Global Co-Operation in the New Millennium
The 9th European Conference on Information Systems
Bled, Slovenia, June 27-29, 2001

AVOIDING FAILURE IN SPI INITIATION

Pouya Pourkomeylian

AstraZeneca R&D Mölndal, S-431 83 Mölndal, Sweden
Viktoria Institute, Box 620, 495 30 Göteborg, Sweden
Tel.: +46 31 776 17 96, Fax: +46 31 776 37 30
Pouya.pourkomeylian@astrazeneca.com

ABSTRACT

Software Process Improvement (SPI) is a recognised systematic approach for improving the capability of software organisations. Such initiatives have met with a number of difficulties such as: scaling the SPI initiatives, setting realistic goals, the complexity of organisational changes, and the organisational culture. For organisations with no earlier experience with SPI, the first initiative might therefore run the risk of being the last. Therefore understanding SPI and the risks involved in such initiatives may help organisations to run SPI activities more successfully. This paper shows the results of a collaborative research project in which the first SPI initiative in an organisation was analysed based on a framework that maps the characteristic features of SPI. On the basis of our findings we argue that the first SPI initiative: 1) should be a learning process focusing on learning SPI practice, 2) should satisfy organisational goals rather than routinely follow a normative model for reaching a maturity level, 3) should be organised as a project aiming to improve a few software processes based on practitioners' ideas and needs, and 4) should include analysis to identify the most characteristic feature of the SPI initiative and the risks.

Keywords*: Software Process Improvement (SPI), SPI Initiation*

1. INTRODUCTION

Software Process Improvement (SPI) is a systematic approach to improve software processes in organisations. This approach was developed by the Software Engineering Institute (SEI) inspired by the work of Watts Humphrey (1989). The basic idea of SPI is to focus on software processes as social institutions with a complex interplay of people, methods, tools, and products (Aaen *et al.* 2000).

SPI initiatives start with an assessment to identify the organisations' current software process problems. The improvement activities should then be planned and performed on the basis of the assessment's findings and other goals of the organisation. The improved new software processes should then be institutionalised in the entire organisation to become part of the practitioners' daily work. Many organisations have been inspired by the concept of SPI and started SPI initiatives. Achieving success with SPI has, however, proven to be a difficult challenge. Many organisations do not succeed in performing their improvement activities, others have difficulties with implementation of new processes in the organisation (Tryde *et al.* 2000). Different factors such as scaling the SPI initiative, setting realistic goals, the complexity of organisational changes, and

the organisational culture have made it difficult to achieve success in SPI initiatives (Goldenson and Herbsleb 1995, Herbsleb *et al.* 1997, Mashiko and Basiili 1997, Johansen and Mathiassen 1998). For an organisation with no earlier experience in SPI (a novice organisation), the first initiative might as a consequence run the risk of being the last. It is therefore crucial for such an organisation to find answers to some key questions before starting an SPI initiative: What are the most characteristic features of the first SPI initiative? How should a novice organisation organise, plan, and conduct an SPI initiative? Does a novice organisation have a fair chance of succeeding in its first SPI initiative?

According to (Aaen *et al.* 2000), organisations that start SPI efforts should find inspiration and guidance in the literature. They argue that these organisations should avoid the pitfalls that have led to failure in other organisations and should learn from successful initiatives that have bearing on their own situation. However, following this advice is not easy: the SPI literature is extensive and is growing and there are no authoritative sources outlining the underlying rationale of SPI (Aaen *et al.* 2000). A large body of knowledge about SPI has become available during the last years, including specific models (Paulk *et al*. 1993, Kuvaja 1994), concepts to support practical use of the models (McFeely 1996, Zahran 1998), experience reports (Goldenson and Herbsleb 1995, Johansen and Mathiassen 1998), and critical evaluations (Curtis 1994). A survey of SPI literature and a MAP of the key ideas in SPI are presented by (Aaen *et al.* 2000). They provide a conceptual MAP, which describes three fundamental aspects of SPI including nine ideas. According to these authors SPI is based on a number of ideas that offer specific answers to specific concerns. SPI has three fundamental concerns: the *management* of SPI, the *approach* taken to guide the SPI initiatives and the *perspective* used to focus attention on the SPI goals. Table 1 contains a survey of the MAP described by (Aaen *et al* .2000) and provides an overview of the key ideas involved in SPI.

Concern	Idea	Aspiration	Pitfalls
Management of SPI	Organisation	Create a dedicated effort adapted to the conditions of the organisation	Inadequate resources, emphasis and co-ordination
	Plan	Plan goals, activities, responsibilities and co-ordination	Loss of motivation. Diversity or deadlock
	Feedback	Measure and assess benefits	Opportunism, and loss of relevance
Approach to SPI	Evolution	Learn by experience and employ stepwise improvements	Burnout and inertia
	Norm	Seek dedication and legitimacy	Hastiness and fundamentalism
	Commitment	Ensure dedication and legitimacy	Goal deflection and gold plating
Perspective in SPI	Process	Integrate people, management and technology	Customer disinterest
	Competence	Empowerment through competence building	Turf guarding
	Context	Establish sustainable effort	Machine bureaucracy

Table 1: *The SPI MAP with aspiration and pitfalls (Aaen et al. 2000b)*

Based on their concept the management of SPI initiatives builds on three ideas: 1) the SPI activities are *organised* as dedicated efforts, 2) all improvement efforts are carefully *planned* and 3) *feedback* on effects on software engineering practices are ensured. The approach to SPI initiatives is guided by three additional ideas: 1) SPI is *evolutionary* in nature, 2) SPI is based on idealised, *normative* models of software engineering and 3) SPI is based on a careful creation and development of *commitments* between the actors involved. Finally, the perspective on the SPI target is dominated by three ideas: 1) SPI is focused on software *processes*, 2) the practitioners' *competencies* are seen as the key resources and 3) SPI aims to change the *context* of the software operation to create sustainable support for involved actors.

Using the MAP, this study analyses one SPI project done as the first SPI initiative in a novice organisation. The main research question is: Which were the most characteristic features of this SPI initiative? It is hoped that this analysis will help other novice organisations to find ways to increase their chances of success and minimise the risks of failure.

The next section discusses the research approach. Section 3 presents the case. Section 4 presents the results of the SPI initiative analysed and discusses the findings according to the research question stated above and section 5 concludes the paper by presenting some lessons.

2. THE RESEARCH APPROACH

In this study we have combined action research in combination with case study and analysed one SPI initiative that was carried out from April 1999 to May 2000 and aimed to improve the software organisation's software processes. By SPI initiation in this study means all activities performed for planning, organising and improving new software processes. These activities do not include implementation of new processes in the organisation. The author has been the driving force behind the SPI initiative and has actively participated in activities to initiate, organise, plan, and conduct the SPI initiative during the one-year period. In this study the author reflects on the SPI initiative and tries to make conclusion about, lessons useful for understanding the field of SPI and support the practice of the first SPI initiative in novice organisations.

Based on the MAP we used a workshop to analyse the conducted SPI project at the software organisation. The workshop was conducted as a structured brainstorm with an SPI expert and the author who was the project manager of the SPI project. First we listed the three main fundamental concepts of SPI and the nine SPI ideas. Second we defined the aspirations based on the MAP (Aaen *et al.* 2000) for every idea. Next step we determined the extend to which these ideas were followed in the conducted SPI project. Then we described why every specific SPI idea was or was not performed and described every situation for each idea. On the basis of these information we evaluated the effects that performing or not performing every specific SPI idea had on our SPI project. We also identified the risks for every SPI idea in our project and tried to understand what each risk could have caused if it had happened.

3. THE CASE

This study was conducted at AstraZeneca, one of the world's leading pharmaceutical companies. AstraZeneca is a research-driven organisation with a formidable range of products designed to fight disease in important areas of medical need. The company was formed in April 1999 by the merger of Astra AB and Zeneca Group PLC. AstraZeneca has a strong research base and powerful product portfolio, designed in seven areas of real medical need – cancer, cardiovascular, central nervous system, gastrointestinal, infection, pain control and anesthesia, and respiratory. AstraZeneca is world number three (1999) in ethical pharmaceuticals and has more than 50,000 employees world-wide. There are research and development (R&D) centers of excellence in Sweden, UK and the USA and R&D headquarters in Södertalje, Sweden. The company has some 10,000 R&D personnel and a US $2 billion R&D investment in 1999, extensive global sales and marketing network, employing over 25,000 people, and 12,000 people employed in production in 20 countries.

3.1. The Software Organisation

This research started before the merger between the two companies in an IS organisation called Clinical Research and Information Management (CRIM) at the former Astra Hässle in Sweden and continued later in the new IS organisation, which then changed its name to Development IS (DevIS). DevIS supports clinical and pharmaceutical projects, Regulatory Affairs and Product Strategy and Licenses at AstraZeneca R&D Mölndal. DevIS is also responsible for influencing the development of the global clinical research processes

and IS/IT tools in AstraZeneca. DevIS comprises 90 people including contractors, most of whom have backgrounds in IS/IT.

Many regulatory authorities require that pharmaceutical companies and their software organisations comply with GXP (Good *Manufacturing* Practice, Good *Clinical* Practice, and Good *Laboratory* Practice) rules. GXP rules are the authorities' quality requirements to pharmaceutical companies for ensuring patient health, the quality of processes (e.g. clinical studies or software development) and the quality of products (e.g. tablets or software). As a software organisation in the pharmaceutical business, DevIS must address many quality requirements. One fundamental requirement is that DevIS must be able to show the authorities, by documented evidence, that software development activities (e.g. software change control, software validation, and data processing and storage) are being performed in compliance with quality requirements. Therefore every software project regulated by GXP requirements should carefully apply all quality rules and be able to show by documented evidence that the software is compliant with the related GXP requirements. The company long ago adopted standard operation procedures that explicitly describe the company's software quality rules. These standard operation procedures should be applied for all information systems regulated by GXP requirements.

Employees of DevIS are basically engaged with software development, software maintenance and software operation activities. The software development activities occur in two forms: 1) development of totally new software products (software development) and 2) developing or changing existing software products (software maintenance). A typical software development project at DevIS is scheduled to take between six months and one year and includes analysis, design, construction, testing, and validation. Software maintenance activities can consist of changes in the code or developing a completely new application for existing software products. Software products in DevIS include the software and all related documentation (e.g. user requirement specification, test plan, validation plan, validation report, user manuals etc.).

3.2. The Problem Area

The results of a problem analysis performed in early 1999 in one of CRIM's largest software development groups showed a need for improving software project disciplines and providing guidelines to understand the standard operation procedures and GXP rules. The director of DevIS initiated an improvement project called Software Process Improvement at CRIM (SPIC) (whose name was changed to SPID after the merger (Software Process Improvement at DevIS)) to understand the existing problems and improve the organisation's software processes. The following figure illustrates a rich picture of the SPID project.

The SPID project was initiated, organised, planned, and performed during the period of April 1999 to May 2000 aiming to improve DevIS's software processes. A maturity assessment using a modified CMM-based (Capability Maturity Model) assessment method, QBA (see Arent and Iversen 1996), showed that DevIS was by then a level one organisation and addressed improvement possibilities in all analysed KPAs (Key Process Areas). An improvement report based on the assessment's findings and other findings from earlier improvement initiatives at DevIS addressed six improvement activities. The steering committee of SPID gave priority to the following improvement activities (improvement decision) from the improvement report:

- To establish a minimum documentation level for documenting the results of software projects and create the software documentation process.
- To improve processes for software validation, software change management, and document version control.
- To create a template library including templates for documentation of software development activities, such as: user requirement specification, design specification, test plan, and validation plan.

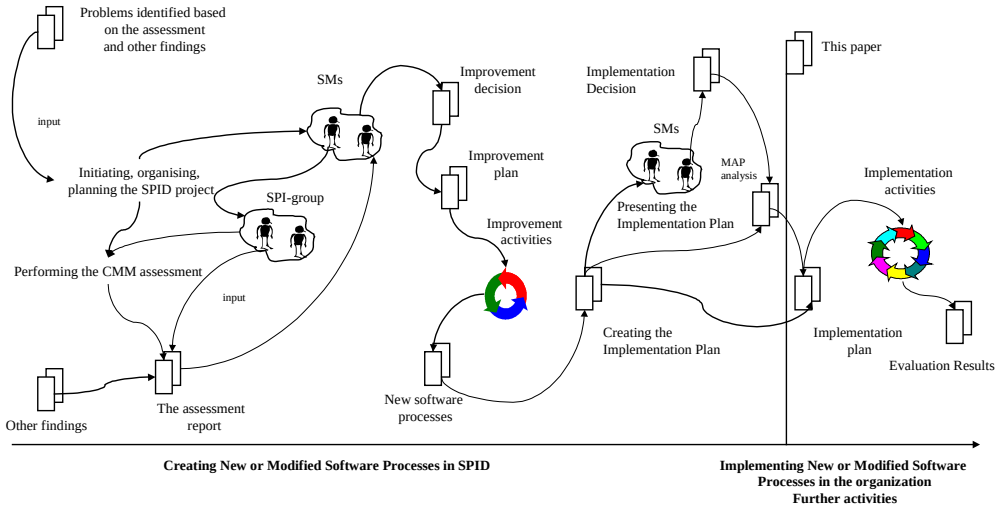

Figure 1: *The rich picture of SPID (Checkland 1990)*

An improvement plan was created and the SPI group (including the author, two SPI consultants and five software engineers) started planning and performing improvement activities over a period of four months, which resulted in creation of new software process guidelines. An implementation plan was then created and presented for the steering committee. The steering committee of the project accepted the new software processes and decided to implement the newly created processes throughout all of DevIS. The implementation activities are scheduled to be arrayed out between August 2000 and June 2001. The implementation activities include among others a trainee program for all practitioners at DevIS and aim to change the context in which the new software processes will work. The implementation phase also includes further improvement activities in which the created processes will be improved on the basis of experiences of using them in practice. This phase will result in a new version of the software process guidelines in June 2001.

4. THE FINDINGS

In this section we discuss the most characteristic features of SPI affecting the SPID project on the basis of the MAP (Aaen *et al.* 2000).

4.1. The Management of SPID

SPID was organised as a project with specific budget and resources. The initial improvement infrastructure of the project was established early in the project, and the roles and responsibilities in the infrastructure were described. The organisation of SPID consisted of: one project manager (the author of this paper) responsible for co-ordinating, planning and performing the project; a steering committee including the software managers and the director of DevIS responsible for allocating resources to the project and deciding on the acceptance of the results of the project; a reference group including software engineers and project managers responsible for giving input to the software improvement activities for creating the new software processes; and a working group including the SPI consultants responsible for documenting the newly created software processes. An SPI plan was created to define the deliverables, milestones, schedules, and goals of the project. Feedback on improvements was not defined explicitly. Because, the reference group was consisted of experienced project managers who could give feedback on the improvements as we went along. It was simply mentioned in the SPI plan that the results of the project (the new software processes) should be tested in two software projects before implementation in the entire organisation.

Organising the SPID as a project gave us the possibility to allocate resources to the project as with any other project at DevIS. This helped to gain the management's commitment to the project during the project's entire life span. It further helped us to structure the project's organisation and the responsibilities. Organising SPI initiatives as regular software projects has been supported in earlier work by (Johansen and Mathiassen 1998, Aaen *et al.* 2000, Zahran 1998, Arent and Norbjerg 2000). Organising SPID as a project brought risks as well. People working in the reference group were very busy with other projects at the same time and this sometimes meant that someone could not deliver what he/she was supposed to deliver at a meeting. However this was a minor problem because the practitioners commitment to the project was very high. Another risk related to organising SPID was co-ordinating the resources and the meetings both with the management and the improvement team. This occurred because the action demanded a grate deal of planning time. Because the highly commitment of the project manager to the project this risk could be managed during the project's life span.

On the other hand, planning the project helped us to understand: what to do, when to perform which activity and who should do what in the project. This helped us to focus on our schedule and the deliverables. But, on the other hand, this sometimes caused stress, especially for the project manager because he had the pressure on himself to deliver results on time. Another risk of detailed planning was that we sometimes felt that we were prisoners in the schedules and deliverables, although we did not let our thoughts and ideas be stopped or disturbed by the depth of the plans. We tried to "reflect-in-our-actions" (see Schön 1987) all the time and changed some plans and some deliverables a couple of times. These changes were either necessitated by other ongoing improvement activities in the company, which had effects on our project, or because we felt that some changes in a specific plan and its deliverables would provide better results for the project in the end. Our plan was a framework for action rather a procedure to follow in detail. This helped us in not losing our motivation, but this is still a pitfall that might cause problems during the implementation activities. Therefore we will create some flexibility in our implementation plan for not loosing motivation.

Not defining the feedback on the improvements in detail actually did not affect the initiation of the SPID. We spent time on other issues, but we missed some feedback during the improvement activities in terms of knowing whether we were on the right track and we could not measure and document quality and progress. Of course we had five project managers in the reference group of the project. This created in short run a checking mechanism, which controlled whether we were on the right track. But in the long run the risks of not defining feedback might caused problems such as uncertainty and even misdirecting the whole project and risk for not getting the desired results.

4.2. Approaches to the SPID

In SPID we decided to improve a few software processes in an evolutionary way by taking one step at a time. The goals were neither to reach any maturity level in the CMM nor to improve several software processes. To understand the current level of software process problems we adapted and performed a modified CMM-based assessment, which helped us to identify our software process problems in a structured way. Because the goal of SPID was not to reach any maturity level in the CMM we decided not to follow the CMM recommendations in our improvement activities. We rather let ourselves get inspiration from the concept of SPI, i.e. improving software processes in a systematic way, than tried to fulfil the CMM's requirements. We based our improvement strategy on the assessment's findings, other quality goals within the organisation, and practitioners' and management's commitment to the project. Without the management's commitment we could neither start the SPI initiative nor get the necessary resources for the project and without the practitioners' commitment to the project we could never have constructed a creative forum for discussion and improvement of the new software processes.

Having an evolutionary approach in SPID helped us to concentrate on a few software processes. We could see the whole picture and did not get lost in the complexity of having several software processes to improve. We could manage the situations well and had time to reflect in our actions to improve our SPI practice. The risk with focusing on a few software processes was rather that the software processes focused on were related to other software processes and software development models. This caused many hours of discussion

to separate other issues from our main scope. We still, however, needed to remind each other of the scope of the software processes several times during our project. There were risks all the time that we might burnout and get tired of maintaining our commitments in having an evolutionary approach to our improvement activities. Another risk with having an evolutionary approach to our project was that, because the improvements were not anchored and maintained as part of the daily practices, the performance increases were limited and invisible for us during the project. Because we could not measure the extend to which these processes were useful in the daily work of the practitioners.

Not aiming to reach a maturity level in the CMM led to a positive reaction among both the management and the practitioners. Because the goal was not just to reach an "abstract" target (reaching a "level" in a model, for management and practitioners at that time) but, solving the organisation's problems by being inspired by a well-known model, which was modified to suit the organisation's situation. We spent much time on modifying the CMM-based model and creating an institutionalisation strategy. This created more motivation and enthusiasm among practitioners and management and strengthened their commitment to the project. But on the other hand, not following the CMM's recommendations during the improvement activities led to creation of an incoherent improvement environment. We had no long-term improvement visions for further improvement activities after SPID. But as a novice organisation we did not wanted to commit ourselves to a long-term vision at that time, when we did not know enough about the CMM and the SPI. This actually did not affect the initiation of the project neither the improvement activities.

One key factor in succeeding in creating new processes was that the management and the practitioners believed in SPID and were concerned about the results of the project. Practitioners working with SPID were almost all persons with in-depth experience in leading software projects and wrestling with software process problems. They knew the importance of solving the software process problems, and this made them very committed to the project and involved in working hard to create processes that could function in practice. This created a good platform for implementation activities. Even if the commitment process was vital for SPID, it could be carried too far. We could become so dedicated to solving problems that we could lose sight of the original goal. This could further lead to a loss of perspective on the long-term improvement program and to the gold planting of solutions to current problems. Therefore we kept reminding each other of the goal of the project during our meetings all the time.

4.3. Perspectives in SPID

At the time this project started DevIS did not have a detailed description of all software processes. This meant different interpretations of any given simple software process activity in the software projects. For instance, the practitioners knew that a software product should be validated before being put into operation, but the interpretation of the software validation process was different in different projects. We knew from earlier improvement activities that a description of the software documentation process was missing in the organisation. Knowing which documents should be created as products of the software projects could help us to identify the activities needed to create them. It could further help us to identify the main processes needed for supporting these activities. We therefore adapted a product perspective in the beginning of the project and focused on identifying the documents needed as the results of a software project. On the basis of this, we identified actors and activities needed for creating each document. After defining the software documentation process we changed our focus and concentrated on processes that should be improved (software validation and software change control). By focusing on the processes we could integrate people, management, and technology, (see Aaen *et al.* 2000). During the improvement process the practitioners' competencies, ideas and experiences were the main input to the improvement work. But we could not empower all practitioners through competence building at that time. Competence building will be a key part of the implementation activities. Within the improvement phase we established suitable efforts and changed a few software processes and created templates for documenting the results of software activities. Because the improvement phase of the project ended before the implementation phase we did not need to change the context in which the software processes should be operated. Within the implementation phase based on newly created software processes a supporting infrastructure for the software processes will be developed.

The risk in focusing on software processes was that these processes were related to the other processes, the software development models, and the software project management models. It required many hours of discussions to separate different issues from the project's main target and focus on the defined goals. Another risk with focusing on processes was that we could have lost the customer perspective and forget their needs. But we had the project managers in the reference group, which even could act as representatives for customers (practitioners who should use the software processes). The benefit of focusing on documents as software product was that we agreed upon the documentation as one type of software products. This helped us later to both satisfy the customers which had requirements on the products of the processes (documents) and identify the activities and processes needed to be in place to support the creation of these documents and the software.

All the improvement activities in SPID were based on the practitioners' ideas and experiences. One problem of focusing on practitioners' competencies for improving software processes was that the whole project was dependent on their input. If a majority of practitioners was not able to join a meeting we cancelled the meeting and had to wait until the next time. This problem has been identified by Johansen and Mathiassen (1998). Another problem was that the practitioners had different experience from different software projects. This caused variations in interpretation of any specific software process activity. Much time was needed to discuss different views and experiences related to one specific software activity or process. However the benefit was that we knew that all the different issues discussed had already been put into practice and shown some degree of usability. On the other hand, we could only empower the project members through competence building. The other practitioners were not actively involved in improvement activities. This will occur within the implementing phase in which the newly created software processes will be implemented in the organisation. One risk in limiting the competence building within the project was that other practitioners might not commit themselves in using the newly created processes. To reduce this risk our implementation strategy includes a training program for all practitioners for giving them the knowledge they need for using these processes. We also will create a supporting infrastructure, which will help the practitioners in practice use of the processes in the software projects. This might even experienced by the practitioners as a controlling function or a machine bureaucracy.

5. LESSONS LEARNED

We have analysed an SPI project based on the most characteristic features of SPI, (see Aaen *et al.* 2000). On the basis of our findings we will argue that performing such analysis may help a novice organisation to understand the nature of SPI activities and further identify the risks involved in such an initiative. In this way the organisation may have a better chance to succeed in planning, organising and running its first SPI initiative. From the perspective of this framework we believe that: a novice organisation has a chance to succeed with its first SPI initiation if the organisation emphasis learning from the concept of SPI, and focuses on a few carefully selected software processes to improved. This study suggests a number of lessons relevant for future SPI projects and the SPI practice.

Lesson one: *The first SPI initiative should aim to learn the concept of SPI and the software process problems.* The first SPI initiative in a novice organisation should have the character of being be a learning process in which the organisation learns about its current software practice and the SPI approach in general while improving software processes. Doing this helped us to critically reflect in SPI activities and learn the concept of SPI by doing it in practice.

Lesson two: *A novice organisation should focus more on the SPI concept than the CMM recommendations.* The first SPI initiative should start by diagnosing the current maturity level of the organisation. The CMM can offer much help in doing this. However for improving the software processes, it is better to rely on the organisation's goals and focus on the SPI activities and learn to improve a few software processes based on the organisational goals rather than just following the CMM as a model. This helped us in gaining management and practitioners' commitment to initiating the project and focusing in learning the SPI as a systematic concept for improving the software processes. But on the long run this will lead to lack of long-term vision for software process improvement initiatives.

Lesson three: *The first SPI initiative should be organised as a project, with specific goals, deliverables, and recourses aiming to improve a few software processes in an evolutionary way, on the basis of the organisation's needs, ideals and practitioners' ideas.* Organising the SPI initiative as a project and having a flexible plan as a framework for conducting the activities was essential for managing improvement activities in our first SPI initiative. For guiding the SPI improvements it is much more important to focus on stepwise improvement and satisfy the organisation's goals than to follow an abstract goal. This supports gaining the management and practitioners' commitment to the project.

Lesson four: It is highly recommendable to conduct a MAP analysis early in the project to identify the most characteristic feature of the SPI initiative, and the risks related to the SPI activities. Doing such an analysis before starting the project based on the MAP see (Aaen et al. 2000) will help the novice organisations to better understand the SPI effort and identify the risks related to the project. Such an analysis should be completed by a risk analysis to calculate the importance of each risk in relation to its probability to happen and its effects on the project if it happens. This will help in better understanding the concept of SPI and planning for taking care of the possible risks in the project.

The three first lessons correspond with (Aaen et al. 2000, Johanssen and Mathiassen 1998, Zahran 1998, Pourkomeylian 2000, Arent and Norbjerg 2000). However, SPID did not adapt the whole MAP in detail. But as the first SPI initiative it is most essential to be focused on delivering results that are visible to the management in the scheduled time. This will ensure dedication and legitimacy for the continuous SPI efforts in the organisation. As a success criterion it should be mentioned that the steering committee of SPID accepted the newly created software processes and decided to implement these processes in the entire organisation. The implementation phase in SPID starts in August 2000.

6. ACKNOWLEDGEMENT

This research was sponsored by AstraZeneca R&D Mölndal in Sweden in collaboration with the Viktoria Institute Gothenburg in Sweden and the Institute of Computer Science Aalborg in Denmark.

REFERENCES

Aaen I., Arent J., Mathiassen L., Ngwenyama O., (2000). A Conceptual MAP of Software Process Improvement. Artikelsamling, Center for Softwareprocesforbedring. Dansk Elektronik, Lys & Akustik.

Arent J., Iversen J., (1996). Development of a Method for Maturity Assessments in Software Organizations based on the Capability Maturity Model, in Department of Computer Science. Aalborg: Aalborg University.

Arent J., and Norbjerg J., (2000). SPI as Organizational Knowledge Creation: A Multiple Case Analysis. Proceedings Conference Proceedings at Maui Hawaii.

Checkland Peter, Scholes Jim (1990). Soft System Methodology in Action, John Wiley & Sons LTD. England.

Curtis, B., (1994). A Mature View of the CMM. American Programmer, 7, 9, 19-27.

Goldenson, D. R., and Herbsleb J., D., (1995). After the Appraisal: A Systematic Survay of Process Improvement, its benefits, and Factors that Influence Success. CMU/SEI-95-TR-009, SEI, Pittsburgh.

Herbsleb, J., *et al*. (1997): Software Quality and the Capability Maturity Model. Communications of the ACM, 40(6), 30-40.

Humphrey, Watts S., (1998). Managing the Software Processes, Addison-Wesley Publishing Company, USA.

Johansen, J., and Mathiassen L., (1998). Lessons learned in a National SPI Effort. EuroSPI 98 Göteborg.

Kuvaja Pasi, Bicego (1994). BOOTSTRAP - a European assessment methodology, Software Quality Journal 3, 117-127.

Mashiko, Y., and V.R. Basiili (1997). Using the GQM Paradigm to Investigate Influential Factors for Software Process Improvement. Journal of Systems and Software, 36. 17-32.

McFeeley Bob (1996). IDEAL: A User's Guide for Software Process Improvement, Software Engineering Institue, Cornegie Mellon University.

Paulk Mark C., *et al*.(1993). The Capability Maturity Model for software Version 1.1, Carnegie Mellon University, Software Engineering Institute.

Pourkomeylian P., (2000). Knowledge Creation in Improving a Software Organization. Proceedings of IRIS 23. 163-180.

Schön Donald A. (1987). Educating the Reflective Practitioner. Jossey-Bass Publishers . San Francisco.

Tryde, S., Nielsen A.-D., & Pries-Heje J., (2000). A Framework for Organizational Implementation of SPI in Practice. In: L. Mathiassen *et al.* (Editors): *Learning to Improve.*

Zahran Sami (1998). Software Process Improvement, Software Engineering Institute, SEI Series in Software Engineering, Addison Wesley Longman.