AlliedSignal Aerospace", 29th Annual Hawaiian International Conference on System Sciences, Vol 1, 3-6 January, Hawaii, pp 673-680.

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

Carver, J. & Basili, V. (2003) "Identifying Implicit Process Variables to Support Future Empirical Work", Journal of the Brazilian Computer Society, October-December, pp 1-10.

Coleman, G. & O'Connor, R. (2007) "Using grounded theory to understand software process improvement: A study of Irish software product companies", Journal of Information and Software Technology, Vol 49 No 6, pp 654-667.

Crowder, K. (2007) "Systems Engineering for Very Small Enterprises, Insight", INCOSE, Vol 10 No 2, pp 28.

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

Crosby, P.B. (1979) Quality is Free: The Art of Making Quality Certain, McGraw-Hill, New York.

Daskalantonakis, M.K. (1994) "Achieving Higher SEI Levels", IEEE Software, July, pp 17-24.

Dion, R. (1993) "Process Improvement and the Corporate Balance Sheet", IEEE Software, July, pp 28-35.

European Commission (2005) "The New SME Definition: User Guide and Model Declaration". http://europa.eu.int/comm/enterprise/enterprise_policy/sme_definition/sme_user_guide.pdf accessed March 2008.

Fitzgerald, B. (1998) "An Empirical Investigation into the Adoption of Systems Development Methodologies", Information & Management, Vol 34 No 6, pp 317-328.

Forward, A. & Lethbridge, T.C. (2002) "The Relevance of Software Documentation, Tools and Technologies", 2002 ACM Symposium on Document Engineering, 8-9 November, McLean, Virginia, pp 26-33.

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

Goede, R. & De Villiers, C. (2003) "The Applicability of Grounded Theory as Research Methodology in Studies on the use of Methodologies in IS Practices", 2003 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists on Enablement Through Technology, 17-19 September, Pretoria, South Africa, pp 208-217.

Goulding, C. (2002) Grounded Theory: A Practical Guide for Management, Business and Market Researchers, Sage Publications, London.

Hansen, B. & Kautz, K. (2005) "Grounded Theory Applied – Studying Information Systems Development Methodologies in Practice", 38th Annual Hawaiian International Conference on Systems Sciences, 3-6 January, Hawaii, pp1-10.

Herbsleb, J., Zubrow, D., Goldenson, D., Hayes, W. & Paulk, M. (1997) "Software Quality and the Capability Maturity Model", Communications of the ACM, Vol 40 No 6, pp 30-40.

HotOrigin (2004) Ireland's Software Cluster: Preparing for Consolidation, Unpublished report, HotOrigin Ltd., Dublin, Ireland.

Humphrey, W.S. (1995) A Discipline for Software Engineering, Addison Wesley, Boston, Massachusetts.

IDA Ireland (2003) IDA Annual Report 2003, Annual Report, IDA Ireland, Dublin, Ireland.

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, International Organisation for Standardisation, Geneva, Switzerland.

Juran, J.M. (1988) Juran on Planning for Quality, The Free Press, New York.

Laporte, C.Y., Alexandre, S. & O'Connor, R. (2008) "A Software Engineering Lifecycle Standard for Very Small Enterprises", in: R.V.O'Connor et al (eds.) Proceedings of EuroSPI, Vol. 16, Springer-Verlag, Germany, pp. 129-141.

Martin, P.Y. & Turner, B.A. (1986) "Grounded Theory and Organizational Research", The Journal of Applied Behavioural Science, Vol 22 No 2, pp 141-157.

Myers, M.D. (1997) "Qualitative Research in Information Systems", Management Information Systems Quarterly, Vol 21 No 2, pp 241-242.

New Oxford Thesaurus of English (2001) New Oxford Thesaurus of English, Oxford University Press, New York.

Norman, G. (2007) A Grounded Theory of Software Process Improvement Model Adoption, Unpublished PhD thesis, West Virginia University, Morgantown, USA.

O'Riain, S. (1997) "An Offshore Silicon Valley: The Emerging Irish Software Industry", Competition and Change: The Journal of Global Business and Political Economy, Vol 2 No 8/9, pp 175-212.

Orlikowski, W. (1993) "CASE Tools as Organizational Change: Investigating Incremental and Radical Changes in Systems Development", Management Information Systems Quarterly, Vol 17 No 3, pp 309-340.

Power, N. (2002) A Grounded Theory of Requirements Documentation in the Practice of Software Development, Unpublished PhD Thesis, Dublin City University, Ireland.

Qureshi, S., Liu, M. & Vogel, D. (2005) 'A Grounded Theory Analysis of e-Collaboration Effects for Distributed Project Management', 38th Annual Hawaiian International Conference on Systems Sciences, 3-6 January, Hawaii, pp 1-10.

Silva, L. & Backhouse, J. (1997) "Becoming Part of the Furniture: The Institutionalisation of Information Systems", in: A.S. Lee, J. Liebenau & J.I. DeGross (Eds.) Information Systems and Qualitative Research, Chapman & Hall, London, pp 389-414.

SEI (Software Engineering Institute) (2007) "Process Maturity Profile - CMMI SCAMPI Class A Appraisal Results Mid-Year Update 2007", http://www.sei.cmu.edu accessed February 2008.

Sheldon, L. (1998) "Grounded theory: issues for research in nursing", Nursing Standard, Vol 12 No 52, pp 47-50.

Staples, M. & Niazi, M. (2006) Systematic Review of Organizational Motivations for Adopting CMM-based SPI, Technical Report PA005957, National ICT Australia, Sydney, 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, Vol 80 No 6, pp 883-895.

Strauss, A. & Corbin, J.M. (1998) Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory, 2nd edition, Sage Publications, New York.

Zahran, S. (1998) Software Process Improvement, Addison Wesley, Boston, Massachusetts.