8

Risk Management*

Kalle Lyytinen
Lars Mathiassen
Janne Ropponen

Abstract. We present a simple, but powerful framework for software risk management. The framework synthesizes, refines, and extends current approaches to managing software risks. We illustrate its usefulness through an empirical analysis of two software development episodes involving high risks. The framework can be used as an analytical device to evaluate and improve risk management approaches and as a practical tool to shape the attention and guide the actions of risk managers.

Introduction

Considerable hopes in improving the performance in software development have been placed in techniques and guidelines that identify, analyze and tackle software risks (Alter *et al.* 1978; Davis 1982; McFarlan 1982; Burns *et al.* 1985; Boehm 1989, 1991; Fairly 1994). Software risks are incidents that endanger a successful development process leading to wrong or inadequate software operation, software rework, implementation difficulty, delay or uncertainty (Boehm 1991). They involve the concept of a consequence which incurs losses, is uncertain and introduces choice (Boehm 1989; Charette 1989; Barki *et al.* 1993).

Research on software risk management has primarily focused on crafting guidelines for specific tasks (Alter *et al.* 1978; McFarlan 1982; Boehm 1989; Charette 1989). This has led to a number of problems. First we have little empirical evidence of the practical

* Published as: A Framework for Software Risk Management. K. Lyytinen, L. Mathiassen & J. Ropponen. In: Journal of Information Technology, Vol. 11, No. 4, 1996.

usefulness of risk management. Second, risk management approaches shape the attention and guide the actions of risk managers in quite different and *ad hoc* ways. Third, risk management approaches have largely ignored the organizational environment in which they are used and their impact on management performance.

Our research on software risk management attempts to address these issues. We have studied risk management practices (Ropponen 1993; Ropponen *et al.* 1993); see also (Van de Swede *et al.* 1993; Neo *et al.* 1994) and we have conducted experiments with risk management approaches (Mathiassen *et al.* 1995); see also (Boehm *et al.* 1984). In addition, we have evaluated and compared classical approaches to risk management (Lyytinen *et al.* 1996). This paper is based on the insights from these studies with the goal to present a general characterization of software risk management as a distinct form of organizational behavior. The framework we present is general enough to cover the entire development process and comprehensive enough to integrate all types of software risks and it can be used both as an analytical device to evaluate, compare and develop risk management approaches and as a practical tool for software risk managers.

The paper is organized as follows. First, we introduce three complementary organizational views on software development. These are synthesized into a framework for software risk management. We use the framework to analyze software risk management and to provide a general understanding of the literature on risk-based software management. We then present two real-life software episodes to illustrate the value of the framework in making sense, organizing and re-interpreting software management practices. Finally, we discuss some implications for research and practice in software risk management.

Organizational perspectives on software development

Contemporary approaches to software risk management share a number of fundamental weaknesses (Lyytinen *et al.* 1996). First, they rely on simplistic environmental models and make no distinctions between different types of risk-generating contexts.

Second, they guide action through *ad hoc* lists of risk resolution techniques that provide a rather weak understanding of the nature

of risk management behavior. Third, they shape the attention of risk managers through specialized or narrowly focused lists of risk factors. We introduce three complementary organizational perspectives on software development. Each perspective is designed to overcome one of these weaknesses.

A hierarchical view of software development

Software development embraces three environments: the system environment in which the software system is to operate, the development environment in which the development takes place, and the management environment which shapes software management activities. The development process is generated within the development environment. Its purpose is to inquire into the system environment, anticipate effective ways to use software and thereafter implement the software system. The software management process forms a second-order process that manages the software development process and its environment. It is generated within the management environment. The concern of this process is to design and sustain an effective development environment so that the first-order process meets its goals. The practical reasons for introducing the three environments and the two related processes are: the dynamics of organizational learning, the management of complexity through separation of organizational concerns, and the management of organizational uncertainty.

Software risks are borne: 1) within the system environment, e.g., users might have no experience using the kind of software being developed (Alter *et al.* 1978; McFarlan 1982); 2) within the development environment, e.g., developers lack experience in analyzing this kind of system environment (Alter *et al.* 1978; Davis 1982; Barki *et al.* 1993); or 3) within the management environment due to managers' bias, laziness, ignorance or inaction which leads to ignoring available information (Keil 1995).

Software development as satisfying behavior

Software development is concerned with designing artifacts. Such design activities fall short of completely rational behavior in which knowledge of the environments is certain and consequences of all acts can be calculated (Simon 1979; Parnas *et al.* 1986). A software developer can at best try to discover satisfactory solutions in relation to some aspiration levels and adjust designs on the basis of his

or her information about the environments and available heuristics at hand. Alternatives are neither given nor readily at hand and the concept of search governed by heuristics is, as a consequence, essential in understanding design and management actives.

Software risks are created by limited resources, skills or information. These handicaps prevent designers successfully seeking or selecting design alternatives, estimating their usefulness, or implementing them effectively. Risk-based management is a means to derive more information about the three environments. Thereby actors decrease environmental uncertainty (i.e., increase actors' knowledge) so as to decide which alternative actions to pursue (i.e., increase actors' intelligence), and dynamically set the aspiration levels (i.e., increase design adaptability). Risk management methods specify search procedures for information gathering, organization, and interpretation to simplify complex decisions under conditions of bounded rationality (Simon 1983).

The structure and dynamics of the environments

Whilst the concept of satisfying behavior is intuitively appealing it does not help much in characterizing the content, size and the structure of search spaces. We need to describe the structure and dynamics of the three environments of software development. We use Leavitt's model of organizational change (1964) to frame the scope and structure of Simonian search spaces in the context of software development. Leavitt's presentation is appealing since it addresses organizational change and uncertainty in a general, but simple manner. Leavitt argues that organizations are multivariate systems consisting of four interacting variables—task, structure, actor, and technology. Leavitt uses the term task to denote organizational *raison d'être* such as manufacturing and servicing. By structure Leavitt means systems of communication, systems of authority and systems of work flow. Actors refer to those participants involved that carry out tasks, and technology is defined as any technical means, know-how and tools to carry out tasks. According to Leavitt, these four variables are highly interdependent and a change in one variable will result in changes in the others.

Software risks arise whenever unexpected threatening or conflicting states occur in or between the four components in any of the three environments so as to increase the likelihood of major loss (Lyytinen *et al.* 1996). Likewise, a risk reduction strategy can

change any of the four variables in the environments. Leavitt's model suggests a systematic way to organize risk identification and resolution activities which is an improvement over earlier, mostly *ad hoc* check lists (e.g., Alter *et al.* 1978; Davis 1982; McFarlan 1982; Boehm 1989; Barki *et al.* 1993). It also articulates some necessary and sufficient features of organizations that effectively cope with software risks. Finally, it clarifies the nature of software risks by relating them to the turbulence and the lack of equilibrium in the socio-technical system.

A framework for software risk management

By marrying the Simonian model of behavior (process view) with the Leavittian model of an organization (structural view) in the context of the hierarchical software development model we obtain the framework depicted in figure 1. It meets some of the requirements we have set for a framework for software risk management: it is simple, general, and comprehensive (cf., Lyytinen *et al.* 1996).

The framework describes the three software development environments symmetrically in terms of the Leavittian model: on each level we distinguish a qualitatively distinct set of tasks, technology, actors and structures. Their constellations can either create or reduce software risks depending on how they are pasted together and what types of behaviors they are able to portray. The differences in the Leavittian components on each level are exhibited by the use of distinct names for the components as exhibited by the terms "system", "project", and "management", respectively. The model recognizes that the three layers of socio-technical systems are closely intertwined by the two change processes that share properties of bounded rationality: the development process and the risk-based management process.

The risk-based management process deals with questions like: do software developers have any experience with the technological platform of the project? It thereby conducts a risk analysis of states and events that may affect the capability of the development environment to carry out the software development task within the set aspiration level (Boehm 1989, 1991; Charette 1989). It can (often unconsciously) change the development environment by enacting heuristics. For example, it can launch experiments with the technological platform. Through such risk resolution activities (Boehm

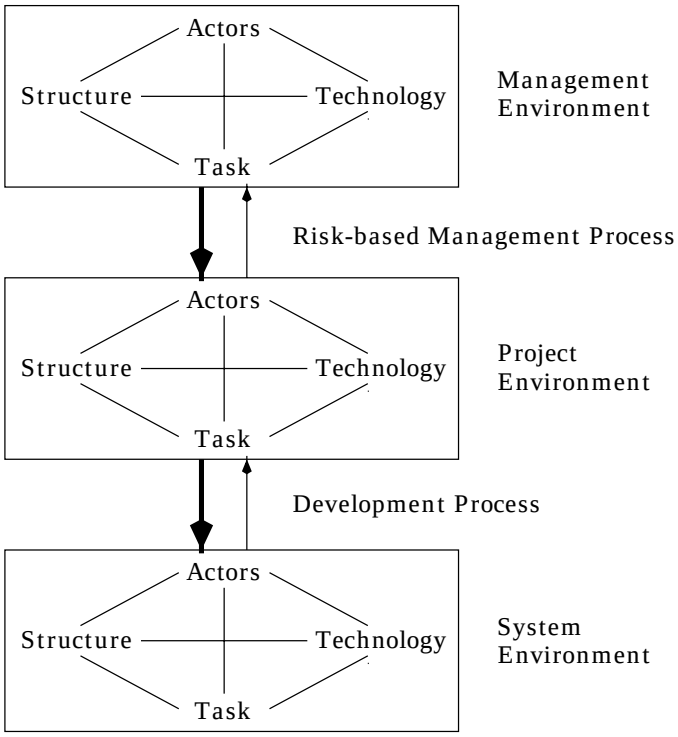

Figure 1. A framework for software risk management.

1989; Charette 1989) one or all of the environments will change (Boehm 1989).

Software risks can be seen to condense into project specific risk profiles. Such profiles are continually shaped by component interactions and changes introduced by the risk-based management process. In different project stages risk profiles vary, and at each development stage the risk profile contains several incidents which can prevent the project from meeting its aspiration levels and thus incur losses.

Risk profiles can be attacked in two ways: actively and skillfully where actors carefully scan the environments, feel responsible and committed, and willfully change aspiration levels; or passively by ignoring risks due to lack of accountability, insufficient organizational commitment, incompetence, information overload, stress, opportunism or laziness. In the latter situation the aspiration levels

decreases *de facto* and *ex post* with-out introducing an explicit choice. These two extreme strategies to deal with software risks—an active and systematic one, and a passive, laissez faire and ad hoc one—are highly dependent on how the actors, technology and structure in the management environment are joined together.

Project task

The system environment constitutes the context in which one has to understand the project task. The task is to describe a system environment in relation to stakeholders' expectations and—within specific time and cost constraints—to develop and implement a new software system by changing the necessary components of the sys-

System task	Is the task well understood?
	Is the task unstructured with many exceptions?
	How many tasks are included?
	What is the impact on user tasks?
	Will actors be critically dependent on the system?
	Is there much unarticulated, tacit knowledge involved?
	What is the extent of tasks?
	How much variation and flexibility are involved in the tasks?
System structure	Is the structure of the organization fit for the system?
	Does the system span over several organizational units?
	Does the system structure change rapidly?
	Do power relations change?
	Is the structure in harmony with the tasks?
System technology	Is the technology appropriate for the technology tasks?
	Is the technology well-proven and reliable?
	What is the life-span of the technology?
	Is the technology new in this organization?
	Is the technology standardized or changing fast?
	Is the cost of the technology prohibitive?
System actors	Do the actors have experience with the technology?
	Are their earlier experiences with IT bad?
	How knowledgeable are the actors?
	Are their expectations realistic?
	Do actors personally benefit from the change?

Table 1. Project task relates to system environment risk factors.

tem environment. Task related risk resolution techniques should not just address the project task as an abstract formulation of the project outcome and its aspiration level. Instead, a full repertoire of heuristics should be available to probe system environment components and their interactions. Some typical questions asked during risk identification are summarized in table 1 (Mumford 1983; Lyytinen *et al.* 1987; Boehm *et al.* 1989; Lyytinen *et al.* 1996).

Project structure

This component refers to the systems of communication, authority and work flow. Systems of communication specify who should be communicating with whom, when, how frequently communications take place, and how formal communications are (Andersen *et al.* 1990; Davis *et al.* 1990). Risks transpire when actors communicate ineffectively, when the appropriate actors are not involved in communications, or when the scope of communications is limited (Curtis *et al.* 1988; Heiskanen 1994). The systems of authority define lines of responsibility between project actors and groups. The more unclear and inappropriate the allocation of responsibilities, the more risks are produced (Lyytinen *et al.* 1996); if the selected organizational form is in disagreement with the actors' expectations and the nature of the task new risks are created (Constantine 1993); in addition, the risk profile changes when the work flow structure does not match the task.

Project technology

Project technology includes available methods, tools and infrastructure to design the software system and eventually to implement it. Systems development methods, quality assurance systems, computers, software, and other equipment are examples of such technologies (Cooprider *et al.* 1991). Technological shortcomings and the dynamic nature of the technology are indisputable sources of risk. Typically, the technological conditions under which the task is carried out change and these changes amplify existing weaknesses in the technology including its inadequate functionality, reliability, efficiency, or user-friendliness.

Project actors

Project actors cover all stakeholders who are involved in the process or can set forward claims to or benefit from the project. Thus, all

participating persons, groups, and other stakeholders including customers, managers, maintainers, development groups and users (Boehm *et al.* 1989) need to be included into the actor set. Actors exhibit several prominent properties that influence a project's risk profile: knowledge and skills, experience, expectations and commitments, and beliefs and values. All these can vary from one actor (group) to another. Moreover they can vary relative to an actor's task, and his or her connection to the project structure and technology.

Project component interactions

Second order considerations examine interactions between task, structure, technology, and actors as major sources of risks. Examples of such considerations are: actors' ability and shortcomings in performing the task, the appropriateness of the structure and the technology in relation to the task, the effectiveness with which the structural arrangements support and utilize the actors' abilities, skills and experience in using the technology, and finally the fit between technology and structural arrangements (Lyytinen *et al.* 1996).

Risk-based software management

The performance of the management environment (cf., figure 1) depends on how actors, structure and technology in the management environment are assembled. Such performance is defined by the available information processing capability to find out critical incidents, to suggest alternatives, to calculate consequences and thereby to resolve software risks.

Management task

The risk-based management task is to search for incidents that can prevent the development environment from performing in a satisfactory manner and to modify the development process by intervening into the development environment. This is indicated by the big arrow pointing downwards from the management environment in figure 1. Software risks are born in all three environments. Table 2 offers a set of generic questions that can be used to search for specific risks in a software project. Table 1 offers more specific ques-

	System	Project	Management
Task	Which tasks are the software supporting?	What are the requirements to the system and its environment?	What is an appropriate project environment?
Structure	Which organization is the software system part of?	How is the project organized?	How are the management activities organized?
Technology	Which technical platform is the software system implemented on?	Which technologies are used to develop the software system?	Which technologies are used to manage the project?
Actor	Who are using the software or affected by it?	Who are involved in or affected by the project?	Who are involved in managing the project?
Relations	How is the fit and what is the dynamics between components?	How is the fit and what is the dynamics between components?	How is the fit and what is the dynamics between components?

Table 2 Generic questions to support risk-based management.

tions focusing on the system environment and hence related to the project task.

The risk-based management task is dynamic: it differs through project phases and between development projects and it involves a feedback loop indicated by the small arrow in figure 1. Through this loop managers become more competent and experienced, the organization integrates new experiences and schemes into its systems of interpretation (Weick *et al.* 1983), and new methods, codifications of procedures, and decision making rules are adopted.

We can observe a large variation in how the risk management task is formulated and accomplished. Sometimes risk-based management forms an integral part of the project management activities. This corresponds to the continuous view of risk management advocated by Boehm (1989, 1991); Boehm and Ross (1989) and Alter and Ginzberg (1978). Sometimes risk-based management forms a

discrete event which relates only to the very early phases of a software project. This corresponds to the discrete view of risk management presented by David (1982) and McFarlan (1982).

Management actor

Risk managers are expected to take deliberate actions to tackle risks. Usually this role is assigned to a project manager. But other project members and groups such as project committees can also carry out risk management tasks. Risk managers form a proactive part of the management environment: they decide what actions should be taken, what risk management technologies should be used, and how the management structure should be organized. They also set the aspiration levels that are honored during searches and consequent risk resolutions.

Finding skillful, open-minded software managers is instrumental in improving the risk management capability (Boehm 1989; Van de Swede *et al.* 1993). Actors' attitudes, psychological make-up and cognitive bias can drastically affect the way in which they deal with software risks. This has been pointed out by such actor features as “escalating commitment” (Keil 1995), “no-problem syndrome” (Weinberg 1986), “group think”, and “inaction” (Weick 1979). Actors' capability can be improved by increasing their awareness of risk management methods, improving their ability to use the methods, and forging values and preferences that deliberately recognize risks, esteem high quality and the lack of errors (cf., Boehm *et al.* 1989; Roberts 1993; Keil 1995).

Management technology

Risk management methods offer heuristics to identify risks, compose a risk profile, evaluate its significance and attack it accordingly. These methods form the Simonian “intelligence” that can be transformed from one environment to another. Most of these methods are codified representations of good management practices. Willcocks and Margetts (1994) convincingly demonstrate that conventional risk management technologies tend to ignore external factors that affect the risk management capability. These include: history like the past performance that shapes actors' expectations; external structural factors like external pressures, constraints, and omissions; internal structural factors like the misfit between the or-

ganizational and IS management structure; and actor related features like past experience, culture and climate (Keil 1995).

Management structure

Communication lines, the structure of authority, and lines of accountability are significant in organizing the risk-based management process. It is important to effectively communicate risks to everyone involved and to reward reporting of omissions and errors (Keil 1995). It is also necessary to define systems of authority for tactical considerations on which risk resolution strategies to apply when (Boehm *et al.* 1989). The management structure is also important to create a sense and discipline of accountability, i.e., that quality makes a difference and that actors are accountable for what they are doing (Rochlin 1993). At the same time, the management structure must be organized to provide adequate and reliable information and a management structure which favors openness and does not punish voicing of problems and concerns (Keil 1995).

Interactions and improvements

The intrinsic and complex relations between the four Leavittian elements are expressed in questions like: how do technologies relate to specific management tasks? what level of experience and competence do the involved actors have in using these technologies? who is responsible for managing these risks and when? Software organizations' effectiveness can be nurtured over time by perfecting the quality of some, or all components, in the management and development environments. Such developments, however, are not the subject of project management, but one important target in managing and orchestrating the larger environment for software delivery (Earl 1989). Analysis of such changes has taken place in discussions of evolutionary models of IT diffusion (Lyytinen 1991). These models are often called maturity models. The weakness of such models is their primary focus on well-defined processes of managing complexity and what could be called lubrication of the organizational machine. As a consequence they tend to ignore the importance of an organic focus to deal effectively with high levels of complexity and uncertainty. But despite their weaknesses, maturity models have a definite value in distinguishing successive levels of how the development capability can mature over time (Nolan 1973; Humphrey 1989; Galliers *et al.* 1991). Thereby, through successive

trials of analysis of weaknesses, maturity models can be used to diagnose what organizations have learned (i.e., their current level of intelligence in Simonian terms) in managing software risks, and to point out that organizations must continually improve their risk management capability (i.e., the necessity to learn to learn (March *et al.* 1991)).

Two cases

In the following, we use the framework to structure and interpret two software development episodes involving considerable risks. In case Alpha, risk management techniques were actively applied and the management environment was organized accordingly. In case Beta, no conscious risk management strategy was followed. We analyzed both cases using published written descriptions (Gjesing 1993; Markus *et al.* 1994; Keil 1995). In addition, we have communicated with the original authors and checked that our interpretations are authentic and valid. Our primary goal is to illustrate the value of the framework in making sense, organizing and re-interpreting software risk management practices.

In case Beta, the risk profile was never fully established. The system environment was, however, a major source of risks and continued to be so throughout the project. At the same time a management structure was installed that did not match with the risk levels and it was populated with actors that were cognitively and emotionally biased. Accordingly, the management environment was never in equilibrium with the risk based management task and it was handled with an inappropriate assembly of actors, management structure and technology. In contrast, risk management was carried out consciously as a kind of one-shot strategy in case Alpha. Risk information was sought, structured and at the end analyzed using quantitative tools. This activity was carried out by an individual project manager. Risks arising from all three environments were systematically detected and planned for and, despite initially high levels of risks, the development process could be managed better than usual.

Case Alpha: advanced banking system

The project task was to develop a general filing and retrieval system for a large bank which would facilitate coordination between bank

employees while working on the same customer case (Gjesing 1993). Another goal was to establish homogeneous working procedures for dealing with customer cases. The overall task was of medium size (approximately 12 man years) and it involved a fair amount of technological novelty and complexity.

The system environment is a large bank. Relevant system actors are the entire bank personnel who are eventually going to use the system. The system structure reflects the division of work and employed coordination mechanisms to manage customer cases. The system (task) stores and retrieves files on customer cases on which the personnel are currently working. The system technology consists of the bank's existing computing facilities supplemented with a new file management system and an electronic mail system.

The development process was split into two stages because of the newness of technology components and the necessary changes in system structure. The timing and quality constraints for the task were tight. During the project's first year a prototype was to be developed and tested within a limited functional area. Based on the experiences gained from this prototype a full-scale system was then to be developed over a period of another two years. The first experimental stage was, as a consequence, crucial for the success of the whole project as further developments depended on the outcomes of this initial stage.

The development environment covers part of the bank's IS department, an organization with about 800 employees operating at two different sites. Both sites develop and maintain the same computing infrastructure and no clear functional division had been made between the sites. The project actors in this case were the project manager accompanied by other project members and user representatives. Overall the project was staffed by 10 full time members. The project structure forms a complex organizational web, because the project members had to be obtained from two departments representing both sites of the IS department, and user representatives from several functional areas. The project technology consisted of the system technology and a new object oriented methodology and a new CASE tool. The project task, in the first stage of the project, was to develop the prototype, to garner sufficient experiences from the design, implementation and use of it, and to produce information for making decisions regarding the continuation of the project.

The project manager worked in a specific management environment. He conducted a risk analysis and managed the development process based on this information. The project manager was inexperienced with managing projects of this size but he was willing to follow risk management principles and committed to the management process. Parts of the risk analysis were conducted in team meetings using the risk analysis steps suggested by Boehm (1989). The risk management technology was the method suggested by Boehm (1989, 1991) and the management structure was based on traditional project management principles.

The project manager saw several threats including new technology, complex project organization, and a new application area. Therefore he executed risk-based project planning in two steps (Boehm 1989, 1991): 1) risk assessment step—consisting of risk identification, risk analysis, and risk prioritization; and 2) risk control step—consisting of risk management planning, risk resolution, and risk monitoring. During risk assessment, a list of 29 risk items were compiled using Boehm's top-ten list. It was supplemented with informal discussions with colleagues, the use of rich pictures (Checkland 1981), soliciting past project experience, and conducting a detailed analysis of the system requirements and success factors. Each risk was further analyzed carefully in relation to the short term goals of the project and assessed quantitatively.

The risk profile included risks associated with: the project task (insufficient requirements, generality of the design); project technology (new CASE tool, difficulty in the implementation of the basic file system and electronic mail system, inefficient automatically generated code); project actors (inexperienced project manager); project structure (technical subcontracts, considerable coordination effort, tight schedule), and actor-technology fit (project members inexperienced with object oriented methodology). Subsequently, risk control activities were carried out with emphasis on risk management planning. Risk resolution tactics were evaluated, further designed, and integrated into the project plan.

The project manager found that this approach shed light on the instrumental role of the risk management technology in improving searches, finding useful information, and in structuring the information. He also identified a number of weaknesses: the approach is based on a pessimistic world view and important risks can be neglected because they are mentioned so often (the-wolf-is-coming-

effect); there is no mechanism for dealing with time; there is no support for which risk resolution strategy to apply and when. These problems indicate weaknesses in pasting together the management environment: actors' cognitive bias (the-wolf-is-coming-effect), poor commitment (pessimistic world-view), lack of structure (time dimension), and poor heuristics (no support for strategy choice). Finally, the project manager argues that these weaknesses can be avoided if one is aware of them (by finding an appropriate technology-actor fit).

Case Beta: expert system to support sales representatives

The project task was to design and implement an advanced expert system that could be used by the sales force to configure error-free computer systems while drafting customer orders (Markus *et al.* 1994; Keil 1995). The project was originally welcomed with big fanfare, promised staggering return on investment, had great visibility in the organization's business strategy, and obtained early management commitment. Moreover, the project had good resources, had good technical expertise at hand, and relied on a careful project planning and user participation strategy during the system roll out (Mumford *et al.* 1989). It also had a number of risk factors that were well recognized from the start such as untried system technology, unique application area, and inexperienced users. The project task was large in size (over 100 man years), its complexity was high, and its implementation scope was broad (Mumford *et al.* 1989).

The system environment is part of a large computer vendor. The relevant system actors are mainly the vendor's sales-representatives who were expected to use the system as a tool to configure the systems they sold. The idea originated from the manufacturing department which realized that most configuration errors are borne in sales and that their expert system used in configuring the manufactured systems could be leveraged also to provide configuration support for sales. The sales department, however, showed little interest in the project originally and was during the initiation of the project developing its price quotation system which it saw as a strategic application (Markus *et al.* 1994). The system structure reflects the traditional division of work in the sales organization. The system task is to prompt a series of questions about the system being sold and thereafter to configure it correctly by looking for mismatched items and components, to lay out processors and other

components in cabinets, to lay out the floor plan, and to do the cabling. The system technology consists of standard hardware available from the vendor, proprietary expert system software to build up the data and rule base, and telecommunication facilities available for the sales-offices. Part of the system technology was not tested before.

The development process consisted of the initial design and implementation phase which lasted from late 1980 to end of 1983. During that period the system was designed, implemented and piloted among a number of our user representatives. Piloting was found necessary to test out the technology and to get users involved. The second, roll out, phase lasted from late 1983 until late 1992. During this period the system was continually adapted and modified and its deficiencies and implementation hurdles were attacked through several larger or smaller implementation measures. After a solid start the project faced chronic implementation problems, despite several extensions and improvements in software, until it was abandoned. The development environment includes the company's sales offices and representatives and during its peak the project staff consisted of 26 technical specialists and management staff. The project structure is complex. It involves the management of the technical system implementation, users through user design teams and coordination with changes in company's organization, product lines, strategy, sales channels and campaigns. The project technology is basically the same as the system technology mentioned above. It was familiar to the technical implementors. The initial project task was to finish the project by December 1982 (Keil 1995).

The management actors are the project managers and other managers involved in making decisions. Most of these were not knowledgeable in developing this kind of system. Those who had experience had not been involved in a similar type of development effort. The management structure was designed so that few people controlled both the development resources and the auditing of system outcomes (Keil 1995). The management structure did not specify responsibilities and lines of communication clearly which later led to ignoring available information and misinforming higher levels of management. The organization had, however, experience in successfully managing projects of this size. In addition, the project strategy suggested active user participation in designing the use of the system. The project was, in summary, not specifically badly

managed when compared to other more successful projects in the organization.

Nonetheless, the project did not apply any risk management technologies and management actors constantly ignored available signals on system risks. A project risk profile was never systematically constructed though some risks were identified such as the novelty of the system technology and organizational implementation difficulty. No specific plans to reduce these risks were devised. During the course of the development process new risks emerged including wrong understanding of sales-representatives' work, poor or lacking functionality, an inadequate and cumbersome user-interface, poor data quality, slow performance, need for linkages with other company's applications and too tight deadlines (Keil 1995; Markus *et al.* 1994). Some of these—especially lack of linkages to other applications, lacking functionality and the inappropriate fit between the system structure and system task (Markus *et al.* 1994)—were, at the end, fatal for the system.

Conclusion

We have presented a framework for software risk management and we have used it to: i) describe the nature of software risk management, ii) provide a general understanding of the literature on risk-based software management, iii) interpret software episodes involving high risks, and iv) provide generic checklists covering the space in which software risks are born. The framework should not be seen as a traditional testable model. Instead, the framework presents, in relation to the published literature so far, a wider and more systematic way to organize software risk considerations.

The framework sheds light on areas to improve software production. Organizations should be attentive to risky incidents that can help to change behaviors and beliefs and the use of risk management methods should be encouraged. Risk management considerations should cover all the four components—task, structure, technology and actor—in all three environments. Table 2 offers a survey of key generic questions that define the space in which specific software risks are born, and table 1 provides more specific questions related to the identification of risks in the system environment.

Such checklists and other risk management methods provide a fast approach to improve the risk management ability of a software organization. Changes in the other components of the management environment are outcomes of evolutions unfolding over longer periods of time: hiring new people, changing organizations' competencies, skills and beliefs, or restructuring the management and development practices. Such drastic interventions pay off only over longer periods of time—but then, in most cases, they do pay off more handsomely.

Acknowledgments

The authors are thankful for useful comments from the editors and reviewers and for discussions with Michael Gjesing, Mark Keil, Karleen Roberts, and Roy Schmidt.

References

Alter, A. & M. Ginzberg (1978): Managing Uncertainty in MIS Implementation. *Sloan Management Review*. Vol. 20, No. 1 (23–31).

Andersen, N. E., F. Kensing, J. Lundin, L. Mathiassen, A. Munk-Madsen, M. Rasbech & P. Sørgaard (1990): *Professional Systems Development: Experience, Ideas and Action*. Hemel Hempstead, Hertfordshire: Prentice Hall.

Barki, H., S. Rivard & J. Talbot (1993): Toward an Assessment of Software Development Risk. *Journal of Management Information Systems*. Vol. 10, No. 2 (203–225).

Boehm, B. W. (1989): *Software Risk Management*. Tutorial. Los Alamitos, California: IEEE Computer Society Press.

Boehm, B. W. (1991): Software Risk Management: Principles and Practices. *IEEE Software*, Vol. 12, January (32–41).

Boehm, B. W., T. Gray & T. Seewaldt (1984): Prototyping Versus Specifying: A Multiproject Experiment. *IEEE Transactions on Software Engineering*. Vol. 10, No. 3.

Boehm, B. W. & R. Ross (1989): Theory-W Software Project Management: Principles and Examples. *IEEE Transactions on Software Engineering*, Vol. 15, No. 7 (902–916).

Burns, R. & A. Dennis (1985): Selecting an Appropriate Application Development. *Database*. Vol. 17, Fall (19–23).

Charette, R. N. (1989): *Software Engineering Risk Analysis and Management*. New York: Intertext Publications, McGraw-Hill.

Checkland, P. (1981): *Systems Thinking, Systems Practice.* Chichester: John Wiley.

Constantine, L. (1993): Work Organization: Paradigms for Project Management and Organization. *Communications of the ACM*. Vol. 36, No. 10 (32–43).

Cooprider, J. G. & J. C. Henderson (1991): Technology Process Fit: Perspectives on Achieving Prototyping Effectiveness. *Journal of Management Information Systems.* Vol. 7, No. 3 (67–87).

Curtis, B., H. Krasner & N. Iscoe (1988): A Field Study of the Software Design Process for Large Systems. *Communications of the ACM*. Vol. 31, No. 11 (1268–1287).

Davis, G. B. (1982): Strategies for Information Requirements Determination. *IBM Systems Journal.* Vol. 21, No. 1 (4–30).

Davis, G. B., A. S. Lee, K. R. Nickles, S. Chatterjee, R. Hartung & Y. Wu (1990): *Diagnosis of an Information System Failure: A Framework and Intrepretive Process.* Management Information Systems Research Center, WP-01-06, University of Minnesota.

Earl, M. (1989): *Management Strategies for Information Technology.* London: Prentice Hall.

Fairly, R. (1994): Risk Management for Software Projects. *IEEE Software.* Vol. 11, No. 3 (57–67).

Galliers, R. D. & A. R. Sutherland (1991): Information Systems Management and Strategy Formulation: The "Stages of Growth" Model Revisited. *Journal of Information Systems.* Vol. 1, No. 2.

Gjesing, M. V. (1993): *Risk-based Project Management—An Example.* The Danish Bank Academy. (In Danish)

Heishanen, A. (1994): *Issues and Factors Affecting the Success and Failure of a Student Record Systems Development Process—A Longitudinal Investigation Based on Reflection-in-Action*. Ph.D. Dissertation, Department of Computer Science, University of Tampere.

Humphrey, W. S. (1989): *Managing the Software Process.* Software Engineering Institute, The SEI Series in Software Engineering. Reading. Massachusetts: Addison-Wesley.

Keil, M. (1995): Pulling the Plug: Software Project Management and the Problem of Project Escalation. *MISQ.* Vol. 19, No. 4 (421–447).

Leavitt, H. J. (1964): Applied Organization Change in Industry: Structural, Technical and Human Approaches. (55–71) in H. J. Leavitt (Ed.): *New Perspectives in Organizational Research.* Chichester: John Wiley.

Lyytinen, K. (1991) Penetration of Information Technology in Organizations: A Comparative Study Using Stage Models and Transaction Costs. *Scandinavian Journal of Information Systems.* Vol. 3 (87–109).

Lyytinen, K. & R. Hirschheim (1987): Information Systems Failures—A Survey and Classification of the Empirical Literature. (257–309) in: *Oxford surveys in Information Technology*. Vol. 4. Oxford: Oxford University Press.

Lyytinen, K., L. Mathiassen & J. Ropponen (1996): Attention Shaping and Software Risk —A Categorical Analysis of four Classical Approaches. Submitted for publication.

March, J., L. Sproull & M. Tamuz (1991): Learning From Samples of One or Fewer. *Organization Science.* Vol. 2, No. 1 (1–13).

McFarlan, W. (1982): Portfolio Approach to Information Systems. *Journal of Systems Management.* Vol. 33, No. 1 (12–19).

Markus, L. & M. Keil (1994): If We Build It, They Will Come: Designing Information Systems that Users Want to Use. *Sloan Management Review*.

Mathiassen, L., T. Seewaldt & J. Stage (1995): Prototyping and Specifying: Principles and Practices of a Mixed Approach. *Scandinavian Journal of Information Systems.* Vol. 7, No. 1 (55–72).

Mumford, E. (1983): *Designing Human Systems.* Manchester: Manchester Business School.

Mumford, E. & B. W. McDonald (1989): *XSEL's Progress—The Continuing Journey of an Expert System*. Chichester: John Wiley.

Neo, B. S. & K. S. Leong (1994): Managing Risks in Information Technology Projects: A Case Study of TradeNet. *Journal of Information Technology Management.* Vol. 5, No. 3 (29–45).

Nolan, R. L. (1973): Managing the Computer Resource: A Stage Hypothesis. *Communications of the ACM.* Vol. 16, No. 7 (399–405).

Parnas, R. & P. Clemens (1986): A Rational Design Process: How and Why to Fake It. *IEEE Transactions on Software Engineering.* Vol. 12, No. 2 (251–257).

Robens, K. (Ed.) (1993): *New Challenges to Understanding Organizations.* New York: MacMillan.

Rochlin, G. (1993): Defining “High Reliability” Organizations in Practice: a Taxonomic Prologue. (11–32) in K. Robens (Ed.): *New Challenges to Understanding Organizations.* New York: MacMillan.

Ropponen, J. (1993): *Risk Management in Information Systems Development.* Licentiate thesis, Department of Computer Science and Information Systems, University of Jyväskylä.

Ropponen, J. and Lyytinen, K. (1996): How Software Risk Management Can Improve Systems Development: An exploratory Study. Accepted for publication in *European Journal of Information Systems.*

Simon, H. (1979): Rational Decision Making in Business Organizations. *American Economic Review*. Vol. 69, No. 4 (493–513).

Simon, H. (1983): *Theories of Bounded Rationality, Behavioural Economics and Business Organization*. Vol. 1–2 (160–176). Cambridge: MIT Press.

Willcocks, L. & H. Margetts (1994): Risk Assessment and Information Systems. *European Journal of Information Systems.* Vol. 3, No. 2.

Weick, K. (1979): *The Social Psychology of Organizing.* Reading, Massachusetts: Addison-Wesley.

Weick, K. & R. Dafi (1983): The Effectiveness of Interpretation Systems. In K. Cameron *et al.* (Eds.): *Organizational Effectiveness: A Comparison of Multiple Models.* New York: Academic Press.

Weinberg, G. M. (1986): *Becoming A Technical Leader—An Organic Problem Solving Approach*. Dorset House Publishing.

Van de Swede, V. & J. van Vliet (1994): Consistent Development: Results of a First Empirical Study on the Relation Between the Project Scenario and Success. (80–93) in G. Wijers *et al.* (Eds.): *Proceedings of the 6th CAiSE Conference.* Utrecht: Springer Verlag.