Coleman G. and O'Connor R., The Influence of Managerial Experience and Style on Software Development Projects, International Journal of Technology, Policy and Management, Vol. 8, No. 1, 2008
DOI - 10.1504/IJTPM.2008.016183

The Influence of Managerial Experience and Style on Software Development Process

Gerry Coleman
Department of Computing, Dundalk Institute of Technology, Dundalk, Co. Louth, Ireland
gerry.coleman@dkit.ie

Rory O'Connor
School of Computing, Dublin City University, Glasnevin, Dublin 9, Ireland
roconnor@computing.dcu.ie

Abstract

This paper presents the results of a study of how software process and software process improvement is applied in actual practice in the software industry using the indigenous Irish software product industry as a test-bed. This study focuses on the role and influence of both the Company Founder and the Software Development Manager on the initial formation of software development process practices. The results of this study contain useful lessons for software entrepreneurs who need to make decisions about process and process change within their organisations as they grow.

1. Introduction

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 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. A number of different models exist as instantiations of how software development can be undertaken. Some of the best known process models include: Waterfall Development, Evolutionary Development and Component-based Development (Sommerville, 2007).

There is a widely held belief that a better software process results in a better software product, with authors such as Humphrey (1995) stating that "*to improve your product, you must improve your process quality*". In support of this Zahran (1998) states that "*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 Software Process Improvement (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.

In the Information Technology domain, 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. 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 '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 Standardisation (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 to 2006 (SEI, 2006), in which time just 1,581 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 authorised for public release and there are appraisals that are not reported or authorized for public release, it is clear that the published figures represent 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 than 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 to 2006 from a population of more than 900 software companies (Enterprise Ireland, 2005). 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 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 the following research question: *What software processes are software companies using?* Preliminary investigation of this question raised the following linked sub-questions: *How are software processes initially established in a software company?* and *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: Section 2 describes the context in which this study was undertaken and presents the research methodology used. Section 3 examines the results, with particular emphasis on two key issues - the background of the software development management and the management style which prevails in the organisation. Section 4 considers the evaluation of the results and Section 5 contains a discussion on the findings, limitations of the study and presents a future research agenda.

2. The Study

2.1 Research Context

A context and scope for the study was set as follows: 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, we decided to limit the scope to software product companies. In addition, given the geographical location of the researchers, we chose to confine the study to indigenous Irish software product companies who naturally operate 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.

2.2 Research Methodology

The methodological approach taken in this study was that of Grounded Theory (Glaser and Strauss, 1967). The aim of grounded theory is to develop a theory from data rather than to gather data in order to test a theory or hypothesis. 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 the continuous interaction between data collection and data analysis (Goulding, 2002). Grounded theory uses qualitative methods to obtain data about a phenomenon and a theory emerges from that data, where that theory is grounded in the reality as represented in the data. According to Strauss and Corbin (1998), the theory that is derived from the data is more likely to resemble what is actually going on than if it were assembled from putting together a series of concepts based on experience or through speculation.

As the objective with the methodology is to uncover theory rather than have it pre-conceived, grounded theory incorporates a number of steps to ensure good theory development. The analytical process involves coding strategies: the process of breaking down interviews, observations and other forms of appropriate data into distinct units of meaning, which are labelled to generate concepts. These concepts are initially clustered into descriptive categories. The concepts are then re-evaluated for their interrelationships and, through a series of analytical steps, are gradually subsumed into higher-order categories or one underlying core category which suggests an emergent theory.

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. 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.

Coleman G. and O'Connor R., The Influence of Managerial Experience and Style on Software Development Projects, International Journal of Technology, Policy and Management, Vol. 8, No. 1, 2008
DOI - 10.1504/IJTPM.2008.016183

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).

2.3 The Study Phases

This study was divided into three distinct phases: firstly a Preliminary Phase (PP) to assist with framing the study and test the interview guide and approach; a Detailed Phase (DP) which developed the initial concepts and categories and enabled evaluation of the theoretical sampling process; and a Final Phase (FP) which further developed the categories and concepts to produce the grounded theory. In total the three phases of the study involved 25 interviews across the 21 companies profiled in Table 1.

Table 1 - Company and Individual Participation Breakdown

Company	*Market Sector*	*Interviewee*	*Phase*
1	Telecommunications	Development Manager	PP
2	Company secretarial	Product Manager	PP
3	Telecommunications	CEO	PP
4	Telecommunications	CTO	DP
5	Telecommunications	Development Manager	DP
6	Compliance Management	Quality Manager	DP
7	Enterprise	Product Manager	DP & FP
8	E-Learning	Development Manager	DP
9	Information Quality	Development Manager	DP
10	Telecommunications	Development Manager	DP
11	Telecommunications	CTO	DP & FP
12	Financial Services	CTO	DP
13	Financial Services	Product Manager	DP
14	Interactive TV	Product Manager	DP & FP
15	Public Sector	Product Manager	FP
16	Medical Devices	CTO	FP
17	Telecommunications	CTO	FP
18	Public Sector	CEO	FP
19	HR Solutions	General Manager	FP
20	Games Infrastructure	Product Manager	FP
21	Personalisation	Technical Director	FP

To generate more detailed information on how the sampling process should progress, a preliminary study phase involving 4 interviews across companies 1-3 was undertaken. 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 4 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 next phase (DP) involved interviews with an additional 11 companies. Closely following the tenets of grounded theory meant that, following the initial open coding, the interviews were then re-analysed and coded axially across the higher-level categories that had emerged from earlier interviews. Any memos or propositions that emerged through the coding process were recorded for further analysis and included as questions in subsequent interviews. A consequence of this was that the interview guide was constantly updated.

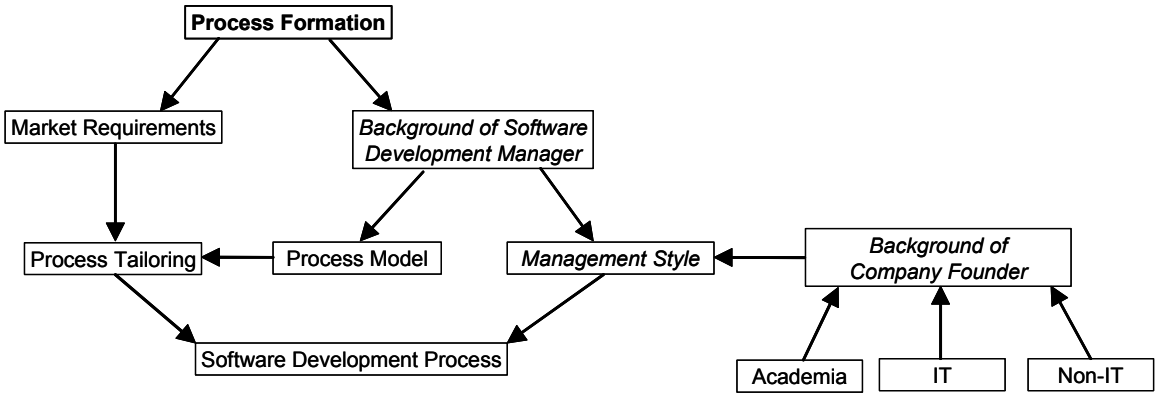
Coleman G. and O'Connor R., The Influence of Managerial Experience and Style on Software Development Projects, International Journal of Technology, Policy and Management, Vol. 8, No. 1, 2008
DOI - 10.1504/IJTPM.2008.016183

To achieve the necessary diversity amongst the study base we carried out an additional 10 interviews (FP). These interviews involved re-interviewing 3 earlier participants and 7 new companies. 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.

3. Study Results

Grounded Theory provides mechanisms 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 that can be used as the fulcrum of the study results. The answers to the research question posed in section 1 are contained in the key conceptual theme *Process Formation*[1] and its associated sub-categories. The theoretical framework can be presented in a pictorial fashion (Figure 1) creating a clear image of how this theme, its categories and subcategories are interrelated.

Figure 1 – Extract of Theoretical Framework



Process Formation is a conceptual theme and is a predecessor of its two categories, *Background of Software Development Manager* and *Market Requirements*. The *Background of Software Development Manager* determines the *Process Model* used as the basis for the company's software development activity and this *Process Model* is then subject to *Process Tailoring*. *The Background of Software Development Manager* coupled with the *Background of Founder* of the company creates an associated *Management Style* and this, in conjunction with the tailored process model, creates the company's initial *Software Development Process*.

3.1 Management Background

One of the key theoretical themes addressed by the research was *Process Formation*. The findings show that this depends on several factors, including the *Background of the Software Development Manager*, essentially the expertise that manager has accumulated over their working and educational lives, the demands of the market in which the company operates, the founder's *Management Style* and the organisational culture

3.1.1 Background of Company Founder

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

The company founders' backgrounds could be categorised as one of three different types, Information Technology (IT), Academia/IT, Non-IT (Table 2). It should be noted that those with an IT background were not all previously employed in the software sector. Also, those from the academia/IT background were essentially researchers within University IT departments who spun-off the company from research work. Those with non-IT backgrounds were a builder, engineer, teacher, geophysicist, TV executive and HR executive. In a number of the companies (1, 4, 11, 12, 16, 17, and 21), the founder or co-founder was acting as Chief Technology Officer (CTO).

Table 2 – Background of Founder

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

Within the 21 study companies, those founded by people from a non-IT background have fewer employees than those with an IT or Academic/IT lineage. Six who were non-IT founded companies have, on average, been in existence for 11 years and have an average staff complement of 50. By contrast, the 5 Academia/IT company figures are 6 years and 66 employees respectively. If employee size was to be used as a measure of success then it could be suggested that that those companies in the study who had a founder from a non-IT background are less successful than those whose founder came from an IT or Academia/IT background, perhaps suggesting greater familiarity with the industry or indeed technology itself.

3.1.2 Background of Software Development Manager

Within the different study companies the title given to the person with responsibility for process definition and implementation varied. In the start-up situation, where process was established, it was the person with responsibility for managing the software development effort who managed the process activity. Within this study, the generic individual with company responsibility for process is referred to as the 'Software Development Manager'.

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

The majority of those interviewed had previously operated in a software development manager, or similar, role prior to joining their current company. From all of the interviews, it was clear that where the software development manager had worked before, what their responsibilities were, what process and process improvement model was used and the company culture shaped the process that the software development manager used in their

current company. The link between the company's original process and the *Background of the Software Development Manager* was highlighted in a number of interviews. The extracts below, are typical of the responses as to why a particular process model was used.

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

"The VP Engineering was the main architect of the product, plus the engineering and product management and all the other things besides and that has been the model until very recently. He brought that with him."

"In terms of technology, I'm the CTO, I was hired [in week 2 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 one are reasonable if not great."

If the managers had a prior positive experience with a particular process model and they understood it particularly well, then they opted for familiarity rather than something novel. This concept of bringing a particular model or tool with them was a common feature of the managers interviewed. For example, the software development manager in companies 8 and 11 brought the RUP (Rational Unified Process) with them, whilst the manager in company 12 brought XP to his current organisation.

3.2 Management Experience

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

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

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

"What the IT experience gave me was a knowledge of how things were done, how I'd seen them done, how I'd seen them done badly… equipped me with information as to what sort of processes I wanted to put in place and what I didn't and why I wanted them."

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

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

In a couple of the companies interviewed, during the start-up phase a senior developer was appointed rather than a software development manager. In very small organisations such as

these, which have 1 or 2 person teams, the senior developer is effectively the software development manager. Subsequently, the practices used by the senior developer, created by their background experience, become the de facto initial process. This extract from Company 2 characterises it best:

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

Whilst the background and experience of the software development manager helps to form the process, prior negative experiences can also work against certain process elements. For example, in relation to the adoption of the ISO9000 and CMM(I) standards, prior negative experience had an influence on software development manager's decisions in their current companies, as exemplified by these interview extracts.

"I worked in companies who were so hung up on ISO 9001. And it just didn't work."

"I did work in an organisation, a manufacturing company, which was ISO and particularly their lab research work. The amount of documentation that was required there seemed to impede things more than anything else."

"It [CMM in previous company] was dire. It just got in people's way. It was designed almost to get in people's way. It wasn't designed to enhance the development process. It wasn't for me."

3.3 Management Style

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

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

manager. Nonetheless, there was one excellent example from an Expansion company which showed how *Management Style* affected the initial software process and how it was managed.

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

3.3.1 Management Approaches – "Command and Control"

In three of the Start-up companies, the *Management Style* is very directive, which can be characterised for this study as a 'command and control' management approach, with strong similarities to McGregor's (1985) 'Theory X' style. This type of 'command and control' style was illustrated by company managers who closely supervised their staff, lacked trust in their staff's abilities and made decisions without consultation. Some examples of how managers exercised 'command and control' are illustrated by these interview extracts.

Company 1 directing their staff on why they needed to follow SPI: *"So we were telling people this [SPI] is for the growth of the company so it's for everybody's good to go along with it and embrace it."*

Company 3, one of the smallest interviewed, has a very 'hands-on' CEO who also adopts a 'command and control' *Management Style*: *"If a guy isn't delivering, we just don't want him in the company. You encourage him to leave or structure an exit for him."*

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

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

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

"If you end up with process-type activity, which is purely known to the developers on the project, and is a language they speak among themselves, it becomes unhelpful, because it can be used as a defence for not getting things done. Once that develops, from a business management point of view, it would be very difficult to dismantle that attitude subsequently if you have a team of 5 people who adopt that approach to their work. You are almost in a position, where it's like a spoiled batch, you would nearly be better to toss out that batch and start again, cause if you feel it's not what you want within the team it would be very difficult to get it out."

There is a fear here of loss of control and power which is an element in the 'command and control' style. From the study, this fear is primarily confined to the smaller companies. It may

be the case that as companies increase in size, more managerial staff are required to deal with the teams involved and the founder and software development manager realise they have little option but to allow others to take control thus challenging their fears with respect to delegating authority. But other managers also showed suspicion of developers within their teams as is evident in the following examples.

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

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

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

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

3.3.2 Management Approaches – "Embrace and Empower"

In opposition to 'command and control' structures, many company managers within the industry operate what can be characterised for this study as an 'Embrace and Empower' regime, which has strong similarities to McGregor's (1985) 'Theory Y' style. In this context the opinions of subordinates are valued and included as part of software development policy. Also there is greater evidence of trust in development staff and their ability to carry out tasks with less direct supervision. Overall there is greater delegation of responsibility, more participation by staff in decision-making, and, more generally, an environment where consensus prevails.

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

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

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

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

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

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

4. Support for Findings

On completion of a grounded theory study it is likely that the findings will be at variance with published studies in related areas (Strauss and Corbin, 1998). Therefore the developed theory should either be integrated with the existing work or act as a critique of it (Goulding, 2002). On completion of the investigation and data analysis, the literature has a major role both in confirming findings and using the findings to highlight where the literature is incorrect or only partially explanatory (Strauss and Corbin, 1998). But an extensive trawl of the literature should only be done after the grounded theory has been formulated as, "*running to the published literature to validate or negate everything one is finding hinders progress and stifles creativity. Used as a analytic tool it can foster conceptualisation*" (Strauss and Corbin, 1998. The grounded theory presented in section 3 and 4 advocates a consideration of factors, other than merely technology, in SPI programmes. Therefore this research, in conjunction with a review of the Software Engineering literature, also examined and discovered support for the theory in the Information Systems, Human, and Social factors disciplines and from academics, practitioners and other industry commentators.

4.1 Software Development Manager

One of the theoretical categories raised by this research is that the initial development process used by a software start-up is based on the experiences of the software development manager.

In a Northern Irish context, McFall et al. (2003) found that often companies are managed by entrepreneurs and directors who know the processes well and subsequently act as mentors to other members of staff.

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

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

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

4.2 Management Style

Management Style describes the way a leader discharges their administrative functions and motivates and communicates with their staff (Buchanan and Huczynski, 1985). Among the study practitioners interviewed, *Management Style* varied between 'command and control' approaches and 'embrace and empower'. In software start-ups many managers encourage all employees to be involved in all aspects of development (Kelly and Culleton, 1999). Whilst numerous organisations retain this culture of involvement, many large companies delegate responsibility for software process to a dedicated process group. In smaller companies and start-ups, senior management often allow their developers to have a significant influence over the way they work. By contrast, some organisations enforce process on their employees. This 'command and control' *Management Style* has its opponents who believe that efforts to force developers to work according to procedures developed by those not immediately responsible for results have failed (Beck and Boehm, 2003). For example, XP proposes a strategy of decentralised decision making (Beck, 2000). As a result, agile development methodologies thrive in 'embrace and empower' environments where staff are empowered and the organisation can be said to be thriving on chaos, whereas plan-driven approaches are more suited to a situation dominated by policy and procedure (Boehm and Turner, 2003). Nevertheless, some companies may struggle with adopting an agile development approach as many managers, particularly those at senior level, are reluctant to surrender the feeling of control that Gantt charts and other plan-driven process documents provide (Cohn and Ford, 2003).

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

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

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

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

5. Discussion and Conclusions

One of the key theoretical themes addressed by the research was *Process Formation* which is related to how process is formed or created within a software product company. The findings show that this depends on several factors. The main one relates to the *Background of the Software Development Manager*. This describes how the expertise accumulated by the person tasked with managing the initial software development effort dictates what the start-up software process will be. The final shape the process model takes will be influenced by

additional factors, including the demands of the market in which the company operates, their own and the founder's *Management Style*, and the culture of the organisation. In addition, whether the *Management Style* used within the organisation is controlling and directive, or consensual and involving, will further influence how closely developers adhere to the firm's defined working methods.

There is an absence of published material describing how process is initially formed in software product companies. This research provides a new contribution to this area. Using evidence from practice, a theory has been generated which explains the factors which influence the first software process a company will use. In a further departure from standard practice, the research explains how SPI is not solely technology-centred but rather is affected by wider human and organisational factors. This suggests that SPI studies which concentrate purely on procedural and bureaucratic adherence, and thus neglect the human and organisational dimension, are flawed by failing to take account of key pieces in the SPI jigsaw. Consequently, this research offers support to the argument that people issues, amongst other factors, must be considered in SPI initiatives.

The findings from this research indicate that human and social factors have a major role to play in SPI. However, this is an aspect that has largely been ignored in SPI studies within software engineering. Whilst Information Systems studies do attempt to take the human and social dimension into account when examining methodological and process issues, the same cannot be said for Software Engineering research. This socio-cultural element should be explored further, within software engineering, to get a full picture of its role in SPI

5.1 Implications

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

Process Formation is primarily of relevance to software start-ups. The study has revealed that software process in a start-up situation is a nebulous concept in that organisations will use whatever works to support their immediate business objective. Typically this business objective is survival. Getting a product to market as quickly as possible may mean the difference between survival and decline. But any small software company suffers from having limited resources and is focused merely on 'doing things' rather than 'doing things right'. The resources are simply not available to explore the best way to develop software for

that organisation at that time. As a result, start-ups depend largely on the experience of the person acting as Software Development Manager whose expertise and know-how can help them meet their deadlines and reach the next stage of development. For companies like this who, by necessity, typically have a skeleton process in place, any attempt to interest them in SPI will be somewhat redundant. However, agile methods, as have been seen in this study, do have a lot to offer such organisations. Start-ups are product-driven and, with very small development teams, often developer-led. Agile methods too are product-driven and developer-led. Because of the confluence between these two factors, there is more value in offering start-up companies 'software practice improvement' rather than software process improvement. Only then when survival has been achieved and development approaches have somewhat stabilised, should the issue of SPI be examined.

5.2 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. This study, though the reality of the situation could be potentially different to that described, 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.

5.3 Future Research

One of the contributions of this work is a grounded theory explaining how software process is initially established. 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 a counter-balance to 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.

References

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

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

Coleman G. and O'Connor R., The Influence of Managerial Experience and Style on Software Development Projects, International Journal of Technology, Policy and Management, Vol. 8, No. 1, 2008
DOI - 10.1504/IJTPM.2008.016183

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

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

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

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

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

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

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

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

Coleman G., O'Connor R., Using grounded theory to understand software process improvement: A case study of Irish software product companies, *Information and Software Technology*, Vol. 49, No. 6, pp. 531-694

Crosby, P.B., 1979, *Quality is Free: The Art of Making Quality Certain*, McGraw-Hill, USA.

DeMarco, T. and Lister, T., 1999, *Peopleware: Productive Projects and Teams*, 2nd Edition, Dorset House, New York.

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

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

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

Hansen, B. and Kautz, K, 2005, 'Grounded Theory Applied – Studying Information Systems Development Methodologies in Practice', in Proceedings of 38th Annual Hawaiian International Conference on Systems Sciences, Big Island, HI.

Herbsleb, J. and Goldenson, D., 1996, 'A Systematic Survey of CMM Experience and Results', in *Proceedings of the 18th International Conference on Software Engineering*, Berlin, Germany, pp. 323-330.

Humphrey, W.S., 1995, A Discipline for Software Engineering, Addison Wesley, Boston, MA

Humphrey, W.S., 2002, *Winning With Software: An Executive Strategy*, Addison Wesley, Boston, MA.

Humphrey, W.S., Snyder, T. and 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.

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

Kautz, K., 1998, 'Software Process Improvement in Very Small Enterprises: Does it Pay Off?', in *Software Process Improvement and Practice*, Vol. 4, No. 4, pp. 209-226.

Kasse, T. and McQuaid, P.A., 1998, 'Entry Strategies into the Process Improvement Initiative', in *Software Process Improvement and Practice*, Vol. 4, No. 2, pp. 73-88.

Keeni, G, 2000, 'The Evolution of Quality Processes at Tata Consultancy Services', in *IEEE Software*, July/August 2000, pp. 79-88.

Kelly, D.P. and Culleton, B., 1999, 'Process Improvement for Small Organisations', in *IEEE Computer*, October, pp. 41-47.

Laporte, C.Y. and Trudel, S., 1998, 'Addressing the People Issues of Process Improvement at Oerlikon Aerospace', in *Software Process Improvement and Practice*, Vol. 4, No. 4, pp. 187-198.

Lycett, M., Macredie, R.D., Patel, C. and Paul, R.J., 2003, 'Migrating Agile Methods to Standardised Development Practices', in *IEEE Computer*, June, pp. 79-85.

Martin, P.Y., Turner, B.A., 1986, Grounded Theory and Organizational Research, The Journal of Applied Behavioral Science, Vol. 22. No. 2., 141-157.

Mathiassen, L., Axel Nielsen, P. and Pries-Heje, J., 2001, 'Learning SPI in Practice', *Addison Wesley Professional Articles*,

McFall D., Wilkie F.G., Mc Caffery F., Lester N. and Sterritt R., 2003, 'Software Processes and Process Improvement in Northern Ireland', in *Proceedings of 16th International Conference on Software and Systems Engineering and their Applications (ICSSEA)*, CNAM, Paris, France, December.

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

Nisse, D., 2000, 'Leadership, Army Style', in *IEEE Software*, March/April, pp. 92-94.

Potter, N.S. and Sakry, M.E., 2002, *Making Process Improvement Work: A Concise Action Guide for Software Managers and Practitioners*, Addison Wesley, Boston, MA.

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>

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 and Murphy, R, 2007, An exploratory study of why organizations do not adopt CMMI, The Journal of Systems and Software, Vol. 80, No. 6, 883–895.

Sommerville, I., 2007, *Software Engineering*, 8th Edition, Addison Wesley, Reading MA.

Royce, W., 2005, 'Successful Software Management Style: Steering and Balance', in *IEEE Software*, September/October, pp. 40-47.

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

Sutton, S.M., 2000, 'The Role of Process in a Software Start-up', in *IEEE Software*, July/August, pp. 33-39.

Strauss, A., Corbin, J.M., 1998. Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory, 2nd Edition, Sage Publications.

Yamamura, G., 1999, 'Process Improvement Satisfies Employees', in *IEEE Software*, September/October, pp. 83-85.

Zahran, S., 1998, *Software Process Improvement: Practical Guidelines for Business Success*, Addison Wesley, Boston, MA.

Biographical notes

Dr. Gerry Coleman is a member of the Department of Computing and Mathematics at Dundalk Institute of Technology. He received his Ph.D. in Software Engineering from Dublin City University. Prior to taking up an academic position, he worked for a number of years as a Business Analyst and Project Manager in the software development departments of several large financial institutions. He also worked as a Senior Consultant in the Centre for Software

Coleman G. and O'Connor R., The Influence of Managerial Experience and Style on Software Development Projects, International Journal of Technology, Policy and Management, Vol. 8, No. 1, 2008
DOI - 10.1504/IJTPM.2008.016183

Engineering with responsibility for research and technology training and transfer. His research interests include software process improvement, software development methods and software quality, with particular emphasis on small software companies.

Dr. Rory O'Connor is a senior lecturer in Software Engineering at Dublin City University. He received a Ph.D. in Computer Science from City University (London) and an M.Sc. in Computer Applications from Dublin City University. He has previously held research positions at both the National Centre for Software Engineering and the Centre for Teaching Computing, and has also worked as a software engineer and consultant for several Irish and European technology organisations. His research interests are centred on the processes whereby software intensive systems are designed, implemented and managed. He is also interested in technology adoption issues and the process whereby Information Technology tools and techniques are evaluated and selected in a commercial setting.