Communications of the IIMA

Volume 5 | Issue 1 Article 8

2005

Towards an Agile Approach to Software Process Improvement: Addressing the Changing Needs of Software Products

Ian Allison
Nottingham Trent University

Follow this and additional works at: http://scholarworks.lib.csusb.edu/ciima

Part of the Management Information Systems Commons

Recommended Citation

Allison, Ian (2005) "Towards an Agile Approach to Software Process Improvement: Addressing the Changing Needs of Software Products," *Communications of the IIMA*: Vol. 5: Iss. 1, Article 8.
Available at: http://scholarworks.lib.csusb.edu/ciima/vol5/iss1/8

This Article is brought to you for free and open access by CSUSB ScholarWorks. It has been accepted for inclusion in Communications of the IIMA by an authorized administrator of CSUSB ScholarWorks. For more information, please contact scholarworks@csusb.edu.

Towards an Agile Approach to Software Process Improvement: Addressing the Changing Needs of Software Products

Ian Allison
Nottingham Trent University, Burton Street, Nottingham, NG1 4BU, UK
Phone:(44+) 115 8482273 Fax: (44+) 115 8486518 E-mail: ian.allison@ntu.ac.uk

ABSTRACT

This paper highlights the need for greater agility and flexibility in the process improvement activity. The ideas presented here relate to aspects that have arisen from a case study of a global software product organisation. A model of how software process improvement can be agile in nature is proposed. This model is intended as indicative of the issues that need consideration by IS management. The model highlights the need to learn to improve through situated practice within an organisational framework that supports the needs of the business.

Key words: Software Process Improvement; Agile Software Development; Emergent Process; Situated Practice.

INTRODUCTION

The packaged software market is a considerable global industry. Despite the significance of this market, there is a tendency in the information systems (IS) and software engineering literature to focus upon the development of software within internal IS functions or the producers of technical and systems software. These software organisations face rapid changes in their markets, with globalisation of the economy, changes in the competition through deregulation and pressures from customers with increasing demands. Packaged software producers differ to internal IS functions. Packaged software firms function under an intense time to market pressure relative to internal IS functions (Carmel & Sawyer, 1998). They are under pressure to innovate, and upgrade their existing product set. These time pressures are increasing with release cycles reducing in time. To prosper in the face of these pressures software package organisations need to become flexible and highly responsive. The field has begun to develop methods and tools to support this responsiveness. Organisations also need to produce a quality product otherwise the commercial benefits are lost. However, there is a paucity of work looking at how software quality can also be responsive to the context.

It is widely recognised that the quality of software is related to the processes used to create the software (e.g. Paulk, Weber, Curtis, & Chrissis, 1995). In an attempt to create a sustained approach to improving the quality of the software processes used by an organisation the practice of software process improvement was developed. Software process improvement (SPI) facilitates the identification and application of changes to the development and management activities in order to improve the product. Much of the current understanding of software process improvement has been derived from the work of the Software Engineering Institute (SEI).

Unlike the software engineering literature that tends to be restricted to a rational, deterministic view of change, here the ongoing nature of the software processes is placed at the heart of the improvement. By looking at SPI as an emergent rather than deterministic activity the design and action of the change process are seen to be intertwined and shaped by their context (Allison & Merali, 2003; Mathiassen, Pries-Heje, & Ngwenyama, 2002). Building on this understanding, this paper proposes that the process improvement activity should also be contextually responsive rather than externally pre-determined.

SOFTWARE PROCESS IMPROVEMENT

One attempt to address the problems in software quality has been the development of process maturity models for assessing the current capability, and directing the improvement in defined stages. This normative, deterministic perspective of improving the activity of software development has dominated the software process improvement literature. The purpose of these models is to provide a benchmark against which to assess organisational competence, and then to identify the stages for improvement. Software process improvement, though, is a complex activity. So whilst it has been shown that the application of the process maturity models can bring significant business benefit, criticism of these models shows that they rigidly state the practice to be adopted and that organisations face difficulty in implementing the prescribed actions (Bollinger & McGowan, 1991; Gray & Smith, 1998; Goldenson & Herbselb, 1995; Ravichandran & Rai, 2000a). Gray & Smith (1998) criticise any normative maturity models as being synthetic in nature as they are based on an ideal organisation that never existed. Critics of process maturity models suggest that whilst they represent a good initial attempt to improve process capability and provide a useful reference point that they should not be used prescriptively. Organisations, however, tend to try and adopt the key process areas in a mechanistic fashion and consequently have problems following the recommended sequence (Card, 1991).

Consequently, Curtis (2000) states that although many software development organisations learn how to implement successful process improvement programmes others struggle through without making lasting improvements. So whilst there are a number of spectacular results from some case studies the business benefit of SPI in general is questioned, even when apparent progress is made through the maturity levels. The focus on improving the process emphasises the internal activity over the external competitive position (Jung, Hunter, Goldenson, & El-Emam, 2001). In the SPICE (ISO 15504) trials approximately 60 percent did not observe a major impact on the organisation (reported in Conradi & Fuggetta, 2002). The concern is that the process has become more important than the quality of the product or any resultant business benefit.

This internal focus is an untenable position for commercial software producers, where reacting to the market is seen as more important than following a prescribed process. Software production is operating in a turbulent technical and commercial environment, where the competition and technology are fast changing, so there is uncertainty about the software products (Mellis, 2001). Despite the documented benefits some leading commercial software producers do not, therefore, follow software process improvement according to the maturity models or quality standards (Conradi & Fuggetta, 2002). The aim of the process oriented paradigm is a stable and reliable software process, as a result of software process improvement based on rigorous project management and process control. The idea of a stable, repeatable software process contrasts with the environment of commercial software product development. Microsoft has found the need to balance autonomy for development teams within an overall framework that enables the synchronisation and stabilisation of the software (Cusumano, 1997). Fast, innovative development is required compared with rigorous and slow, which makes stability a risk to competitive potential. So, 'it is from the vantage point of innovator that CMM seems most lost' (Bach, 1994, p.16), as Card (1991, p.103) puts it: 'maturity is necessary but not sufficient for true improvement', as the real prize is increased competitiveness not a better maturity assessment score.

In order to manage the software development process within the dynamic technical and business environment, the software industry must continuously evolve and improve its software development practices. Recent research and practice have recognised that software process improvement needs a long-term view and 'must be considered a repeating cycle' (Curtis & Paulk, 1993, p.386). From this perspective many authors have argued for a continuous software process improvement process (e.g. Briand, El Emam, & Melo, 1999; Birk & Rombach, 2001). The primary emphasis in the literature has been on developing frameworks to support the 'how to' of continuous improvement. A principal example of such models is the IDEAL model created by the SEI as an iterative model of SPI to complement the CMM (McFeeley, 1996). This literature has developed alongside, and in support of, the deterministic view that the normative models bring. The research in this area has recognised the difficulties of managing the introduction of new processes but provides little understanding of these difficulties.

The problem is that much of the process modelling and assessment work has forgotten that software is the product of people not of a conceptualised process. Hosking & Anderson (1992, p.6) argue that the:

> *...ability to instigate, plan and direct all forms of change has been taken for granted, creating a present-day misinterpretation appropriately termed 'the illusion of manageability'. By this we mean that the extent to which managers are able to direct or predict change processes is overestimated, and the way in which they might contribute to change is fundamentally misconceived.*

Indeed, they state that there is considerable evidence to suggest that change is far from controllable and can only be influenced to a limited extent; the intended purpose of the intervention is often overcome by unexpected or unintended outcomes. So the challenge is to understand how SPI can be undertaken not as a predictable or designed causal outcome, but as an emergent process developed from the relationship between people and their context. To further our understanding of software process improvement, it is necessary therefore to move beyond deterministic perspectives of technological and organisational change. What is required is an understanding of change that reflects a more complex, dynamic and unpredictable world.

An action based view of knowledge leads us into seeing 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. Thus, if we are to understand how continuous change occurs we need to take into account the organisational 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. A provisional model is proposed to inform IS management of the issues, reflecting an understanding of SPI as an emergent activity.

INFOSERV - A CASE VIGNETTE

InfoServ (a pseudonym) is a leading global information services company operating in over 60 countries. By 2003 the company had over 13,000 employees worldwide with an annual turnover of £1.2 billion, indicating the strength of the company and its ability to adapt to the demands of its market. A division in their UK headquarters develop a number of software packages, with one principal global market analysis product. A longitudinal case study was undertaken that established how and why the software processes changed over a ten year period, with a specific focus on a two year SPI project (Allison, 2005). The process improvement initiative was in the software development unit associated with this market analysis package. A summary of the case is given here to set the context for the proposed agile model, rather than a full case study analysis.

Within the SPI activity at InfoServ it was recognised that the changes were not all based on pre-planned solutions. Indeed, the maturity models played no part in the definition of the process changes. Planned actions were identified according to perceived needs, but often these were changed or abandoned in favour of improvements identified through practice. The actions and outcomes from the formal SPI project were only part of the story, with changes occurring through the ongoing practice and improvisations of the practitioners. The innovations in the software process were primarily based on the reflective considerations of the individuals involved during the practice of developing the software. Some ideas were externally derived but even these initiatives were directly related to the needs identified at the project level. This view of innovations contrasts with much of the traditional SPI literature, and with aspects of recent studies such as the diffusion of innovation literature, which assumes that ideas are mostly externally generated.

The process of improvement needs to account for these reactive, reflective changes if the processes are to be improved not just extemporised. One of the difficulties at InfoServ was the uncoordinated manner in which they allowed the SPI project to progress. The sanctioning of the tasks through management channels was sometimes insufficient to overcome other constraining factors in the context. The infrastructure at InfoServ was minimal, but further support mechanisms might have helped to maintain and coordinate the actions.

There is a need to promote sustainable development of the processes by integrating the experiences of the developers, their learning through action, and sharing that learning. The learning processes that informed the SPI activity were ongoing, not simply delivered via training. Training was seen to assist in the identification of suitable innovations, but not all initiatives from training were incorporated. Rather it was when a need was clearly answered, often serendipitously, within a training event that it was incorporated into the practice.

At InfoServ, this dissemination of ideas occurred through ongoing negotiation. In this sense then, conflict should not be regarded as a negative aspect of organisational life, rather debate should be encouraged. For the debate to engender a common feeling of improvement, trust between the actors is important.

Involvement of all the developers was a fundamental strand of the philosophy of the SPI initiative at InfoServ. This involvement was an attempt to develop a spirit of togetherness, reflecting the literature that suggests this is helpful. Here examples were evident of how individuals, and sub-cultures, both enabled and resisted changes to the processes through the continuous exercising of power. Power should not necessarily be understood negatively or as solely in the hands of those in authority, but as a means used by the individual seeking to fulfil their self-interest. It is enabling and productive as well as constraining; it is what enables the existence of different positions within the organisation. From this perspective then power is part of all organisational relations and is the process through which organisations are sustained, reproduced and changed.

The norms of the organisation, as understood at local levels of communities-of-practice, shaped the retention of existing ideas or the introduction of new ideas. These habits and traditions were drawn on by agents to sanction their actions. However, actors were not passively moulded by their culture, fitting with Giddens (1984) view that agency takes place with knowledge and practical consciousness. The process-in-use within any community-of-practice acted as a norm: informing, guiding, and organising future practice. Such norms thereby sustained existing approaches.

The norms changed as they were challenged through experience, negotiation within the group, and through the introduction of new ideas from other sources.

Within packaged software organisations product quality and customer delivery on schedule are more important than following a prescribed process. At InfoServ there were a number of key examples of process changes that occurred because they were important for the business, but this link was not explicitly made, and therefore not measured, by the organisation. By seeking process changes that enable an organisation to achieve good quality product development rather than process rigidity per se, key business measures should improve. The judgement of this improvement can be achieved through client-based assessment, reflecting the value of the improvement to them (Harkness, Kettinger, & Segers, 1996).

Changes in the process-in-use at InfoServ were seen to occur through different forms of innovation, reflecting Brown & Eisenhardt's (1997) proposal that successful innovation involves improvisation, communication, experimentation, and choreographed transitions. Finding a way to facilitate this level of inventiveness within the software process is an important lesson from this case study. The theoretical development provides a step towards that understanding through the recognition of the situated nature of the improvement. The following section will pick up and develop the implications from the case to suggest the how this might be taken forward to inform future practice.

AN AGILE PERSPECTIVE OF SPI

The lessons discussed above from the case resonate with the contemporary views of agile software development, where evolution and appropriateness are seen to be paramount. The purpose of this section is therefore to explore how these agile concepts can inform the SPI practice. This section contributes to this development by linking agile methods with the SPI concepts. The suggestions will link findings from the study to other recent developments in the SPI literature that have adopted a similar position.

The agile software development manifesto (see Cockburn, 2002, p.213, bold in original) highlights four values:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

Whilst the items on the right have value, the authors of the manifesto value the items on the left more. Similarly for process improvement, it is argued that the SPI activity should be agile rather than following deterministic and normative based approaches.

An agile approach to SPI would be responsive and flexible to local needs, encourage innovation in the process, build SPI projects around those who are motivated, encourage self-organising competent teams, and promote sustainable development of the processes. Figure1 conceptualises the features considered necessary to support an agile approach to SPI. Each of these features is discussed below by drawing on the existing literature and the findings from the case. The basis for these discussions is that, rather than continuing the quest to find a perfect software process, researchers need to recognise that 'every individual software organisation must develop its own practices of continuous self-improvement and establish an appropriate improvement infrastructure' (Birk & Rombach, 2001, pp.34-35).

This discussion is not intended as another prescriptive model but to highlight aspects and issues that need consideration. Above all, agility in the process is an attitude of mind not just another formula to follow. Nor is the purpose of this model to replace existing continuous improvement approaches, but to complement them. Indeed, the ideas encapsulated within the discussion reflect the principal change management factors that have been identified as important in successful SPI projects (Humphrey, 1989; Curtis & Paulk, 1993; Zahran, 1998). These factors (summarised in Table 1) are integral to the agile approach discussed below, which addresses each aspect of the above model.

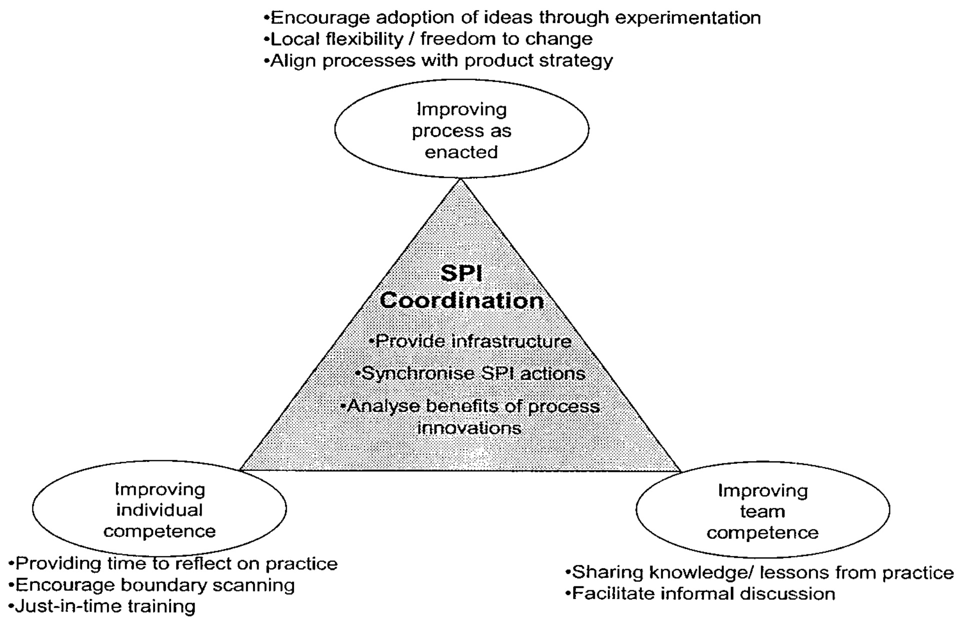
•Encourage adoption of ideas through experimentation
•Local flexibility / freedom to change
•Align processes with product strategy
Improving process as enacted
SPI Coordination
•Provide infrastructure
•Synchronise SPI actions
•Analyse benefits of process innovations
Improving individual competence
•Providing time to reflect on practice
•Encourage boundary scanning
•Just-in-time training
Improving team competence
•Sharing knowledge/ lessons from practice
•Facilitate informal discussion

Figure 1: A model of an agile SPI method.

Improving processes as enacted
Improvement in process management is 'a continuous effort to design an efficacious process by understanding the relationship between process configurations and process outcomes and embedding the knowledge in the process through routines and formal process definitions' (Ravichandran & Rai, 2000b, p.202). In the context of a package software organisation the outcome of the process is directly associated with its product base. However, the danger is that because of commercial pressures organisations 'prevent, hamper, and even abort well-planned, low-key SPI efforts. Further, many of the cited SPI frameworks are hardly suited or applicable in their present forms' (Conradi & Fuggetta, 2002, p.93). The software engineering community therefore need to understand and develop improvement approaches that have constant change at the core.

Factors in ensuring a successful SPI programme
Changes must be aligned with business strategy and priorities should be based on the business context
Commitment to improve and investment required
must come from the top
SPI requires the consensus of all stakeholders and they need to have ownership of the change
Current processes must be understood
Change must be continuous
Improvement infrastructure is required to identify and manage the implementation of changes.
The results of the SPI should be monitored and lessons learned.

Table 1: Factors in ensuring a successful SPI programme.

'The real issue is product innovation not process stabilization and refinement' (Conradi & Fuggetta, 2002, p.95). To be responsive and flexible to local needs process innovation needs to be aligned to the business needs. This does not imply that the process improvement activity should be 'driven' by the business strategy but that changes in the software process should be informed by the planned product base. The need to focus on product improvement as well as process improvement has recently been recognised as essential by one organisation reaching CMM level 5 (McGarry & Decker, 2002), but in a product based organisation this link is fundamental to the successful improvement of the process from the outset.

Nonaka & Takeuchi (1995) state that organisations must maintain a highly adaptive and flexible approach to new product development because it rarely proceeds in a linear or static manner and instead involves an iterative, dynamic and continuous process of trial and error. Project teams can be organised in organic structures that grow and shrink according to the needs of the product under development. Ayas (1996) shows that self-managed teams, organised into a dynamic structure that changes with the need of the product, share knowledge through association with different colleagues. Organisations should be willing to give a high degree of autonomy to such teams especially as they cope with ambiguity, fluctuation and creative chaos in the start up phases of new product development where little prior knowledge exists and taking initiatives and risks are necessary. The concept of self-organising, however, raises the question of who is responsible for the decisions about resource priorities and how to avoid an idiosyncratic approach to the process improvement, especially in the intrinsic political nature of the SPI activity. This aspect will be discussed further in the coordination section below.

Explicitly moving away from a software process philosophy of stability to one that is adaptive and flexible to local concerns will encourage a change in the actors' intentions, as the norms of the organisation will change from remaining consistent to finding the most appropriate means of undertaking a task. This is not to suggest that methodologies or process life-cycles should be abandoned, rather the aim should be to produce a successful product rather than following a pre-defined methodology at all costs. Indeed, the objective is not change for its own sake, as there is a need to balance between optimising the current approaches and experimenting with new ideas (Van Solingen, 2004). Yet, we have to move away from the belief that every instance of a process will be the same, towards recognition that individuals should be the judges of what is good practice.

The primary means for adjusting the process to suit the product development needs will be through learning from practice as individuals identify adjustments in the life cycle. Gasston & Hallorn (1999) advocate that, through leadership, norms should be established that will encourage learning. This view concurs with that of Conradi & Fuggetta (2002) who contend that SPI is about learning not control. Nevertheless, the reasons for adapting the defined process in each case should be understood so as to enable consideration as to whether the adaptation is just for this instance or a lesson for future projects. Learning from the action of developing the products will encourage the innovations in the process to be based on practice not just introduced in isolation. In part this implies we need to capture the existing practice and share it, but also teams should seek out ideas that are appropriate to its needs and avoid simply introducing a new technique because it is available. Much of this innovation capability lies with the individuals and is discussed below, but aspects can be encouraged at the organisational level.

To achieve this ongoing learning from practice we need to do more than just recognise that it happens gradually, we need to encourage innovation in the process. Encouragement to innovate, whether by bringing in new ideas from the software profession or developing approaches locally, requires a suitable mixture of experimentation, training, boundary scanning, and time to reflect on the current experience. The case showed that experimentation assisted both in the individual learning and in persuading others in the team as the idea was tried in practice. This sharing helps the competence of the team to improve not just individuals.

Improving individual competence

The enactment of the software development practices depends upon the existing knowledge of practitioners. No amount of process improvement activity will produce good quality software if the people are not able to understand how the latest approach can be used. So if processes are to improve the individual's knowledge of them needs to be developed. The knowledge of practitioners is changed through ongoing practice. However, the learning process is not the same for everyone, even if they shared similar experiences, as learning is also influenced by the actors' original knowledge base and their interaction with networks of external practitioners. The ability to reflect on their own work and competence, and critically assess them is an essential feature of improving the quality of the work. The creativity to improvise and the imagination to bring in ideas from outside their own prior knowledge are facets of the developers' competence that are important, but relatively under developed skills in software engineering.

To encourage the reflection and introduction of external ideas, organisations can create a learning environment, whereby the IS staff are encouraged to continuously augment their knowledge. By doing so, companies will benefit through learning from previous experiences and increasing their ability to take on advanced techniques or new ideas. Learning should be continuous. Organisations sanction this learning by developing programmes and incentives to encourage this to happen. Professional development, however, is not always well served by external training courses. Timely and tailored training can be helpful in introducing new concepts, but apprentice-like learning can support the implicit sharing of skills and knowledge within a team.

Ciborra (1994) shows that reflecting-in-action helps organisational actors to gain insight into the background context of the change thus helping to reshape and restructure the organisational context: that of business policy and software development. Mapping Ciborra's (1994, p.21) findings on to process improvement, we can see that changes that are 'developed close to and serve the grassroots of the organisation' will enable the creation of locally appropriate but significantly beneficial processes. Giddens (1984) shows that human knowledge is developed through actors reflexively monitoring their own and others' actions and the consequences, both intended and unintended. This reflexivity happens naturally through the action, but software management can support this by providing time and resources during and after projects to consider the lessons. Continuous improvement requires the opportunity, and encouragement, for reflection on previous activities.

Whilst learning through reflection occurs as a natural part of action, time is an important factor in the degree of learning. This interpretation is supported by Mustonen-Ollila & Lyytinen (2003) who found that innovations occurred when the project members had slack time available. So to create an environment responsive to the development needs, it is necessary to provide sufficient slack time to encourage actors to put their views into practice, but this time needs to be managed. Time on its own is not sufficient, organisations need to encourage a willingness to challenge existing practice and seek innovative solutions to current problems.

Improving team competence

From an organisational learning perspective software process management should consider promoting learning and sharing within and between teams more explicitly too. This helps to enable the reuse of innovations across members of a team and across teams.

Knowledge sharing across a team can occur explicitly through the codification of that knowledge or implicitly through socialisation. The case showed that the latter helped to change the norms, and thus the process-in-use as shared meaning was developed through socialisation: shared standards, understanding and experiences. Through interaction a common language emerges across each community-of-practice. Each community needs to find ways of sharing ideas, requiring a level of trust across the organisation. It has been shown that the emergence of the process occurs through negotiated practice. Not all changes are fully accepted, with evidence of resistance and power play. However, attempts to avoid all conflict are mistaken, as small doses of conflict within the team can help to communicate ideas more forthrightly (Cockburn, 2002).

Where project teams remain distinct it is necessary to enable the knowledge transfer through documentation of the lessons or cross training. However, this is not to suggest that such knowledge can be transferred en-masse, rather it means that making the tacit reflections more explicit will reinforce the structures of signification within (and across) communities-of practice. The sharing of understanding gained from experience in projects can be embedded in formal processes, but is more readily shared by the exchange of people, ideas and mutual experiences. One way to promote this is through knowledge management, but perhaps a more active form of sharing would be by establishing networks of professionals across organisational groups. Truex, Baskerville, & Klein (1999) suggest that back channel communications (virtual discussion groups, white boards, etc.) can enhance this sharing. An organisation's ability to manage its knowledge base is an important management skill.

SPI Coordination

To support the learning and innovation activity, the SPI initiative can be supported through a suitable infrastructure. The SEI view of setting up process action teams under the remit of a software engineering process group (SEPG) is a model that can be adapted to suit this more emergent view of SPI. Rather than the SEPG being the sole identifiers of process areas that need to be introduced, they could act as a facilitating group. By supporting through training, encouragement, resources, and progress management the SEPG can enable the introduction of significant changes to the defined process. The group can also act as a research and development group, facilitating boundary scanning, thus helping an organisation to be more aware of and ready to take on external innovations.

Working with the SEPG, process action teams are useful vehicles for supportive development and introduction of innovations. These can be formed from the available body of developers. Yet whilst all developers are shown to be reflexive in nature, and thus change their own practice through that reflection, not all actively support any organisational initiative. It is suggested therefore that any planned activity as part of a SPI project should be centred around those who are intent on participating. So whilst all can be involved, and drawn into the discussions, forcing everyone to be involved may be counter-productive. Maintaining an open dialogue about the innovations will help to engage those who are less interested, and help to facilitate the negotiation required to bring about the change. So whilst continuous personal development and continuous improvement are key to incremental improvement, focusing on selected process areas and managing improvements in those areas is more likely to be beneficial.

The SEPG could also act as the control group for process actions, synchronising SPI actions over time. New initiatives can be incorporated into the improvement action plan, thus maintaining legitimacy for the SPI project. Also, any lessons from previous development activities can be incorporated into the action plan. The resources can be prioritised by the group to suit the business needs, as well as personal motives. Jakobsen (1998) suggests seeking early success opportunities as this fuels further innovation. Software practitioners will be motivated by the successful application of their knowledge and seek further opportunity to learn, so early successes need to be seen to be useful by the team.

One way to improve the relevance of the SPI activity is to align it with the emergent business strategy. Both planned and improvised improvements could be coordinated to focus upon the areas that are expected to be important to the business to engender a sense of purpose over the longer term. Thus, changes to the software process would enable the organisation to deliver products suitable to its customer base allowing the process improvement to be judged on product related benefits, as well as the internal process benefits.

It is useful to make an assessment of benefits in the early stages of innovations and to assess the associated risk with this change (Gibson, 1998). So to encourage ongoing innovation the usefulness of the changes should be analysed by identifying and collecting key metrics for feedback on the process. The primary measures for package development organisations are best associated with customer values, which may be collected through surveys or through direct measures such as sales revenue or help desk statistics. Also, to give a more immediate review of the impact, software development metrics can be collected. It is suggested that the SEPG, along with any business strategy group, discuss the key targets for improvement and then actions, measures and success can all be associated with those goals.

CONCLUSION

Previously, SPI has been mainly understood as something to be engineered. Yet this literature has been criticised because it fails to understand the micro-dynamics of software practice. Here the ongoing relationship between software process improvement and product development is seen as a constant and fundamental aspect of software practice. It is suggested that SPI can be better understood as emergent, situated change rather than a rational, predetermined activity that transforms an organisation from one state to another. By building on the continuous process improvement models in the literature, an emergent view provides an alternative way of looking at the improvement activity.

The ideas presented here for recognition of the need for agility in SPI relate to aspects that have arisen from a longitudinal case study at InfoServ. The model presented is intended as indicative of the issues that need consideration by IS management, but the ideas are not an exclusive list therefore the views should not be seen as prescriptive. The model highlights the need to learn to improve through situated practice within an organisational framework that supports the needs of the business, reflecting the learning intensive nature of SPI recently identified elsewhere. The case extends the understanding in the existing literature by showing that this learning is not orchestrated but occurs through situated practice and forms part of the integration of the product development and process improvement. Drawing on an active view of learning, where individuals learn through reflection-in-action, process improvement has been understood to occur through ongoing sensemaking. In turn, as this individual learning is shared through communities-of-practice the process-in-use becomes the norm, which is drawn on the next time. In time the espoused theory changes to reflect these norms.

Issues facing IS managers are outlined in a proposed model of agile SPI, reflecting the understanding of SPI as an emergent activity. The purpose of this model is to highlight the need for greater flexibility in the process improvement activity, set within an organisational framework that supports the needs of the business and individual needs. The model relates primarily to packaged software organisations, but can be interpreted for other domains. In presenting these suppositions, it is intended to provoke debate and to challenge the existing software engineering hegemony. Further development and testing of these ideas are required; these tasks form part of the planned future research materialising from this paper.

REFERENCES

Allison, I. (2005) Software Process Improvement as Emergent Change: a Structurational Analysis, *PhD Thesis*, University of Warwick.

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.

Ayas, K. (1996) Professional Project Management: A Shift Towards Learning and a Knowledge Creating Structure, *International Journal of Project Management*, 14(3), 131-136.

Bach, J. (1994) The Immaturity of the CMM, *American Programmer*, 7(9), 13-18. Birk, A and Rombach, D. (2001) A Practical Approach to Continuous Improvement in Software Engineering, In: Wieczorek, M. & Meyerhoff, D. (eds.) *Software Quality: State of the Art in Management, Testing and Tools*, 34-45, Berlin, Germany: Springer-Verlag.

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

Briand, L. El Emam, K. & Melo, W. (1999) An Inductive Method for Software Process Improvement: Concrete Steps and Guidelines. In: El Emam, K. & Madhvji, N.H. (eds.), *Elements of Software Process Assessment and Improvement*, 113-132, Los Alamitos, CA: IEEE Computer Society Press.

Brown, S.L. & Eisenhardt, K.M. (1997) The Art of Continuous Change: Linking Complexity Theory and Time-Paced Evolution in Relentlessly Shifting Organizations, *Administrative Science Quarterly*, 42(1), 1-34.

Card, D. (1991) Understanding Process Improvement, *IEEE Software*, 8(4), 102-103.

Carmel, E. & Sawyer, S. (1998) Packaged Software Development Teams: What Makes Them Different? *Information Technology and People*, 11(1), 7-19.

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.

Cockburn, A. (2002) *Agile Software Development*, Boston, MA: Addison-Wesley.

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

Curtis, B (2000) The Global Pursuit of Process Maturity, *IEEE Software*, 17(3), 76-78.

Curtis, B. & Paulk, M (1993) Creating a Software Process Improvement Program, *Information and Software Technology*, 35(6/7), 381-386.

Cusumano, M.A. (1997) How Microsoft makes Large Teams work like Small Teams, *Sloan Management Review*, Fall, 9-20.

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

Gibson, R. (1998) Software process improvement: innovation and diffusion. In: Larsen, T.J. & McGuire, E.B. (eds.) *Information Systems Innovation and Diffusion: Issues and Directions*, 71-87, Hershley, PA: Idea Group Publishing,.

Giddens, A. (1984) *The Constitution of Society*, Cambridge: Polity Press.

Goldenson, D. & Herbselb, J. (1995) *After the Appraisal: A Systematic Survey of Process Improvement, its Benefits, and Factors that Influence Success*, Technical Report CMU/SEI-95-TR-009, Pittsburgh, Pennsylvania : Carnegie Mellon University/ Software Engineering Institute.

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.

Harkness, W.L., Kettinger, W.J. & Segers, A.H. (1996) Sustaining Process Improvement And Innovation in the Information Services Function: Lessons Learned at The Bose Corporation, *MIS Quarterly*, 20(3), 349-367.

Hosking, D.M. & Anderson, N. (1992) *Organizational Change and Innovation: Psychological Perspectives and Practices in Europe*. London: Routledge.

Jacobsen, A.B. (1998) Bottom-up Process Improvement Tricks, *IEEE Software*, 15(1), 64-68

Jung, H., Hunter, R. Goldenson, D.R. & El-Emam, K. (2001) Findings from Phase 2 of the SPICE Trials, *Software Process Improvement and Practice*, 6, 205-242

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

McGarry, F. & Decker, B. (2002) Attaining Level 5 in CMM Process Maturity, *IEEE Software*, 19(6), 87-96.

Mellis, W. (2001) Process and Product Orientation in Software Development and their Effect On Software Quality Management. In: Wieczorek, M. & Meyerhoff, D. (eds.) *Software Quality: State Of The Art In Management, Testing And Tools*, 3-15, Berlin, Germany: Springer-Verlag.

Mustonen-Ollila, E. & Lyytinen, K. (2003) Why Organizations adopt Information System Process Innovations: a Longitudinal Study using Diffusion of Innovation Theory, *Information Systems Journal*, 13(3), 275-297.

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

Paulk, M.C., Weber, C.V., Curtis, B., & Chrissis, M.B. (1995) *The Capability Maturity Model: Guidelines For Improving The Software Process*, Reading, MA: Addison-Wesley

Ravichandran, T. & Rai, A. (2000a) Quality Management in Systems Development: an Organizational System Perspective, *MIS Quarterly*, 24(3), 381-415.

Ravichandran, T. & Rai, A. (2000b) Software Process Management: An Organisational 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.

Truex, D., Baskerville, R. & Klein, H. (1999) Growing Systems in Emergent Organizations, *Communications of the ACM*, 42(6), 117-123.

Van Solingen, R. (2004) Measuring the ROI of Software Process Improvement, *IEEE Software*, 21(3), 32-38.