The current issue and full text archive of this journal is available at
www.emeraldinsight.com/1741-0398.htm

An investigation into software development process formation in software start-ups

Gerry Coleman
Department of Computing, Dundalk Institute of Technology, Dundalk, Ireland, and
Rory V. O'Connor
School of Computing, Dublin City University, Dublin, Ireland

Abstract

Purpose – This paper reports on the results of an investigation into how the software development process is initially established within software product start-ups.

Design/methodology/approach – The study employs a grounded theory approach to characterize the experiences of small software organizations in developing processes to support their software development activity. Using the indigenous Irish software product industry as a test-bed, the authors' examine how software development processes are established in software product start-ups and the major factors that influence the make up of these processes.

Findings – The results show that the previous experience of the person tasked with managing the development work is the prime influencer on the process a company initially uses. Other influencers include the market sector in which the company is operating, the style of management used and the size and scale of the company operations.

Originality/value – The model has particular implications for start-up software product organisations that wish to successfully manage their product development from an early stage.

Keywords Computer software, Product development, Ireland

Paper type Research paper

Introduction

For many small and start-up 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 provide assistance in making estimates, developing plans and measuring quality. 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 including, waterfall, evolutionary and component-based developments (Sommerville, 2007), exist as instantiations of how software development can be undertaken.

Small software companies, and in particular start-ups, are creative and flexible in nature and are reluctant to introduce process or bureaucratic measures which may hinder their natural attributes (Sutton, 2000). In addition small and start-up companies have very limited resources and typically wish to use these resources to support product development.

Journal of Enterprise Information Management
Vol. 21 No. 6, 2008
pp. 633-648

1741-0398
DOI 10.1108/17410390810911221

This research set out to explore the following research question:

RQ1. How are software processes initially established in a software company?

To attempt to answer this it was necessary to address two further questions:

RQ2. What software processes are software companies currently using?

RQ3. How do the operational and contextual factors, present in organisations, influence the content of software processes?

Background

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

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 software process improvement (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 per cent of senior managers as opposed to 5 per cent of developers. 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.

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. 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. There is evidence that empowering development practitioners, and allowing them to take ownership of the processes they use, motivates SPI success (Baddoo and Hall, 2003).

Research methodology

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 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, as they are orientated towards how individuals and groups view and understand the world and construct meaning out of their experiences. Also, a particular strength of qualitative research is its ability to explain what is going on in organisations (Avison *et al.*, 1999).

Of the qualitative methodologies available, we believed grounded theory (Glaser and Strauss, 1967) offered the best mechanism for achieving the research objectives. The emphasis in grounded theory is on new theory generation. This manifests itself in such a way that, rather than beginning with a pre-conceived theory in mind, the theory evolves during the research process itself and is a product of continuous interplay between data collection and analysis of that data (Goulding, 2002). According to Strauss and Corbin (1998), the theory that is derived from the data are 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. We chose grounded theory as the method of enquiry for the following reasons:

- Given the lack of an integrated theory in the literature as to how software processes are formed, 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. Software development is labour-intensive and software process relies heavily on human compliance for its deployment.
- It is an established and credible methodology in sociological and health disciplines (e.g. nursing studies, psychology), and a burgeoning one in the IT arena. This study provided an opportunity to apply a legitimate and suitable methodology to the software field.

Since the initial launch of grounded theory, the Glaser and Strauss alliance gradually separated until each was developing a different version of the methodology. First in 1990 (Strauss and Corbin, 1990) and in a follow-up (Strauss and Corbin, 1998), Strauss, now in conjunction with Corbin, created an updated version of grounded theory with extended coding systems. 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. This study employed the Strauss and Corbin (1998) approach. 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). 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 methodology to examine, the use of "systems thinking" practices (Goede and de Villiers, 2003), software inspections (Seaman and Basili, 1997; Carver and Basili, 2003), process modelling (Carvalho *et al.*, 2005), requirements documentation (Power, 2002), and virtual team development (Sarker *et al.*, 2001; Qureshi *et al.*, 2005). Hansen and Kautz (2005) used grounded theory to study the use of development practices in a Danish software company and concluded that it was a methodology well-suited for use in the IS sector.

From a software process perspective, the role of individual actors, and their environmental surroundings and conditions, weighs heavily on how the process is practiced. We believed that grounded theory, whilst handling the contextual and situational factors, could facilitate and support the gathering and analysis of those human experiences and highlight the associated interrelationships with other human actors.

Study setting

The 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 creating a software process, 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.

Conducting the grounded theory study

Despite the research questions being clearly defined, the theoretical sampling approach of grounded theory means it is unclear in advance exactly the types of practitioners and companies that need to be interviewed during a study to meet the research objectives. As a result, the study was divided into three phases, a preliminary phase 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 three phases of the study involved 25 interviews with senior company personnel across the 21 companies profiled in Table I.

At the outset, to generate more detailed information on how the sampling process should progress, a preliminary study phase, involving four interviews across companies 1-3 was undertaken. This phase highlighted two issues in particular which would steer the immediately subsequent sampling activity. Firstly, analysis of the software companies' target market indicated that the intended list of companies, in the full study, should incorporate as many sectors as possible. Secondly, a specialist qualitative analysis tool, which supported the grounded theory approach, was essential. Having investigated the range of tools which are used for data management in qualitative research, Atlas TI (Muhr, 1997), a tool designed specifically for use with grounded theory, was selected.

The next phase of the study (Phase 1) involved interviews with an additional 11 companies. Though a number of theoretical concepts emerged during the early fieldwork, the researchers decided to re-evaluate the study progress following the interview with company 14. This analysis indicated that the range of companies interviewed should be diversified. This approach is in accordance with both Strauss and Corbin (1998) and Goulding (2002), who advocate diversity in the data gathering and "staying in the field" until no new evidence emerges. We also believed that to conclude the sampling process at this point would constitute premature closure, a mistake often associated with grounded theory (Glaser, 1992).

Company no.	Market sector	Total no. of employees	No. employees in s/w development	Interviewee
1	Telecommunications	6	3	Development manager
2	Company secretarial	50	20	Product manager
3	Telecommunications	10	3	CEO
4	Telecommunications	70	30	CTO
5	Telecommunications	12	6	Development manager
6	Compliance management	100	40	Quality manager
7	Enterprise	150	100	Product manager
8	e-Learning	120	70	Development manager
9	Information quality	27	9	Development manager
10	Telecommunications	15	12	Development manager
11	Telecommunications	160	110	CTO
12	Financial services	35	23	CTO
13	Financial services	130	90	Product manager
14	Interactive TV	60	40	Product manager
15	Public sector	150	90	Product manager
16	Medical devices	19	9	CTO
17	Telecommunications	70	35	CTO
18	Public sector	3	3	CEO
19	HR solutions	30	15	General manager
20	Games infrastructure	40	20	Product manager
21	Personalisation	50	40	Technical director

Table I.
Subject company profile

To achieve the necessary diversity amongst the study base we carried out an additional ten interviews (Phase 2). Three of these interviews involved re-interviewing earlier participants, a technique available to grounded theory studies and supported by (Goulding, 1999) as it allows for a comprehensive checking and verification process of the data already analysed, and seven additional companies. These seven additional companies (companies 15-21) were specifically selected as their business sectors helped extend the scope of the study and ensured that theoretical categories were not being established on an excessively narrow basis. 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.

Results and discussion

The grounded theory categories and the various relationships were then combined to form the theoretical framework for process formation as shown in Figure 1.

Within the theoretical framework, each node is linked by a precedence operator, with the node attached to the arrowhead denoting the successor. No relationship types other than precedence are contained within the framework and the network is read from left to right. The tildes (~) represent codes that were renamed or merged with other codes during the analysis process. The root node of the framework, process formation, is a conceptual theme and is linked to several key conceptual categories.

Theoretical categories

In relation to the factors influencing process formation, the study highlighted a number of theoretical categories (Table II) (From hereon the categories produced by the study are denoted in italics.). Each of these categories can be linked to quotations within the interviews and these provide support and rich explanation for the results.

In the study companies, the title of the person with overall responsibility for software process differed, from software development manager to chief technology officer (CTO), director of engineering, or product development manager. For reasons of simplicity and clarity, the generic title software development manager has been used in

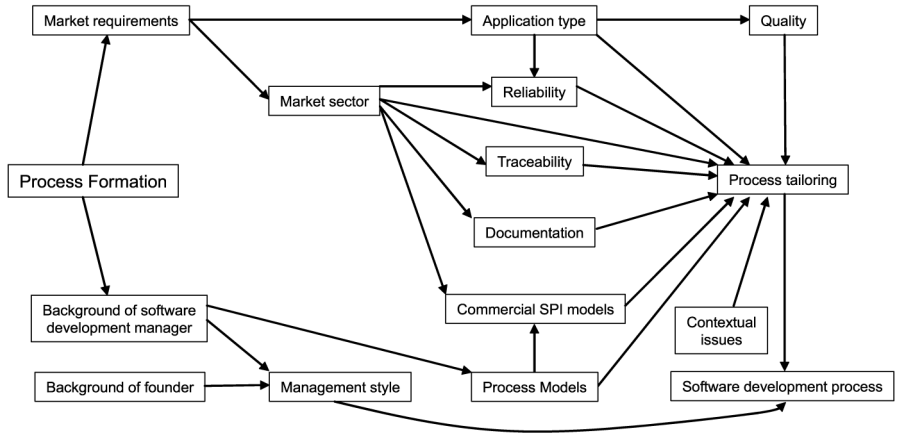

Figure 1.
Process formation network

this study. The *background of software development* manager determines the *process model* used as the basis for the company's software development activity and this *process model* is then subject to *process tailoring*. The *background of software development manager* coupled with the *background of founder* of the company creates an associated *management style* and this, in conjunction with the tailored process model, creates the company's initial *software development process*.

Background of the software development manager

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

The majority of those interviewed had previously operated in a software development manager, or similar, role prior to joining their current company. From all of the interviews, it was clear that where the software development manager had worked before, what their responsibilities were, what process and process improvement model was used, and the company culture, shaped the process that the software development manager used in their current company. This comment, from the development manager in company 8, is typical of the responses as to why a particular process model was used:

> For software development we have used the Rational Unified Process (RUP). The reason is that the guy we took into head up our technology area brought that with him.

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

Impact of managerial experience. In addition, all of the managers brought with them something less tangible, namely "experience". This is defined within this study simply

Theme	Category
Process formation	*Background of software development manager*
	Background of founder
	Management style
	Process tailoring
	Market requirements

Table II.
Process formation – main theoretical categories

as "knowing what to do in a given situation". One manager when asked about how he managed to grow the software development activity in his current organisation stated:

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

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

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

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

> At the beginning we looked at what sort of environments people had worked in before, and what sort of process they used, and we tried to import and adapt them.

But beyond the *background of software development manager*, the impact of culture or more specifically *management style* also dictates how the process is formed and 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.

Management style

Background of founder. The company founders' backgrounds could be categorised as one of three different types, information technology (IT), academia/IT, non-IT (Table III).

It should be noted that those with an IT background were not all previously employed in the software sector. Also, those from the academia/IT background were essentially researchers within university IT departments who spun-off the company from research work. Those with non-IT backgrounds included a builder, engineer, teacher, geophysicist, TV executive, and HR executive. In a number of the companies (1, 4, 11, 12, 16, 17, and 21), the founder or co-founder was acting as CTO. For a more detailed discussion on the founder's impact on management style please refer to Coleman and O'Connor (2008).

Management style and process formation. There was a sharp diversity between the *management styles* adopted within the different study companies. Some companies tend to be more enforcing of process allowing little deviation whilst others give the developers more latitude within it. During this study, whilst it was clear that *management style* helped the initial formation of the process, it also had an impact on how the process was implemented on an ongoing basis. From the extracts therefore, it

Table III.
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

was not possible to divorce completely *management style* issues at *process formation* from more recent management initiatives which influenced ongoing process adherence. Nonetheless, there was one excellent example, from a manager in one of the larger companies, 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 C [the CEO] because he was the guy who originally wrote the code.

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.

The software development manager in company 1 directed his staff on why they needed to follow process:

> So we were telling people this [process] 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 and who stated:

> 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. The manager in company 9, which was in a growth state of development, typified this thus:

> If [process non-compliance] is happening constantly, then every week it's highlighted in the team meetings and the staff member must explain why. And to be honest it's a bit brutal but 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, from the CEO in company 3, represents many 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.

Other managers also showed suspicion of developers within their teams as is evident in this example quotation from the development manager in company 5:

> 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 fine 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 has 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.

Though management style has a major influence on process formation, there is no clear indication from the study whether companies with this sort of directive style are more or less successful, in general business terms, than those with a more consensual management approach.

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, there is greater evidence of trust in development staff to carry out tasks with less direct supervision, greater delegation of responsibility, and, a more generally, consensual environment.

The quality manager in company 6, one of the largest companies in the study, consults widely with his staff in relation to process usage:

> If our customers are recommending that we change code review, I go away and send an email out to all my 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. 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 often feel 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. Senior engineers have more status in an organisation like this, as this extract from the CTO in company 12 shows:

> If you have one guy working on a piece of consultancy with 15 years experience, he understands the principles of how we work. He doesn't need someone else interfering. So you may as well just let him do the job.

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. However, 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.

Market requirements

The *market requirements* of the target market also have a fundamental effect on the establishment and use of the software process in an organisation. Software companies release products into specific *market sectors*. Within this research, *market sectors* are treated as a subset of *market requirements*. For example, almost all applications used by companies in regulated *market sectors* will have particular requirements and the nature

of regulation means that the process used to create these applications must guarantee this. Other *market sectors* such as telecommunications also require applications which can meet high-availability demands. However, *market requirements*, such as a need for high *reliability*, extensive *documentation* or, as is often the case, speed of delivery, can transcend multiple *market sectors*.

Process and regulation. Probably the best example in this study of *market sector* influencing the process occurred in the case of company 6, whose products are bought by pharmaceutical companies and this meant that its processes had to cater for this from day 1. The quality manager stated: "The most important thing is the market we are producing to. We wouldn't sell without a good quality process."

Within the confines of this regulatory environment, company 6 has very little latitude and flexibility in the process they can use, as the companies to whom they sell must, under Food and Drug Administration (FDA) rules, audit their suppliers. The quality manager reports that:

> Because we produce for the pharmaceutical industry, every single client does a detailed 2-day audit of our software processes, the quality of our products, etc. So when they in turn are audited, by the FDA, they will have an audit report to show them.

This audit is conducted to satisfy regulatory compliance as the pharmaceutical companies themselves must show, not only are their own products compliant, but also that how they are made complies with the regulatory guidelines. It also means that the software producer in this sector must have appropriate *documentation* for all its products for all stages of the development process. Any changes made during development must be recorded for *traceability* and subsequent audit purposes. This imposes a rigour on the process which other companies may be able to avoid. Company 16 operate in the medical space and business expansion plans will mean a move to a regulatory environment. As the CTO outlines:

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

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

Process and application type. Beyond regulated industries, the *application type* may require the system to be constantly available, thus placing a huge requirement on high *quality* and *reliability*. Sectors such as telecommunications and banking can require such systems. The manager in company 4, who develop systems for the mobile telecommunications domain, best personifies this:

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

As the comment makes clear, this industry imposes penalties on late delivery and demands service level agreements. As a result any process, which produces products for this sector, must take account of this from its inception. Another business area that has its own unique demands is the public or governmental sector. Several companies have experienced this with the following extract, from the development manager in company 5, exemplifying things best:

> Take for example the system we are developing for the police force. They have very strict documentation standards which we follow, and that involves a full functional spec, a full UML design, a very tightened development process, and a testing process. So in that case, they are putting certain demands on us, in terms not only of what we do, but the way we do it.

In summary, the above examples illustrate how the development process must be geared to provide the necessary services required by the market.

Process tailoring

Though, in process terms, the software development manager brings with them a wealth of experience to their new organisation, some of that may have been gathered in organisations that were much different in nature, which means that some *process tailoring*, to reflect their new environment, was necessary. The process models used in the study companies were typically based on one of the standard industry development models, waterfall or RUP, or the development methodology XP. All of the companies tailored the model, generally by dropping some of the practices contained within it and adding some new practices which reflected their own particular operating context. *Process tailoring* such as this leads to a proprietary development model which, although possibly based on a standard, is considered more suited to the company's business. The product manager in company 14 provides a good example:

> We took the RUP and at this stage probably very little of our process resembles it. We didn't need all of the detail that was in it, so as a small company, we have changed it around to suit our own needs.

Process tailoring – influencing factors. In every case, however, contextual issues such as company and team size, project size, team expertise, development environment, etc. in addition to the *background of software development manager* and the *market requirements*, were the main inputs to the tailoring process. It is important to recognise that when using process models, as part of process formation, most organisations scale down. Practices are routinely removed. The CTO in company 12 put it most succinctly:

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

Despite its application to the initial software process a company uses, process tailoring is something that occurs throughout the lifetime of the organisation concerned. On every occasion that an improvement to the process is made, contextual issues act as inputs to the improvement process.

Conclusions

This research has addressed factors which influence the formation of software process in software product companies. How process is formed is primarily of relevance to

start-up and early-stage software firms. The study has revealed that in a start-up situation organisations use whatever software process works to support their immediate business objective. 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. Agile methods, such as XP, 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 of these two factors, we believe there is more value in offering start-up companies "software practice improvement" rather than SPI. Then when companies have achieved something of a sustainable base, and development approaches have somewhat stabilised, should the issue of SPI be examined.

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 are established and grow. Because the findings show that the initial process is determined by who acts in the software development manager role, this has clear implications for the hiring policy of the software start-up. Software founders and entrepreneurs must take cognisance of the qualities required by the individual who will fill the development manager role. Whilst it is not essential that the development manager has previously worked in the hiring company's market sector, it would be of significant benefit if they fully understand the demands of that market and the requirements for products sold in that market. For example, where companies are entering a regulated market, such as one governed by the FDA, they will need to put in place processes which reflect the compliance levels imposed by the regulator for auditability, traceability and comprehensive documentation. It would therefore be a major plus for the start-up if they employ a software development manager who has experience of documentation-driven process models as, because there is a lesser process learning-curve for this manager, the software process will meet compliance demands more seamlessly, and competitive advantage can be gained.

Conversely, where there are no external standards or controls imposed, and the founder's desire is to have a product ready for market as early as possible, then employing a software development manager who is familiar with "lean" or agile product-driven development approaches would likely achieve business objectives more quickly than someone from a plan-driven development background. Managerial experience, which can help companies overcome or avoid early process issues, the ability to successfully tailor a process, and *management style*, which can create an appropriate organisational culture, should also be factored into the recruitment decision. Thus, employing the "right" software development manager for a software start-up has far-reaching implications for that organisation and could help determine its long-term health and potential success.

Limitations of the study

Grounded theory as a qualitative research method, using semi-structured interviews, centres on respondents' opinions. This opinion is the respondents' view or perception of what is taking place, which of course, may be at odds with reality. In many instances there may be no supporting evidence to verify the opinion expressed. However, researchers must accept the veracity of what respondents say during the study

interviews (Hansen and Kautz, 2005). Notwithstanding the issues surrounding semi-structured interviews, the opinions of the participants are vital. In this study, even though the reality of the situation could be potentially different to that described, it is the managers' perception of what is happening and it is on this perception that they base their decisions. It is these actions and interactions, arising from the participants' opinions, beliefs and perceptions, which are essential to a grounded theory study.

Future research

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

This study concentrated in one geographical location. A study which examines practices in other countries would provide further validity for this research and indicate if the findings can be replicated elsewhere or if they are peculiar to the Irish context. As much software is developed outside the software product company domain, a study including a wider range of software development from bespoke software solutions to the in-house software departments of non-software companies could be counter-balanced against this work.

References

Armour, P.G. (2001), "Matching process to types of teams", *Communications of the ACM*, Vol. 44 No. 7, pp. 21-3.

Avison, D., Lau, F., Myers, M. and Nielsen, P. (1999), "Action research", *Communications of the ACM*, Vol. 42 No. 1, pp. 94-7.

Bach, J. (1998), "Microdynamics of process evolution", *IEEE Computer*, February, pp. 111-3.

Baddoo, N. and Hall, T. (2003), "De-motivators for software process improvement: an analysis of practitioners' views", *The Journal of Systems and Software*, Vol. 66 No. 1, pp. 23-33.

Baskerville, R. and Pries-Heje, J. (1999), "Grounded action research: a method for understanding IT in practice", *Accounting, Management & Information Technologies*, Vol. 9, pp. 1-23.

Bersoff, E. (1994), "Anatomy of a software start-up", *IEEE Software*, January, pp. 92-100.

Carvalho, L., Scott, L. and 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. 113-27.

Carver, J. and Basili, V. (2003), "Identifying implicit process variables to support future empirical work", *Proceedings of 17th Brazilian Symposium on Software Engineering (SBES 2003)*, pp. 5-18.

Coleman Dangle, K., Larsen, P., Shaw, M. and Zelkowitz, M.V. (2005), "Software process improvement in small organisations: a case study", *IEEE Software*, November/December, pp. 68-75.

Coleman, G. and 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. 531-694.

Coleman, G. and O'Connor, R. (2008), "The influence of managerial experience and style on software development projects", *International Journal of Technology, Policy and Management*, Vol. 8 No. 1, pp. 91-109.

Glaser, B. (1992), *Basics of Grounded Theory Analysis: Emergence vs Forcing*, Sociology Press, Mill Valley, CA.

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

Goede, R. and de Villiers, C. (2003), "The applicability of grounded theory as research methodology in studies on the use of methodologies in IS practices", *Proceedings of the Conference of the South African Institute of Computer Scientists and Information Technologists, Gauteng, South Africa*, pp. 208-17.

Goulding, C. (1999), "Grounded theory: some reflections on paradigm, procedures and misconceptions", Technical working paper, University of Wolverhampton, Wolverhampton.

Goulding, C. (2002), *Grounded Theory: A Practical Guide for Management*, Business and Market Researchers, Sage, Thousand Oaks, CA.

Hansen, B. and Kautz, K. (2005), "Grounded theory applied – studying information systems development methodologies in practice", *Proceedings of 38th Annual Hawaiian International Conference on Systems Sciences, Big Island, HI.*

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", *IEEE Software*, July, pp. 11-23.

Kelly, D.P. and Culleton, B. (1999), "Process improvement for small organisations", *IEEE Computer*, October, pp. 41-7.

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

Muhr, T. (1997), *Atlas TI User's Manual*, Scientific Software Development, Berlin.

Nisse, D. (2000), "Leadership, army style", *IEEE Software*, March/April, pp. 92-4.

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

Qureshi, S., Liu, M. and Vogel, D. (2005), "A grounded theory analysis of e-collaboration effects for distributed project management", *Proceedings of 38th Annual Hawaiian International Conference on Systems Sciences, Big Island, HI.*

Sarker, S., Lau, F. and Sahay, S. (2001), "Using an adapted grounded theory approach for inductive theory building about virtual team development", *The Data Base for Advances in Information Systems*, Vol. 32 No. 1, pp. 38-56.

Seaman, C. and Basili, V. (1997), "An empirical study of communication in code inspections", *Proceedings of the 19th International Conference on Software Engineering, Boston, MA, May*, pp. 17-23.

Sommerville, I. (2007), *Software Engineering*, 8th Ed., Addison Wesley, Reading MA.

Strauss, A. and Corbin, J.M. (1998), *Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory*, 2nd ed., Sage, Thousand Oaks, CA.

Strauss, A. and Corbin, J.M. (1990), *Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory*, 1st ed., Sage, Thousand Oaks, CA.

Sutton, S.M. (2000), "The role of process in a software start-up", *IEEE Software*, July/August, pp. 33-9.

Voas, J. (1999), "Advice for those bitten by the startup bug", *IT Professional*, May/June, pp. 38-45.

Yamamura, G. (1999), "Process improvement satisfies employees", *IEEE Software*, September/October, pp. 83-5.

About the authors

Gerry Coleman is a Member of the Department of Computing and Mathematics at Dundalk Institute of Technology. He received his PhD 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 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. Gerry Coleman is the corresponding author and can be contacted at: gerry.coleman@dkit.ie

Rory V. O'Connor is a Senior Lecturer in Software Engineering at Dublin City University. He received a PhD in Computer Science from City University (London) and an MSc 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.