Scandinavian Journal of Information Systems

Volume 13 | Issue 1 Article 5

2001

Assessing Software Processes: Low Maturity or Sensible Practice

Peter Axel Nielsen
Aalborg University, Denmark, pan@cs.aau.dk

Jacob Nørbjerg
Copenhagen University, jmen@diku.dk

Assessing Software Processes:

Low Maturity or Sensible Practice

Peter Axel Nielsen[a] and Jacob Nørbjerg[b]

Department of Computer Science, University of Aalborg[a]

Department of Informatics, Copenhagen Business School[b]

Abstract

Software Process Improvement efforts rely heavily on the use of software capability maturity models. These models are used to identify problems in an organization's software processes and point to, and prioritize, necessary improvements. Using models for this purpose will, however, automatically turn our attention to certain problems and issues and ignore others. In this paper we argue that the maturity models provide only one perspective on software processes and software process problems. We demonstrate how we, by looking at the organizational context of and the conflicts and uncertainties facing software projects may uncover alternative interpretations of software practices, and identify other problems. The implications for SPI and for the use of maturity models are discussed.

Keywords:

Software process improvement, CMM, organizational change, organizational politics.

1.Introduction

Organizations developing computer-based information systems struggle with unexpected incidents, unstable requirements, staff shortages, rapid technological developments, unclear or conflicting statements from users and customers. These and other problems contribute to uncontrollable projects, runaway budgets, and delayed products of inferior quality.

Over the years software professionals have come to accept this as unavoidable conditions for information systems development but they have also realized that systems development organizations must improve their ability to control these risks and uncertainties and manage their software projects. As a consequence the field of Software Process Improvement (SPI) developed. Integrated in SPI is the concept of - software process - maturity; a measure or expression of the strengths and weaknesses of an organization's software development and management practices. Maturity is defined through so-called maturity models; e.g. the Capability Maturity Model or CMM (Paulk et.al. 1993) and Bootstrap (Kuvaja et. al. 1994). The models define maturity in terms of levels, where each level is described through a specific set of management and development practices and procedures. The higher the maturity level, the better the expected, overall performance in terms of accuracy of schedules and budgets, product quality, and productivity. The models support SPI initiatives in two ways: First, they can be used to assess an organization's maturity level by comparing the organization's development and management practices with the model, and second they can help prioritize improvement initiatives by pinpointing the changes needed to reach a higher maturity level.

The idea of software process maturity emerged at the Software Engineering Institute (SEI) in the early 1980s (Humphrey 1989), and the institute's Capability Maturity Model (CMM) is among the first and most influential of the maturity models. The CMM describes maturity in terms of five levels, from level 1 (Initial), characterized by the absence of even basic project management practices to level 5 (Optimizing) with organization-wide management and development practices and extensive use of process and product measurements to monitor and continuously improve performance. The model, which describes the levels in great detail to reduce ambiguity, was developed by the SEI in close collaboration with the US software industry. It is accompanied by training programs for assessors as well as detailed guidelines and procedures for assessments and maturity evaluations (Dunaway & Masters 1996, Paulk et.al. 1993).

The significant contribution of the CMM is its embodiment of an improvement strategy. According to the CMM an organization at a certain level shall focus on the improvements needed to reach the next higher level first; thus, an organization at level 1 should implement the basic project management practices specified at level 2, before attempting to undertake the more demanding implementation of organization-wide development and management standards required to reach level 3 and so on.

The SEI developed the CMM on a direct contract from the U.S. Department of Defense (DoD) that wanted an instrument to evaluate potential contractors, and the DoD now requires software vendors to be at level 3 or higher for large or complex projects before they can be considered for DoD projects (Saiedian & Kuzara 1995). This has of course created much interest in the CMM especially in the US software industry, particularly as early assessments indicated that over 80% of US software organizations were below this level (Goldenson & Herbsleb 1995).

Other organizations in the US and elsewhere who are not involved in DoD contracts are, however, also using maturity models as a way to improve their software process capability. This, in turn, has lead to increased interest in maturity models and a number of competing models have emerged during the 1990's; e.g., the Bootstrap project funded by the European Union (Kuvaja et. al. 1994), and the SPICE model (Enam et. al. 1998), intended to encompass all previous models. SEI's CMM has had significant influence on the concept of maturity and all the subsequent models, however, and we will use this model as our "reference" in the rest of this article.

Considering the size and importance of the software industry, it is no wonder that there is considerable debate and research about the concept of maturity in general and the maturity models in particular. Some researchers suggest to improve the maturity models by adding new sets of practices and procedures; e.g. Sawyer et. al. (1997), suggest a three-level maturity model for requirements processes, and others suggest to include organizational learning and knowledge management capabilities in the models, since these

seem to be related to software process maturity (Baskerville & Pries-Heje 1998) or to the success of improvement projects (Stelzer et al. 1998).

Another – more fundamental – discussion concerns how maturity models and maturity assessments are used to guide SPI. The maturity assessment process may itself be flawed due to immature assessment techniques, the questions used or the training of assessors (Bollinger & McGowan 1991, Smith et. al 1994, O'Connel & Saiedian 2000) and maturity models may not be applicable across different types of software producing organizations or in all countries (Baskerville & Pries-Heje 1998, Edgar-Newill 1994, Mathiassen & Sørensen 1996, Sharp et al. 1999, Velden et al. 1996). Some authors have gone even further and challenged the maturity idea as such. High maturity levels require, according to the founding fathers of the models, documented processes and extensive use of process and product metrics to control and continuously improve performance, but some critics argue that today's successful software production depends on innovative capability, creativity and the ability to adapt to a rapidly changing environment, not on standardized processes and detailed measurements. Hence higher maturity levels may actually be harmful instead of beneficial to a software organization (Bollinger & McGowan 1991, Bach 1994, Bach 1995, Kohoutek 1996).

The debate about the maturity models and the maturity concept has, however, suffered from a general lack of systematic research into the theoretical and empirical foundations of the maturity concept and the maturity models. Such research should aim at establishing the models' applicability and validity more firmly.

In this paper we will show how the maturity models represent only one possible perspective on software processes The models embody basic software engineering ideas of sound development and management practices which can be found in any textbook on software engineering; e.g. (Pressman 2000, Sommerville 2001). Thus, a maturity model based assessment of an organization's software processes, will focus on the presence – or absence – of these practices and ignore other aspects of software processes. We will argue, however, that by looking at the software processes in their organizational context and drawing on theories of organizational conflicts and politics we can draw another – and equally valid – picture of an organization's software processes and how to improve them.

We evaluated the practices of a group of software project managers (PM) in a small software producing organization from the point of view of the CMM. This did not constitute a complete maturity assessment of the organization but the PMs' accounts of their own practices are sufficiently complete to allow us to assess their practices as immature (level 1) using the concepts and criteria from the CMM. If we include the organizational and technical conditions under which the PMs work in our evaluation of their practices, however, then we come to see these as both sensible and rational ways to cope with the contradictions and uncertainties the PMs must deal with in their day-to-day effort to complete their projects.

By further relating our observations to other studies of organizational behaviour in software development we show that the PMs' practices, and the organizational conditions underlying them, are by no means local or incidental but caused by fundamental structural conflicts and contradictions that can be found in most organizations. By drawing on general theories of organizational conflicts and politics we finally argue that this organizational perspective on software practice questions the feasibility of the improvements recommended in the CMM based evaluation.

2.The research approach

The research results presented here build on data collected through a three year action research project in a Danish software organization. Action research is an approach that through active intervention simultaneously attempts to achieve practical value for a client organization and to contribute to scientific knowledge (for more details about action research, see f. ex. Checkland (1991) or Avison et al. (1999)). The project lasted from 1997 to 1999 and through this period the authors, together with other researchers and external consultants were engaged in the organization's SPI project. We participated in regular meetings in the internal group responsible for the SPI project and cooperated with developers and project managers in specific improvement initiatives. We have documented the research by tape-recordings and minutes of all meetings where we have been present, by tape-recorded interviews with project managers and middle managers, and in field notes and diary entries. The documentation also consists of reflective papers

written during the action research project (e.g., Iversen et al. 1999).

The primary data source for this article are interviews with 7 out of 10 project managers about their software development practices. The interviews took place from June to August 1997. Following Patton's (1990) techniques for qualitative interviewing we used an open-ended interview guide and the interviews were tape-recorded and subsequently transcribed. The original purpose of the interviews was to identify the project managers' perceptions about the company's software process problems (Iversen et al. 1999) and much of the interview text therefore deals directly with process problems. Our present focus is however on the organizational and political aspects of software processes and we have therefore re-read the interviews from this perspective, working closely with the text trying to let it speak for itself. Each of us has carefully read the text and noted what we found to be significant. We have compared our notes and settled on categories of issues. We have then read the text again looking for quotes confirming or disconfirming our categories. The final categories and the quotes illustrating the categories form now the basis of the case description. We will not claim that this resembles an impartial grounded theory approach as put forward by Strauss & Corbin (1990), because our long-time engagement with the company may well have lead us to form interpretations that others would not have been able to.

Secondary data sources come from our continued action research effort after the interviews. This has allowed us to closely observe the project managers in other situations, to view their actions in a long-term perspective, and to work with several viewpoints on how to interpret their perceptions and their actions. We have at many occasions tried to influence project managers and others; e.g. top-management as well as developers and this serves as a context against which we judge our interpretation of the interview text. The context helps us to triangulate our findings and it provides a broader perspective for making sense of the project managers' statements.

3.The Case

The interviews took place in the development department of a company that develops leading edge measurement instruments and systems. A typical product consists of one or more measurement instruments - microphone, thermometer, accellerometer or other, sometimes with embedded software – connected to a PC with analysis and presentation software. Most projects have both a hardware and a software part but they are run as integrated projects under a common project manager. Larger projects can be divided into separate hardware and software sub-projects, each with their own project manager.

The department is managed by a technical director who reports directly to the board. The department and the projects work closely together with marketing, sales, and production departments.

The company had been through a long re-orientation and downsizing process due to increased competition and set-backs in one of the company's major markets, prior to our engagement. Immediately before our entry the development division went through a Bootstrap assessment where it was concluded that most of the company's software process problems concerned project management, configuration management, testing, the development process model, and requirement specification.

In the following descriptions of the project managers' practices we will focus on three broad process areas: estimation and planning, resource allocation, and requirements management. These areas include a major part of the project managers' responsibilities, they are discussed in depth in most of the interviews and they vividly illustrate the uncertainties and organizational contradictions they must cope with.

3.1 Estimates and Planning

The company does not maintain a database of hard experience data to support the estimation and planning of the projects. The project managers therefore base their estimates and plans on past experience and the judgment and estimates of team members. The project managers use the plans to get an overview and control over activities and deadlines. They do not, however, expect schedules to hold and they know they have to reschedule often.

The planning process is however often disturbed by directives from management, time pressure and uncertainty about the requirements as illustrated in this extract from an interview with an experienced project manager:

> "The time schedule was decided beforehand: We had to finish by a certain month so I really didn't have [estimation and planning]

> problems. I made some rough plans for the requirement specification phase. [Some] were right, others were completely wrong. So I stopped doing that. The requirement specification phase has been much longer than planned. We should have been ready by April/May but were not completely finished until August. … I've used a lot of time on setting up schedules in this project; [we've] agreed on some rough plans; and then we've tried to navigate from those [plans]."

This project manager might not agree with the original schedule set by management, but he did not openly discuss it. Furthermore, his own original estimates for the requirements specification phase proved to be far too optimistic. He managed, however, to re-schedule and control the project anyway and at the time of the interview he was confident that he would deliver within one month after the originally set date. This estimate proved to be right.

Other project managers are less fortunate and struggle continuously with unrealistic schedules:

> "We got five days to write the requirements specification for [X]. That sort of set the stage for the whole project. We had thought of at least four months to [investigate] different types of users and verify concepts; but no, [the management] thought that we could save a lot of time by declaring 'You've five days'."

The project actually did manage to write a specification in five days, but it was very superficial and the project never came to believe completely in it, according to the project manager.

Project X was originally scheduled to produce a version 1.0 of the product in 10 months. It took the project group 16 months to produce a beta-version and another three months to complete version 1.0.

The above examples show that the estimation and planning processes are heavily influenced by the product delivery plan produced by the technical director together with the other directors. The plan outlines features of next years' products and tells the sales department when demo versions of new products are to be ready, when they can begin selling the product, and when delivery actually will take place. The schedule lists the products' general features and a first step in a project is to write a more detailed product description based on this feature list. The following extract from an interview with a project manager shows an important aspect of this.

> "... [management] announces next year's products before there is a requirements specification or anything. That means that we launch projects based on mere headlines because we have to."

The project managers realize that this is a perfectly reasonable approach from a marketing and sales point of view. Without a launch schedule the sales representatives would be unable to plan their sales efforts and cater for the customers' needs and demands. It is a difficult situation to bring the project managers in, however. On the one hand they understand the need for a product launch schedule, but they cannot make their project plans based on it. Even worse, they are required to estimate a development project from product feature "headlines" only.

When a project is underway, there is immense pressure to deliver on time. The technical director is under pressure from the other divisions and from top management and he presses the projects to deliver as expected. The sales representatives on their side are under pressure from the market because of their commitments to deliver to the customers according to the launch schedule.

Summarizing the above we can say that the project managers really try to estimate and schedule their projects; they know basic planning and management techniques and estimate, as well as possible, based on past experience, but they find that this doesn't help them cope with the management problems they face. The result can be a rather cynical attitude towards planning as expressed in the following quote from a manager of a delayed project:

> "You gave your best shot and after some time you're "beyond time schedules" because it didn't matter, you just had to complete."

3.2 Resource Allocation

Resource allocation in the company concerns two issues: what projects to start and the allocation or removal of manpower called resources to or from projects.

The company faces a severe shortage of software developers with the specialized knowledge about measurement instruments and analysis algorithms required to develop the company's products. There is therefore an ongoing struggle between project managers to have enough developers with the right qualifications allocated to projects.

> "When we started we were two technical project managers … one for Windows software and one for [signal processing] software. There were eight [developers] on Windows software and five on [signal processing]. That was 20 less than what we'd asked for."

Project managers find other ways to deal with the shortage. Developers cannot be allocated to a project that is not officially started, but some project managers are nevertheless able to create so called 'drawer projects'. A 'drawer project' is a project initiated by a project manager without explicit approval from the technical director. If the project manager eventually succeeds to obtain formal support for the project it will emerge from the project managers drawer and officially start. Several of the project managers we interviewed reported that their – now officially approved – projects were begun as drawer projects.

> "It has been running since … it's one of the political spheres … in principle … officially it was started a month ago [July 1997]. Unofficially, it … started in late February."

It is of course almost impossible to apply proper processes in drawer projects as they are a cover-up of what is actually going on.

It is not clear how project managers manage to allocate resources to drawer projects, but it appears that at least some of these projects have been able to obtain a semi-official status. The following extract concerns a project that had been running for years and had produced several versions of the product without being officially defined as a project, and without a plan and estimates:

> "Interviewer: To go on with this project you needed project team members with particular qualifications. These qualifications were not available in the market and you couldn't move people around in the company. You then [hired] new people and trained them for half a year. Then they were ready to enter the project.
> Project manager: That's exactly the situation here."

The project managers perceive the resource allocation process as sometimes arbitrary. One project manager reports about his difficulties getting his project underway:

> "You see, it was a direct order from management on how to prioritize.[Unfortunately], our key programmer was tied up in maintenance work ... so there was very little work done until after summer."

To overcome this problem the project manager had to convince the technical director to reallocate developers from other projects. This is always a problematic thing to do partly because it may hurt relations to other project managers, partly because the developer himself may resent being moved.

A final quotation illustrates what another project manager perceives as a paradox concerning resource allocation: producing an estimate requires manpower that he can't get without an estimate.

> "[Getting resources] ... it is like a vicious circle. If there isn't anybody on the project, then ...it's hard to make an estimate. And without an estimate you don't get any people, right?"

3.3 Requirements definition and management

The company's official project model prescribes that projects produce a requirements specification to be approved by the project's steering committee. This committee is also supposed to approve all later changes to requirements. There is, however, considerable variation in how individual projects find, describe, and later change and manage requirements. At the time of the interviews a few projects were in the process of adapting a new requirements definition process with systematic use of scenarios, use cases and prototypes. This approach spread gradually to other projects over the following years and was incorporated in the company's quality system by 1999. At its best therefore, requirements definition is based on project seminars to brainstorm ideas, peer reviews, visits to customers, and testing prototypes internally and at customer sites. The project managers reported that this could be a time consuming process, but that it resulted in detailed, and very useful specifications.

Other projects take requirement specification more lightly:

> "It has been characteristic for this project from day one that it has been absolutely informal. It is probably the biggest 'drawer project' ever. … [Through the whole project] there has been no formal requirement specification."

The project concerns a system to support certification of audio devices according to five different national standards. It never produced a requirements specification, but used the national certification standards as a substitute and managed to produce several releases of the system. The standards do not, however, mention central aspects of a computerized system, e.g., the user interface.

The pressures put on the projects to meet deadlines mean that they are not always able to systematically manage requirements. They perceive themselves as being under the combined pressure from the technical director who wants them to meet the set dead-line and a marketing department that tries to push as many features as possible into the product without concern for time and development costs. For the project managers it becomes more important to meet deadlines with a functioning product than implementing all requirements and they are therefore prepared to simply strip requirements.

> "Interviewer: What happens when you approach the dead-line?
> Project manager: You'll have to adjust your own ambitions for the project. How many nice-to-have features do you throw away?"

Requirements changes are formally approved in the steering committee as noted above. In practice, however, the project managers assume responsibility for requirements changes, eventually with subsequent approval in the committee. The following project manager thus displays extreme confidence in his overview of the market and his ability to make the right decisions regarding requirements:

> "No, [the changes] are approved by the project manager. I take the decision. I don't necessarily go out and ask a large part of the market … OK, I have the insight into market needs that it requires. "

The project managers partly blame this practice on the delays inflicted by the slow and bureaucratic approval process. To avoid these delays the project managers might also simply postpone the formal requirements review and sign-off procedure for as long as possible.

4. Evaluating practice

The practices described above are no doubt problematic. However, as we will show in this section, the interpretation of the practices, the problems identified, and the solutions suggested, depend heavily on the framework used to assess them.

4.1 A CMM perspective

The CMM defines maturity in terms of five levels as described earlier. Each level describes a set of development practices and management procedures that must be in place and followed for an organization to qualify for that maturity level. CMM level 2

describes processes within the following six general areas

- software project planning
- requirements management
- software project tracking and oversight
- software configuration management
- sub-contractor management
- software quality assurance.

The descriptions of the project managers' practices in the previous section fall within the first two of these and a comparison with the specification of CMM level 2 reveals severe weaknesses within both areas - the following evaluation is based on Paulk et al. (1993), pp. 59-60):

Software Project Planning

The CMM requires that projects follow a systematic and documented estimation and planning process, and that all parties commit themselves to approved plans and schedules. The plan and the schedule should include available and allocated resources.

This requirement is, as we saw above, not met in the company's software projects. The project managers do make estimates and plans, but they see their effort as being confounded by fixed delivery dates, and insufficient resources, both with regard to time and people, to perform requirements analysis and subsequent estimation and planning in a proper way. Furthermore, resource allocation seems to be ad hoc and political and not linked to estimates and schedules.

As a result, the project managers themselves don't believe in the – documented – plans, but simply try to deliver as soon as, and when possible. In this process they may rely on their own, unofficial development plan and their agreements with developers.

Requirements Management

Software requirements should be documented and approved and all changes to requirements should be carefully controlled, according to the CMM. Changes to requirements should be reflected in plans and schedules.

We described above how not all projects document and control requirements. Some projects don't have a requirements specification while others keep the specification fluid for as long as possible. Requirements may be changed or removed at the project manager's discretion without previous approval from the steering committee.

Thus, a CMM assessment of the company's software processes would result in recommendations for considerable changes within these two areas [1]. One thing in particular would need to be improved: namely the way commitments are established and maintained. The concept of commitment forms the basis for several of the CMM's recommendations and is intended to capture the mutual agreements about; e.g. plans, resources, schedules etc. that actors in software development projects make. Humphrey explains the elements of making a commitment in this way:

1. The person making the commitment does so willingly.
2. The commitment is not made lightly; that is, the work involved, the resources, and the schedule are carefully considered.
3. There is agreement between the parties on what is to be done, by whom, and when.
4. The commitment is openly and publicly stated.
5. The person responsible tries to meet the commitment under all circumstances, even if help is needed.
6. Prior to the committed date, if it is clear that it cannot be met, advance notice is given and a new commitment is negotiated (Humphrey 1989, p.70)

In the CMM these elements are operationalized in the form of several practices and procedures. It is obvious that commitments are not made and maintained in this way in the company. The project managers are for example ready to accept a fixed delivery date or a project schedule based on "shaky grounds" and to work according to their own 'unofficial' plan until it becomes obvious that the 'official' plan is unrealistic. Upper management, on the other hand, will not commit itself to the project managers' schedules but enforces a fixed deadline.

4.2 Organizational practice

The CMM's assessment and advice is sensible and useful from a pure software engineering perspective:

The company's projects do suffer from time and budget overruns and there are defects in the products. So there is room for improvements. Our involvement with the organization makes us, however, question the CMM assessment and the feasibility of the ensuing recommendations.

Through the interviews and through our intervention into the company we have come to see it as an organization characterized by: contradictory demands, structural conflicts, limited resources, uncertainty and change, and we will argue that other, more complex explanations of the process problems are needed in order to properly identify feasible and sensible solutions, if such solutions exist at all in all cases. We will now discuss how the specific organizational conditions in the company make the project managers behave in the unsystematic and seemingly irrational way described above.

The project managers face contradictory demands. They are for example under considerable pressure from the marketing and sales departments and the technical director to deliver according to the launch schedule already communicated to the sales force. On the other hand, everybody wants fault-free code with high usability that meets the complex needs of the customers. These requirements are contradictory in the sense that they cannot be fulfilled simultaneously by all projects. To the project managers, the products' quality depends on their ability to experiment with the technology and the requirements, but experimentation creates planning uncertainties, and they are therefore less comfortable with a fixed delivery date.

> "Interviewer: Can one say that there is a contradiction between you and your need to experiment and management's need to have a [delivery] date.
> Project manager: Yes, that's the paradox."

Project managers consequently understand estimates as political statements and delivery dates as something to be continuously negotiated. Some even – in their own words – "collect excuses" in case something goes wrong in their project.

There are conflicting interests in the organization; particularly concerning the limited development resources. It is in the interest of the project managers to have adequate resources. This will effectively enable them to handle some of the other uncertainties they face. The technical director, on the other hand, wants to provide the projects with as few resources as possible in order to be able to start more projects or reduce costs. It is in the interest of the project managers to have developers with specific competencies allocated to their projects, but the technical director needs to maintain a flexible work force where competence can be moved around depending on need.

The limited resources create conflicts among the project managers and between them and upper management. The project managers' competition for available resources takes both subtle and outspoken forms. When the competition is latent the project managers will try to influence the technical director's decisions about staffing of projects. It is not uncommon that the technical director has promised a project manager a developer, but no developer is available. When the competition is more manifest a project manager might go to the technical director and argue that he should have a developer from another project under less pressure.

There is uncertainty in any development process. A promising new, but crucial technology may be delayed or it turns out to be less useful than anticipated. Some requirements may turn out to be much more complex to realize than expected. The marketing department might change its mind; it proves impossible to find and hire needed resources, etc. The project managers live with these uncertainties and they attempt to be on top of the situation, but they are often taken by surprise. To reduce uncertainty they need time to experiment and systematically search for new and relevant information, but this collides with the processes defined in the company's quality system and the demand to deliver on time:

> "It is impossible to follow the models in practice. Instead you do what comes natural – to make a kind of ... iterative prototype development. This should be legalized instead of turning [us] into some kind of criminal – a 'closet criminal'."

There is change in the projects' environment. The organizational structure is changing when new

managers are hired and others leave. Project creation or termination leads to changes in the development department and fluctuations in product sales or size of market segments may reshape the whole company. To a project manager significant changes also occur when resources and thereby knowledge leaves the company.

> "Almost all knowledge [relevant to this project] had disappeared from the company [by] January 1997. For a long time in the beginning of 1997 there were literally no development activities [in the project]. New personnel, without deep knowledge of [the application domain] were hired in the second quarter. Training them was a major effort ..."

From this perspective we come to see the project managers as competent actors in a highly contradictory and complex organizational environment. They want to produce usable products without too many defects, and reasonably close to an acceptable delivery date. Fluid or incomplete requirements, 'drawer projects', illicit prototypes, and rough, missing or outdated plans are, in this light, not simply immature software practices, but can be understood as strategies to protect their project and themselves and to ensure the success of the projects they are responsible for in an environment of uncertainty, contradiction, and conflict.

4.3 Organizational politics

Above we interpreted the project managers' practices as means to steer their project through contradictory demands, organizational conflicts and uncertainties regarding product features, technology and staffing. Thus we saw the practices as symptoms of underlying causes embedded in the company.

What we have described may be seen as irrational behaviour from individual project managers or others in the company. We will argue, however, that what we have seen in "the company" is merely the outcome of organizational processes that can be found in many software organizations. Studying at these processes and the theories that explain them can give us a deeper understanding of the project managers' practices and also reveal problems and limitations in the CMM's and other maturity models' perspective on organizations.

In a recent study Linberg (1999) observes that developers find that management sends conflicting signals about the relative importance of schedule, time, quality, and cost in software projects. Kautz et al. (2001) present a similar example where such a problem situation was resolved by a strategy which is not included in the CMM. The developers in Linberg's study also report about deliberate initial underestimation of projects in order to secure project approval. The inevitable ensuing delays and cost overruns therefore came as no surprise to the developers. The study further shows that the developers and managers do not agree on what it means for a project to be successful. When asked to assess the success or failure of a project, the developers considered factors such as interesting and challenging work and the quality of the end-product, at least as important as meeting budgets and deadlines.

Keil & Robey (1999) have interviewed IS auditors about the handling of troubled IS projects. To handle troubled IS projects requires that somebody communicates the bad news to somebody else who can do something about it, but the message may be delayed because nobody wants to transmit or act upon news of a troubled project out of fear of the consequences. "Blowing the whistle" – as Keil & Robey call it – or terminating a troubled project may be perceived as "career suicide" because of the vested interest and prestige in the project, and the bad news may simply be ignored by managers with the power needed to act on troubled projects.

> "The would-be whistle blower must wield sufficient power to challenge [the] conviction [that project completion is critical]." (Keil & Robey 1999, p. 83)

Both Linberg's and Keil & Robey's studies supplement our interpretation of the project managers' practices. In each their way they open up for alternative explanations of the project managers' practices, than those offered by a maturity assessment. Following Linberg we can see how different actors in the company hold different views on projects and how to manage them; i.e. the PMs and management do not agree upon how to schedule a project or how to handle uncertainties. Following Keil & Robey we can also see that bad

news are rarely communicated in the company and that differences of opinion are rarely voiced in front of the technical director. In fact, there is a cover-up of the different perceptions in form of the 'drawer projects' and some project managers collect excuses to defend their self interest and their projects.

The question is what causes this kind of behaviour and what – if anything – can be done to change it. The concept of commitment discussed above is certainly one way to explain the behaviour. Most of our observations in the company, as well as those reported by Linberg, resp. Keil & Robey can be explained through reference to lack of commitment to an open and rational planning and management process. In the case of the company it is obvious that a dedication to open and shared commitments and an agreement on goals and ways to meet them are necessary prerequisites for improvements.

This interpretation of the situation is however based on a conventional notion of commitment assuming that commitment among others is an all positive phenomenon - as discussed and challenged by Abrahamson (2001) - and leaves open the question of what lies behind the missing dedication to commitments. This question cannot be explained within the framework offered by maturity models. We need to turn to other, more complex perspectives on organizations and organizational behaviour.

We find that the concept of organizational politics, though broad and without clear and concise definitions, can offer some explanations. Drory & Romm (1990) write that theories on organizational politics "indicate that formal organizational processes such as decision and policy making, goal setting, and resource distribution are not conducted predominantly by rational considerations which represent the best interests of the organization." (p. 1133). They draw a comprehensive picture of organizational politics encompassing such elements as self-serving behaviour, acting against organizational goals, concealment of motives, informal behaviour, uncertainty in decision making, and organizational conflicts.

Through an extensive literature survey Drory & Romm (1990, p. 1147) come to define organizational politics as a combination of the following three elements: influence, informal means, and conflict. Our descriptions of the practices of the PMs clearly demonstrate the presence of all three elements of organizational politics.

We can go even further based on Knights & Murray's (1994) study of conflicting management priorities in information systems development. Knights & Murray argue that IS organizations are dominated by: competing, politicizing, and conflicting groups. The conflicts are, however, not caused by simple power struggles, personal ambitions or 'turf guarding', but are the results of conflicting views on what is best for the organization. These views influence and are at the same time shaped by personal ambition, departmental loyalties, different world views, and structural conflicts over priorities

> "… it is impossible and misleading to separate off the albeit problematic pursuit of self or sectional interests from those of the organization itself. Rather, it is through the construction, negotiation and reappraisal of self, collective and organizational interests that the fragile reality of an organization is sustained, reproduced and changed."
> (Knights & Murray 1994, p. 29)

Therefore – according to Knights and Murray – there can be no right or universally valid organizational goal or strategy. Broken down to the day-to-day business of producing software and information systems, this means that there is no over-all goal within which to define and prioritize work; there is only an ongoing political struggle.

In this light then, we can argue that the project managers in the company see themselves as perfectably capable to determine what is in the best interest of the company and their environments behaviour as obstacles for successful projects. Other actors; e.g. the technical director, the marketing department etc. may have other ideas of what is 'best' and are therefore not necessarily ready and willing to accept the project managers' views, priorities, actions etc.

For the project managers it is therefore important to do what they can to maintain and enlarge the space in which they have the power to act in what they see as the best interest of the company in more or less open conflict with other actors; i.e. they invent 'drawer projects' to create a space for exploring exciting new possibilities, they circumscribe or short circuit 'bureaucratic' approval and change control procedures

to maintain project momentum, and they use whatever means possible to get the resources they need for their project.

4.4 Changing practice

This is not to say that there isn't room for improvements in the company's software processes. But how can the situation be improved? The PMs' strategies are simplistic and serve to obscure the underlying problems rather than expose and improve them. A maturity assessment on the other hand will just point to a number of weak processes and produce a prioritized list of necessary improvements.

In our view the uncertainties and structural conflicts facing the project managers must be surfaced and removed as a precondition for implementing any change. Kautz et al. (2001) demonstrate how this may be done when process consultants consciously assume the role of political agents. However this approach may not always be feasible.

Furthermore, from Knights and Murray's (1994) perspective on organizational politics we can interpret CMM-inspired improvements as a way to make development projects more transparent to, e.g., higher levels of management and thus reduce the scope of the project manager's control. To willingly accept such a change requires that the project managers trust other groups in the company to accept their perception of what is 'best' for their projects. Based on previous experience, however, the project managers may assume that this is not the case, and that any sign of openness therefore will be exploited by others at the project managers' expense. Adopting this perspective we may therefore expect that improvement initiatives will be accompanied by new attempts to maintain project control as manifested by 'drawer projects', uncontrolled requirements changes, withholding of information and other strategies.

5. Conclusion

Models of software process maturity provide a particular way to assess software processes with a focus on the lack of maturity and lack of rationality. In this article we have shown that by seeing organizations as political, rather than rational, we can provide an explanation of software – and in particular project management – practices that is different from the one offered by the CMM and similar maturity models.

It is not easy to base advice for SPI from our analysis. One approach could be to supplement a traditional maturity assessment with an investigation of the organizational contexts of software practices to uncover and remove 'organizational causes' and obstacles for improvement. Even this approach may, however, be problematic. From a political perspective SPI itself can be seen as a way to change the balance of power in a software producing organization, a change which may inspire new struggle over the control of software projects.

The purpose of maturity models is to guide assessments and change of software practices. They are based on a traditional software engineering perspective and like any model, they rest upon certain assumptions about reality, exposing certain aspects of software processes and organizations at the expense of others. Thus, using the maturity models to assess software organizations will turn our attention to certain problems and recommendations and leave out others. We have demonstrated, through an example, that some of the issues overlooked in a maturity assessment, but surfaced by looking at the software practices from another perspective, may prove to be severe obstacles for the implementation of improvements. This does not falsify the CMM or similar maturity models, but does, in our view, underline the need for more research aiming at understanding the theoretical underpinnings of the maturity models and their practical implications to better understand under which conditions we can use the models to guide improvement efforts.

Notes

[1] A complete CMM assessment falls outside the scope of this paper, but it would expose weaknesses within several of the other level 2 areas too.

References

Abrahamson, P.(2001). Rethinking the Concept of Commitment in Software Process Improvement, in this volume.

Avison, D.,. Lau, F., Nielsen, P. A, M. Myers, M. (1999). Action Research. Comm. ACM.

Bach, J. (1994). The Immaturity of the CMM. American Programmer, 7(9), pp. 13-18.

Bach, J. (1995). Enough about Process: What We need are Heroes. IEEE Software, 12 (March), pp. 96-98.

Baskerville, R. and Pries-Heje, J. (1998). Managing Knowledge Capability and Maturity. In: Larsen, T.J., Levine, L. and DeGross, J.I. (eds.) Information Systems: Current Issues and Future Changes, IFIP, Laxenburg: pp.175-196.

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

Checkland, P. (1991). „From Framework Through Experience to Learning: The Essential Nature of Action Research," in Information Systems Research: Contemporary Approaches and Emergent Traditions (ed. H.E. Nissen et al.), Elsevier, North-Holland, pp. 397–403.

Drory, A. and Romm, T. (1990). The Definition of Organizational Politics: A Review, Human Relations, vol. 43, no. 11, p. 1133-1154.

Dunaway, D. K. and Masters, S. (1996). CMM-Based Appraisal for internal process Improvement (CBA IPI): Method Description. Technical Report: CMU/SEI-96-TR-007, Software Engineering Institute, Pittsburgh, Pennsylvania.

Edgar-Nevill, V.M.A. (1994). Evaluation of the SEI software capability model within an information systems context; in pursuit of software quality. In: Ross, M., Brebbia, C.A., Staples, G. and Stapleton, J. (eds.) Software Quality Management II. Managing Quality Systems. Vol.1, Comput. Mech. Publications, pp.263-278.

Enam, K. E., Drouin, J.-N., and Melo, W. (1998). The Theory and Practice of Software Process Improvement and Capability Determination, Los Alamitos, CA: IEEE Computer Society Press.

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

Humphrey, W.S. (1989). Managing the Software Process. Addison-Wesley, Pittsburgh, Pennsylvania.

Iversen, J.I., Nielsen, P.A. and Nørbjerg, J. (1999). Situated Assessments of Problems in Software Development, DATA BASE for Advances in Information Systems, vol 30, no 2, p. 66- 81.

Kautz, K. Hansen, H. W. and Thaysen, K.(2001). Understanding and Changing Software Organisations: An Exploration of Four Perspectives on Software Process Improvement, in this volume

Keil, M. and Robey, D. (1999): Turning Around Troubled Software Projects: An Exploratory Study of the Deescalation of Commitment to Failing Courses of Action, Journal of Management Information Systems, vol. 15, no 4, pp. 63-87

Knights, D. and Murray, F. (1994). Managers Divided, John Wiley & Sons, Chichester.

Kohoutek, H.J. (1996). Reflections on the capability and maturity models of engineering processes. Quality and Reliability Engineering International, 12 (3), pp. 147-155.

Kuvaja, P., Similä, J., Krzanik, L., Bicego, W., Saukkonen, S., and Koch, G.(1994). Software Process Assessment and Improvement: The Bootstrap Approach, Oxford: Blackwell Publishers.

Linberg, K.R. (1999). Software Developer Perceptions about Software Project Failure: A Case Study, The Journal of Systems and Software, 49, pp. 177-192.

Mathiassen, L. and Sørensen, C. (1996). The capability maturity model and CASE. Information Systems Journal, 6, pp. 195-208.

O'Connel, E. and Saiedian, H. (2000). Can You Trust Software Capability Evaluations, IEEE Computer, vol 33, no 2, pp. 28 –35.

Patton, M.Q. (1990). Qualitative Evaluation and Research Methods. Sage Publications, New York, 2nd edition.

Paulk, M.C., Curtis, B., Chrissis, M.B., and Weber, C.V. (1993). Capability Maturity Model for Software, Version 1.1. 93-TR-024. Software Engineering Institute, Pittsburgh, Pennsylvania.

Saiedian, H. and Kuzara, R. (1995): SEI Capability Model's Impact on Contractors, IEEE Computer, 28, pp. 16 – 26.

Sawyer, P., Sommerville, I. and Viller, S. (1997). Requirements Process Improvement Through the Phased Introduction of Good Practice. Software Process – Improvement and Practice, 3, pp. 19-34.

Sharp, H., Woodman, M., Hovenden, F. and Robinson, H. (1999). The role of 'culture' in successful software process improvement. In: Proceedings 25th EUROMICRO Conference. Informatics: Theory and Practice for the New Millennium, IEEE Computing Society, pp.170-176.

Smith, W.L., Fletcher, R.I., Gray, E.M. and Hunter, R.B. (1994). Software process improvement: the route to software quality? In: Ross, M., Brebbia, C.A., Staples, G. and Stapleton, J. (eds.) Software Quality Management II. Managing Quality Systems. Vol.1, Comput. Mech. Publications, pp.193-211.

Stelzer, D., Mellis, W. and Herzwurm, G. (1998). Technology Diffusion in Software Development Processes: The Contribution of Organizational Learning to Software Process Improvement. In: Larsen, T.J. and McGuire, E. (eds.) Information Systems Innovation and Diffusion: Issues and Directions, Idea Group Publishing, Hershey, USA: pp.297-344.

Strauss, A., Corbin, J. (1990). Basics of Qualitative Research. Grounded Theory Procedures and Techniques. Sage Publications, Beverly Hills, CA, USA.

Velden, M.J.v.d., Vreke, J., Wal, B.v.d. and Symons, A. (1996). Experiences with the Capability Maturity Model in a research environment. Software Quality Journal, 5, pp. 87-95.