A Conceptual MAP of Software Process Improvement

Ivan Aaen, Jesper Arent, Lars Mathiassen, Ojelanki Ngwenyama
Department of Computer Science, Aalborg University, 9220 Aalborg, Denmark

Abstract
Software organisations have for many years struggled to mature engineering practices using a variety of approaches. Over the last decade a new approach, known as software process improvement (SPI), has emerged and become widely used in the software industry. In this paper we position SPI in the landscape of initiatives that can be used in software organisations to mature their operations. A map is offered describing the characteristic features of SPI initiatives, the benefits and risks related to such initiatives, and the relations to complementary approaches to professionalise the industry. The map highlights management, approach, and perspective as three main concerns of SPI and lists three key ideas for each of these concerns. The map is based on an extensive survey of the SPI literature combined with experiences from SPI practice. Practitioners can use the map strategically to make decisions on whether to initiate SPI initiatives, to integrate SPI efforts with other improvement initiatives, and, more generally, to create and manage improvement programs based on SPI ideas. Researchers can use the map to identify key questions and areas of knowledge that can fruitfully inform SPI theory and practice.

Key words:
Strategic management, software process improvement.
1. Introduction

A new set of ideas on how to improve quality and productivity within software engineering has developed over the last decade under the notion of Software Process Improvement (SPI). Inspired by the work of Watts Humphrey (Humphrey 1989; Humphrey 1992; Humphrey 1988), a large body of knowledge on SPI has become available including specific models (SPICE (Emam, Drouin et al. 1998), Bootstrap (Kuvaaja, Similä et al. 1994), CMM (Paulk, Curtis et al. 1993), QIP (McGarry, Pajerski et al. 1994), and QSM (Weinberg 1992-97)), concepts and frameworks to support practical use of the models (Caputo 1998; Grady 1997; McFeeley 1996; McGuire 1999; Wiegers 1996; Zahran 1998), experience reports (Brodman and Johnson 1995; Diaz and Sligo 1997; Goldenson and Herbsleb 1995; Haley 1996; Hayes and Zubrow 1995; Humphrey, Snyder et al. 1991; Johnson and Brodman 1996; Larsen and Kautz 1997; Paulish and Carleton 1994; Sakamoto, Kishida et al. 1996; Wohlwend and Rosenbaum 1994), and critical evaluations (Bach 1994; Bach 1995; Bollinger and McGowan 1991; Brodman and Johnson 1994; Curtis 1994; Fayad and Laitinen 1997; Herbsleb, Zubrow et al. 1997; Humphrey and Curtis 1991; Kohoutek 1996; Ould 1996). In light of these developments earlier quality models are being revised to include process improvement, e.g. ISO9000:2000.

Today, SPI has become one of the dominant approaches to improve quality and productivity in software engineering. Many organisations have committed themselves to long term improvement programs and an increased concern for key practices like requirements management has emerged within the industry. According to Paulk it is, however, “important to remember that software process improvement occurs in a business context. There may be many other crucial business issues being worked on at the same time; there may even be a Total Quality Management initiative under way. Since CMM-based improvement is an application of Total Quality Management principles to software, the synergy of aligning these initiatives seems obvious” (Paulk 1996).

Aligning SPI with other ongoing initiatives in the software organisation is thus an important issue. Software organisations therefore need to understand the characteristic features of SPI approaches and know how they differ from other approaches. How does SPI, for example, relate to the introduction of new methods, to the use of computer aided software engineering, to software factories, and to ISO certification? And how does SPI, on a more general level, relate to Total Quality Management (Deming 1982; Zultner 1993), Business Process Reengineering (Davenport 1993; Hammer and Champy 1993) and other forms of organisational change (Applegate 1994)? Software organisations may need guidance to address such questions in order to make informed decisions on whether to initiate SPI initiatives, to effectively integrate SPI efforts with other improvement initiatives, and, more generally, to create and manage successful improvement programmes based on SPI ideas.

The SPI research community has so far not been very successful in helping to answer such practical questions, despite the fairly extensive literature on SPI. Firstly, there are considerable variations between the authoritative sources on SPI, for example among proposed models and guidelines. Secondly, there is considerable room for interpretation when bringing SPI ideas into practice, adding further variation to what is considered SPI efforts. Thirdly, there are many overlaps and possibly schisms between SPI and other approaches within software engineering, for example software measurement (Jones 1997) and systematic reuse of experiences (Basili, Caldiera et al. 1992).

Our research aims to remedy this situation by offering a general overview of SPI ideas. In doing so we provide a survey of state-of-the-art knowledge on SPI and we position SPI in the landscape of strategic thrusts that can be initiated to mature software organisations (see also Austin and Paulish 1993; Paulish 1993; Paulish and Carleton 1994; Thomson and Mayhew 1997)). The resulting map of SPI addresses the following questions: (1) What are the characteristic features of SPI initiatives? (2) How do SPI initiatives compare to other improvement approaches? (3) What are the key benefits and risks related to SPI initiatives? This map is abstract, i.e. it presents key concepts underlying SPI, and it is rather normative, i.e. its focus is on how the literature advise practitioners to organise and conduct SPI initiatives. In conclusion we identify gaps in the existing literature and potential areas for future research.

The map is structured based mainly on theoretical insights from the literature on SPI supplemented with experiences in practising SPI in close collaboration with software organisations (Johansen and Mathiassen
2. A Conceptual MAP

Improvement efforts must address a number of concerns for how social and technical interventions – i.e. efforts intended to change practices in an organisation - are designed, managed, and conducted. The differences between practical efforts and between theoretical approaches stem from the way in which these concerns are addressed. We distinguish between three fundamental concerns: the principles that are used to manage the intervention, the approach taken to guide the intervention process, and the perspectives used to focus attention on the intervention target. We denote these concerns: management (M), approach (A), and perspective (P)—and hence MAP. The MAP captures and structures the key ideas that the literature offers on how to mature software processes. It is focused on SPI as a strategic thrust that software organisations engage in. Available techniques, practical issues, or implications for the profession and society are complementary issues that fall outside the scope of the paper.

SPI is based on a number of ideas that offer specific answers to three fundamental concerns as summarised in Table 1. A specific improvement effort can, in these terms, be said to follow SPI ideas to the extent that the basic concerns are addressed as suggested. The MAP is distilled from a comprehensive study of the SPI literature and from experiences practising SPI in software organisations. In identifying the SPI literature we have systematically researched the key software engineering journals, the available books on SPI, and papers published in computer and information systems journals and conferences. In selecting references we have systematically researched the references included in the specialised SPI literature and in the available surveys of SPI literature (Fuggetta and Picco 1994; Paulk 1999).

Table 1: A conceptual map of the key ideas underlying SPI

<table>
<thead>
<tr>
<th>Concern</th>
<th>SPI ideas</th>
</tr>
</thead>
<tbody>
<tr>
<td>Management of intervention process</td>
<td>Organisation, plan, feedback</td>
</tr>
<tr>
<td>Approaches to intervention process</td>
<td>Evolution, norm, commitment</td>
</tr>
<tr>
<td>Perspectives in intervention process</td>
<td>Process, competence, context</td>
</tr>
</tbody>
</table>

Management of an SPI intervention effort is based on three ideas: a dedicated and adapted organisation of SPI activities in a dynamic fashion relying primarily on projects; goals, activities, and responsibilities of the overall intervention as well as specific improvement efforts are carefully planned; and feed-back is ensured through systematic measurements and assessments of the effects on software engineering practices. The approach to intervention in SPI is guided by three additional ideas: it is evolutionary in nature focusing on experiential learning and stepwise improvements; it is based on idealised, and a priori defined normative and stable models of software engineering; and it suggests that careful building and development of commitments between the involved actors is essential to ensure dedication and legitimacy. Finally, the perspective of the intervention process is dominated by three ideas: the main lever for improving quality and productivity is the software process that integrates people, management and technology; the building of software developers competencies are seen as the key resource for the software process; and the intention is to change the context of the software operation to establish sustainable support for the actors in the software process.

These nine ideas are intrinsically related and together they form a conceptual map of SPI, which will be elaborated in the following sections. Although there is a large body of literature on SPI that addresses some of the issues we present in the MAP, this literature is fragmented and there is no integrated analysis of it. In Table 2 we present key elements of this literature categorised according to our MAP. Some elements are general references to SPI that address most of the MAP. The other elements are categorised based on their primary focus. The table shows that the literature is very unevenly spread across the categories. This observation is discussed in section 6.
In Sections 3, 4, and 5 we examine each of the basic concepts of the MAP and integrate them with the existing literature. We approach this analysis from the perspectives of theory and practice. First we analyse and contrast each idea with alternative ideas, then we discuss the opportunities and risks involved in practising the idea as part of improvement efforts in

<table>
<thead>
<tr>
<th>MAP Part</th>
<th>Key references</th>
</tr>
</thead>
<tbody>
<tr>
<td>Management (Section 3)</td>
<td>Goldstone and Harbel, 1993; Harbel, et al., 1997; Johnson and Mathisen, 1998; Mathisen and Paulsh, 1997</td>
</tr>
<tr>
<td>Respect (Section 5)</td>
<td>Gold, 1994.</td>
</tr>
</tbody>
</table>
software organisations. These opportunities and risks are general in nature and each specific organisation will have to make its own judgements on which actions can help them take advantage of opportunities or avoid risks. Our ambition is that researchers and practitioners can use the MAP to integrate and make better use of the SPI ideas that exist in the literature.

3. Management of SPI

Clearly, not every organisation that has attempted software process improvement has been successful (Herbsleb, Zubrow et al. 1997). SPI is a challenging and complex change process, where effective change management will be key to success. A number of researchers have pointed to factors such as staff turnover, organisational changes, progress monitoring, and resource dedication as being important to SPI success (Goldenson and Herbsleb 1995; Herbsleb, Zubrow et al. 1997; Johansen and Mathiassen 1998; Mashiko and Basili 1997). Many of these are directly related to management of SPI initiatives. Essential to the effective management of SPI initiatives is a proper organisation of the improvement effort (section 3.1), a comprehensive plan for the effort (section 3.2), and the collection of feedback on the effects of the effort (section 3.3).

3.1 Organisation

The literature is almost unison in recommending that SPI efforts be assigned to dedicated organisational units. As Humphrey sees it: “If software process improvement isn’t anybody’s job, it is not surprising that it doesn’t get done! If it is important enough to do, however, someone must be assigned the responsibility and given the necessary resources” (Humphrey 1989). In line with this several authors provide guidance on how to establish and sustain a process group as the focal point of a software process improvement programme (Fowler and Rifkin 1990; McFeeley 1996; Paulk, Weber et al. 1995). Comprised of people with strong managerial and technical skills this group should be established early to become the main part of the improvement infrastructure. Such qualifications are generally scarce meaning that the SPI initiative may experience problems recruiting and keeping the right people as these people will also be wanted for other pressing purposes in the organisation.

One of the lessons learned from SPI practice is that disintegrated, asynchronous improvement is not only inefficient but also ineffective for solving organisation-wide problems (Humphrey, Snyder et al. 1991). There should be an organisational focal point to plan, co-ordinate, integrate, and implement organisation-wide process improvements. Some researchers have provided examples of how the establishment of a strong and effective infrastructure for continuous improvement can support SPI in different organisations (Basili 1992; Haley 1996; Herbsleb, Carleton et al. 1994; Herbsleb, Zubrow et al. 1994; Sakamoto, Kishida et al. 1996). These cases also show the importance of organising SPI initiatives as dedicated efforts adapted to the organisation. One way to do this is to organise improvement initiatives as projects, where resources are allocated specifically to SPI initiatives and outcomes of the initiatives are specified as project deliverables (Johansen and Mathiassen 1998). This leads to a separation and profiling of the effort from ongoing activities in the organisation and thereby an increased visibility.

Organising SPI as a dedicated effort adapted to the organisation – (i.e. as a project), gives way to a number of opportunities. Firstly it ensures that process adaptation can take place with due consideration to the practicalities experienced in the line organisation. This involves engaging experts from relevant parts of the line organisation who will define working procedures that fit the organisation and its concerns. Secondly resource adaptation can be taken into account. When the effort is established as a project with specified deliverables, the allocation of appropriate resources will be an integral part of the planning process for the project. Thirdly organising the effort as a project with defined deliverables derived from the goals of SPI will increase the controllability of the project and its outcomes.

Process improvement requires a long-term investment that calls for the involvement of top management to establish a well-functioning SPI organisation. When management cannot or will not make needed process improvements, SPI champions may attempt to improve software processes bottom up without an established SPI organisation (Humphrey 1995; Jakobsen 1998). Eventually, applying such a bottom-up approach might open top management’s eyes to the benefits of SPI, and result in a more dedicated SPI organisation. Another alternative would be to centralise the effort to a separate group or to a quality assurance or methods
management, SPI project members, and affected practitioners. (5) The plan can be used as a vehicle for communicating progress to ensure proper visibility and insight in the SPI effort.

Fowler and Rifkin (1990) and McFeeley (1996) provide general guidelines and templates for how to create action plans for managing the SPI effort. Humphrey et al. (1991) also report on a case where action plans were used to manage the SPI effort.

Few would argue against having a plan to manage change initiatives. But improvement activities at times seem to take place without any major plans. In practice we see unplanned and arbitrary improvements where organisations follow opportunities as they emerge almost by accident. We see isolated improvements where changes happen without attention to synergy and without relation to other changes and possibly even in contradiction with other changes. Creating and using plans in SPI initiatives will help to build a common understanding among relevant parties about what is to be done and with what purpose.

A plan is no panacea. Plans can be uncoordinated, meaning that they are not adjusted to other ongoing concerns in the organisation. This could entail that people are not available when they are needed or that other activities overshadow or even contradict planned improvement activities. Another risk occurs when the focus on plans takes a dimension where the improvement activities are relegated to oblivion. Finally a strong insistence of plans and plan adherence may lead to a loss of motivation, killing people’s commitment, or to a loss of flexibility where there will be little room or ability for improvisation.

3.3 Feedback

The universal raison d’être for SPI approaches is to change existing software practices in order to achieve improvements in quality and productivity. But how do the participants know whether they achieve the objectives set out in the improvement plan, how do they know how much the organisation has benefited from the changes, and how do they create stability and orientation in the change process? Such questions point to the need to gather feedback concerning the effect of the SPI effort. There is a need early on to obtain visible results backed up with data if possible, to keep the effort in focus, and to motivate and sustain interest in the SPI initiative (Gray and Smith 1998; Herbsleb, Zubrow et al. 1997; Raynus 1999).
Measurement programs that focus on the beneficial outcomes of the change are required to visualise the progress of the SPI effort compared to a baseline, and to demonstrate the extent to which the goals of the effort are met. The measurement programs may also serve to manage the change process, providing feedback from the improvement efforts initiated in the organisation to the people responsible for managing SPI. Several approaches exist that seek to get this kind of feedback, i.e. measuring the benefits of the SPI effort (Briand, Differding et al. 1996; Carleton, Park et al. 1992; Debou, Lipták et al. 1994; Ebert 1998; Ebert, Liedtke et al. 1999; Florac, Park et al. 1997; Grady 1992; Henry, Rossman et al. 1995; Jones 1996; McGarry, Pajerski et al. 1994; Rozum 1993; Zahran 1996). There are also several reported experiences with measuring the effect of SPI efforts (Brodlman and Johnson 1995; Diaz and Sligo 1997; Dion 1992; Dion 1993; Haley 1996; Herbsleb, Carleton et al. 1994; Herbsleb, Zubrow et al. 1994; Iversen and Mathiassen 2000; Johnson and Brodlman 1996; Sakamoto, Kishida et al. 1996). The Goal Question Metric approach (Basili 1992) seeks to make measurements operational by focusing the SPI effort on specific business goals rather than using underlying generic models as norms for the effort (Mashiko and Basili 1997). However, experiences with measurements indicate that it is difficult to establish useful metrics programs that can help measure the benefits from a business perspective. It is therefore advisable to establish a baseline through a series of dedicated improvement projects starting with relatively simple indicators and giving high priority to the practical use of the data (Iversen and Mathiassen 2000; Johansen and Mathiassen 1998; Rozum 1993).

There are alternatives to measuring the effects of the SPI effort. One option is to strive for the achievement of abstract goals. This is what people do when they for example state a goal to go for level 3, 4 or 5 in the CMM model without getting an understanding of the benefits to be obtained by such a move. This way the improvement goal becomes elusive and it is difficult to mobilise the organisation in the effort. Another option is to rely on people’s perception of the effects. It is without question difficult to measure effects of SPI and there will be a temptation to settle with a consensus in the organisation that the effects are there without proper measurements. This approach may maintain commitment to some degree, but it only provides little information to manage the change process. A third alternative would be to rely on religious impulse presenting the CMM with almost religious appeal. Relying on this impulse may help in building commitment towards the SPI effort, but based on a moral claim it is susceptible to loss of faith in the particular objectives of the effort.

Feedback may produce at least three opportunities for the SPI effort. Firstly it can provide legitimacy to the effort and the resources spent by pointing to positive outcomes. Practitioners as well as managers in the organisation will be more appreciative and protective of the effort given that a return of investment can be demonstrated. Secondly measurements serve as vital instruments for control of the effort. Measurements may demonstrate the efficiency and effectiveness of SPI strategies, tactics, and the changes made to engineering practice. Thirdly as suggested above measurements used prudently may contribute to maintain motivation, commitments, and legitimacy.

Several risks are involved in measuring effects though. Measurements are per se regarded to be a problematic area of software engineering. Software organisations experience difficulties in establishing well-grounded and justifiable measurements that are both relevant and meaningful. In other words it may be hard to argue for the validity of the measurements. Another problem is to ensure verifiability in order to establish that they are trustworthy, accurate and reliable. Verifiability means that the measurements should be repeatable and comparable. Conformance to these requirements may be very difficult for many SPI efforts. On top of these challenges we find the more subtle hazards of opportunism on behalf of organisational units or individuals. To some it will be tempting to use measurements as an opportunity for advancing or protecting particular interests. Opportunism may lead to attempts to include irrelevant data into the measurement program or even to fraudulent or pretended measurements. Keeping individual metric data private is essential to reduce such problems.

4. Approach to SPI

The approach addresses how to obtain real changes in the software process. The first idea is that changing the complicated working processes of software engineering should be done in an evolutionary rather
than revolutionary manner (section 4.1). The second idea is that norms should be utilised to guide and control the results of this process of change (section 4.2). Finally the third idea asserts that software process improvement should draw heavily on committing people to the changes they will be affected by (section 4.3).

4.1 Evolution
A widespread idea in SPI approaches is the use of an evolutionary approach where changes are implemented by a sequence of changes over a period of time instead of in one single, dramatic transformation. SPI approaches generally emphasise stepwise yet incremental improvements within a limited set of process areas. These incremental changes are continuous, concerted, and accumulative following Deming-like Plan-Do-Check-Act cycles of assessing, experimenting, androlling out at the individual, project, and organisational level using perceived needs as one important driver. At the conceptual level several authors advocate using an evolutionary approach to SPI (Basili and Green 1994; Fowler and Rifkin 1990; Johnson and Brodman 1996). Likewise a number of authors report on experiences from using this approach (Arent, Iversen et al. 2000; Hayes and Zubrow 1995; Herbsleb, Carleton et al. 1994; Humphrey, Snyder et al. 1991; Jones 1996; Larsen and Kautz 1997; Willis, Rova et al. 1998; Wohlwend and Rosenbaum 1994).

However, alternative approaches exist in other areas of organisational change. Business Process Reengineering or Process Innovation is an example of a revolutionary approach that focuses on the implementation of deliberate and fundamental change in business processes to achieve breakthrough improvements in performance (Davenport 1993). A Business Process Reengineering initiative starts with a relatively clean slate rather than from the existing process, and is generally a discrete top-down initiative. An alternative change approach is technology-push, where organisational actors are encouraged by management or experts to acquire a specific technology (e.g. Computer Aided Software Engineering) as a way to achieve impressive improvements in performance. The complexities of software processes would presumably inhibit improvements based on revolutionary strategies or technology-push.

An evolutionary approach to SPI offers several opportunities. The opportunity to involve practitioners in identifying, designing, and implementing changes is an important success factor in SPI (Goldenson and Herbsleb 1995; Humphrey 1989). People, who participate in developing their own future work process, will likely be more willing and motivated to change their existing practice. The opportunity for experience based learning is another important aspect of SPI. Carrying out stepwise incremental improvements increase the opportunity to learn from experiences, successes and failures when projects experiment with new or modified processes. Finally, keeping and leveraging the best elements of the existing process is possible in evolutionary SPI. As Paulk states: “Begin with the “as is” process, not the “should be” process, to leverage effective practices and co-opt resisters” (Paulk 1996).

Still, an evolutionary approach involves risks. One risk is a limited or invisible effect per change. The focus in SPI is on the accumulation of incremental changes to gain performance increases rather than on the interjection of an immediate, large change. However, implementing incremental changes to a limited set of process areas might not readily yield immediate, visible, and large improvements. SPI professionals may be in a situation where they cannot measure any effect due to measurement uncertainties. Another risk occurs if the incremental improvements are not anchored and maintained in the daily practices. If this happens the performance increase may be even more limited and invisible. The first wave of changes will probably be noticed, but the second wave may either pass by unnoticed or make people forget what the first wave brought about. Still, dealing with incremental changes generally means low risks as compared to the risks involved in revolutionary change approaches.

4.2 Norm
Many software organisations approach the road to improved performance by adopting an existing norm for how to improve software processes. A norm-based approach to SPI provides the basis for carrying out capability assessments (Daskalantonakis 1994; Dunaway and Masters 1996; Iversen, Johansen et al. 1998; Raynus 1999; Sanders 1998) of the existing processes according to a professional standard indicating an ideal configuration of processes. As important, these norms also provide a basis for formulating a strategy aiming to fulfil the gaps between
In the 1990s a host of software process norms have emerged from a number of schools, for example, CMM – the Capability Maturity Model (Humphrey 1988; Humphrey 1995; Konrad, Chississ et al. 1996; Paulk 1995a; Paulk, Curtis et al. 1993; Paulk, Weber et al. 1995; Paulk, Weber et al. 1993), Bootstrap (Haase, Messnarz et al. 1994; Kuvaja and Bicego 1994; Kuvaja, Similä et al. 1994), SPICE (Dorling 1993; Emam, Drouin et al. 1998; Rout 1995) and the new ISO9000:2000. Similarly norms have been suggested for areas related to software development, such as, the People Capability Maturity Model (Curtis, Hefley et al. 1995), the Software Acquisition Capability Maturity Model (Ferguson, Cooper et al. 1996), and the Systems Engineering Capability Maturity Model (Bate, Kuhn et al. 1995). In an effort to provide an overview of these norms a number of surveys and comparative studies have been published (Garcia 1997; Paulk 1995b; Saiedian and Chennupati 1999; Thomson and Mayhew 1997; Tingley 1997). The emergence of all these models has led to a call for more focus on the practical use of models and less focus on developing even more models (Wiegers 1998).

However, critics and proponents have engaged in a debate on norm-based improvements and alternative approaches have been suggested (Bach 1994; Bollinger and McGowan 1991; Curtis 1994; Fayad and Laitinen 1997; Herbsleb, Zubrow et al. 1997; Humphrey and Curtis 1991; Kohoutek 1996; O’Connell and Saiedian 2000). One alternative approach to using norms is the Goal Question Metric approach, which relates business goals with questions and metrics to specify a measurement system for quality improvement (Basili 1992; McGarry, Pajerski et al. 1994). Iversen et al. describe an alternative to norm-based assessments called problem diagnosis that deals with eliciting problems perceived by project managers to see the specific and unique features of the software processes in a company (Iversen, Nielsen et al. 1998).

A norm-based approach to SPI includes several opportunities. Clearly stated and well-understood visions and goals are often mentioned as success factors in SPI initiatives (Arent, Iversen et al. 2000; Fitzgerald and O’Kane 1999; Goldenson and Herbsleb 1995). A norm-based approach provides the opportunity to create a vision of a future state and explicate goals in accordance to the norm and tailored to the organisational context. Furthermore the norm support organisations in comparing experiences and achievements from process improvement efforts (Brodman and Johnson 1994; Hayes and Zubrow 1995). However, one risk involved in using a norm to formulate a vision is aiming too high and developing an overly ambitious strategy where changes may not be experienced as evolutionary. Another important possibility with norm-based SPI is benchmarking. By applying a professional standard, which is widely used throughout the world, a company has the possibility to compare itself against other companies and profile itself accordingly. Finally, a norm provides criteria for prioritising improvement areas and implementing stepwise improvements focusing on a limited number of improvement areas at a time.

A norm-based approach introduces several risks. The risk of developing an overly ambitious strategy based on the norm has been pointed to above. Following the norm for the norm’s sake with little regard to the actual need is also a risk. Finally it can be difficult to obtain reliable results of the assessment process and hence difficult to compare norm and reality (Emam and Madhavji 1995). When this happens the norm will be of little use for identifying the progress achieved and areas that need more attention.

4.3 Commitment

Salancik describes commitment as a state of mind that holds individuals in a line of behaviour (Salancik 1977). As such commitment may greatly influence the outcome of large-scale organisational change, and the concept is essential for SPI (Arent, Iversen et al. 2000; Grady 1997; Humphrey 1989; Humphrey 1997; Paulk 1996). Writers on commitment in SPI argue that in order to change the performance of an organisation, senior management must actively support the change initiative with resources and attention.

Grady lists seven business and five organisational aspects that influence management commitment (Grady 1997). Three business aspects of strategic importance are: Vision, Strategic focus, and Core competence; and four business aspects of tactical importance are: Customer perception, Market share, Product cycle time, and Profitability. Of the five organisational aspects two are strategic: Organisational maturity, and Process improvement infrastructure; and three are tactical: Organisational inertia, Stability, and Cost/Time alignment. Any of these aspects if not managed effectively can distract managers enough to
jeopardise an SPI project.

Are there any alternatives to commitment-based improvement? One possibility is to base the improvement effort on power. Paulk, however, points out that mandating top-down that everyone should follow the new processes is a common recipe for failure (Paulk 1996). One could also imagine personal initiatives to improve practices considered problematic by individuals. Even though this might lead to improvements of individual capabilities, such discrete personal initiatives, without sponsorship and co-ordination, might also lead to islands of excellence rather than predictably improved organisational capability. If the entire organisation shows commitment towards SPI, people will be motivated to share new ideas and experiences, try out new practices, and work together to reach challenging goals (Jakobsen 1998).

Although the commitment process is vital for SPI, it can be carried too far. Managers and practitioners may become so dedicated to solving current problems, that they lose sight of the original goal. This can further lead to a loss of perspective of the long-term improvement program, and to gold plating solutions that are sophisticated beyond what is appropriate for the current situation.

5. Perspective in SPI

SPI offers specific perspectives on the intervention target: software processes as they are practiced. How should one focus attention, and what kinds of means are useful? On this level, the entire body of knowledge on software engineering is potentially relevant. Within SPI the focus is on software processes (section 5.1), the software developers competencies are considered the key resource (section 5.2), and the intention is to develop a supportive and sustainable context for software engineering (section 5.3).

5.1 Process

The software process denotes the integration of what, how, and with what means people work to produce specific products. CMM defines a software process as a set of activities, methods, practices, and transformations that people use to develop and maintain software and the associated products (Paulk, Weber et al. 1995). Software process improvement is aimed at maturing the software process—meaning that the software process becomes better defined and more consistently implemented throughout the organisation—and this results in increased process capability, i.e. the range of expected results with respect to quality and productivity to be achieved by following the process.

A process is described as an institution, i.e. a significant and firmly established set of practices within the organisation. Changing existing or establishing new software processes requires the “building of infrastructure and corporate culture that support methods, practices, and procedures so that they are the ongoing way of doing business, even after those who originally defined them are gone. As a software organisation gains in software process maturity, it institutionalises its software process via policies, standards, and organisational structures” (Paulk, Weber et al. 1995).

The idea in the process perspective is to provide a holistic perspective on software engineering that is useful for improving the profession. An alternative could be to focus on discrete parts:

Product focus. Even though the conventional wisdom seems to be that improved processes lead to improved products, this position can be questioned. Software processes are extremely complex and it would be easier to focus on product quality. On the other hand software quality as a concept is hard to define clearly and all parts of the software process contribute one way or the other to achieve desired qualities in the end product. A product focus will therefore not likely provide clear indications on where and with what means improvements may be achieved.

Method and tool focus. Studying how specific methods and tools perform in projects or in the organisation would be another alternative. It allows for focused observations of fine-grained process elements resulting in recommendations on whether a given method or tool should be exchanged with alternatives or whether it should be altered or used in a different way. But focusing on methods and tools leads to a simplistic understanding where interdependencies between process elements are not sufficiently understood or perhaps even completely ignored. This implies that methods and tools should be studied within an integrative framework that helps understand their practical use in their organisational context. Only an integrative, holistic framework, e.g. based on processes, can provide the basis for sound
analyses of inter-dependencies.

People focus. Another alternative would be to use perspectives in which behavioural changes are seen as the main driving force in improvements (Bach 1995). In contrast to meticulous scrutiny of product quality or detailed analysis of how methods and tools function this perspective focus on the practitioners that perform processes. Practitioners are assumed to have a good sense of where improvements can and should take place and they want to take active part in improvement efforts. This perspective does not call on assessments of software process capability nor on measurements of the effects of improvement efforts. Also, there is no direct focus on the potential and actual effect of using technology, be it methods or tools. The software process remains largely intangible in this approach and improvements are seen as intrinsic parts of professional practices.

These alternative perspectives constitute a loosely coupled system where people, methods, tools, and products can be studied separately or where interdependencies are limited to a few relationships. They express the basic idea of SPI which is to focus on software processes as social institutions with a complex interplay of people, methods, tools, and products (Fowler and Rifkin 1990; McGarry, Pajerski et al. 1994; Raynus 1999). Adopting this systems view is likely to lead to more viable solutions (Haley 1996). First, because it involves a holistic view on software engineering seeing the software process as a system thereby allowing for an understanding of complex interdependencies. Second, because it leads to a situated understanding seeing the processes and their changes from a use perspective in which the specific conditions for improving software operations play a major role (Bjerkenes and Mathiassen 2000; Gray and Smith 1998). And finally, because it calls for a participatory view on process improvement by virtue of the very definition of software processes.

Still, the process perspective involves several risks. Competent and experienced people within the organisation are set aside for mainly internal purposes. Few of the customers will appreciate this strive for an improved process and recognise this as something they will benefit from in a longer perspective. A second risk is that existing software processes may prove too difficult to change. Although the SPI effort strive for institutionalising new processes it must not be forgotten that existing processes are already institutions. A third risk is underestimating the people element degrading practitioners to be merely instruments in the software process. This may lead to problems committing the practitioners to a new software process. Finally, the broad focus on people, methods, tools, and products may lead to an under-utilisation of new technical infrastructures that can be instrumental in provoking changes to the existing software process.

5.2 Competence

Even the best methods and tools require competent people to be of any use, and competent people are therefore a key ingredient of any well-functioning software process. Several authors have recognised this factor, e.g. Boehm who identifies people as the top risk for software development (Boehm 1988) and Humphrey who argues that talented people are the most important element in any software organisation (Humphrey 1989). Conditions differ from project to project requiring people engaged in a project to follow — but also competently adapt — an established software process. Ideas on how to develop competencies are therefore needed as an essential part of a successful software process improvement effort.

The development of supportive infrastructures with suitable methods and tools and of sustainable management structures needs to be complemeted with appropriate skills and responsibilities amongst software engineers (Hutchings, Hyde et al. 1993; Pressman 1994). Specific models have been developed to support such efforts. In these models competence building is addressed both on the organisational and project level (Curtis, Heffley et al. 1995) and on the individual level (Humphrey 1995).

One conceivable alternative to relying on competence building and empowerment as the main instrument for process adherence is bureaucratisation, i.e. building the software process as a bureaucracy in which rules and hierarchical management structures are key instruments for improvement and sustainability. A bureaucratic strategy may focus on institutionalising structural support through heavy use of technology and by customising powerful technical infrastructures to each new project. Such infrastructures are intended to embody the software process and instruct and guide practitioners (Aaen, Bettcher et al. 1998). This approach does not, however, allow for discretion and adaptation—the process will be pre-programmed and
Competent professionals as defined in this section can understand and appreciate the process and individually and collectively they will be able to use their discretion and adapt to the calls of the situation. This way projects provide opportunities to build, supplement and transfer knowledge and routines among developers. In that way individuals will be participants in a learning organisation where good and bad experiences contribute to the continuous development of the software process to suit contemporary and future needs.

Competence building makes it possible to delegate responsibilities to where insights into problems and opportunities reside. In organisations where common values are not sufficiently strong competence building may lead to a number of problems: (1) Loss of corporate control resulting from weaker overall co-ordination. (2) Goal deflection where overall goals for the software process gives way to particular objectives or ideals. (3) Turf guarding where protection of particular interests gets in the way of organisational or project goals. Building competencies without empowering people to exercise their competence is on the other hand risky as it might lead to staff turnover.

5.3 Context
Adhering to a predefined software process provides a context for software engineering in which the process can be improved on a general level while parts of the process are adapted to specific needs. The context provides an environment for each element of the software process making it clear why things are done, how they are done, and when they are done. The context also provides a setting for the software engineers, it supports the introduction of newcomers, it defines requirements for training, it establishes opportunities and constraints for process variations and adaptations, and it establishes a framework for the customer/supplier relationship (Bjerkness and Mathiassen 2000). Level 3 of the CMM aims at establishing and maintaining such a context (Palk, Weber et al. 1993).

The context represents the standard software process of the organisation and by this the relatively stable basis for customising software processes to particular projects. The context thus embodies the capability of the software process and in that sense it represents the very essence of institutionalisation. The context is where individual and organisational competencies and infrastructures merge through training, documented procedures, a repertoire of methods and tools, and other kinds of support.

One alternative to this is to let go of the organisational standard software process and address every project as a unique process. This alternative will call for superior qualifications of the participants, the process overheads will likely increase, and the possibilities for learning from project to project will be defied. Classic alternatives would be to rely on heroes (Bach 1995) or on widespread usage of technology. These alternatives have been tried for years with few documented successes.

A sustainable and supportive context offers several opportunities for a sound and strong software process: best practices can be identified, systematic reuse can be supported, new employees can be introduced to traditions and practices, training programs can be offered, and a professional software engineering culture can be developed and reinforced.

The biggest risk connected to a strong focus on the context is undoubtedly conservatism—the possible ossification of the process where practices are kept unchanged after their justification has disappeared. In the same vein there is a risk for ritualisation where activities are performed because they are part of tradition rather than because they are needed.

6. Contributions to SPI Theory and Practice
SPI has become a commonly used strategy to improve quality and productivity in software engineering. But the literature on the subject is both rich, varied, and quite difficult to overlook and interpret. In this paper we set out to build a comprehensive framework to integrate concepts that are foundational to Software Process Improvement theory and practice. Our study offers a general overview over SPI ideas in the form of a conceptual MAP that is independent of the available normative models for conducting SPI. The MAP, which is summarised in Table 3, was developed based on an extensive literature survey supplemented by experiences from a set of longitudinal action research engagements in Danish companies implementing SPI (Johansen and Mathiassen 1998). The MAP organises and integrates concepts and fundamental assumptions...
that underlie the emerging body of knowledge on SPI. The three categories of our framework, Management, Approach, and Perspective, signify fundamental concerns that we have encountered in the practice of SPI. We believe that both researchers and practitioners can use this MAP to analyse and guide existing SPI research and practice. In the following we outline some of the contributions of the MAP in these two arenas.

6.1 Contributions to SPI Theory

The conceptual MAP contributes to SPI theory and research in several ways: First, it presents a comprehensive framework that integrates fragmented streams of SPI research to facilitate the development of a coherent body of knowledge. Second, the MAP can support and guide future research initiatives in SPI by indicating prominent areas in current SPI literature. Finally, it can orient new researchers who are interested in conducting research in specific areas of SPI, by providing them with a starting point for their efforts.

As we have shown in Table 2, current SPI literature can be categorised within the MAP according to contributions. This analysis reveals some serious gaps in the research on SPI. Currently, the SPI literature focuses mainly on aspects related to norms for classifying software organisations and feedback, that is how to assess whether an organisation is compliant with a specific norm. However, it is important to realise that compliance does not automatically lead to success. Norms are partial models, and do not cover all aspects relevant for business success. An organization may comply with the specific practices of a process standard, but still fail to meet the needs of the organization. Interpreting norms to fit the organization’s culture and business context is important for norm-based improvements. Areas such as the organisational context, management commitment, the intervention process, and the building of competence have not received adequate attention by researchers. There are several research questions about these areas that need to be investigated. How do contextual features of an organisation enable or constrain SPI initiatives? Although the SPI literature outlines an evolutionary path to progress and strategies for ascertaining the level achieved, there is almost no discussion about how organisational conditions might impact SPI change initiatives. Even though SPI is an organisational change mechanism the literature is woefully uninformed by organisational change theory. Thus, current approaches to SPI overlook many issues of organisational change, such as organisational learning, culture and politics, that can impact on how change is perceived, enacted and institutionalised.
Another area that the SPI literature fails to address effectively is management of the SPI change initiative. SPI change initiatives are by nature long term engagements; it takes an organisation several years to move from level 1 to 5 on the CMM scale. Managing this large-scale, long-term change is a challenge to even highly sophisticated managers. How do managers maintain commitment to long term organisational transformation in a dynamic environment where everyday issues continue to compete for attention? What are the key levers of change in software organisations? How should the change process be organised? To what extent and how should top-down and bottom-up approaches be combined? How can organisational fatigue be avoided? Are incentive schemes effective tools for SPI? All these and others are important questions of management upon which the SPI literature is silent.

Further, very little attention is paid to strategies for formulating and communicating the vision for SPI change throughout the organisation. Organisational change research points out that the vision for the new organisation must be understood and shared by the majority of the members of the organisation if the change initiative is to be effective (Hart and Quinn 1993). This not only helps to build much needed commitment but also relieves the change agents from the burden of continually arguing the case for specific change activities.

Lastly, the role of competence needs much more research. What are the areas of expertise that are important to SPI? How do managers acquire this expertise? Should they engage consultants or build up the expertise in-house? Here again the SPI literature is silent. Change agents involved in SPI need a deep appreciation of organisational change issues. A key problem for managers however, is the scarcity of experts in organisational transformation.
scale, complexity and duration of SPI projects present managers with the problem of acquiring and maintaining the appropriate expertise.

6.2 Contributions to SPI Practice

The MAP can help software managers make informed decisions on SPI initiatives. It provides an overview of concerns, ideas, and knowledge related to SPI. This kind of overview can help managers survey their situation and anticipate and strategize on how to deal with major issues. More generally, the MAP is intended to serve as a practical framework for the management of improvement programs based on SPI ideas.

There is considerable room for interpretation when bringing SPI ideas into practice. It is always necessary to adapt to the specifics of an organisational environment and to start out from existing traditions for software development and management. As a consequence, the ways in which SPI is practised vary greatly. The authors have collaborated with one organisation that emphasised a rigorous application of CMM ideas. This organisation organised extensive in-house training of CMM specialists to take charge of the improvement efforts and performed formal CMM assessments across the divisions of the organisation. Another organisation decided to develop an in-house standard for professional project management inspired by CMM level 2, but each of the six key process areas was slightly modified and four new key process areas were added to reflect particular needs. A third organisation opted to push norm-driven assessment ideals into the background. The improvements in this case were driven by problem diagnoses that were carried out in close collaboration with project managers. This strategy was not based on any general maturity norm, but it led to a number of improvement activities to which the project managers felt highly committed. Thus the first organisation was driven by a general norm, the second was driven by an adapted norm, and the third was problem driven rather than norm driven.

These examples illustrate how SPI initiatives use norms differently as part of their approach to SPI (see Table 3). Variations can also be found related to other concerns and ideas of the MAP. Such differences between SPI theory and practice can lead to discussions of how many of the nine ideas must be followed for an initiative to qualify as a ‘true’ SPI effort. A less dogmatic approach would be to use the examples to illustrate how the MAP can be used on a practical level to manage SPI efforts. First, the MAP can be used as a constructive framework to design SPI initiatives by suggesting 9 ideas of concern. Second, the MAP can serve as a diagnostic tool to evaluate ongoing SPI efforts: which ideas are practised well, and which ideas could improve an SPI effort? We invite software managers to use the MAP in these ways to incorporate SPI ideas into their strategic thinking and practice.

Acknowledgements

This research has been partially sponsored by the Danish National Centre for IT Research. We wish to thank the editor, Karl Kautz, for valuable comments and the software organisations, practitioners, and researchers participating in the Danish SPI initiative.

References


Konrad, M., Chrissis, M. B., Ferguson, J. et al. (1996) Capability Maturity Modeling at the SEI. Software Process - Improvement and Practice, 2, 1, 21-34.


