Communications of the IIMA

Volume 6 | Issue 1 Article 2

2006

Learning to Improve Software Processes: Making Sense of Practice

Ian Allison
Nottingham Trent University

Yasmin Merali
University of Warwick

Learning to Improve Software Processes: Making Sense of Practice

Ian Allison
Nottingham Trent University, Nottingham, NG11 8NS, UK.

Yasmin Merali
University of Warwick, Coventry, CV4 7AL, UK.

ABSTRACT

Software Process Improvement (SPI) programs are frequently considered to be planned in nature. However, there is recent evidence to suggest that SPI can be understood as a form of learning. Drawing on the organizational learning literature, this paper proposes an active learning perspective of improvements in processes. This view recognizes the various actors in the project to be reflective in their actions, making sense of the current context and thus designing their use of the process to best suit their needs at the time. The changes in the processes emerge through ongoing adjustments, experimentation and improvisation as developers and managers seek to improve their product development.

Key words: Software Process Improvement; Reflective Practitioners; Organizational Learning; Sensemaking

INTRODUCTION

The robustness of an organization's software processes are often considered to be a key factor in the resultant quality of an organization's information systems. Through software process improvement (SPI) programs the current methods used are refined or altered to ensure future systems are developed to a high standard. Traditionally, following the Software Engineering Institute (SEI)'s lead, programs of improvement were determined from a predefined model (e.g. the Capability Maturity Model (CMM)). Evidence shows that a number of organizations achieved benefits as a result of this adoption (Herbselb, 1997). Consequently, the majority of the literature has focused on developing such models (Hansen, 2004).

Not all companies have found this approach to software process improvement to be beneficial with many abandoning SPI programs or avoiding norm-based models like CMM (Herbselb, 1997; Conradi and Fuggetta, 2002). These models are also the subject of criticism for the rigidity of the process areas and their underlying deterministic assumptions about implementation (Bollinger and McGowan, 1991). They are also seen to be inflexible and lack an emphasis on people (McFeeley, 1996).

The SPI literature, therefore, has started to explore how processes change through time by considering process improvement in practice. So, building upon this recent literature, this paper will show that processes emerge through situated practice, as the various actors make sense of their own actions, thus enabling the learning required to improve the processes to achieve better quality products. A longitudinal case study based on a software vendor is drawn upon to provide evidence and examples of how processes improve through improvisation as practitioners reflect on their actions, and in doing so enhance the resilience of the end products.

LEARNING IN SPI

The SPI literature has begun to recognize that both competence of an organization's members and their ability to learn are important aspects of improving the quality of products and processes. Mathiassen et al (2002, p.7), for instance, suggest that:

> *SPI is driven by knowledge about practices and perceived needs, insights gained during the improvement process, software industry standards, and state-of-the art methodologies and tools. SPI efforts also depend*

on the implicit, individual knowledge of participants. However, the general idea is to make knowledge explicit and to share knowledge.

This section discusses how organizational learning theory informs our understanding of how process improvements occur. An understanding of learning and the use of knowledge as a process will be explored, showing that to understand how process improvements occur we need to encapsulate the way that organizations learn to improve over time. An action based view of knowledge helps us to see SPI as a process of sensemaking, where agents are seen to draw on their knowledge during the action, and learn through reflection on their experience.

Lyytinen and Robey (1999) cite the failure of organizations to learn from their own previous experience as a reason for the recurrence of problems in IS development. They argue that learning from experience runs against the common practice because the incentives are not there to do this. When failure occurs the pressure is on IS management to externalize the problem, identifying deficiencies in the infrastructure, or its supply, rather than poor internal application. The result of this pressure is that improvements in systems development approaches have focused on the adoption of new tools, technologies and techniques from outside, such new development methods. Lyytinen and Robey (1999) argue though that external knowledge may not be transferable into an organization as it is based on the experience of others and, even when it is transferable, external knowledge often adds little competitive benefit as it is in the public domain. So, this external emphasis does not produce the experiential learning necessary to avoid making the same mistakes with the next set of tools and technologies. To deliver improvement an organization needs to create knowledge from the inside and thereby feed the continuous improvement activity (Nonaka and Takeuchi, 1995).

New methods or development technology are readily identified when employees are knowledgeable about current processes. Fichman and Kemerer (1997) confirm that organizations are more likely to innovate or take on innovation when the skills required are close to those in the organization. Successful application of ideas increases confidence and knowledge, encouraging further changes (Allison, 1999). Competence of software developers can encourage commitment to projects and process improvement. When software practitioners 'understand and appreciate the process, they are empowered to use their discretion and adapt the process to meet the needs of both the situation and their customers' (Aaen et al, 2002, p.34). Without such ability or readiness to learn, an organization is likely to face difficulties in delivering change and adapting to the demands of the market, and will therefore lose out in a competitive environment. Ravichandran and Rai (2000), for instance, suggest that an IS group's ability to learn needs to be nurtured, as acquiring technical know-how places significant demands on developers. Organizations seek to address this learning by developing programs and incentives to encourage this to happen.

These human resource management strategies do not offer immediate solutions. Rather, Scarbrough (1998, p.221) states that whilst this resource based view of knowledge traces 'a clear and compelling causal path from endogenous characteristics of the firm through its product and innovation portfolio to competitive performance', there are problems with tapping knowledge as knowledge is distributed and embedded in different social groups (communities-of-practice). So, he challenges the desire to orchestrate the distribution of knowledge in a formalized way, arguing that it tends to institutionalize the communities-of-practice from which competence emerges. The problem, therefore, is one of integration and not just a problem of combining, sharing or making data commonly available (Scarbrough, 1999).

Many recognize the separation of tacit understanding of individuals from the explicitly documented knowledge captured in formal systems, such as software process models and standards. The problem with seeing knowledge as only the explicit dimension is that much of organizational knowledge is not in its formal systems (the espoused theory) but in its beliefs, habits, routines, and memory of its members (the theory-in-use) (Argyris and Schön, 1996). This is why Choo (1998) encourages organizations to find ways to move the tacit to the explicit through sharing experiences, stories, reconfiguring existing knowledge and through internalization of the tacit by experience. Pourkomeylian (2001) suggests that externalization through socialization occurs in software process improvement, for instance through the creation of SPI plans and process descriptors. Whether it is possible to consciously share tacit knowledge in this way is debatable, but it recognizes the importance of the dynamic, social element in the sharing of knowledge. What it misses is the reflective nature of learning through action. Senge (1990) states that organizations are continuously changing through active implementation and reflection on their theory-in-use. Argyris and Schön (1996, p.16), likewise, state that:

individuals within an organization experience a problematic situation and inquire into it on an organization's behalf. They experience a surprising mismatch between expected and actual results of action and respond to that mismatch through a process of thought and further action that leads them to *modify their images of organization or their understanding of organizational phenomena and to restructure their activities so as to bring outcomes and expectations into line, thereby changing organizational theory-in-use.*

McGuire and Randall (1999), also, argue that the software development activities involve multiple iterations and feedback loops that result in a maturing sequence of mental models for the human actors. Process models and software engineering features can be understood as frameworks for learning (Mathiassen, 1998). These can be drawn on by the developers in sharing and adopting ideas. Mathiassen (1998) uses Schön's concept of reflection-in-action to show how software developers learn by making use of their past experiences and adapting these to fit the current situation. Reflection-in-action assists organizational actors to gain insight into the background context of the change, thus it helps to reshape and restructure the organizational context. Schön (1983) considers that we show ourselves to be knowledgeable through our intuitive actions and decisions in everyday life. Our knowing is often tacit, as seen in our patterns of action and feeling for the given activity. Schön (1983, p.49) therefore posits that 'our knowing is *in* our action'. So our work life depends upon our tacit knowing-in action: the ability to recognize phenomena, patterns and symptoms. So the know-how is in the action.

If we recognize knowing-in-action, Schön (1983) claims that we should also recognize that we sometimes think about what we are doing. This thinking implies that we consider what we are doing during the action so as to improvise and respond to the context. This 'reflection-in-action hinges on the experience of surprise' (Schön, 1983, p.56); when things please us or lead to undesired results we may respond by reflecting-in-action. At such moments, Schön considers that reflection interactively focuses on the intuitive knowing that is implicit in the action. A practitioner's reflection helps to surface and correct previous tacit understanding; they may reflect on the tacit norms and appreciations which support a decision. Practitioners also reflect on their knowing-in-practice through a post-activity review: they think back on a project. March and Olsen (1976, p.56) see this experiential learning as sensemaking, in which 'individuals and organizations make sense of their experience and modify behavior in terms of their interpretations.'

So, successful improvement involves learning, iteration and the emergence of new ideas, because processes 'are improvisational in that they combine real-time learning through design iterations and testing' (Eisenhardt and Tabuzi, 1995, p.108). The integration of any new innovation is not simple and requires intensive learning; the adoption should not be seen as a stand alone action (Lyytinen and Damsgaard, 2001). Lyytinen and Damsgaard (2001) also point out that the timescales in the implementation of any innovation are not necessarily short, so recognizing the time-based factors in the emergent change is necessary. So we need a context and process based model that helps us to understand this emergent change as occurring through reflective learning. It is therefore necessary to accommodate iterative experimentation, use, and learning in any model of change.

SPI AS SENSEMAKING

The challenge, then, is to understand process change as an emergent, active learning process developed from the relationship between people and their context. Orlikowski (1996, p.65) argues that a situated change perspective helps to explore the ongoing practices of organizational actors that emerge 'out of their (tacit and not so tacit) accommodations to and experiments with everyday contingencies, breakdown, exceptions, opportunities, and unintended consequences that they encounter'. Each action either changes or reproduces existing organizational properties and practices. As these amendments are sustained over time then fundamental changes occur:

The recurring story is one of autonomous initiatives that bubble up internally; continuous emergent change; steady learning from both failure and success; strategy implementation that is replaced by strategy making; the appearance of innovations that are unplanned, unforeseen, and unexpected; and small actions that have surprisingly large consequences. (Weick, 2000, p.225)

The concepts of situated change are founded on the social theories of sensemaking and improvisation. To engage in sensemaking is to frame the boundaries of the problem, and impose coherence upon it to allow us to decide how the situation needs to be changed. We are always in the middle of complex situations as there are no absolute starting points: sensemaking is an ongoing activity. In improvised change the solution applied is refined to fit the need on each occasion. To do this, organizations need to create new knowledge and to draw on the competencies available. Sensemaking helps us to understand situated, purposeful change in dynamic organizations in a different way to the deterministic view. Taking this perspective can help us to understand the unfolding changes in SPI through the actions of the human actors, whether intended or not. The intention of the key actors and their design for the change are important aspects of understanding the reasons for and the consequences of the changes.

Weick (1995) considers sensemaking to be the process of constructing an interpretation of the unknown by active agents. Whilst not all authors agree, he argues that action results from sensemaking, as interpretation is only part of sensemaking not synonymous with it: 'sensemaking is about authoring as well as reading' (Weick, 1995, p.7). It is an activity or process of invention not simply a discovery of 'the interpretation' (ibid). Individuals try to make sense of their experience. Even when the experience is complex and ambiguous they impose order, attribute meaning and provide explanations (March and Olsen, 1976).

Sensemaking is an ongoing activity. We are always in the middle of complex situations; there are no absolute starting points. Weick (1995) shows that problems do not present themselves as givens, but need to be constructed from complex, fuzzy, puzzling and uncertain situations. To engage in sensemaking is to frame the boundaries of the problem, and impose coherence upon it to allow us to decide in what direction the situation needs to be changed: There is a strong reflexive quality to this process. People make sense of things by seeing a world on which they have already imposed what they believe. People discover their own inventions, which is why sensemaking understood as invention, and interpretation understood as discovery, can be complementary ideas (Weick, 1995, p.15).

Interpretation and invention, or improvisation, are at different ends of a continuum (Weick, 2001). With improvisation greater modification of the original model is implied than with interpretation which is seen as just giving meaning to the original. Improvisation is a mixture of the pre-composed and the spontaneous (Weick, 2001). It does not materialize out of nothing: Ciborra (1999) points out that the improvisation is planned not haphazard, as people practice they develop the skills and understanding necessary. Weick (1995) develops the idea of enactment to show that we receive stimuli as a result of our own action. We do something (say to improve our context) which then helps us to do our job better so we continue to improve it. Sensemaking is, therefore, more than simply a cognitive process, but one that exists within the body. Mingers (2001, p.123) therefore argues that the physical embodiment of our action 'suggests that much that we "know", in the sense that we are able to undertake particular actions and activities, is essentially tacit, habitual, and beneath our consciousness'. Knowledge is therefore learnt through action by practice and habituation.
The integration of any new innovation is not simple and requires intensive learning; the adoption should not be seen as a stand alone action (Lyytinen and Damsgaard, 2001). Software process improvement technologies and methods require interpretation and will evolve through time (Swanson, 1994). Lyytinen and Damsgaard (2001) point out that technologies do not diffuse in a homogenous and fixed social ether but that a complex set of contextual constraints shape these changes. Thus, this paper explores how continuous change occurs by taking into account the contextually situated actions of the various actors and how these interacted to enable the improvement in the process.

RESEARCH METHODOLOGY

A single-case study was adopted for exploratory research and to develop theory. The case organization, InfoServ (a pseudonym), is a leading global information services company with over 13,000 employees and an annual turnover of £1.2 billion. This study focuses on the Market Analysis Package (MAP) software team based in the GeoMarketing (UK) division over a ten year period. The division combines data and software products for the market analysis purposes. The MAP product is the division's flagship product and directly contributes half of their sales. As the purpose of this study was to understand the process of improving the software process, the unit of analysis was taken as the continuous software process improvement activity rather than the organizational unit, allowing the case to be compared at a later date with another improvement initiative (Miles and Huberman, 1984).

The methodology for this study has been based on longitudinal study and processual analysis (e.g. Pettigrew, 1997). The overall stages of the research were based on Eisenhardt's (1989) roadmap of how to undertake case study research, thus enfolding the literature rather than being theory led. Both in-flight and historical data was captured through participant-observation for over a year enabling the development of a more holistic response to the events (Jogensen, 1989). In this case wide-ranging access was given to people and information. The role was undertaken overtly, also enabling open interviews to be conducted, attendance at meetings, access to a wide range of documents, notes to be taken immediately and interviews to be taped for later transcription. Two forms of interview were used: a set of 29 formal semi-structured interviews covering all the development staff within the software development team and their line management and a set of 27 review meetings software managers to discover their intentions for, and reactions to, product and process developments. Internal data collected by the software group was also acquired for use in the analysis of the perceived efficacy of the SPI activities.

The analysis was based upon a contextual, structurational framework (see Allison and Merali, 2003 for details). The concepts and themes developed were drawn from the data using an inductive approach. The findings in this paper were therefore grounded in the data and supported by the literature rather than been literature driven. To ensure that the account is plausible two forms of feedback have been sought: on the case accuracy and interpretations, and the framework and analysis. Informants have reviewed the accuracy of the case study data and agreed the interpretation of the data.

PROCESS EMERGENCE: A LEARNING PERSPECTIVE

The organization established an SPI program following two years' experience of using processes defined as part of an ISO 9000 initiative. These processes had been defined in a group quality manual. Over the two years from the definition of the process individual teams had returned to previous practice, had devised or adopted additional practices, and had adapted the processes in the manual to suit their needs. However, one key reason for initiating the SPI program was that a major release of a new package had significant defects. Here though we will focus on what happened after this point, by looking at how the processes changed and individuals learned through practice. The nature of process improvement at InfoServ has been shown to fall into three types of change: planned, improvised and unintentional (Allison and Merali, 2006).

A Brief Synopsis

Planned changes occurred through action teams following an initial SPI meeting to identify areas for change, such as the need to introduce an automated testing tool. Improvisation came about through personal desire to change aspects of the development process. This happened when individuals saw an opportunity to try something different and so would experiment with the idea to test it and to communicate the benefits to others. Additionally, through ongoing practice changes to the process would 'bubble-up', often unintentionally, as people learned to do things differently. Small adjustments were made as a consequence of the ongoing interpretation and application of practiced processes to develop new software products. They occur as a consequence of the ongoing interpretation and application of practiced processes to develop new software products. So, whilst the actor is conscious of their action, they do not design the action to change the process. Innovations in the software process were based on the reflective considerations of the individuals involved during the practice of developing the software.

The analysis of the case illustrated that the learning occurs through reflection-in-action during the enactment of the software process. Individuals learn, share lessons and draw on this shared knowledge to undertake the process of developing products. Figure 1 identifies separate elements of the learning activity to be discussed. This is not to imply that these elements can be separated other than for analytical purposes, rather that learning and practice occur together, synergistically informing each other. The model reflects the dualistic view of action and context, where the individual draws on the organizational context to undertake their practice and through that practice changes the context. The discussion highlights the linkage between the two facets.

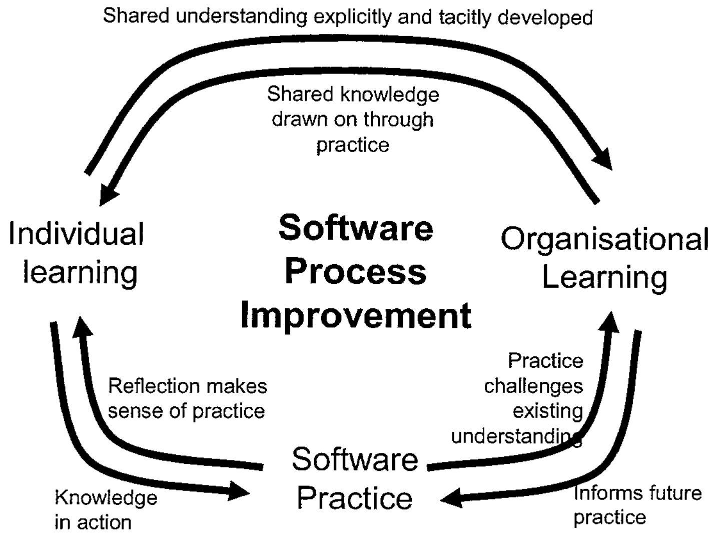
Shared understanding explicitly and tacitly developed
Shared knowledge drawn on through practice
Individual learning
Software Process Improvement
Organisational Learning
Reflection makes sense of practice
Practice challenges existing understanding
Software Practice
Knowledge in action
Informs future practice

Figure 1: SPI as learning

As discussed above, sensemaking is considered to be more than a cognitive process, but rather it is embodied in our physical action: actors improve their software process which helps to improve the product development, and so continue to refine their process. Actors therefore learn through action, bringing prior experience and conceptual knowledge from other sources to support their action. This learning is akin to the model of learning through use developed by Daft and Weick (1984), and Kolb's (1984) experiential learning model.

Learning to change is often taken to be a transfer of information from those who know to those who do not. This 'reifies knowledge and de-emphasizes its collective nature as well as social processes of knowing' (Swan and Newell, 2000, p.592). This view implies that prior knowledge is the only thing that is important, which tends to reinforce what is already done in the organization. MacDonald (1998, p.40) argues, on the contrary, that this knowledge sharing is 'a process which cannot be directed and controlled'. At InfoServ their local adoption was influenced by the context, the desire of the developer, and the perceived relevance of the innovation for the product development. The assimilation and application of ideas in the case, whether from internal or external sources, were seen to be related to the perceived value of the method or idea. The value related to their level of trust of the sources as well as the perceived usefulness or applicability of the technique.

At InfoServ, product and process innovation occurred through individual experimentation. One example was the creation of a component from the MAP software that encouraged the software management to explore the adoption of component-based development. More specifically in the area of process innovation the adoption of both the Critical Chain planning and test estimation approaches followed external training. In each case these ideas were assimilated through an individual who was able to identify how these approaches would help to resolve existing issues that they were facing. The techniques were introduced through improvisation and experimentation: each idea was reinvented for the needs of the team, debated with other members of their community-of-practice before exposing it to members of the wider team, and then evaluated in given instances to test its usefulness. The willingness to put their own resources into trying out new ideas combined with the creativeness to achieve a useful result were critical personal attributes. The experimentation often allowed the individual to explore an idea without the pressure to succeed: the result could be thrown away if it did not work out. Experimentation can be encouraged, but the instances observed did not result from any organizational scheme but from individual capability and desire. In both the examples above it was the motive to change the current approach that encouraged them to put their own time into the experiment. This type of successful assimilation reflects Ciborra's (1999, p.137) understanding of smart improvisation as an action 'which contributes to the individual or organizational effectiveness' rather than that of a novice or someone just acting extemporaneously which has no effective link with the demands of the situation.

Formal training and development programs are not required to enable external ideas to be incorporated into an organization. Merali (2002) shows that individuals who are part of a fluid, external network of professionals will reflect on those interactions and seek to apply ideas appropriate to the perceived needs of their own software processes and products. The danger is when external innovations are introduced en-masse because they have worked elsewhere. Lefebvre et al (1995) show that the level of benefits derived from the adoption of new ideas is dependent upon the level of penetration of those approaches within the organization, which requires that the solution applied is refined to fit the need on each occasion. This is a form of bricolage; that is, it makes use of whatever resources and skills are at hand, and therefore contrasts to the engineering view of change where projects are not started until all resources are identified and available (Weick, 2001).

Bricolage encourages the use of existing tools and routines by people at the operational level to solve new problems. Local cues are used to obtain ad hoc solutions that bubble-up serendipitously from the normal daily activity. Thus, these changes 'emerge from the enactment and reinforcement of local innovation' (Ciborra, 1994, p.16). However, this is not to suggest that the development of the process or product is entirely random. By combining the approaches of bricoleurs with those of engineers there are no limits to the possible implementations of software engineering ideas (Dahlbom and Mathiassen, 1995).

Ciborra (1994) finds that only by encouraging improvisation through tinkering, or bricolage, and having a willingness to fail will innovation occur. When new ideas were introduced at InfoServ not everyone fully understood them, not even those who were introducing the idea, but people were willing to experiment with ideas to see if they had value. The changes to the software process therefore continued to emerge as individuals considered how their actions were supporting the development of the software products.

Individuals' mental models matured through the enactment of the process, as highlighted by the technical architect, who recognized that he had a maturing view of design that took into account perceived mistakes as well as successes: 'I know a lot more about designing large, relatively large software projects in C++ than I did before doing this one. There is a whole host of design approaches which I would no longer take (sic.)'. This maturing of understanding was evident across the team. Individual learning is linked to the organizational learning that takes place as a result of the organization's experience with any innovation (Lefebvre et al, 1995).

Such organizational learning is more than simply documenting new processes but is about improving the knowledge of members of the organization by sharing experiences. Indeed, Conradi and Fuggetta (2002) found that developers considered formally documenting processes ineffective in transferring knowledge. So whilst, software process models documented in manuals can be changed to reflect current practice, to incorporate new ideas from outside or in an attempt to develop a more mature process, such manuals at best only reflect the desired practice of a software development group. Truex et al (2000) suggest that methods are discarded early in the development process and practice is often inconsistent with the defined methodology. So, even if we assume that teams attempt to instantiate the process as defined in a particular project, by looking at the defined processes our understanding of the process is limited to the espoused theory.

Argyris and Schön (1996) show that individuals maintain their own theory-in-use by interpreting the espoused theory and other previous experience. It is the theory-in-use that the individual draws on to respond to the particular problem faced. We therefore cannot assume that the espoused theory is the same as the theory-in-use, and that it is the same as the action within the development activity. However, it is as the theory-in-use changes and becomes the norm that the espoused theory also changes, in the way that the defined processes were rewritten or communicated to new members of the team, reflecting Argyris and Schön's (1996) concept of double loop learning.

Sharing understanding can therefore be supported through process documentation but this has a limited role in communicating understanding. We therefore need to recognize that knowledge emerges from 'patterns of social relations and dynamic practice' (Scarbrough, 1998, p.227). Knowledge can be viewed as an emergent property of organizational interaction with the wider environment, and in terms of social practice and relations. As organizations learn through practice the norms of their communities-of-practice emerge. As practice unfolds it challenges those norms and refines them through individual learning, and then future practice (and related process change) is informed by the norms as individuals draw on them to sanction actions.

These norms can be seen in terms of individuals learning to routinely follow the process. Whilst there was some resistance to following the processes as defined, as the actual processes used became the norm people knew what was expected of them and therefore as one developer put it: 'it gets instinctive...[and so they] follow [the] method...despite themselves'. The application of object oriented programming was a good example of this routinization.

The move to a C++ development environment was difficult because knowledge and skills in this area were in short supply. The level of prior understanding varied and therefore some developers found the abstraction and encapsulation discipline difficult, sometimes returning to old habits. However, through repeated use, listening to others in design meetings and subsequent inspection meetings, and both formal and informal training their personal understanding developed, albeit to a varying extent. The improvement was evident to those looking back at software developed at different times.

So as ideas are shared across a group, tacitly and explicitly, the degree of systemness increases within the communities-of-practice through shared meaning. Giddens (1999) defines social systems as the reproduced relations between actors or collectives, organized as regular social practices. These collectives are not unified, but draw on commonly understood rules to communicate and act. So there is a greater ability to draw on similar rules as ideas are shared across communities-of-practice. This shared knowledge resides with individuals, but the team's capability increases through an improved ability to access knowledge held within the community and a greater understanding of its relevance to their needs. As this capability improves, so the team's agility to respond to development triggers improves. As norms are drawn on within the practice, then further experience will challenge the understanding embedded in those norms – at the individual level but through discussion and negotiation the norms will evolve – and those norms will be used to inform participants' future practice. The case showed that sharing ideas to develop group solutions is difficult, as individual motives tend to engender a personal perspective. Trust between the participants enables this sharing and learning to happen.

CONCLUSION

Previously, SPI has been mainly understood as something to be engineered. Even continuous process improvement approaches, such as IDEAL, take little account of the organizational context. Thus, if we are to understand how continuous change occurs we need to take into account the organizational context it is occurring in, the actions of the various actors and how these interact to enable the learning and knowledge creation said to be required to improve the process.

The improvement at InfoServ was shown to be a process of emergent change. Reflection by developers during their development tasks informed their process improvement. The changes were derived from innovations they felt to be directly relevant to their work, and would address problems in the software or the life cycle. Ideas that were tried out as experiments to address a particular need during a development activity were communicated to other developers through a shared experience of the product development.

Future work must therefore continue to develop this understanding of software process improvement as emergent change. To do so it will be necessary to develop a stronger theoretical framework to understand how this change happens so as to draw out the different features of the change in a more precise and theoretically informed way. To achieve this further empirical data will be required. The lessons from this study also need to be taken on in terms of what the impact of a more emergent, agile perspective of SPI would mean for practice and how this can be best supported (Allison, 2005). An agile perspective that highlights the need to learn to improve through situated practice within an organizational framework would support the ongoing needs of the business, reflecting the learning intensive nature of SPI.

REFERENCES

Aaen, I., Arent, J., Mathiassen, L. & Ngwenyama, O. (2002), Mapping SPI Ideas and Practices, In: Mathiassen, L. Pries-Heje, J. & Ngwenyama, O. (eds.) Improving Software Organizations, 23-46, Upper Saddle River, NJ: Addison-Wesley.

Allison, I. & Merali, Y. (2003) Software Process Improvement: Towards an Emergent Perspective, In: Levy, M. Martin, A. & Schweighart, C. (eds.), Proceedings of the 8th UKAIS Conference, 9 – 11th April, University of Warwick.

Allison, I. & Merali, Y. (2006) Intra-team Software Process Emergence: Resilence Through Improvisation, In: Hapeshi, K. & Tomlinson, A. (eds.), Proceedings of the 11th UKAIS Conference, 10– 11th April, University of Gloucestershire.

Allison, I. (1999) Information Systems Professional Development: A Work-Based Learning Model, *Journal of Continuing Professional Development*, 2(3), 86-92.

Allison, I. (2004) Software Process Improvement as Emergent Change: a Structurational Analysis, *PhD Thesis*, University of Warwick.

Argyris, C. & Schön, D.A. (1996) Organizational Learning II: Theory, Method and Practice, Reading: MA : Addison-Wesley.

Bollinger, T. B. & McGowan, C. (1991) A Critical Look at Software Capability Evaluations, *IEEE Software*, 8(4), 25-41.

Choo, C.W. (1998) The Knowing Organization: How Organizations Use Information To Construct Meaning, Create Knowledge and Make Decisions, New York: Oxford University Press.

Ciborra, C. (1994) The Grassroots of IT and Strategy, In: Ciborra, C. & Jelassi, T. (eds) Strategic Information Systems: a European Perspective, 3-24, Chichester: John Wiley & Sons.

Ciborra, C.U. (1999) A Theory of Information Systems Based on Improvisation, In: Currie, W.L. and Galliers, R. (eds.), Rethinking Management Information Systems, 136-155, Oxford: Oxford University Press.

Conradi, R.& Fugetta, A. (2002) Improving Software Process Improvement, *IEEE Software*, 19(4), 92-99.

Daft, R.L. & Weick, K.E. Toward a Model of Organizations as Interpretation Systems, *Academy of Management Review*, 9(2), 284-295.

Dahlbom, B. & Mathiassen, L. (1995) Computers in Context: the Philosophy and Practice of Systems Design, Cambridge, MA: Oxford : NCC Blackwell.

Eisenhardt, K.M. (1989) Building Theories from Case Study Research, *Academy of Management Review*, 14(4), 532-550.

Fichman, R.G. & Kemerer, C.F. (1997) The Assimilation of Software Process Innovations: an Organizational Learning Perspective, *Management Science*, 43(10), 1345-1363.

Gasston, J. & Halloran, P. (1999) Continuous Software Process Improvement Requires Organizational Learning: An Australian Case Study, *Software Quality Journal*, 8(1), 37-51.

Giddens, A. (1999) Elements of the Theory of Structuration, In: Elliott, A. (ed.) The Blackwell Reader in Contemporary Social Theory, 119-130, Oxford: Blackwell Publishers Ltd.

Gray, E.M. & Smith, W.L. (1998) On the Limitations of Software Process Assessment and the Recognition of a Required Re-orientation for Global Process Improvement, *Software Quality Journal*, 7(1), 21-34.

Hansen, B.H., Rose, J. & Tjornehoj, G. (2004) Prescription, description, reflection: the shape of the software process improvement field, *International Journal of Information Management*, 24, 457-472.

Herbselb, J., Zubrow, D., Goldenson, D., Hayes, W., & Paulk, M. (1997) Software Quality and the Capability Maturity Model, *Communications of the ACM*, 40(6), 30-40.

Hollenbach, C., Young, R., Pflugrad, A. & Smith, D. (1997) Combining Quality and Software Improvement, *Communications of the ACM*, 40(6) 41-45.

Jorgensen, D.L. (1989) Participant Observation. Newbury Park, CA: Sage Publications Inc.

Kolb, D.A. (1984) Experiential Learning, Englewood Cliffs, NJ: Prentice Hall.

Lefebvre, E, Lefebvre, L A, & Roy, M (1995) Technological penetration and organizational learning in SMEs: The cumulative effect, *Technovation,* 15 (8), 511-522.

Lyytinen, K. & Damsgaard, J. (2001) What's Wrong with the Diffusion of Innovation Theory? In: Ardis , M.A and Marcolin, B.L. (eds.) Diffusing Software Products and Process Innovations (IFIP TC8 WG8.6 Fourth Working Conference, 173-204, April 7-10th, Banff, Canada), Norwell, MA: Kluwer Academic Publishers.

Lyytinen, K. & Robey, D. (1999) Learning Failure In Information Systems Development, *Information Systems Journal*, 9, 85-101.

MacDonald, S. (1998) Information For Innovation: Managing Change Form An Information Perspective, Oxford: Oxford University Press.

March, J.G. & Olsen, J.P. (1976) Ambiguity and Choice in Organizations, Olso, Norway: Scandinavian University Press.

Mathiassen, L. (1998) Reflective Systems Development, Online at http://www.cs.auc.dk/~larsm/rsd.html, Accessed on 5/12/2001.

Mathiassen, L., Pries-Heje, J. & Ngwenyama, O. (2002) Improving Software Organizations: from principles to practice, Boston, MA: Addison-Wesley.

McFeeley,B (1996) IDEAL: A User's Guide For Software Process Improvement (CMU/SEI-96-HB-001), Pittsburgh, PA: Software Engineering Institute/ Carnegie Mellon University.

Merali, Y. (2002) The Role of Boundaries in Knowledge Processes, *European Journal of Information Systems*, 11(1), 47-60

Miles, M.B. & Huberman, A.M. (1994) Qualitative Data Analysis: An Expanded Sourcebook (2nd ed.). Thousand Oaks, CA : Sage Publications, Inc..

Mingers, J. (2001) Embodying information systems: the contribution of phenomenology, *Information and Organization*, 11, 103-128

Nonaka, I. & Takeuchi, H. (1995) The Knowledge-Creating Company, New York: Oxford University Press.

Orlikowski, W.J. (1996) Improvising Organizational Transformation Over Time: A Situated Change Perspective, *Information Systems Research*, 7(1), 63-92.

Pettigrew, A.M. (1997) What is Processual Analysis? *Scandinavian Journal of Management*, 13(4), 337-348.

Pourkomeylian, P. (2001) Knowledge Creation in Improving a Software Organisation, In: Ardis, M. & Marcolin, B. (eds.) IFIP TC8 WG8.6 Fourth Conference, 205-224, Norwell, MA: Kluwer Academic Publishers.

Ravichandran, T. & Rai, A. (2000) Software Process Management: An Organizational Learning Perspective, In: Hansen, H.R., Bichler, M., & Mahrer, H. (eds.) Proceedings of the 8th European Conference on

Information Systems, 202-209, 3rd-5th July, Vienna, Austria: Vienna University of Economics and Business Administration.

Scarbrough, H. (1998) Path(ological) Dependency? Core Competencies from an Organizational Perspective, *British Journal of Management*, 9(3), 219-232.

Scarbrough, H. (1999) The Management of Knowledge Workers, In: Currie, W.L. and Galliers, R. (eds.), Rethinking Management Information Systems, 474-496, Oxford : Oxford University Press.

Schön, D.A. (1983) The Reflective Practitioner: How Professionals Think In Action, New York: Basic Books.

Suchman, L.A. (1987) Plans and Situated Actions: the Problem Of Human-Machine Communication, Cambridge: Cambridge University Press.

Swan, J. A. & Newell, S. (2000). Linking Knowledge Management and Innovation, In: Hansen, H.R., Bichler, M., and Mahrer, H. (eds.) *Proceedings of the 8th European Conference on Information Systems*, 591-598, 3rd-5th July, Vienna, Austria: Vienna University of Economics and Business Administration.

Swanson, E.B. (1994) Information Systems Innovation among Organizations, *Management Science*, 40(9), 1069-1092.

Truex, D., Baskerville, R. & Travis, J. (2000) Amethodical Systems Development: The Deferred Meaning Of Systems Development Methods, *Accounting, Management, and Information Technology*, 10, 53-79.

Weick, K.E. (1995) Sensemaking in Organizations, Thousand Oaks, CA: Sage Publications Inc.

Weick, K.E. (2000) Emergent Change as a Universal in organizations, In: Beer, M. and Nohria, N. (eds.) *Breaking the code of change*, 223-242, Boston, MA: Harvard Business School Press.

Weick, K.E. (2001) Making Sense of the Organization, Oxford: Blackwell Publishers.