16 SOFTWARE PROCESS MATURITY AND ORGANIZATIONAL POLITICS

Peter Axel Nielsen
Department of Computer Science
Aalborg University
Fredrik Bajers Vej 7
DK-9220 Aalborg
Denmark

Jacob Nørbjerg
Department of Informatics
Copenhagen Business School
Howitzvej 60
DK-2000 Frederiksberg
Denmark

Abstract

The discipline of information systems development has witnessed a tremendous interest in improving software processes. It is believed that improving a systems development organization's software processes will alleviate problems with productivity and systems quality. In this paper we explore the limitations of the theories and models behind software process maturity. Through an action research project with a systems development organization, we illustrate how maturity models ignore issues of structural conflicts and contradictory demands manifest in most organizations. This limits the models' ability to explain software practice and thus limits their usefulness for guiding organizational change processes.

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, etc. These and other problems contribute to uncontrollable projects, runaway budgets, and delayed products of inferior quality. The idea that systems development organizations can and will become better at managing their software projects is captured in the concept of software process *maturity*.

The meaning of maturity is spelled out in detail in various *maturity models*. These models are intended to support improvement initiatives in software producing organizations by (1) determining the organization's maturity level through comparing the organization's development and management practices with the model and (2) guiding the change process by pinpointing the improvements needed to increase maturity; i.e., reach a higher level of maturity.

The capability maturity model (CMM) is one of the first and most influential of the maturity models. The model describes maturity in terms of five levels. Level 1 (initial) is characterized by the absence of even basic project management practices. Level 5 (optimizing) is characterized by organization-wide management and development practices and extensive use of process and product measurements to monitor and continuously improve performance. The levels are described in great detail in order to reduce ambiguity and the model is accompanied by training programs for assessors as well as detailed guidelines and procedures for assessments and maturity evaluations (Dunaway and Masters 1996, Paulk et al. 1993).

A number of competing models have emerged during the 1990s; e.g., the Bootstrap model in Europe (Kuvaja et al. 1994), and the SPICE model (Enam et al. 1998), intended to embrace all previous models. The CMM has had significant influence on the concept of maturity and all subsequent models, however, and we will use this model as our frame of reference in the rest of the article.

Further developments of the maturity models have been suggested. Sawyer et al. (1997) have suggested a three-level maturity model for requirements processes, and others suggest adding organizational learning and knowledge management capabilities to the models (Baskerville and Pries-Heje 1999; Stelzer et al. 1998).

Other researchers have raised more fundamental questions, challenging the idea of software process maturity as such and the use of model-based assessments to guide improvements. First, some argue that the assessment process itself may be flawed due to immature assessment techniques (Bollinger and McGowan 1991; O'Connel and Saiedian 2000; Smith et al. 1994).

Second, it has been argued that the effects of increased maturity on organizational performance across different types of software producing organizations is still unverified especially in organizations outside the U.S. (Baskerville and Pries-Heje 1999; Edgar-Nevill 1994; Mathiassen and Sørensen 1996; Sharp et al. 1999; Velden et al. 1996). Some authors have gone even further and

challenged the general validity of the maturity idea. The dominating maturity models associate high maturity levels with documented processes and extensive use of process and product measurements 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 the organization (Bach 1994, 1995; Bollinger and McGowan 1991; Kohoutek 1996). The discussions about these issues have, however, suffered from a lack of systematic theoretical and empirical research into the concept of maturity and its practical application—including more explicit descriptions in the models of the models' own theoretical and empirical base.

The recommendations in the maturity models are based on basic software engineering ideas of sound development and management practices, and they seem both sensible and feasible. We do, however, find that there is a need for more systematic research into the models' theoretical and empirical foundations in order to establish their validity and limitations more firmly.

In this paper, we study the practices and perceptions of project managers in a small software producing organization. In section 2, we describe our research approach. Section 3 presents the case and describes and discusses the project managers' practices both related to the CMM and in a broader organizational context. In section 4, we relate our observations to relevant theories of organizational politics and discuss the limitations of the CMM. In section 5, we conclude that the CMM in its current form ignores relevant organizational issues, which, we will argue, limits the value of CMM-based assessments and recommendations. Further research is then outlined.

2. THE RESEARCH APPROACH

Our study is fundamentally action research. We have worked with the company's project managers and software process improvement group, and actively participated in their improvement efforts from 1997 to 1999. We have throughout tried to balance the dual purposes of action research contributing both to the action in the company and to research (Avison et al. 1999). Checkland (1991) advocates that in action research there should be a framework being tested. Our framework has been the CMM and the body of knowledge of software process improvement (representative sources: Humphrey 1989; Paulk et al. 1993). We have documented our action research in tape recordings and minutes of all meetings where we have been present, in tape-recorded interviews with project managers and middle managers, and in our own field notes and diaries produced

during and after meetings. The documentation also consists of the reflective papers written during the endeavor and thus influencing both our framework and the actions we advocated in the company.

In this article, we focus on the project managers. We draw on two sources. First, we conducted seven interviews with seven out of 10 project managers in June, July, and August, 1997. The purpose was to elicit the project managers' perceptions of problems with software processes in the company. Following the techniques for qualitative interviewing we used an open-ended interview guide (Patton 1990). After the first interview, we modified and improved the interview guide. The interviews were tape-recorded and subsequently transcribed. In our analysis of the qualitative data, we have followed Patton's ideas. We have worked closely with the text trying to let it speak for itself (Patton 1990).

As action researchers, we are not only observers; we are also involved in the situation. We cannot avoid bringing in our own beliefs and experience when we interpret and re-experience the interview text. Much of the interview text deals directly with software process problems while our focus in this article is on the organizational politics of software processes. Both of us have read the text carefully with this focus and noted text we found to be significant. We have compared our notes and settled on a set of categories of issues. We have then read the text again looking for quotations confirming or disconfirming the categories. The final categories and quotations illustrating the categories now form the basis of the case description in section 3. We will not claim that this is anything like a grounded theory approach. We, as action researchers, may well have formed interpretations that others would not have been able to form.

Second, we have continued our action research two years after the interviews were conducted. That has allowed us to look closely at 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 on many occasions tried to influence project managers and others. Our shared practice serves as a context against which we judge our interpretation of the interview text. That helps us triangulate our findings and it provides a larger perspective for making sense of the project managers' statements.

3. THE CASE

The interviews took place in the Research & Development (R&D) Division of a company that develops leading edge instruments and systems. The instruments are built from dedicated hardware with embedded software and they are connected to a PC with analysis and presentation software. Most projects inte-

grate both hardware and software; project management is also integrated. Very large or complex projects are further divided into hardware and software projects, each with their own project manager.

The technical director manages the R&D. The R&D and its projects collaborate closely with the Marketing and Sales Division and with the Production Division.

Prior to our action research, the company had been through a long reorientation and downsizing process due to increased competition and setbacks in one of the company's major markets. Immediately before our entry, the R&D went though a Bootstrap assessment (following Kuvaja et al. 1994). It was concluded that most of the company's software process problems concerned project management, configuration management, testing, development model, and requirements specification.

We describe the project managers' practice in section 3.1. In section 3.2, we assess their practices relative to the necessary capabilities at CMM level 2, and in section 3.3, we discuss the project managers' own interpretations and explanations of their practices. In section 3.4, we compare the two interpretations of the project managers' practice. Further, we interpret the project managers' practices in terms of strategies to produce and deliver products under difficult conditions of contradictory demands and organizational conflicts.

3.1 Project Management Practice

We focus on three areas of project management: estimates and schedules, requirements management, and resource allocation. These illustrate most clearly the relationships between the project managers and the surrounding organization.

3.1.1 Estimates and Schedules

The project managers seek to maintain an approach to estimation and scheduling based on explicit plans and experience. They do this to get an overview and control of activities and deadlines. They do not expect schedules to hold and they know that they often have to re-schedule.

A project manager negotiates estimates and schedules with the project team based on a list of features and requirements for the product. Directives from management, time pressure and uncertainty about the requirements, however, often disturb the planning process as illustrated in this extract from the interview with an experienced project manager:

> The time schedule was set on beforehand: We had to finish by a certain month so I really didn't have [estimation and planning] problems. I've made some rough plans for the requirement specification. They proved to hold or not to hold at all. So I stopped doing that [i.e., planning]. The requirement specification phase has been much longer than planned. We had expected to complete in April or May, but in reality we didn't complete until August.…I've used a lot of time on setting up time schedules; I've performed according to some rough plans we agreed on; and we've tried to navigate from these [plans].

The project manager does not question the overall schedule set by management. His plans and estimates are based on ad hoc and experience-based planning. The estimate for the requirement specification phase did not hold. He was, however, able to schedule and re-schedule throughout the project. He managed the project anyway and at the time of the interview he was confident that he could complete the project not later than a month after the original estimate. This later proved to be right.

For other project managers, the conditions are not as favorable. Many projects struggle with unrealistic time schedules set by management:

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

Product X was very complex and the project manager and the project had an initial estimate of four months for the requirements specification alone. They actually managed to write a specification in five days, but it was very superficial and the project never came to believe in it completely.

The estimation and planning processes are strongly regulated by the product launch schedule produced by the technical director together with the other directors. The plan outlines features of next year's products. It tells the Sales Department when they can expect delivery of each product, when they can begin selling the new product, and when they will have demo versions available. Tentative product features are stated in the schedule and a first step in a project is to write a more detailed product description that the Sales Department can use for promotion. The following extract from an interview with a project manager shows an important aspect of this.

> [The management] tries to announce what products will be out next year before even the requirements specifications are made or anything. That means that we've projects starting from mere headlines [because] you have to.

From a marketing and sales point of view, this is a perfectly reasonable approach. Without a launch schedule, it would be impossible for the sales representatives to plan their sales efforts and to cater for the customers' needs and demands. It is, however, a difficult situation to impose on the project managers. 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 asked to estimate a project to develop a product that is only described as headlines.

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 in turn puts pressure on the projects to deliver as expected without much consideration for their difficulties. 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 the planning techniques, but they find that these techniques don't help them to cope with the problems they face. Their frustration can be formulated as in the following extract: "You gave your best shot and after some time you were beyond the time schedule because it didn't matter—you just had to complete."

3.1.2 Requirements Management

We group the processes through which requirements are found, described, prioritized, and later changed and managed as requirements management.

Many project managers and team members focus tremendously on the requirements, while the projects' environment often neglect the details of requirements and hence also requirements management. Most projects try to follow the company's development model where requirement specification plays a major role and some work systematically with scenarios, using cases and prototype testing internally and at customer sites in the requirements definition process. This process is time consuming but results in detailed, and—according to the project managers—very useful specifications.

Other projects take requirements specification more lightly. One project develops a system to support measurements according to five different national certification standards. The project manager explains the requirements management in the following way:

> It has been characteristic of 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.

This project never produced a requirement specification but used the five national certification standards instead, according to the project manager. These standards do not, however, mention central aspects of a computerized system, e.g., the user interface.

A "drawer project" is a project initiated by a project manager without explicit approval from the technical director. If the project manager succeeds in obtaining formal support for the project, it will emerge from the project manager's drawer and officially start. Several of the project managers we interviewed reported that their (now officially approved) projects began as drawer projects. It is, of course, almost impossible to apply proper processes in a drawer project, since it is a cover-up of what is actually going on.

The pressure put on the projects to meet deadlines reduces their ability to define and manage requirements systematically. For instance, when a deadline approaches, a project manager may simply take matters in his own hands and strip requirements. One project manager describes it this way: "Then you have to be clear. You'll have to adjust your own ambitions for the project. How many nice-to-have features are thrown away?" The project managers see themselves in a situation where the marketing department tries to push as many features as possible into the product without necessarily considering time and development costs. On the other hand, the technical director wants the projects to deliver on time. For the project managers, it becomes more important to meet deadlines with a product that works than implementing all requirements and they are, therefore, prepared to remove requirements.

The change of requirements must be approved in the project's steering committee according to the company's official procedures. In practice, however, the project managers assume responsibility for requirements change, sometimes with subsequent approval in the committee. The following statement by a project manager displays extreme confidence in his overview of the market and his ability to make the right decisions regarding the 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 the market needs that it requires.

The project managers partly blame the slow and bureaucratic approval process for this practice. To avoid the delays caused by this process, the project managers will also often postpone the formal requirements review and sign-off for as long as possible.

3.1.3 Resource Allocation

Resource allocation concerns two problems: which projects to start and the allocation of manpower to projects. There are far too few software developers with the specialized knowledge required to develop the company's products and there is a constant struggle between project managers to get sufficient developers with the right qualifications allocated to projects:

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

Project managers find other ways to deal with this. Developers cannot be allocated to a project that is not officially started, but to be on the safe side, a project manager can always create a drawer project. The following extract shows this: "It has been running since… it's one in the political spheres… in principle…officially it was started a month ago [July 1997]. Unofficially, it…started in late February." The project managers perceive the resource allocation process as sometimes arbitrary or paradoxical. They are requested to estimate their manpower needs but doing so is an effort that requires manpower: "[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?"

Other project managers were, however, able to hire and train new people as needed. In the following statement, a project manager refers to a project that had been running for years without officially being recognized as a project, and without a plan and estimates. The manager was, however, able to hire and train new people for the project.

> *Action researcher:* To start this project you needed project team members with particular qualifications. These qualifications are not available in the market and you cannot move people around in the company. You then [hired] new people and trained them for half a year. Then they are ready to enter the project.
> *Project manager:* That's exactly the situation here.

Another project stalled for a long time because a key programmer was temporarily allocated to do maintenance on another high priority project. To overcome the loss of a key resource, the project manager went "shopping" for

developers in other projects. This is always a problematic thing to do, partly because it may hurt relations with other project managers, partly because the developer himself may resent being moved.

3.2 A CMM Perspective

The company has an elaborate quality management system that is followed, at least formally, but our interviews show that the company's software practices (at the time of the interviews) are definitely immature according to the CMM. Estimates and plans are not produced and managed systematically. Requirements management is ad hoc and resources are not allocated according to plans and priorities but through a highly political process.

The company's software processes thus fail to satisfy at least three out of the six key process areas required to be a CMM level 2 organization. Additionally, there are weak processes in configuration management and sub-contractor management, but here it suffices to look at how the company's processes conform to the CMM's requirements for the three areas above (the following is based on Paulk et al. 1993, pp. 59-60).

Requirements Management:

Goal 1: *System requirements allocated to software are controlled to establish a baseline for software engineering and management use.*
A baseline of requirements is rarely established. On the contrary, requirements are often held fluid until the very last day of the project.

Goal 2: *Software plans, products, and activities are kept consistent with the system requirements allocated to software.*
Some projects try to do this in their own way. Many projects fiddle with requirements to meet delivery deadlines or handle other contingencies.

Software Project Planning:

Goal 1: *Software estimates are documented for use in planning and tracking software projects.*
Many projects estimate their effort, but the delivery date is often fixed beforehand. The estimates are documented in the project plans.

Goal 2: *Software project activities and commitments are planned and documented.*
Activities are planned in most projects and documented in the project plans. Most projects have ways of establishing and maintaining internal commitments. Many projects do not handle external commitment well. Altogether, few commitments are documented.

Goal 3: *Affected groups and individuals agree to their commitments related to the software project.*

All agreement to commitment is unsystematic, and external commitment often fails with unnecessary uncertainty as a result.

Software Project Tracking and Oversight:

Goal 1: *Actual results and performances are tracked against the software plans.* The amount of running code is measured against the deadline, but in most cases there are no means of early discovery of a project in trouble.

Goal 2: *Corrective actions are taken and managed to closure when actual results and performance deviate significantly from the software plans.* Corrective actions are taken, but mostly based on experience and gut feeling.

This evaluation is supported by a Bootstrap assessment conducted half a year before we entered the company. The Bootstrap assessment has slightly different maturity measures, but nevertheless the assessors concluded that the following processes needed to be improved: software development model, description of software processes, requirements specification, project management, module and integration testing, and configuration management. That coincides largely with our CMM evaluation.

Based on our interviews, CMM would recommend that the software practice should be improved considerably in at least three out of the six key process areas at CMM level 2. One thing in particular needs to be improved; namely, the way commitments are established and maintained. Humphrey (1989, p.70) explains the elements of making commitment:

1. Commitments are made willingly.
2. Commitments are not made lightly; they are carefully considered.
3. There is agreement on what, whom, and when.
4. Commitments are stated openly and publicly.
5. The person responsible tries to meet the commitment.
6. Prior notice is given if a committed date cannot be met.

The company clearly does not demonstrate such dedication to commitments. According to the CMM, it should be established.

3.3 A Perspective from Organizational Politics

There is no doubt that the company's software development suffers from budget and schedule overruns, high workload, firefighting, and quality problems. The recommendations produced by a maturity assessment are, therefore, sensible, i.e., to install and follow systematic processes. However, the only *explanation* of the problems offered by an assessment is the low maturity of the pro-

cesses, in other words, that the company lacks proper and systematic software practices.

The CMM certainly points to a number of problems or areas where the company's software processes should and possibly could be improved. That is useful, but it is our contention that CMM provides a partial view on the problems and necessary improvements. CMM points out that the practices aren't sufficiently rational. In particular, commitments should be formed in a more explicit, open, and public way. That is an admirable desire; but we find it hard to be explicit, open, and public under the conditions in the company and we believe that attempts to install such a change will create resistance. In our interpretation of the company's software processes, we would have to look for an explanation of the resistance to change in the political reality in the company. We thus need deeper, more complex explanations of the process problems in order to identify feasible and sensible solutions.

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. We will now discuss how these specific organizational conditions make the project managers deliberately or implicitly choose less systematic and rational processes than advocated by the CMM.

The project managers face *contradictory demands*. They are, for example, under considerable pressure to deliver on time. Delivery on time pervades the whole company. The marketing division requires it because it needs to plan the launch on the market. The sales representatives expect it because they have already made commitments to the customers. Thus the technical director puts a lot of pressure on projects. On the other hand, everybody wants error-free running code with high usability that meets the complex needs of the customers. These requirements are contradictory in the sense that they cannot be fulfilled at the same time in all projects. Balancing these requirements is not easy, however, and it creates problems and conflicts in many projects:

> *Action researcher:* 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.

For the project managers, the product quality depends on their ability to experiment with the technology and the requirements. However, experimentation creates planning uncertainties, and they are, therefore, less comfortable with a fixed delivery date. Project managers consequently understand estimates as political statements and delivery dates as something to be continuously negotiated.

There are *structural conflicts*, beyond mere contradictory demands and these form fundamental conditions for the company and its projects. In particular, there are conflicts around the *limited development resources*. The limited resources create conflicts among the project managers as well as between the projects and the technical director. The project managers' competition over available resources takes many 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 new developer, while 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 an additional developer from another project that is under less pressure.

It is in the interest of the project managers to have as many resources as possible. This will effectively enable them to handle some of the 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 manager 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 needs.

There is *uncertainty* in any development process. A promising technology turns out to be less optimal than anticipated. Some requirements may turn out to be much more complex to realize than anticipated. The marketing department might change its mind. It is impossible to find and hire needed resources. Such conditions are inherent in systems development. The project managers live with the uncertainties. They try 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 requirement to deliver on time, as discussed above.

There is *change* in the project's environment. The organizational structure is changing while new managers are hired and others leave. When a project is created or terminated, major changes happen in the R&D division. When a product sells well or not so well, changes are instantiated. When a market segment increases or decreases, the organization changes as a consequence. To a project manager, significant changes also occur when resources and thereby knowledge leaves the company: "It happened, really, that in a period until January 1997 almost all [relevant] knowledge disappeared from the company. Some areas in the company have never recovered from this knowledge drain."

From this perspective, we have come to see the project managers as competent actors in a highly contradictory and complex organizational environment. They want to produce something that is usable, at an acceptable level of quality,

and reasonably close to an acceptable delivery date. Thus, their behavior is not only an example of immature software practices, but can be understood as strategies to maintain control of their own situation and ensure the success of the projects they are responsible for, in an environment of uncertainty, contradiction, and conflict.

The project managers' strategies are also counter-productive even from their own perspective. They are caught in a game out of their control. There is a cover-up of the actual performance in the projects. There is protection of resources. There is protection of self. These strategies are simplistic and cannot change the situation for the project managers. Their behavior reinforces the same conditions that they oppose.

4. CMM AND ORGANIZATIONAL POLITICS

The practices we have described in the R&D division can be found in other companies as well, and so can the problems with time and budget overruns, quality, etc. The CMM and similar maturity models will largely explain such practices as lack of maturity and recommend installing systematic management processes based on mutual commitments. While this is useful, it is also too limited a view. To properly understand these practices, we need another perspective than that offered by the CMM and other maturity models. We need a perspective that allows us to understand the problems as symptoms of underlying causes embedded in the organization and to see the project managers' practices as means to steer their project through contradictory demands, organizational conflicts, and uncertainties regarding product features, technology, and staffing.

Other empirical studies produce similar descriptions of the conditions for software project management. In a recent study, Linberg (1999) observes that developers found that management sends conflicting signals about the relative importance of schedule, time, quality, and cost in software projects. The developers reported 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 did not agree on what it means for a project to be successful. Consequently, they did not agree on what it means to fail.

Keil and Robey (1999) interviewed IS auditors about the handling of troubled IS projects. Handling a 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" on a troubled project is perceived as "career suicide" or there may be so much vested interest and prestige in the project that the bad news is simply 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 and Robey 1999, p. 83). Both Linberg's and Keil and Robey's studies show how important aspects of the project managers' actions cannot be understood solely as lack of maturity or of rational and systematic processes. Following Linberg, we can certainly see how different actors in the company hold different views on projects and how to manage them, but that these are rarely voiced in front of the technical director. In effect, there is a cover-up of the different perceptions. Following Keil and Robey, we can also see how bad news are rarely communicated in the R&D division.

Thus, we can measure the software process maturity by means of the CMM, but there are also relevant and important aspects of the project managers' practices that we cannot measure or understand within the framework of software process capability and maturity. We find the theories of *organizational politics* to offer a relevant alternative perspective. Drory and 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 behavior, acting against organizational goals, concealment of motives, informal behavior, uncertainty in decision making, and organizational conflicts.

Through an extensive literature survey, Drory and Romm (1990, p. 1147) come to define organizational politics as a combination of the following three elements: influence, informal means, and conflict. Based on this, we can characterize the practices of the R&D division as being under the influence of organizational politics.

We can go even further based on Knights and Murray's (1994) study of conflicting management practice in information systems development. Here they argue that IS organizations are dominated by competing, politicizing, and conflicting groups. The conflicts are, however, not founded in simple power struggles, personal ambitions, or "turf guarding," but in deeper layers 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 and Murray 1994, p. 29).

Therefore—according to Knights and Murray—there is 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 overall goal within which to define and prioritize work; there is only an ongoing political struggle about what constitutes such a goal. We take this to be a challenging and opposite view on systems development than that offered by the CMM.

In this light then, we see that the project managers in the company perceive themselves as perfectly capable of determining what is in the best interest of the company, and their organizational environments as obstacles for successful projects. They will, therefore, do what they can to enlarge the space in which they have the power to act in whichever ways they see appropriate.

The CMM advises a rationalistic way out of this, but organizational politics points out that a rationalistic way may not work. The CMM advocates insight into and control with projects, but organizational politics offers an explanation of why organizational actors might perceive any sign of openness as a weakness to be exploited by others in the organization.

5. CONCLUSION

In this article, we have argued and illustrated that the CMM offers a view on organizational practice that is sometimes too limited. In our case, we have found that many other aspects than lack of maturity and rationality are relevant and important in order to understand software development practice. Through an analysis of interviews with project managers, we have shown that by seeing organizations as political, rather than rational, we can provide a valid complementary explanation to the maturity models' interpretations of development practices. We have argued this for the case with which we have been involved, but it is our contention that our findings apply to other systems development organizations as well.

The CMM and other maturity models are good for objectively assessing software processes. The theories of organizational politics are good for explaining other aspects of organizational behavior. Organizational politics is,

therefore, not an alternative to the maturity models; it should be seen as complementary to these models.

To successfully change and improve software practices we have to find a synthesis between the maturity models' perspective and that of organizational politics. It is possible that the CMM and other maturity models would be greatly enhanced by having a better idea about organizational reality whereas theories of organizational politics when used in software process improvement would be greatly enhanced by having some idea about how to intervene in and change practice. Here we have merely established the need for such a synthesis. We will leave it for further research to explore.

6. ACKNOWLEDGMENTS

The Danish National Centre for IT Research supported the action research project. We thank the company and their software improvers for the collaboration during 1997-1999. We thank our colleagues, I. Aaen, K. Kautz, L. Mathiassen, and two anonymous reviewers for helpful comments on previous versions of the article.

7. REFERENCES

Avison, D., Lau, F., Nielsen, P. A., and Myers, M. "Action Research," *Communications of the ACM* (42:1), 1999, pp. 94-97.

Bach, J. "Enough About Process: What We Need Are Heroes," *IEEE Software* (12), March 1995, pp. 96-98.

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

Baskerville, R., and Pries-Heje, J. "Managing Knowledge Capability and Maturity," in *Information Systems: Current Issues and Future Changes*, T. J. Larsen, L. Levine, and J. I. DeGross (eds.), Laxenburg, Austria: IFIP Press, 1999, pp. 175-196.

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

Checkland, P. "From Framework Through Experience to Learning: The Essential Nature of Action Research," in *Information Systems Research: Contemporary Approaches and Emergent Traditions*, H-E. Nissen, H. K. Klein, and R. Hirschheim (eds.), Amsterdam: Elsevier/North-Holland, 1991, pp. 397–403.

Drory, A., and Romm, T. "The Definition of Organizational Politics: A Review," *Human Relations* (43:11), 1990, pp. 1133-1154.

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

Edgar-Nevill, V. M. A. "Evaluation of the SEI Software Capability Model Within an Information Systems Context: In Pursuit of Software Quality," in *Software Quality Management II: Managing Quality Systems, Volume 1*, M. Ross, C. A. Brebbia, G. Staples, and J. Stapleton (eds.), Ashurst, UK: WIT Press, 1994, pp. 263-278.

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

Humphrey, W. S. *Managing the Software Process*, Reading, MA: Addison-Wesley, 1989.

Keil, M., and Robey, D. "Turning Around Troubled Software Projects: An Exploratory Study of the De-escalation of Commitment to Failing Courses of Action," *Journal of Management Information Systems* (15:4), 1999, pp. 63-87.

Knights, D., and Murray, F. *Managers Divided.,* Chichester, England: John Wiley & Sons, 1994.

Kohoutek, H. J. "Reflections on the Capability and Maturity Models of Engineering Processes," *Quality and Reliability Engineering International* (12:2), 1996, pp. 147-155.

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

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

Mathiassen, L., and Sørensen, C. "The Capability Maturity Model and CASE," *Information Systems Journal* (6), 1996, pp. 195-208.

O'Connel, E., and Saiedian, H. "Can You Trust Software Capability Evaluations," *IEEE Computer* (33:2), 2000, pp. 28 –35.

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

Paulk, M. C., Curtis, B., Chrissis, M. B., and Weber, C. V. *Capability Maturity Model for Software, Version 1.1. 93-TR-024,* Software Engineering Institute, Pittsburgh, PA, 1993 (http://www.sei.cmu.edu/publications/documents/93.reports/93.tr.024.html).

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

Sharp, H., Woodman, M., Hovenden, F., and Robinson, H. "The Role of 'Culture' in Successful Software Process Improvement," in *Proceedings of the Twenty-fifth EUROMICRO Conference—Informatics: Theory and Practice for the New Millennium*, Los Alamitos, CA: IEEE Computing Society Press, 1999, pp. 170-176.

Smith, W. L., Fletcher, R. I., Gray, E. M., and Hunter, R. B. "Software Process Improvement: The Route to Software Quality?" in *Software Quality Management II: Managing Quality Systems, Volume 1*, M. Ross, C. A. Brebbia, G. Staples, and J. Stapleton (eds.), Ashurst, UK: WIT Press, 1994, pp. 193-211.

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

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

About the Authors

Peter Axel Nielsen is currently an associate professor in Information Systems at the Department of Computer Science at Aalborg University. Over the past years he has been engaged in understanding information systems develop-

ment practice and the use of methodologies. His research interests include analysis and design techniques, object-orientation, and software process improvement. He is co-author of a book on object-oriented analysis and design and a forthcoming book on software process improvement. Peter can be reached by e-mail at pan@cs.auc.dk.

Jacob Nørbjerg holds a M.Sc. and a Ph.D. in computer science from the University of Copenhagen. His research interests include software process improvement, systems development as a work process, and the design, construction, and use of information systems in organizations, particularly distributed organizations. Prior to his present assignment at Copenhagen Business School, he worked at the Center for Tele-Information, Technical University of Denmark, and at the Department of Computer Science, University of Copenhagen, where he participated in an international research project about CSCW and software engineering. Jacob can be reached by e-mail at jacob@cbs.dk.