9

Attention Shaping*

Kalle Lyytinen
Lars Mathiassen
Janne Ropponen

Abstract. The paper examines software risk management in a novel way in which managers address risks through sequential attention shaping and intervention. These risks represent incongruent states within and between the four components of Leavitt's model: task, structure, technology, and actors. Such a state can in the extreme case lead to a failure of developing or implementing the software, and thus to a major loss.

Based on this model, we analyze four classical risk management approaches: McFarlan's portfolio approach, Davis' contingency approach to requirements determination, Boehm's approach to software risk management, and Alter & Ginzberg's implementation approach. The analysis shows how these approaches embody significant behavioral and ideological differences in the role and possible actions of risk managers during software development. This suggests that managers should combine approaches to contingently install comprehensive risk management practices under different environmental conditions.

Keywords: software development, risk management, organization theory, content analysis

1. Introduction

The information systems field has been plagued by system failures (Lyytinen *et al.* 1987). To combat these, several approaches for man-

* Published as: Attention Shaping and Software Risk: A Categorial Analysis of Four Classical Approaches. K. Lyytinen, L. Mathiassen & J. Ropponen. In: ??? Submitted for Publication

aging software development[1] risks have been developed during the last two decades. In general, they help identify, analyze, and tackle *software risks* (Charette 1989; Boehm *et al.* 1989; Boehm 1991; Fairley 1994), *implementation risks* (Lucas 1981; Alter *et al.* 1978; Keen *et al.* 1978; Lyytinen 1987; Kwon *et al.* 1987), *project portfolio risks* (McFarlan 1982; Earl 1987), or *requirements risks* (Davis 1982; Burns *et al.* 1985) by formalizing risk oriented correlates of success and deriving from them a set of managerial principles (Boehm 1991). The approaches serve as a means to identify and manage incidents which form threats to successful software operation, or cause software rework, implementation difficulty, delay, or uncertainty.

It is surprising to notice how diverse these approaches are. They often address similar situations with different tactics and different situations with similar tactics. This raises questions like: how do these approaches relate to one another? why do they differ so much? which concepts underlie these approaches? do the approaches embody any theoretically reasoned theory of risk and risk oriented behavior? Overall, the area seems to lack theories which help relate risk management approaches and explain to what extent, how, and why they vary. The lack of such notions is due to weaknesses in current ways to investigate software risks. Most students of software risk management have not been attentive to a growing body of organizational theories of risky behavior, uncertainty absorption, and organizational change (March 1988; Galbraith 1977; Perrow 1984; Roberts 1993). Yet, by drawing upon such theories we can nurture systematic accounts of how managers attack software risks. Moreover, a majority of works in software risk management have been normative and provide relatively weak theoretical analysis of the nature of risk management (Boehm 1989, 1991; Charette 1989; McFarlan 1982).

The goal of this paper is to overcome some of these weaknesses. We contrast decision theoretic notions of risk with the behavioral one suggested by March *et al.* (1987) and demonstrate how the latter concept has more fidelity in characterizing risk management approaches. The key insight of the behavioral perspective is that software risk managers, rather than following a rational calculus, attempt to master their environments in order to avoid major losses. We use Leavitt's (1964) theory of organizational change as a means to explore the attention shaping (gestalts), heuristics (pattern

matching), and intervention embedded in doing so. More specifically, we use categorical analysis to describe the attention foci, heuristic rules, and intervention content of four classical approaches: Boehm (1989, 1991), Davis (1982), McFarlan (1982), and Alter *et al.* (1978).

After a brief overview of the use of risk concepts in managerial theory and software development, we summarize Leavitt's organizational model and justify its use as a means to analyze the content and logic of different risk management approaches. After this, we conduct a categorical analysis of the four classical risk management approaches. We conclude by raising questions about the studied approaches, by suggesting how they can be combined to reduce managerial bias, and by proposing some new avenues for empirical research in software risk management.

2. Risk management

2.1. Risk management in managerial theory

In the rational decision theory the risk concept reflects the variation in the distribution of possible outcomes, their likelihoods, and their subjective values (Arrow 1965). A risky alternative is one for which the variance is large; and risk is one of the attributes which along with the expected value of the alternative, are used in evaluating alternative gambles. The risk concept is thus embedded in the idea of rational choice affected by the expected return of the alternative. The theory expects that decision-makers deal with risk by first calculating alternatives and then choosing one alternative among the available risk-return combinations thus behaving in a rational manner (Yates 1992).

The decision theoretic view is not, however, consistent with empirical studies of how managers deal with risks (March *et al.* 1987; Bromley *et al.* 1992). In these studies, the calculus managers follow is shown to be less precise. First, uncertainty of positive incomes is not an important aspect of risk; a risky choice is one that contains a threat of a very poor performance (March *et al.* 1987; Fischoff *et al.* 1984). Second, risk is not a probability concept; it deals with the magnitude of the bad outcome (Bell 1985). Accordingly, managers act in a loss-aversive manner instead of a rational as predicted by the traditional theory (Kahnemann *et al.* 1982; Arrow 1965). Finally, though quantities may be involved in assessing

the level of risks, there is little desire to reduce risk to a single construct.

The behavioral view of risk has several implications. First, when risks involve great losses managers seek to avoid risks rather than just accept them (Cyert *et al.* 1963). They make fast decisions to avoid risks, negotiate uncertainty absorbing contracts, or just delay decisions if possible (MacCrimmon *et al.* 1986). Second, managing risks is not seen as gambling, but mastering the environments so as to bring the risks under control (MacCrimmon *et al.* 1986; cf. Adler's (1980) concept of "risk makers"). Third, managers neither understand, nor care to use precise probability estimates: crude characterizations are used to exclude certain possibilities from the decision (Fischoff *et al.* 1981) and thus make the managerial process a sequential pruning exercise instead of a one-shot decision.

2.2. Risk management in software development

In the literature dealing with software risk (Boehm *et al.* 1989; Charette 1989; Boehm 1991), risks are primarily defined following the decision theoretic view. Software risks form speculative gambles which have both a loss and profit associated with them. Yet, advice on how to manage such risks pay lip-service to this concept, i.e. most risk management approaches deal solely with negative outcomes and how to avoid them. Thereby the central insight of the decision theoretic view (the importance of considering the whole distribution of possible outcomes) becomes obscured.

The behavioral perspective (March *et al.* 1987) seems more appropriate in explaining current practices of managing software risks for several reasons. Software risk management approaches focus on ambiguous losses (Boehm 1991), they rely on multidimensional, qualitative models that make the management task simple and practicable, and they seek to avoid risks rather than optimize their expected returns by suggesting a plethora of means to control the environment. Examples of such arrangements abound in the literature including: investments in methods and standards, ignoring user resistance through "sign-off" practices, or preferring standards for interfaces (Boehm *et al.* 1989). Finally, risk management approaches exclude bad alternatives rather than optimizing expected returns (Boehm 1991; Charette 1989).

Behavioral interpretations invite us to study how software developers enact and maintain images of their task and environments through limited, selective, often unquestioned, and relatively stable cognitive strategies. The strategies help them make sense of the situations, exclude bad choices, and control the environments (Ciborra *et al.* 1987; Seely Brown *et al.* 1991). Cognitive strategies—as formalized in risk management approaches—provide us with concrete examples on how images of software development are built, what their content is, how they are to be enacted, and what their principal structure is.

Risk management strategies distinguish between the realm of managerial action and inquiry which is driven by attention shaping patterns and theories of managerial intervention, and the realm of the real world where the actual software development takes place, see figure 1. Risk management strategies use observations from the past, they learn from analogical situations, and they use deductive reasoning (March *et al.* 1991) to detect risky incidents. Over time, observations are generalized by crafting specific theories of cause-effect chains into generic risk items. These risk items guide managerial inquiry and shape the scanning of the environments for circumstances which call for managerial action. The risk items are combined with risk identification and analysis techniques to scan and make sense of the environments. In addition, risk management approaches feature a repertoire of risk resolution techniques. These are derived from local causal theories on how risky incidents affect software development and how interventions affect development

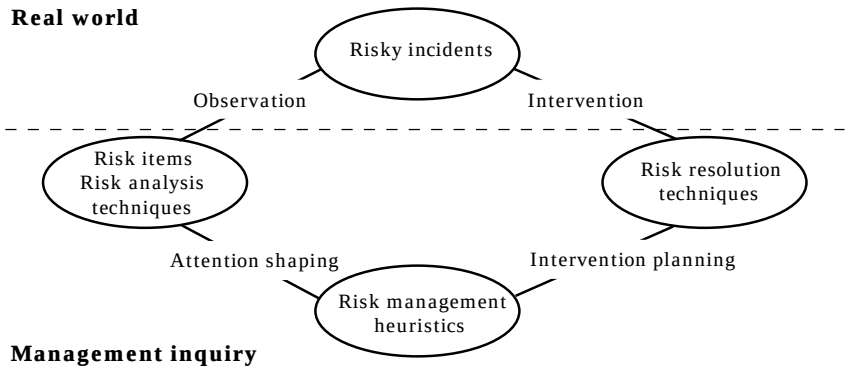

Figure 1. Risk Management Strategy.

trajectories. The techniques help formulate schematic plans for interventions that decrease the likely impact of risky incidents, or avoid it altogether. The plans are formulated using a set of heuristics that show how risk managers should relate observed risky incidents to effective managerial interventions.

This concept of risk management strategy implies a number of crucial questions: 1) which aspects of the situation are examined and inductively generalized into risk items? 2) which risk items are contained in different approaches? 3) which risk resolution techniques are proposed? and 4) how do available heuristics help match risky incidents with risk resolution techniques.

3. A Leavittian model of software risks

3.1. Application of Leavitt's theory of organizations

Applications of organization theory in the study of software development have become common (Beath 1983, 1987; Swanson 1984; Swanson *et al.* 1990; Nidumolu 1995; Saarinen *et al.* 1993b). These studies draw primarily on theories which take into account task uncertainty (Galbraith 1977), complexity, or participant's opportunism (Williamson 1975) as important organizational design parameters and apply them to problems of software development organization and control. Yet, none of these have examined in detail the sources and forms of software risks, nor the contents of suggested interventions into development situations. To do so we need a simple, but yet sufficiently rich organizational model of software development and software risk.

We chose Leavitt's open systems model of organizational change (Leavitt 1964) which has been widely used in classifying schools of organizational change in the management literature for several reasons[2]. First, even though software development organizations are less persistent than normal organizations, they share important characteristics with them. Common characteristics include structuring principles, environmental adaptation and scanning techniques, leadership patterns, and change dynamics. This all warrants the use of Leavitt as a basis for developing a classification scheme for risk management strategies. Second, Leavitt's model was originally developed in the organizational theory as an attempt to achieve a synthesis of major dimensions and dynamics of organizational change. It sought to cover all major schools and simultane-

ously play out their differences (Leavitt 1964). This is quite similar to what we are trying to do with software risk management approaches. Third, Leavitt's theory has been used extensively in the IS literature including design methods (Mumford 1983) and implementation research (Keen 1981; Kwon *et al.* 1987), and it is therefore widely known. Fourth, Leavitt's model is sufficiently broad as to take into account the most essential aspects in software development which are at the focus of software risk management approaches. Fifth, it shares virtues of a good organizational model: it is simple, and it is reasonably well defined to be applicable for analyzing risk management approaches. Finally, as earlier research using the model shows, it can be extended[3] to accommodate new dimensions should any such need arise.

In Leavitt's model (1964) organizations form multivariate systems consisting of four interacting components —task, structure, actor, and technology. These components can easily be translated into well known elements of software development: actors cover all stakeholders including users, managers, and designers; structure denotes project organization and other prevailing institutional arrangements; technology means development tools, methods, and hardware and software platforms; and task signifies expected outcomes in terms of goals and deliverables (see figure 2).

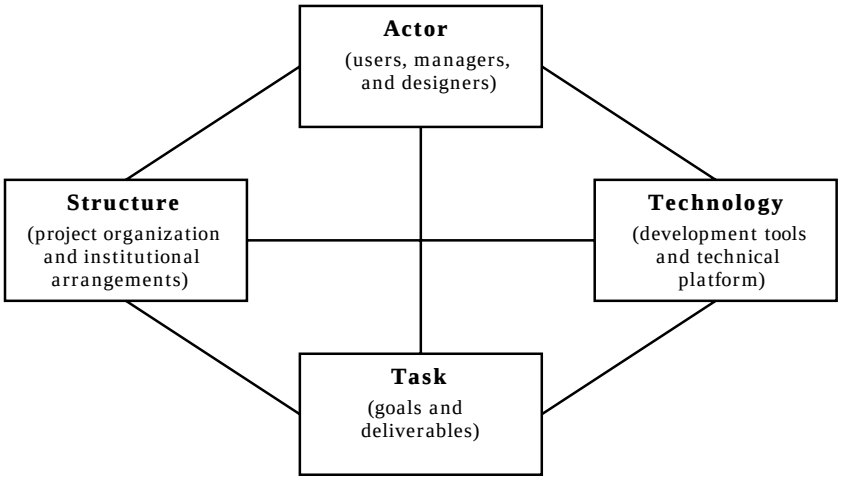

Figure 2. Software development and Leavitt's model.

Leavitt's model originates from open systems theory. Its key conjecture is that model components are strongly interrelated: change in one component will have effects, planned or unplanned, on the others. This mutual dependency is depicted in figure 2 by the edges that connect every component with the others thus creating its well known diamond shape. The model also postulates—again based on the open system equilibrium notion—that if one component's state is incongruent with others, this will create considerable dysfunctional effects and decrease the performance of the whole. Some intervention is consequently needed to re-establish the balance. But an unbalance, retained over longer periods of time, will induce oscillations in the system which, in the end, will decrease its capability to accomplish its task, and eventually undermine the existence of the whole system.

The connection between the Leavittian model and the concept of software risk can be stated as follows: a change in any Leavittian component in a systems development process can create disturbances which, in the extreme, can lead to a failure of the software change, otherwise known as a loss. Such major disturbances should be understood as those risky incidents which increase the difficulty in estimating what the performance of the development project is likely to be in terms of product or process (Nidumolu 1995). According to Leavitt's model these incidents always fall into one or several Leavittian elements—such as technology and its reliability—and their connections—such as a person's capability to deal with the technology. Active "gestalts" of likely risky incidents help managers identify the risks, estimate their severity, and, once observed, orchestrate interventions into one or several of the Leavittian components using risk resolution techniques.

3.2. A Leavittan view of software risks

Leavitt uses the term *task* (Ta) to describe an organization's raison d'etre (Leavitt 1964). In software development a task is normally defined through project deliverables and process features, i.e. a development task dictates *what* software developers should accomplish and *how* (Blokdijk *et al.* 1987). Several task related features increasing the exposure to risk have been identified: task size or complexity (Lyytinen 1987; Curtis *et al.* 1988; Oz 1994; Waters 1993; Zmud 1980; Beath 1987), task uncertainty (Burns & Dennis 1985; Lucas 1982; Turner 1992; Waters 1993; Nidumolu 1995), specificity

of design (Saarinen *et al.* 1993b), the stability of the task (Walters 1993; Saarinen *et al.* 1993a), or the ambiguity or existence of the task description (Beynon-Davis 1995; De Salabert *et al.* 1995), or the limits to what is known (Jones *et al.* 1992). Process related features include unrealistic targets (Sabherwal *et al.* 1996) or pressures to get the system working (Sabherwal *et al.* 1996). In general, these can be classified along two task related properties (Mathiassen *et al.* 1992): task complexity which represents the amount of relevant information available to carry out the task, and task uncertainty which is defined by the availability, validity, and reliability of the task related information. The higher the amount of available information about the task, or the lower its validity and reliability, the higher is the development risk.

By *structure* (S) Leavitt means systems of communication, systems of authority, and systems of work flow (Leavitt 1964). Though not making it explicit, Leavitt denotes by structure both the normative dimension, i.e. values, norms, and role expectations, and the behavioral dimension, i.e. the actual patterns of behaviors as actors communicate, exercise authority, or work. Software risks associated with structure have been discussed in relation to: task (Beath 1987), technology (Curtis *et al.* 1988), or actors (Markus *et al.* 1994). Neglect of structural dimension is likely to generate considerable risks in developing the system in time and cost (Curtis *et al.* 1988; Thambain *et al.* 1986; Nidumolu 1995; van Swede *et al.* 1994; van Genuchten 1991), in getting the requirements right (Nidumolu 1995; Beath 1987), or getting the system accepted (Davis *et al.* 1992; Markus *et al.* 1994).

Leavitt classifies structural features into three subsystems which are all relevant here: systems of communication, systems of authority, and systems of work-flow. Systems of communication specify who communicates with whom and how formal communications should be (Andersen *et al.* 1990; Davis *et al.* 1985). Risks can accordingly transpire when actors communicate ineffectively, infrequently, or are not willing to voice their concerns (Newman *et al.* 1990; Curtis *et al.* 1988; Henderson *et al.* 1988; Beynon-Davis 1995). Systems of authority define lines of responsibility between actors and thereby establish organizational power bases. These are central in reducing some risk drivers like requirements or technological uncertainty (Nidumolu 1995; Neo *et al.* 1994; Beath 1987). The lack of control structures has in some occasions lead to major disasters like

the fall of the London Exchange's Taurus system (Beynon-Davies 1995), or the Confirm system (Oz 1994). Risks can also be borne through informal interactions that unfold in networks of interrelated actors (Curtis *et al.* 1988; Andersen *et al.* 1990; Borum *et al.* 1993). Systems of work flow describe how the development processes are organized over time and space and between actors. Risks increase when the work flow is not well defined or its complex dependencies increase uncertainty and coordination costs (Humphrey *et al.* 1991).

Actors (A) represent individuals or groups of stakeholders who can set forward claims or benefit from software development. Actors include customers, managers, maintainers, developers, and users (Boehm *et al.* 1989). Examples of actor related software risks abound: personal shortcomings (Keil 1993; Boland 1992), lack of commitment and skill (Beynon-Davies 1995), differences among stakeholders (Wilkocks *et al.* 1994), wrong expectations (Ginzberg 1981), false beliefs (Hirschheim & Newman 1990), non-existent or unwilling users (Grover *et al.* 1988), unethical professional conduct (Oz 1994), personnel turnover, personal politics, and opportunism (Borum *et al.* 1993; Markus *et al.* 1994; Saarinen *et al.* 1993b; Keen 1981; Grover *et al.* 1988), or dysfunctional effect of actors' learning on project performance (Brooks 1974). Empirical studies also stress the importance of individual talent and experience in reducing task or technology related risks (Curtis *et al.* 1989; Henderson *et al.* 1992).

By *Technology* (T) Leavitt denotes "tools—problem solving inventions like work measurement, computers, and drill presses". He also goes on to point out that there is "some uncertainty about the line between structure and technology" (see also Gervin 1981). In line with the concept of "problem solving inventions" we include in technology methods, tools, and infrastructure to develop and implement the software system (Cooprider *et al.* 1991). These technologies can create considerable risks, especially if they are unreliable, inefficient, non-standardized, non-compliant, or have functional limitations (Genuchten 1991; Willkocks *et al.* 1994; Oz 1994; Sabherwal *et al.* 1996).

Considering each of the four components individually provides a valuable, first order understanding of sources of risks and their resolution. Equal important are, however, *relations* which deal with interdependencies between task, structure, technology, and actors.

Task-Actor interdependencies (Ta-A, A-Ta) focus on the actors' ability and shortcomings in relation to achieving the task, the ability to specify and analyze the task and its problems, and the inclination to make shortcuts. Typically such risks relate to the actor's experience in carrying out or specifying the development task, their disagreement about the task (see e.g. Curtis *et al.* 1989; Benyon-Davis 1985; Lyytinen 1988), or opportunism (Sabherwal *et al.* 1996).

Task-Technology interdependencies (Ta-T, T-Ta) clarify how technologies fit with the task, and how misfits can create considerable risks. Several contingency models have formulated rules to select an appropriate software development technology for a given task (Mathiassen *et al.* 1992; Saarinen *et al.* 1993a). Studies in software engineering (Curtis *et al.* 1988; Jarke *et al.* 1993; Genuchten 1991; Ciborra *et al.* 1987) stress the importance of deploying appropriate technology components, such as CASE-tools, e-mail or configuration tools to cater for task variation. Recently, some empirical studies have shown that the level of technological sophistication tends to increase the task complexity (Sørensen 1993) and thereby the development risk.

Task-Structure interdependencies (Ta-S, S-Ta) deal with how project organization can be instrumental in carrying out the development task, and how a misfit between the structure and the task can bring about risks. These concerns have lead to contingency models of the interactions between the development task and the institutional arrangements (Boehm 1988; Ciborra *et al.* 1983; Lehtinen *et al.* 1986; Parnas *et al.* 1986; Beath 1987). The conclusion from these works is that inappropriate arrangements can lead to bad or unsatisfactory outcomes (Nidumolu 1995; Saarinen *et al.* 1993b; Beath 1987).

Actor-Technology interdependencies (A-T, T-A) address risks which are created by improper matching of people with technology, or by introducing untried technologies. Problems of matching inexperienced users with unknown and complex technologies are well documented in the empirical literature on software risks (Barki *et al.* 1993; Keil 1995; Smolander *et al.* 1990; Kendall *et al.* 1992). Technologies may also lower professional esteem, motivation, and change career paths (Orlikowski 1993). A recent empirical study discovered that specific technological choices like machine architecture correlate with some actor related risks such as gold plating (Ropponen *et al.* 1996).

Actor-Structure interdependencies (A-S, S-A) focus on interactions between the structure and the actors. Typical concerns are: incentive schemes and sanctions, values and beliefs, and how actors' behaviors and expectations are in concordance with the prevailing organizational structure. This area has been a traditional focus of socio-technical design that seeks to develop jobs that are varied and demand multiple skills, and thereby meet human values (Mumford 1983). These concerns have been raised also in several systems design textbooks though no empirical studies are available (Lundeberg *et al.* 1981; Andersen *et al.* 1990; DeMarco *et al.* 1987; Henderson & Lee 1992). Recent studies have also focused on the interactions between stakeholder participation and process structures—in particular the system of workflow (Sabberwal *et al.* 1995), or on the problems related to actors being located in different places (Sabherwal *et al.* 1996).

Technology-Structure interdependencies (T-S, S-T) deal with interactions between the organizational structure and technologies. The logic here is that the structure is affected by the technology (or vice versa): an inappropriate structure, given the technology, or inappropriate technology, given the structure, will create considerable disturbances (Caufield 1989; Schmidt 1992). Typical concerns are: how work flows (Curtis *et al.* 1992), how systems of authority and communication lines are affected by technological choices, and what risks these choices create. Technologies can be in conflict with the systems of authority (Markus 1983; Markus *et al.* 1983), or they can prevent creation of lines of communication (Mumford 1983). The use of a highly formalized specification technique assumes, for example, formal sign-on procedures and this can undermine effective communication with users (Bjørn-Andersen *et al.* 1993).

3.3. Summary

Software risk items and their resolution techniques can be organized using Leavitt's model. Our analysis demonstrates that all Leavitt's components—the task, structure, technology, and actors—and their combinations have been recognized widely as risk items and also suggested as targets for risk resolutions in past research. Appendix 1 synthesizes what risk items each Leavittian component can include, and what resolution techniques have been suggested to resolve them. These were drawn from the published literature on IS failures and software risk management.

4. A categorical analysis of four approaches

We apply Leavitt's model in interpreting how risk management approaches shape managers' attention as follows: we classify lists of risk items as suggested by risk management approaches in terms of how they fall into different Leavittian components. In a similar vein, we characterize repertoires of risk resolution techniques in terms of how they cover Leavitt's components. We also analyze how the approaches translate these components into managerial action by the use of heuristics which are interpreted to embody knowledge of cause-effect chains assumed in the observed socio-technical system. By comparing these codings we find out to what extent the espoused attention shaping patterns and intervention heuristics are similar, to what extent they are different, and how the approaches link managerial perceptions of development situations with managerial deliberations. In doing so we disclose how risk management approaches theorize about changes in Leavittian elements and how these affect software risk. This analysis also shed light on differences in the attention foci of risk management approaches and question why approaches exclude some aspects of the situation from managerial inquiry at the cost of putting others in the foreground. These exclusions reflect theoretical assumptions about valid cause-effect chains in development situations. Yet, the choices are ideological in the sense that they reflect specific social interests, and serve as myths, as in most cases they are accepted without further justification or empirical validation (Berger *et al.* 1967; Hirschheim *et al.* 1991).

4.1. Research method

Categorical[4] analysis attempts to make valid inferences from studied texts to their underlying meaning in terms of pre-specified set of categories (Weber 1985). In our situation, this analysis technique helps clarify the content of theories of software development situations underlying software risk management approaches. In particular, categorical analysis reveals the following aspects in each approach (for a similar approach see Beath *et al.* 1994):

1) its *definition of risk* solicits the risk concept as explicitly defined in the approach.
2) the *risk management rationale* clarifies which justifications are provided for the use of the approach.

3) the *underlying risk metric* clarifies whether risk is defined by using single or multiple constructs together with the applied measurement scale.
4) the *risk behavior* clarifies whether the approach assumes a rational or a loss-aversive stance.
5) the *number of and coverage of risk items* clarifies the number of risk items recognized in the approach. This can be interpreted as a coarse indicator of how fine grained the approach is in distinguishing alternative sources as risky incidents.
6) the *focus of risk items* reveals the focus of the approach in terms of how different Leavittian components are covered. By counting the number of items per component and their relative proportions we can determine how focused or balanced the risk identification patterns are.
7) the *number and coverage of risk resolution techniques* can be used as a coarse indicator of how fine grained and rich the approach is in suggesting interventions. Multiple usage indicates as opposed to single usage that the same risk resolution technique can be used to resolve problems associated with several risk items.
8) the *focus of risk resolution techniques* reveals how different Leavittian components are covered by risk resolution techniques. By counting the number of risk resolution techniques per component and analyzing their relative proportion we can judge how focused or balanced the risk resolution patterns are.
9) the *heuristics* link risk items with specific risk resolution techniques. This addresses how risk items and risk resolution techniques are related and the structure and the level of detail of the rules by which this is done.
10) its *scope* analyzes when the approach should be applied and under what circumstances it should be applied. Continuous approaches are used throughout the development process, while discrete approaches only are applicable in specific phases or contexts.

We followed these steps in our analysis of the four classical risk management approaches (Alter *et al.* 1978; Boehm 1989, 1991;

Davis 1982; McFarlan 1982). The motivation for choosing these four classical risk management approaches is their widespread use and typical coverage of risk management issues. They are seen by many as the classics in the field (see Lucas 1981; Saarinen *et al.* 1993a). Each approach and the mapping of its risk identification and risk resolution technique into Leavitt's categories are explained in appendices 2 through 5. How the coding and analysis was carried out is represented in appendix 6 which also evaluates the reliability and validity of the coding.

4.2. How are the four approaches different?

The results of the analysis are summarized in table 1. The table reveals several important differences in the four risk management approaches.

The Concept of Risk. We can observe prominent differences between the approaches. McFarlan focuses on risks related to obtaining some or all the goals that are relevant in setting up the project. Boehm's concept is more restricted and focuses on avoiding losses for some or all stakeholders. Alter & Ginzberg also discuss failure, but only the failure to implement the system organizationally. Davis' view (due to its focus on requirements specification) deals exclusively with the difficulty of obtaining a correct and complete understanding of the task. These differences in the notion of software risk explain diversity in the way risks are treated in these approaches. There is a clear difference between, on the one hand, Boehm's and Alter & Ginzberg's approaches, and, on the other, the approaches of Davis and McFarlan. The former focus entirely on possible losses, but provide no means to measure the magnitude of loss. As a consequence these approaches cannot be used to compare risk levels of different projects. The latter set up a contingency model in which they match situational risk items to a repertoire of risk resolution techniques. In doing so, they abstract situational risk items into a simple rank order metric (instead of the nominal scale metric used with Boehm and Alter & Ginzberg) and thereby compare different projects. Moreover, they (though somewhat implicitly) look at the combination of costs and benefits, i.e. they are more in line with the rational decision ideal.

Risk Item Focus. We notice major differences in how the approaches mold the attention of software risk managers. Whilst Boehm is balanced in that all four Leaviattian components perceive

equal weight, the three other approaches are more concentrated in their focus. Davis concentrate on actor and task related risks; McFarlan's approach sees risk items to be primarily task related; and Alter & Ginzberg focus on actor related risk items. It is also worth noticing that the two contingency approaches, i.e. Davis and McFarlan (for operational reasons) enlist quite few, high level risk items, whereas the two other approaches enlist several items on varying levels of detail and thus offer a more fine grained vocabulary and classification scheme.

Risk Resolution Focus. The four approaches propose different parts of the environments as subjected to management control. Accordingly, they embody distinct and biased causal theories of how software development processes evolve. Boehm's focus is on traditional software engineering solutions and he exhibits a clear attachment to technological fixes. His approach tells software managers that software risks are better managed by applying new or complementary technologies for software development and by disciplining the process. Davis' focus and bias are different. He assumes that the task and actor elements are fixed and that they establish the primary source of risk during requirements specification. Therefore, risk resolution tactics must seek to configure the best possible combination of technology and structure to match with the observed risk level. Davis propose that these elements are under the control of the project manager, and he or she can shape them as suggested in the approach. Alter & Ginzberg are primarily concerned with implementation risks. These are perceived as products of actor's attitudes and beliefs, and they are combined with a detailed and balanced set of risk resolution techniques that cover all Leavitt's components. Finally, McFarlan emphasizes risk items associated with inappropriate task specification and the risk resolution techniques aim at submitting the software development organization under better managerial control. McFarlan thus advocates a managerial fix. He argues that any organizational structure in the software development is malleable and can be reshaped by swift management action.

It is worth noting that there is great variation in the number of resolution techniques offered. Alter & Ginzberg's and Boehm's approaches are more elaborate, but the variation and content of the techniques are less systematic, and they use the same risk resolution technique to resolve different risk items. This reflects the as-

sumption that some techniques are generic and not necessarily specific to any of the risk items. Both Davis and McFarlan propose specific techniques which are applicable only to one risk item. Interestingly neither of these contingency approaches offer resolution techniques to modify the task. This can be a symptom of lack of systematic analysis. But it can also be seen as a different perception of the role of the software manager where the task is given and the challenge is to change the rest of the socio-technical system to fit this need.

Heuristics. In general, suggested heuristics are simple and they are not thoroughly articulated. They allow for creative adaptation and managerial improvisation to meet the demands of varying situations. Two very different alternative structural forms are suggested. Boehm and Alter & Ginzberg compile a list of risk resolution techniques for each observed risk item (like a menu list). No rationale is provided why these techniques come handy for a given risk item. In a way they form an ideological myth which has to be accepted based on the face value (Trice *et al.* 1984). In addition, no technical rules are given of how to compose a set of risk resolution measures given the observed risk item list. In contrast, the contingency models of Davis and McFarlan offer a relatively systematic approach with selection rules to compose a comprehensive risk resolution strategy. These approaches, too, are superfluous in providing theoretical reasoning behind the suggested rules.

Scope. This feature deals with the applicability of the risk management approach across the development trajectory. Discrete approaches deal with situations at specific points of the software development trajectory. Continuous approaches cover larger segments of the trajectory while sometimes adopting a very specific point of view. Sometimes these approaches cover the whole span of the development trajectory. Again, we can observe differences between the approaches. Davis' and McFarlan's approaches are discrete. McFarlan's approach is used by functional and IT managers during the project initiation phase. This specific context explain its focus on structural aspects which is the viewpoint advocated by management. Davis addresses the task of avoiding incomplete and wrong requirements and sees the technology and structure as controllable targets at specific phases and contexts. In contrast, Boehm's and Alter & Ginzberg's approaches are continuous. Boehm's approach spans most parts of software development start-

Type of Analysis	Alter & Ginzberg	Boehm	Davis	McFarlan
Risk definition	"Uncertainty as to whether the project or parts of it can be completed at all" (p. 23)	"Risk exposure is the probability of an unsatisfactory outcome times the loss to the parties if the outcome is satisfactory . . . Unsatisfactory outcome is multidimensional: . . . budget overruns, wrong functionality, user interface shortfalls, . . . poor quality software."	"Difficulty in arriving at correct and complete requirements " (p. 5)	"Exposure to . . . Failure to obtain all, or any of the benefits . . . Costs of implementation that vastly exceed planned levels . . . Time for implementation that is much greater than expected . . . Technical performance significantly bellow estimate . . . Incompatibility of system with hardware and software" (p. 13)
Risk metric	Multidimensional construct, nominal scale	Multidimensional construct, nominal scale	Single construct, rank order scale	Single construct, rank order scale
Risk behavior	Loss-aversive	Loss-aversive	Rational	Rational
Risk management rationale	Avoid failure in organizational implementation	Avoid cost overrun, delays, rework, overkill, poor quality of software	Increase likelihood of obtaining useful information requirements	Maximize fit between project management approach and level of risk
Risk Items	# 8	# 10	# 3	# 3
Coverage of Risk items				
Actor	6.5[1] (1)	2 (2)	2	0.5 (1)
Task	0	3.5 (1)	1	2
Technology	1	2.5 (1)	0	0.5 (1)
Structure	0.5 (1)	2	0	0
Risk Item Focus	Actor	Balanced	Actor and Task	Task
Risk Resolution Techniques	# 16 with multiple usage	# 36 with multiple usage	# 4 with single usage	# 29 with single usage

Table 1. Categorical Analysis of Four Risk Management Models.

ing from requirements specification and ending up with configuring and implementing the software system. The intention is to provide new tools for project managers to complement their traditional

Type of Analysis	Alter & Ginzberg	Boehm	Davis	McFarlan
Coverage of Risk Resolution techniques				
Actor	5 / 15	4 / 4.5 (2)	0	2 (2)
Task	2 / 3	2 / 4 (0)	0	0
Technology	3 / 8	21 / 30 (0)	2 (4)	7
Structure	6 / 17	9 / 10.5 (2)	2 (4)	20 (2)
Risk Resolution Focus	Balanced	Technology and structure	Technology and structure	Structure and technology
Heuristics	• A set of resolution techniques proposed for each risk item • No rules for the composition of resolution techniques • The same resolution technique applies to different risk items	• A set of resolution techniques proposed for each risk item • No rules for the composition of resolution techniques • The same resolution technique applies to different risk items	• Four strategies for requirements determination • Selection of strategy based on the risk level • Each strategy composed of risk resolution techniques	• Eight strategies for project design • Selection of strategy based on risk items • Each strategy involves a number of risk resolution techniques • No rules for composing resolution techniques
Scope	Continuous	Continuous	Discrete	Discrete

Legend for the table
= number of distinct risk items and resolution techniques identified in each approach
x / y = number of distinct entries / number of all entries
(z) = number of relation entries i.e. entries related to two Leavitt variables

Table 1. Categorical Analysis of Four Risk Management Models (continued).

method and tool kits. Alter & Ginzberg cover the whole development trajectory, including organizational implementation of the system. Their scope is narrow, the approach is meant to be used by all actors who have a stake in organizational acceptance of the system, and it complements in this way Boehm's approach.

5. Discussion

From the behavioral perspective, risk management approaches can be interpreted as articulations of micro theories of development situations that seek to improve managerial behaviors in situations with threats of a major loss. These theories—in line with studies in the sociology of knowledge—embed ideological bias in selecting con-

cepts and conjectures. The exclusions are products of assumptions dealing with cause-effect chains within development situations. These choices are ideological in the sense that they reflect specific social interests and are, in most cases, given without further justification or empirical validation (Berger *et al.* 1967). The analysis of the four classical approaches supports this conclusion.

Our study addresses weaknesses in the current risk management literature: its bleak understanding of how risk management approaches relate to one another and of what types of risks software developers can and should address *in toto.* Risk management approaches differ in terms of suggested risk items and risk resolution strategies. Some of them deal primarily with actor risks (Alter & Ginzberg), task risks (McFarlan), interaction risks (Davis), or a more balanced set of risks (Boehm). None of the examined approaches concentrate on technology risks or structure related risk. We conjecture that none of these approaches alone is capable of addressing the majority of pitfalls during software development. None of them generates enough requisite variety (Ashby 1956) to understand the environments and distill relevant risk related information from it.

The analysis also reveals that the approaches adopt alternative mechanisms (or gestalts) to organize the risk management approach. Some of them—like Boehm's or Alter & Ginzberg's model—offer fairly detailed and fine grained risk items and risk resolution techniques—whereas Davis' and McFarlan's model offer few risk items and relatively few risk resolution techniques, which, however, consist of several items. The latter two also suggest rank order metrics for measuring risk whereas the former only suggest simple characterizations of risk exposure.

The categorical analysis reveals that the use context and dynamics are distinct in these approaches. Therefore, they are not exclusive alternatives. Instead, they can be effectively combined to increase the variety generated by the control system as Ashby suggests. For example, Alter & Ginzberg's approach adds a generic risk concern (actor related risks) which is largely missing in Boehm's approach. Therefore, these could be combined as to yield two complementary viewpoints in a more comprehensive risk management approach using Leavitt's model as an organizing framework. In the same manner, Davis' approach and McFarlan's approach can be combined with Boehm's approach to cater for specific contexts in the

risk driven spiral model (Boehm 1988). McFarlan's approach can be used before embarking on the spiral while thinking of the project set-up and instituting the organizational environment. Clearly, in such an environment task and structural issues dominate. Davis's approach can be used to plan and monitor the risks associated with the first cycle in the spiral for which Boehm does not offer any risk driven plan. Overall, based on Leavitt's model it is feasible to organize a contingent risk planning approach in which the development cycles are organized as rows and Leavittian components as columns. By using values such as low, moderate, and high for each entry in the table we can build up risk profiles for each development cycle and select the risk management approach accordingly.

The applied technique for analyzing risk management approaches has several limitations. First, Leavittian theory adds an additional layer of theoretical complexity to an area which does not necessarily need one. This problem can be overcome by crafting practical approaches using other terms and vocabularies closer to the every day experience of software managers. The second limitation concerns the validity and reliability of our coding. Though we achieved very high levels of reliability in codings, and did not find the classification scheme to be too limited, the technique must be tested with other approaches and with different analysts to achieve higher levels of confidence. We have taught our framework to several students and asked them to analyze risk management approaches. They have done so without any greater difficulty which suggests that the categorical approach is easy to learn and to use.

The paper has several implications for research and practice. Our analysis suggests that risk management needs to be examined as an instance of organizational uncertainty absorption and management. This aspect has only been touched upon here through concepts of attention shaping and heuristics. Ultimately, we are dealing with problems of designing complex social systems where we can only achieve satisficing (Simon 1979, 1983; Galbraith 1977) solutions. Therefore, one fruitful avenue is to explore in more depth the process of risk management as satisficing behavior (Lyytinen *et al.* 1996). Another question is how organizations learn from their risky incidents and how these are inductively generalized into causal theories of development situations. Again, Leavitt's model can serve as a valuable asset in analyzing how local knowledge is structured, organized, and validated. Our conclusion from probing the four clas-

sical approaches is that their coverage is not complete and their items are not systematically inferred. Therefore, one research task ahead is to develop a more encompassing and comprehensive risk item and risk resolution taxonomy. One step towards this end is presented in appendix 1. Moreover, one author has been engaged in developing a systematic risk management approach based on the principles outlined in this paper which has been adopted as part of the EuroMethod—a standard developed by and for the European Commission for contracted software development (Euromethod 1996).

For empirical studies we identify three major challenges. First, the Leavittian framework can be used to conduct a meta-analysis of how software initiatives have succeeded or failed with varying constellations of task, actor, structure, and technology risks. Second, we can use the framework to carry out ethnographic investigations on software risk management practices. In these studies we should describe and analyze, in more detail, how contextual features such as personal anxiety, organizational protocols, and incentives affect the scope and direction of risk management, and how they interact with available written protocols for risk management. Third, we can examine the structure of employed risk resolution sets and understand how software managers seek to address observed risks at various stages of software development.

The paper conveys a useful message to practice. It reveals that practitioners should be cautious with regard to the expected miracles of risk management. Risk management approaches are not complete nor risk-proof and they are not the silver bullet that will cure the pitfalls of software development. Their value is that they put high premium on learning, understanding development situations, and on the sane principles of management: be open, fear the worst, and honor the complexity of social and technical change.

Acknowledgments

This paper has largely benefited from Roy Schmidt's constructive criticisms, and literature suggestions. Thanks go also to the associate editor and two reviewers for their detailed and insightful suggestions of how to improve the paper.

Notes

1. We understand software development quite broadly to cover require-

ments analysis, design, implementation, and organizational adoption of software systems. This definition is broader than usually assumed in software engineering and associated discussions of risks (Boehm 1991; Charette 1989), but it coincides quite well with the notion of information systems development as expressed in standard textbooks (Davis & Olson 1985), or literature surveys on IS development (Lyytinen 1987; Iivari 1991).

2. We acknowledge many of the criticisms proposed against Leavitt's model. These include its focus on the static structure, ignorance of the environment, applicability to temporary organizations, and the nature of the distinctions it suggests for analyzing organizations. Many of these are valid if one wants to develop a valid theory of temporary organizations or organizational processes operating in uncertain and complex environments. But this is not our goal in the paper: we want to use Leavitt's model as a classification framework in developing interpretations of how different software risk approaches refine their organizational interventions. In such an endeavor, many criticisms raised against Leavitt's model are not valid. An alternative would have been to analyze risk management behaviors through some dynamic social process models like agency theory or transaction costs. By doing so we would, however, have lost the simplicity of Leavitt's model in analyzing the **content** of attention shaping patterns. Typically, dynamic models either focus on manager's psychological behaviors like in escalation theory (Staw & Fox 1974; Staw & Ross 1987; Keil & Mixon 1993), or on processes as they unfold and maintain or disrupt the "equilibrium" (Gersick 1995; Mohr 1982; Newman & Robey 1993). Such models do not tell us **what** managers perceive or **how** they enact their perceptions during software development processes. Such descriptions of how processes change are complementary to accounts of changes in attention focus over time.
3. Such extensions have been made. For example, Kwon & Zmud (1987) augment the model with the concept of environment and environmental factors. Davis & Olson (1985) add to the model the concept of organizational culture. We shall ignore such extensions in this paper as no environmental or cultural aspects were identified in the risk management approaches.
4. Normally this method is called content analysis. But we use the term categorical analysis as suggested by one of the reviewers as it better conveys the idea underlying the research technique.

5. Use of the value .5 is due to “relational” risk items and risk resolution techniques which look at the interactions and dependencies between components. See appendix 6 for a more detailed explanation.

References

Adler, S. (1980): Risk-Making Management. *Business Horizons,* Vol. 23, No. 2 (11–14).

Alter, S. & M. Ginzberg (1978): Managing Uncertainty in MIS Implementation. *Sloan Management Review*, Fall.

Andersen, N. E., F. Kensing, M. Lassen, J. Lundin, L. Mathiassen, A. Munk-Madsen & P. Sørgaard (1990): *Professional Systems Development: Experience, Ideas, and Action*. Englewood Cliffs, New Jersey: Prentice-Hall.

Arrow, K. J. (1965): *Aspects of the Theory of Risk Bearing.* Helsinki: Yrjö Janssonin Säätiö.

Ashby, W. R. (1956): *An Introduction to Cybernetics*. London.

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

Beath, C. M. (1983): Strategies for Managing MIS Projects: A Transaction Cost Approach. (133–147) in *Proceedings of the 4th International Conference on Information Systems*. Houston, Texas.

Beath, C. M. (1987): Managing the User Relationship in Information Systems Development Projects: A Transaction Governance Approach. (415–427) in *Proceedings of the 8th International Conference on Information Systems*. Pittsburgh, Pennsylvania.

Beath, C. M. & W. Orlikowski (1994): The Contradictory Structure of Systems Development Methodologies: Deconstructing the IS-User Relationship in Information Engineering. *Information Systems Research,* Vol. 5, No. 4 (350–377).

Bell, D. (1985): Disappointment in Decision Making. *Operations Research,* Vol. 33 (1–27).

Berger, P. & T. Luckmann (1967): *The Social Construction of Reality—A Treatise in the Sociology of Knowledge.* London: Penguin Books.

Beynon-Davis, P. (1995): Information Systems Failure: the case of LASCAD project. *European Journal of Information Systems*, Vol. 4, No. 3 (171–184).

Bjørn-Andersen, N. & L. Markus (1993): Power over Users: Its Exercise by Systems Professionals. *Communications of the ACM,* Vol. 30, No. 6.

Blokdijk, A. & P. Blokdijk (1987): *Planning and Design of Information Systems.* Academic Press.

Boehm, B. W. (1988): A Spiral Model of Software Development and Enhancement. *Computer,* May (61–71).

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

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

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).

Boland, R. (1992): *In Search for Management Information Systems: Explorations of Self and Organization.* Unpublished working paper, Weatherhead School of Management, Case Western Reserve University.

Borum, F. & J. Christiansen (1993): Actors and Structure in IS Projects: What Makes Implementation Happen. *Scandinavian Journal of Management Studies.*

Bromiley, P. & S. Curley (1992): Individual Differences in Risk Taking. (87–132) in J. F. Yates (Ed.): *Risk Taking Behavior*. Chichester: Wiley.

Brooks, F. (1974): *The Mythical Man-Month*. Englewood-Cliffs, New Jersey: Prentice-Hall.

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

Caufield, C. (1989): *An Integrative Research Review of the Relationship between the Technology and the Structure—A Meta-analytic Synthesis.* Ph.D. thesis, University of Iowa.

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

Ciborra, C. & G. Bracchi (1983): Systems Development and Auditing in Turbulent Contexts: Towards a New Participative Approach. (41–52) in E. M. Wysong *et al.* (Eds.): *Information Systems Auditing.* Amsterdam: North Holland.

Ciborra, C. & G. F. Lanzara G (1987): Formative Contexts of Systems Design. (27–52) in H. Klein *et al.* (Eds.): *Information Systems Development for Human Progress.* Amsterdam: North-Holland.

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).

Curtis, B., M. Kellner & J. Over (1992): Process Modeling. *Communications of the ACM,* Vol. 35, No. 9 (75–90).

Cyert, R. & J. March (1963): *A Behavioral Theory of the Firm*. Englewood Cliffs, New Jersey: Prentice-Hall.

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

Davis, G. B. & M. H. Olson (1985): *Management Information Systems—Conceptual Foundations, Structure, and Development*. McGraw-Hill.

Davis, G. B., A. S. Lee, K. R. Nickles, S. Chatterjee, R. Hartung & Y. Wu (1992): Diagnosis of an Information System Failure: A Framework and Interpretive Process. *Information & Management,* Vol. 23, No. 2 (293–318).

DeMarco, T. & T. Lister (1987): *Peopleware—Principles of Project Management*. Englewood-Cliffs, New Jersey: Prentice-Hall.

DeSalabert, A. & M. Newman (1995): Regaining Control: The case of Spanish Air Traffic Control System-SACTA. (1171–1180) in G. Doukidis *et al.* (Eds.): *Proceedings of the 3rd European Conference on Information Systems.*.

Earl, M. (1987): *Information Management Strategy*. Englewood-Cliffs, New Jersey: Prentice-Hall.

EuroMethod (1996): *Euromethod Version 1*. European Commission. Information available at http://www.fast.de/Euromethod/.

Fairley, R. (1994): Risk Management for Software Projects. *IEEE Software,* May (57–67).

Fischoff, B., S. Lichtenstein, P. Slovic, S. Derby & R. Keeney (1981): *Acceptable Risk*. New York: Cambridge University Press.

Fischoff, B., S. Watson & C. Hope (1984): Defining Risk. *Policy Sciences,* Vol. 17 (123–139).

Galbraith, J. (1977): *Organization Design*. Reading, Massachusetts: Addison-Wesley.

Genuchten, M. van (1991): Why is Software Late? An Empirical Study of Reasons for Delay in Software Development. *IEEE Transactions on Software Engineering,* Vol. 17, No. 6 (582–590).

Gersick, C. (1991): Revolutionary Change Theories: A Multilevel Exploration of the Punctuated Equilibrium Paradigm. *Academy of Management Review*, Vol. 16, No. 1 (10–36).

Gervin, D. (1981): Relationhips between Structure and Technology. (3–38) in P. Nystrom *et al.* (Eds.): *Handbook of Organizational Design*. London: Oxford University Press.

Ginzberg, M. (1981): Early Diagnosis of MIS Implementation Failure: Promising Results and Unanswered Questions. *Management Science,* Vol. 27, No. 4 (459–478).

Grover, V., A. L. Lederer & R. Sabherwal (1988): Recognizing the Politics of MIS. *Information & Management*, Vol. 14 (145–156).

Henderson, J. & S. Lee (1992): Managing I/S Design Teams: a Control Theories Perspective. *Management Science,* Vol. 38, No. 6 (757–777).

Hirschheim, R. & M. Newman (1991): Symbolism and Information Systems Development: Myth, Metaphor, Magic. *Information Systems Research,* Vol. 2, No. 1 (29–62).

Humphrey, W. (1989): *Managing the Software Process*. Reading, Massachusetts: Addison-Wesley.

Humphrey, W., T. Snyder & R. Willis (1991): Software Process Improvement at Hughes Aircraft. *IEEE Software,* Vol. 11, No. 7 (11–23).

Iivari, J. (1991): A Paradigmatic Analysis of Contemporary Schools of IS Development. *European Journal of Information Systems,* Vol. 1, No. 4 (249–272).

Jarke, M. & K. Pohl (1992): Vision Driven System Engineering. In N. Prakash *et al.* (Eds.): *Information Systems Development Process*. Amsterdam: North Holland (3–20).

Jones, M. & G. Walsham (1992): The Limits of Knowable: Organizational and Design Knowledge in Systems development. (195–212) in K. Kendall *et al.* (Eds.): *The Impact of Computer Supported Technologies on Information Systems Development.* Amsterdam: North-Holland.

Kahnemann, D. & A. Tversky (1982): Variants of Uncertainty. *Cognition,* Vol. 11 (143–157).

Keen, P. G. W. (1981): Information Systems and Organizational Change. *Communications of the ACM,* Vol. 24, No. 1 (24–33).

Keen, P. G. W. & S. Scott-Morton (1978): *Decision-Support Systems: An Organizational Perspective*. Reading, Massachusetts: Addison-Wesley.

Keil, M. (1995): Pulling the Plug: Software Project Management and the Problem of Project Escalation. *MIS Quarterly,* Vol. 19, No. 4 (421–448).

Keil, M. & R. Mixon (1994): Understanding Runaway IT projects: Preliminary Results from a Program of Research based on Escalation Theory. In *Proceedings of 27th HICSS*. IEEE Computer Society Press.

Kendall, K., K. Lyytinen & J. DeGross (Eds.): *The Impact of Computer Supported Technologies on Information Systems Development*. Amsterdam: North-Holland.

Kwon, T. H. & R. Zmud (1987): Unifying the Fragmented Models of Information Systems Implementation. (227–251) in R. Boland *et al.* (Eds.): *Critical Issues in Information Systems Research*. Chichester: Wiley.

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

Lehtinen, E. & K. Lyytinen (1986): Seven Mortal Sins of Systems Work. (63–79) in P. Docherty *et al.* (Eds.): *Information Systems for Organizational Productivity—Participation and Beyond*. Amsterdam: North-Holland.

Lucas, H. (1981): *Implementation—The Key to Successful Information Systems*. New York: Columbia University Press.

Lucas, H. (1982): Alternative Structures for the Management of Information Processing. In Goldberg *et al.* (Eds.): *On the Economics of Information Processing*, Vol. 2. New York: Wiley.

Lundeberg, M., G. Goldkuhl & A. Nilsson (1981): *Information Systems Development: A Systematic Approach*. Englewood Cliffs, New Jersey: Prentice-Hall.

Lyytinen, K. (1987): Different Perspectives on Information Systems: Problems and Solutions. *ACM Computing Surveys*, Vol. 19, No. 1 (5–44).

Lyytinen, K. (1988): Expectation Failure Concept and System's Analysts' View of Information System Failures: Results of an Exploratory Study. *Information & Management*, Vol. 14 (45–56).

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

Lyytinen, K., L. Mathiassen & J. Ropponen (1995): A Framework for Software Risk Management. *Journal of Information Technology*, Vol. 11, No. 4.

MacCrimmon, K. & D. Wehrung (1986): *Taking Risks: The Management of Uncertainty*. New York: Free Press.

March, J. (1988): Variable Risk Preferences and Adaptive Aspirations. *Journal of Economic Behavior and Organizations*.

March, J. & Z. Shapira (1987): Managerial Perspectives on Risk and Risk-Taking. *Management Science*, Vol. 33.

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

Markus L. (1983): Power, Politics and MIS implementation. *Communications of the ACM,* Vol. 26 (430–444).

Markus, L. & D. Robey (1983): The Organizational Validity of Management Information Systems. *Human Relations,* Vol. 36, No. 3 (203–226).

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

Mathiassen, L. & J. Stage (1992): The Principle of Limited Reduction in Software Design. *Information, Technology, and People,* Vol. 6, No. 2.

McFarlan, W. (1982): Portfolio Approach to Information Systems. *Journal of Systems Management,* January (12–19).

Mohr, L. B. (1982): *Explaining Organizational Behavior*. San Fransico: Jossey-Bass.

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

Neo, B. S. & K. Leong (1994): Managing Risks in Information Technology Projects: A Case Study of Tradenet. *Journal of Information Technology Management,* Vol. 15, No. 3 (20–45).

Newman, M. & F. Noble (1990): User Interaction as an Interaction Process: A Case Study. *Information Systems Research,* Vol. 1, No. 1 (89–113).

Newman, M. & D. Robey (1992): A Social Process Model of User-Analyst Relationships. *MIS Quarterly,* Vol. 16, No. 2 (249–265).

Nidumolu, S. (1994): The Effect of Coordination and Uncertainty on Software Project Performance: Residual Performance Risk as an Intervening Variable. *Information Systems Research,* Vol. 36, No. 3 (191–219).

Orlikowski,W. (1993): CASE Tools as Organizational Change: Investigating Incremental and Radical Changes in Systems Development. *MIS Quarterly,* Vol. 17, No. 3 (309–340).

Oz, E. (1994): When Professional Standards are Lax—the Confirm Failure and its Lessons. *Communications of the ACM,* Vol. 37, No. 10 (29–36).

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

Perrow, C. (1984): *Normal Accidents: Living with High-Risk Technologies.* New York: Basic Books.

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

Ropponen, J. & K. Lyytinen K (1996): How Software Risk Management can Improve Systems Development —An Explorative Survey. Accepted for publication. *European Journal of Information Systems.*

Saarinen, T. & A. Vepsäläinen (1993a): Managing Risk of System Implementation. (91–117) in T. Saarinen: *Success of Information Systems—Evaluation of Development Projects and the Choice of Procurement and Implementation Strategies*. Ph.D. thesis, Helsinki School of Economics and Business Administration.

Saarinen, T. & A. Vepsäläinen (1993b): Procurement Strategies for Information Systems. (118–141) in T. Saarinen: *Success of Information Systems—Evaluation of Development Projects and the Choice of Procurement and Implementation Strategies*. Ph.D. thesis, Helsinki School of Economics and Business Administration.

Sabberwal, R. & D. Robey (1995): Reconciling Variance and Process Strategies for Studying Information Systems Development. *Information Systems Research,* Vol. 6, No. 4 (303–323).

Sabherwal, R & J. Elam (1996): Overcoming Problems in Information Systems Development by building and sustaining Commitment. *Accounting, Management ,and Information Technologies,* Vol. 5, No. 3/4 (283–309).

Schmidt, R. (1992): *Leadership Patterns and Information Technology Usage Patterns in Top Management Teams*. Ph.D. thesis, Indiana University.

Seely Brown, J. & P. Duguid (1991): Organizational Learning and Communities of Practice: Toward a Unified View of Working, Learning, and Innovation. *Organization Science.* Vol. 2, No. 1 (40-57).

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. *Behavioral Economics and Business Organization.* Cambridge, Massachusetts: The MIT Press (160–176).

Smolander, K., V.-P. Tahvanainen & K. Lyytinen (1990): How to Combine Tools and Methods in Practice—A Field Study. (195–214) in B. Steinholtz *et al.* (Eds.): *Proceedings of CAiSE'90*. Berlin: Springer-Verlag.

Swanson, B. (1984): Organizational Designs for Software Maintenance. (73–81) in *Proceedings of the 5th International Conference on Information Systems*. Tucson.

Swanson, B. & C. M. Beath (1990): *Software Maintenance.* Chichester: Wiley.

Starbuck, W. (1983): Organizations as Action Generators. *American Sociological Review,* Vol. 48, February (91–102).

Staw, B. & F. Fox (1977): Escalation: The Determinants of Commitment to a Chosen Course of Action. *Human Relations,* Vol. 30, No. 5 (431–450).

Staw, B. & J. Ross (1987): Behavior in Escalation Situations: Antecedents, Prototypes and Solutions. *Research in Organizational Behavior,* Vol. 9 (39–78).

Swede, D. van & H. van Vliet (1994): Consistent Development: Results of a First Empirical Study in the relation between Project Scenario and Success. (80–93) in G. Wijers *et al.* (Eds.): *Advanced Information Systems Engineering*. Lecture Notes in Computer Science 811. Berlin: Springer-Verlag.

Sørensen, C. (1993): *Adoption of CASE tools—an Empirical Investigation*. Ph.D. thesis, Department of Mathematics and Computer Science, University of Aalborg.

Thambain, H. J. & D. Wilemon (1986): Criteria for Controlling Projects according to Plan. *Project Management Journal,* (75–81).

Trice, H. & J. Beyer (1984): Studying Organizational Cultures through Rites and Ceremonials. *Academy of Management Review,* Vol. 9, No. 4 (653–669).

Turner, J. (1992): A Comparison of the Process of Knowledge Elicitation with that of Requirements Determination. (415–430) in W.W. Cotterman *et al.* (Eds.): *Challenges and Strategies for Research in Systems Development*. New York: Wiley.

Waters, R. (1993): The Plan that Fell Earth. *Financial Times*, 12, March 1993.

Weber, R. (1985): *Basic Content Analysis*. Newbury Park, California: Sage Publications.

Willcocks, L. & H. Margetts (1994): Risk and Information Systems: developing the analysis. (208–230) in L. Willcocks (Ed.): *Information Management: The evaluation of information systems investments.* Chapman-Hall.

Williamson, O. (1975): *Markets and Hierarchies: Analysis and antitrust Implications*. New York: Free Press.

Yates, J. F. (Ed) (1992): *Risk Taking Behavior*. Chichester: Wiley.

Zmud, R. (1980): Management of Large Software Development Efforts. *MIS Quarterly,* Vol. 4, No. 2 (45–55).

Appendix 1: A Leavittian model of Risk Management

Leavitt's component	Risks Item	Risks Resolution Technique
Task	Task complexity • project size • number of parties Task uncertainty • ambiguity • task specificity • wrong functions • continuous change • existence of requirements	Reduce complexity • divide tasks • requirements scrubbing Reduce uncertainty • keep system simple • reduce the scope • use scenarios • use pilots to demonstrate system value • test the system Manage process • carefully plan and manage milestones and new releases
Structure	Systems of communication • inefficient • poor • lack of channels Systems of authority • inappropriate structure • poorly defined responsibilities • inappropriate rewards • inefficient governance structure Systems of work flow • unrealistic schedules • inappropriate work flow and coordination • poor physical arrangements	Improve communications • user participation • user surveys • team meetings • user lead teams • publicize participation results • monitor progress and promote open discussion • focus on critical task related topics Reorganize • project organization • external contracts and outsourcing • user committees and good relationships • formal procedures • user managed decisions and development • cost allocation structures Change work flow • pre-scheduling • cost and schedule estimation • incremental approach • path-analysis • risk-driven project planning • physical arrangements

Table A1-1. A Leavittian model of risk management.

Leavitt's component	Risks Item	Risks Resolution Technique
Actor	Actor pitfalls • lacking/variation • turnover • non-willing/ethical problems • poor or inappropriate beliefs, skills, and experience • political conflicts and power plays	Improve actors • staff with top talent • seek champions • cross training • morale building • user commitment • manage expectations • implementation games • training • role playing • study and screen potential actors
Technology	Pitfalls in technology • complexity • components unreliable • performance shortfalls • technical interfaces, • defects in quality • new and untried Technological uncertainty • high cost and non-adaptability • maintainability • extendibility	Improve technologies • specification standards and methods • task and organizational analysis techniques • information hiding/abstraction and modeling • bench marking • simulation/scenarios • prototyping
Task-Actor	Inappropriate actors for a given task • inability to specify or implement • goldplating	Improve fit • flexible governance structures • task matching • training
Task-Technology	Inappropriate technology for a given task • impossibility to implement or specify • poor performance • technology too expensive	Improve fit • contingency models for software development • manage technology options
Task-Structure	Inappropriate structure for the task • wrong project strategy • wrong control structure	Change the task to fit the structure • requirements scrubbing Change the structure to fit the task • adapt authority and decision structure • modify process model
Actor-Technology	Incompetent/too competent actors for the given technology • actors' experience • available computer science capabilities • gold plating • actors not willing to work with outdated/standard technology	Improve fit • prototyping, • technical analysis • scenario techniques • service assessment • technical training • hire top talent

Table A1-1. A Leavittian model of risk management (continued).

Leavitt's component	Risks Item	Risks Resolution Technique
Actor-Structure	Lack of commitment • wrong incentives • poor responsibilities • false beliefs and values • poor goals	Gain management support • apply appropriate leader-ship tactics • hire with good cooperation and management skills • install team building programs
Technology-Structure	Inappropriate fit • technology not aligned with authority and work-flow, • structure not appropriate for technology	Improve fit • change authority or work flow • adopt/configure new organizational technologies

Appendix 2: Boehm's Software Risk Management Model

Boehm's model (1991) suggests a comprehensive set of steps and guidelines to manage software production risks. The basic model consists of six steps. The most important steps for our analysis are those where one identifies risks and identified risks are planned for (steps 1 and 4). For step 1 Boehm suggests a risk-identification check-list which gives the top 10 primary sources of risks. Other techniques for this step include assumption analysis, decision driver analysis, and decomposition, but these are not analyzed here as they do not increase decision-maker's understanding of the development domain. Proposed check lists are the main means to identify the most likely problems. Boehm's top ten list is exhibited in table A2-1. Boehm also suggests more detailed checklists to analyze further each risk item and its

Risk Item	Coding
1. Personnel Shortfalls	A
2. Unrealistic schedules and budgets	S
3. Developing the wrong functions and properties	Ta
4. Developing the wrong user interface	Ta
5. Gold-plating	A-Ta
6. Continuing stream of requirements changes	Ta
7. Shortfalls in externally furnished components	T
8. Shortfalls in externally performed tasks	S
9. Real-time performance shortfalls	T
10. Straining computer-science capabilities	A-T

Table A2-1. Boehm's Top-ten Risk Item List.

probability (see Boehm 1991 p. 35–36). These lists are mostly based on software development handbooks from the U.S. Department of Defense. We shall not, however, explore these lists here further (though such an analysis could be done easily) as Boehm does not use them to identify major sources of risk.

For each risk item Boehm attaches a set of risk-management techniques that “have been most successful to date in avoiding and resolving the source of risk”. The idea is that after detecting the most important risk items risk-managers can compile the associated set of risk management measures and plans. Boehm's list of risk management techniques for each risk item is illustrated in table A2-2.

Risk Item	Risk Resolution Technique	Coding
1. Personnel Shortfalls	1. Staffing with top talent	A
	2. Job-matching	A-S
	3. Team-building	A-S
	4. Morale building	A
	5. Cross-training	A
	6. Pre-scheduling	S
2. Unrealistic schedules and budgets	1. Detailed, multisource cost and schedule estimation	S
	2. Design to cost	S
	3. Incremental development	S
	4. Software Re-use	Ta
	5. Requirements scrubbing	Ta
3. Developing the wrong functions and properties	1. Organizational analysis	T
	2. Mission analysis	T
	3. OPS-concept formulation	T
	4. User Surveys	T
	5. Prototyping	T
	6. Early user's manuals	T
4. Developing the wrong user interface	1. Task Analysis	T
	2. Prototyping	T
	3. Scenarios	T
	4. User characterization	T
5. Gold-plating	1. Requirements Scrubbing	Ta
	2. Prototyping	T
	3. Cost-benefit analysis	T
	4. Design to Cost	Ta
6. Continuing stream of requirements changes	1. High change threshold	S
	2. Information hiding	T
	3. Incremental development	S

Table A2-2. Boehm's risk resolution techniques.

Risk Item	Risk Resolution Technique	Coding
7. Shortfalls in externally furnished components	1. Bench marking	T
	2. Inspections	T
	3. Reference checking	T
	4. Compatibility analysis	T
8. Shortfalls in externally performed tasks	1. Reference checking	T
	2. Pre-award audits	T
	3. Award-fee contracts	S
	4. Contracts	S
	5. Competitive design	S
	6. Prototyping	T
	7. Team building	A-S
9. Real-time performance shortfalls	1. Simulation	T
	2. Bench marking	T
	3. Modeling	T
	4. Prototyping	T
	5. Instrumentation	T
	6. Tuning	T
10. Straining computer-science capabilities	1. Technical analysis	T
	2. Cost-benefit analysis	T
	3. Prototyping	T
	4. Reference checking	T

Table A2-2. Boehm's risk resolution techniques (continued).

Appendix 3: Davis' Model to Manage Requirements Specification Risks

Davis' model is concerned with selecting procedures that lead to complete and correct information requirements. He argues that one of the reasons for poor performance (and high risks) in systems development is that methods for obtaining and documenting user requirements are presented as general solutions rather than alternative methods for implementing a chosen strategy of requirements determination. Therefore he suggests “packaging” different methods into alternative strategies that are then carefully explained. These strategies are connected to a contingency framework (risk management model in our terminology) which suggests the most successful (least risky) requirements determination strategy for a given situation. The three risk items affecting the requirements specification are depicted in table A3-1. Davis uses these to define the overall requirements process uncertainty (risk level). Davis' model has four strategies which are regarded as most effective on different levels of requirements process uncertainty. Each strategy basically embodies a different technology/structure combination which the risk manager should configure so as to affect the actors to

obtain a good actor-task fit. Hence we get a coding of Davis' requirements specification strategies as shown in table A3-2.

Risk Items	Coding
1. Existence and stability of a set of usable requirements	Ta
2. Ability of users to specify requirements	A
3. Ability of analysts to elicit and evaluate requirements	A

Table A3-1. Davis' risk items.

Requirements determination strategy	Coding
1. Asking from users	T-S
2. Deriving from existing systems	T-S
3. Synthesis from characteristics of the utilizing system	T-S
4. Discovering from experimentation	T-S

Table A3-2. Davis' risk resolution strategies.

Appendix 4: Alter & Ginzberg's Implementation Risk Model

Alter & Ginzberg's implementation risk model focuses on problems associated with the organizational acceptance and implementation of the information system. They argue that implementing any system which leads to organizational acceptance will involve uncertainty from the managerial point of view. Therefore, these uncertainties should be detected and appropriate measures should be taken to minimize their impact. Overall they recognize several sources (factors) for organizational implementation uncertainty. These are depicted in table A4-1.

In suggesting risk resolution strategies, Alter & Ginzberg follow an approach similar to Boehm. For each risk factor they list a set of risk resolution strategies which are classified into inhibiting (I) or compensating (C) strategies. Inhibiting strategies are *ex ante* means to avoid a particular problem, while compensating strategies are *ex post* means to make up for a previous error or problem. Overall, Alter & Ginzberg's strategies lead to the classification shown in table A4-2.

Risk items	Coding
1. Designer lacking experience	A
2. Nonexistent or unwilling users	A
3. Multiple users or designers	A
4. Disappearing users, designers or maintainers	A
5. Lack or loss of support	A-S
6. Inability to specify the purpose or usage pattern in advance	A
7. Unpredictable impact	A
8. Technical or cost-effectiveness problems	T

Table A4-1. Alter & Ginzberg's list of implementation risk factors.

Risk item	Risk resolution strategy	Coding
1. Designer lacking experience	1. Use prototypes (C) 2. Use evolutionary approach (C) 3. Use modular approach (C) 4. Keep the system simple (C)	T S T Ta
2. Nonexistent or unwilling users	1. Hide complexity (C) 2. Avoid change (C) 3. Obtain user participation (I) 4. Obtain user commitment (I) 5. Obtain management support (C) 6. Sell the system (I) 7. Insist on mandatory use, 8. Permit voluntary use (C) 9. Rely on diffusion and exposure (C)	T S S A A A S S S
3. Multiple users or designers	1. Obtain user participation (C) 2. Obtain user commitment (C) 3. Obtain management support (C) 4. Provide training programs (C) 5. Permit voluntary use 6. Rely on diffusion and experience (C) 7. Tailor system to people's capabilities (C)	S A A A S S Ta
4. Disappearing users, designers or maintainers	1. Obtain management support (C) 2. Provide training programs (C) 3. Provide ongoing assistance (C)	A A A
5. Lack or loss of support	1. Obtain user participation (I) 2. Obtain user commitment (I) 3. Obtain management support (I) 4. Sell the system (I) 5. Permit voluntary use (C) 6. Rely on diffusion and exposure (C)	S A A A S S

Table A4-2. Alter & Ginzberg's list of risk resolution strategies.

Risk item	Risk resolution strategy	Coding
6. Inability to specify the purpose or usage pattern in advance	1. Use prototypes (C) 2. Use evolutionary approach (C) 3. Use modular approach (C) 4. Obtain user participation (I) 5. Provide training programs (C)	T S T S A
7. Unpredictable impact	1. Use prototypes (I) 2. Use evolutionary approach (I) 3. Obtain user participation (I) 4. Obtain management support (C) 5. Sell the system (C)	T S S A A
8. Technical or cost-effectiveness problems	1. Use prototypes (I) 2. Use evolutionary approach (I) 3. Use modular approach (C) 5. Keep the system simple (I)	T S T Ta

Table A4-2. Alter & Ginzberg's list of risk resolution strategies (continued).

Appendix 5: McFarlan's Model of Managing a Portfolio of Projects

McFarlan proposes three major risk items (dimensions influencing the inherent risk) related to project organization and management as shown in table A5-1.

Name of the Risk Item	Content of the Risk Item	Coding
1. Project size	Size in cost, time, staffing level, or number of affected parties	Ta
2. Experience with technology	Familiarity of the project team and the IS organization with the target technologies	A-T
3. Project Structure	How well structured is the project task	Ta

Table A5-1. MacFarlan's risk items.

McFarlan classifies general methods (risk resolution techniques) for managing projects into four principal types:

External integration tools include organizational and other communication devices that link the project team's work to the users at both the managerial and the lower levels, see table A5-2a.

Internal integration devices ensure that the team operates as an integrated unit, see table A5-2b.

Formal planning tools help to structure the sequence of tasks in advance and estimate the time, money, and technical resources the team will need to execute them, see table A5-2c.

Formal control mechanisms help managers evaluate progress and spot potential discrepancies so that corrective action can be taken, see table A5-2d.

For each group, a number tools (risk resolution techniques) are suggested.

External integration tools	Coding
1. Selection of user as project manager	S
2. Creation of user steering committee	S
3. Frequency and depth of meetings of this committee	S
4. User-managed change control process	S
5. Frequency and detail of distribution of project team minutes to key users	S
6. Selection of users as team members	S
7. Formal user specification approval process	S
8. Progress reports prepared for corporate steering committee	S
9. Users responsible for education and installation of system	S
10.Users manage decisions on key action dates	S

Table A5-2a. McFarlan's risk resolution techniques for external integration tools.

Internal integration tools	Coding
1. Selection of experienced DP professional leadership team	A-S
2. Selection of manager to lead team	S
3. Frequent team meetings	S
4. Regular preparation and distribution of minutes on key design decisions	S
5. Regular technical status reviews	S
6. Managed low turnover of team members	A
7. High percentage of team members with significant previous work relationships	A-S
8. Participation of team members in goal setting and deadline establishment	S
9. Outside technical assistance	S

Table A5-2b. McFarlan's risk resolution techniques for internal integration tools.

Formal planning tasks	Coding
1. PERT, critical path etc., networking	T
2. Milestone phases selection	S
3. Systems specification standards	T
4. Feasibility study specifications	T
5. Project approval process	S
6. Project post audit procedures	T

Table A5-2c. McFarlan's risk resolution techniques for formal planning tools.

Formal control tasks	Coding
1. Periodic formal status reports versus plan	T
2. Change control disciplines	T
3. Regular milestone presentation meetings	S
4. Deviations from plan	T

Table A5-2d. McFarlan's risk resolution techniques for formal control tasks.

Appendix 6: Description of the Coding and Analysis

General Description and Motivation

Categorical analysis is the process of identifying, coding, and categorizing primary patterns in the data. In conventional qualitative research this means analyzing the content of observations, interviews, or some publicly available data (such as annual reports) to identify trends and solicit major patterns of meaning. In our case, the data set consisted of four articles describing risk management approaches. The goal for using categorical analysis was to develop a systematic representation of the four approaches by using Leavitt's categories and thereby to reveal their varying foci and rationale. Our use of research (or consulting) articles as data sets is original though not unique (see e.g. Beath *et al.* 1994).

Coding Rules

The coding was conducted by classifying every risk item and risk resolution technique using Leavittian components. Before actual coding, we agreed on a number of coding rules. Each risk item and risk resolution technique was assigned to one Leavittian coding scheme. The coding was based on the title of each item plus a careful reading of its content description. Sometimes this lead to a further reading of the item description in the main body of the text (if this was available) to further clarify its meaning. This was motivated by the observation that applying the name of the risk item or the resolution technique as a sole basis for coding information was insufficient.

For example, McFarlan's risk item "Project structure" is denoted in our coding as **Ta** (and not S) because he defines this item as to deal with how well structured the project task is (and not the project structure). We also agreed initially to use relational codings such as **T-A** (read Technology-Actor relationship) in addition to pure component codings to characterize those situations where the source of risk was caused by a misfit between the components. This was motivated by Leavitt's (1964) observation that any organizational problem can cover one or several of the components thus adding the need to consider mutual relationships. An example of such a coding is McFarlan's second risk item "Experience with the technology" which is defined as familiarity with the target technologies. This amendment was necessary as few relational codings for both risk items and risk resolution techniques were detected during the coding. Overall, however, the number of relational codings was surprisingly low. These were counted as .5 while counting the frequencies of risk items in each Leavittian component.

The concern for the validity of distinctions (like between technology and structure) was another important concern in our effort. Because we used Leavitt's components as analytical devices rather than ontological categories we did not regard distinctions to be absolute (in terms of the focus of perception). This idea is also noted by Leavitt when he writes "Although I differentiate structural from technical from human approaches to organizational tasks, the differentiation is in points of relative weightings and underlying conceptions and values, not in the exclusion of all other variables" (Leavitt 1964, p. 56). For our classification task the major concern is therefore the discriminating validity of Leavitt's constructs.

We started with the above coding rules, but were open to add or modify the coding scheme should any such need arise. In particular, we were concerned with how environmental and cultural issues would be covered in our coding scheme, because the omission of these components has been one major criticism against his model. This caution was premature as we found only two risk items (7. "shortfalls in externally furnished components" and 8. "shortfalls in externally furnished tasks" in Boehm's risk item list) out of 24 which we could classify as environmental risks. Both of them could be easily interpreted either as technological risks (7) or structural risks (8), and we therefore did not add a new category "environment" to our coding. We could observe two risk resolution techniques out of 85 that dealt with cultural phenomena. These were Boehm's suggestions to use "team building" and "morale building" in resolving personnel shortfalls. (Team building is also suggested to reduce shortfalls in externally furnished tasks which

suggests a structural reading of this technique). Because the number of such items was so low we decided to code both of them as **A-S** to simplify the coding scheme. In our understanding, this does not violate a "correct" reading of them**.**

During the coding we had to deal also with possible multiple codings of the risk items or risk resolution techniques, i.e. the possibility that the item or the technique could cover two independent Leavittian categories thus signaling an ambiguous definition. We could find only one such instance. It dealt with Boehm's risk resolution technique "Competitive design or prototyping". In further analysis this was broken into two independent resolution techniques where the former dealt with the **S** component and the latter with the **T** component.

One concern in the coding is also the reliability of the codings. In our case four risk items out of 24 (16%) were originally coded differently[1], but with risk resolution techniques only one out of 82 (1 %). In all the codings the level of consensus achieved was ca. 95% during the first round which is a very high coding reliability. In cases where differences in coding were observed the coders discussed their reasons for coding and tried to find a common understanding of how the item should be coded. This was done by consulting once again the original text and interpreting the context of the term to ascribe a correct interpretation for the item. This resulted in codings on which all agreed and which are presented in appendices 2 through 6.

Analysis

The two above steps were conducted independently for all articles by the authors. Obtained results were then compared for commonalties and differences to yield table 1. We counted the frequencies of different types of Leavittian categories for both risk items and risk resolution techniques. These frequencies were then applied to determine the relative risk item and risk resolution focus.

Notes

1. These were in Boehm's list: shortfalls in externally performed tasks, short-falls in externally furnished components , and straining computer-science capabilities; and Alter & Ginzberg's list: lack or loss of support.