Controlling Prototype Development Through Risk Analysis*

By: Richard L. Baskerville
Binghamtom University
Vestal Parkway East
P.O. Box 6015
Binghamton, NY 13902-6015
U.S.A.
rbask@binghamton.edu

Jan Stage
Aalborg University
Department of Computer Science
Fredrik Bajers Vej 7
DK-9220 Aalborg East
DENMARK
jans@cs.auc.dk

Abstract

This article presents a new approach to the management of evolutionary prototyping projects. The prototyping approach to systems development emphasizes learning and facilitates meaningful communication between systems developers and users. These benefits are important for rapid creation of flexible, usable information resources that are well-tuned to present and future business needs. The main unsolved problem in prototyping is the difficulty in controlling such projects. This problem severely limits the range of practical projects in which prototyping can be used. The new approach suggested in this article uses an explicit risk mitigation model and management process that energizes and enhances the value of prototyping in technology delivery. An action research effort validates this risk analysis approach as one that focuses management attention on consequences and priorities inherent in a prototyping situation. This approach enables appropriate risk resolution strategies to be placed in effect before the prototyping process breaks down. It facilitates consensus building through collaborative decision making and is consistent with a high degree of user involvement.

Keywords: Action research, IS project risk management, IS control, IS management issues, IS development strategies, prototyping, iterative design, evolutionary design

ISRL Categories: AI0101, EE0101, EJ, EL, FA, FA02, FA03, FA04

* Blake Ives was the accepting senior editor for this paper.

Introduction

Organizations are using information systems as a lever to improve their flexibility and focus their activities. Examples of this use include cross-functional virtual organizational forms and business process reengineering (Mackenzie 1986; Yoo 1995). The new organizational forms that emerge from the leverage are fundamentally dependent on internetworked systems that must adapt readily to rapid changes in their associated marketplaces. Adaptive information systems require shorter systems development cycles and networks of small, innovative systems. Prototyping is often suggested as a mechanism for improving the effectiveness of analysis and design in loosely structured, high technology development projects needed in adaptive situations. Prototyping alleviates many of the practical problems that arise in requirements definition (Boar 1984; Mason and Carey 1983; Naumann and Jenkins 1982; Zelkowitz 1982) and improves design effectiveness by integrating users directly into the design process (Budde, et al. 1984; Ehn 1988, Grønbæk 1990). This article is specifically concerned with managing evolutionary prototyping projects where iterative development and user evaluation cycles carry the system from initial analysis into operation (Connell and Schafer 1989).

There are different forms of prototyping. There are "throw-away" design prototypes, e.g. mock-ups and user interface prototypes, that have limited functionality and precede the specification process. There are specification prototypes that provide a throw-away working model of an entire system prior to specification and construction. There are design-driven prototypes that provide a prefinalization "test-drive" of a traditionally developed system. The classic prototyping approach is embodied in evolutionary prototypes that begin as design prototypes and cycle through iterative phases of prototype reconstruction and user evaluation until full functionality is achieved (Connell and Schafer 1989).

The advantages of prototyping are widely acknowledged. Systematic use of prototypes relieves many of the problems that occur when information systems development is based on extensive use of specifications (Boar 1984; Ehn 1988; Lantz 1988; Vonk 1990). Prototypes provide users with a concrete understanding of the proposed computer system. They eliminate the confusion and potential for misunderstanding that originate from the interpretation of abstract specifications and replace this with meaningful and direct communication between systems developers and users. Accordingly, prototyping has been numbered among the four potentially fundamental advances in software engineering (Brooks 1987) and has been characterized as an essential component for use in every development project for interactive systems (Martin 1990).

The success of prototyping has been limited by certain problems. These problems include the inefficiency of prototyped computer systems (Lantz 1988), the impracticality of large prototypes (Alavi 1984), the unrealistic expectations created by prototypes (Iivari and Karjalainen 1989), the lack of effective development environments including powerful CASE tools (Luqi 1989; Vonk 1990) and the difficulty of managing the prototyping process (Alavi 1984; Boar 1984; Connell and Schafer 1989).

Solutions to some of these problems are being developed. More powerful computers and development environments are gradually alleviating problems with inefficient and ineffective prototypes. To some extent, these technologies will enable larger prototypes and help developers to satisfy both their own and their users' expectations. However, the prototype project management problem seems to remain as a critical barrier.

Management of prototyping is made problematic by its dependence on iterative activities. The basic management functions of planning and control are limited because plans are supposed to change with each cycle, and control is hampered by lack of meaningful progress measurement coupled with the uncontrolled dependence on user cooperation. The purpose of this article is to contribute a new approach to the management of a prototype development project that improves these management functions and overcomes the last remaining barrier. We focus on risk management as a core decision factor in evolutionary prototyping project management. We detail both the theoretical foundations and the practical implications of this new approach. These theoretical and practical elements are fused in an action research project that validates the approach and contains evidence that this technique allows organizations to fully benefit from the promises of evolutionary prototyping.

Problems in Controlling Prototype Development

The management of information systems development projects that involve prototyping faces two major challenges. The first is to determine the extent to which prototyping, as opposed to more traditional approaches, should be applied in certain situations. The second is to control the course of a prototyping effort once it has been decided to use this approach. Below, the first of these questions is briefly discussed before addressing the second, which is the main scope of this paper.

Prototyping versus specifying

A key challenge in information systems development is to determine the relevance of specific development approaches. The simple answer is to focus on the type of system being developed. In projects where high technical complexity makes software reliability and verification paramount, it has been suggested that the need for specifications makes the prototyping approach redundant (Parnas 1985). Such formulas exclude batch systems, process control systems, and complex systems. But it has been pointed out that most of the evidence proscribing such situations is anecdotal (Iivari and Karjalainen 1989). Further, contrasting evidence shows prototyping can be effective for process control systems (Duke, et al. 1989) and systems for alienated users (Baskerville 1993). Prototyping has also found new roles in object-oriented systems. All components of an object-oriented design should be prototyped (Coad and Yourdon 1991), and prototyping can be critical for specifying complex object-oriented software systems (Gupta, et al. 1989).

A more elaborate solution is to limit the use of prototyping to certain situations. But classifications of such "ideal" situations are problematic. For example, the ideal situations have been characterized by smallness in system size, objectives for interactive application, and an intense concern with the user interface (Connell and Schafer 1989). Further, the organizational setting is thought to require a great deal of user enthusiasm that will persist even after the essential system aspects are settled (Alavi 1984). The main problem with current work regarding ideal prototyping situations is the mechanistic manner in which such situations have been classified. This line of reasoning attempts to provide universal definitions that apply to highly varied organizational needs and settings.

The inclusion of a wider range of project attributes in determining the proper development approach has been more successful. A contingency approach has been suggested (Davis 1982) and subsequently refined (Burns and Dennis 1985). The strength of the contingency approach is that it detaches the developer from methods that are inappropriate for a particular setting. Its main drawback is that it determines the relevance of prototyping or specifying from an evaluation of the software project as a whole.

Management problems in prototyping projects

The prototyping literature suggests at least four key factors that intensify the problems of managing a prototyping effort. First, it can be difficult to get managers, analysts, programmers, and users to agree to the exact objectives of the process. Without such a mutual agreement, user-designer collaboration may splinter, and user enthusiasm can drop off quickly, leading to a shallow analysis and design (Alavi 1984).

Second, the project manager has only limited control over users and their interaction with designers. This can lead to one of two ill-effects. When the users dominate this interaction, they may inflate the project scope in an attempt to "do it all" while they have the resources of the prototyping project available (Connell and Schafer 1989). When the designers dominate, they may deflate the project scope in order to reduce the programming work (Boehm, et al. 1984).

Third, each iteration involving development and subsequent evaluation of a prototype may uncover a multitude of potential revisions and directions of further improvement. This confronts management with a hard decision point concerning the contents of the next prototype (Vonk 1990). If this decision is not made, the process may splinter into different and possibly unproductive directions.

Fourth, it is difficult to comparatively measure the "fit" of each iteration of a prototype. Thus, project progress and nearness-to-completion cannot be gauged or accurately reported. This can lead to "over-evolved" prototypes, sacrificing low-cost flexibility for elegant and efficient program structures. Alternatively, the prototype may be delivered prematurely with either a design that is unsuitable to the users or programs that are poorly tested, documented, or tuned (Connell and Schafer 1989).

Existing work on prototyping management reflects two simple mechanisms to resolve the problems. These mechanisms impose essential limits on either the duration or scope of prototyping. Limiting the duration means that the time spent on prototyping is confined to a strictly limited period (Martin 1990). Limiting the scope of prototyping means that the prototypes are applied to very narrow areas of the development process (Lantz 1988). Rather than employing prototyping as an overall approach to systems development — as with evolutionary prototyping — only narrow design and design-driven prototypes are used in an otherwise traditional project. Limiting scope or duration are each mechanisms that side-step the project management problems of prototyping. This not only constrains the role of prototypes in systems development, but also the potential benefits demonstrated by ongoing prototyping research. Successful prototyping projects require both functionality and data storage facilities. A simple implementation of a very limited part of the system is not enough to entice real user interaction (Grønbæk 1990).

Risk Analysis in Prototyping

This section describes our risk analysis approach in a prescriptive fashion. Aside from the prototyping literature, this approach can be traced to three streams of risk analysis thought. These streams are security risk analysis, contingency management, and the spiral model.

The first stream of literature originates from the information systems security community. In this stream, risk analysis is employed to produce knowledge about the security of a system under development (Baskerville 1991). Many published methodologies adopt or adapt the approach (Badenhorst and Eloff 1990; Courtney 1977; Fisher 1984,; Fitzgerald 1978; Parker 1981), large organizations practice variations (Saltmarsh and Browne 1983), and it is part of a U.S. Federal Information Processing Standard (NBS 1979). These contributions are all based on a conventional approach to risk analysis. This approach involves two major elements of risk: P, the probability of an exposure's occurrence, and C, the economic cost or loss attributed to such an exposure. The risk R related to this exposure is then calculated as the product of the two elements: $R = P * C$.

A major drawback of conventional risk analysis is the use of monetary units as a means for measuring the severity of a risk. Its basic data values (risk probabilities and loss estimates) are highly interpretive; they are usually gleaned whole by the professional from an unstructured study of the complex multivariate organizational landscape. These values are then manipulated with very positivistic formal and logical mathematical operations. But if the original estimations are invalid, then the probability arithmetic that follows these is complete nonsense. Considered as a scientific or statistical approach, risk analysis seems a shallow exercise in simple guesswork (Baskerville 1991). It merely provides an attractive scientific structure in which to frame the guesswork. Still, it has survived as an important practical technique in information systems security. The key reason is that the simple probability arithmetic allows the security problem to be expressed in a calculus that is familiar to management and in monetary terms that permit comparison with capital opportunity costs. The probability arithmetic is the language for expressing a subjective, but well-founded, professional opinion.

The second stream of literature regards contingency approaches to systems development and management. The determination of systems development methods can be dependent on various organizational factors, and an organization can select the exact approach dynamically based on various organizational contingencies (Davis 1982). Aside from the development method, the decision regarding whether to undertake a systems development project can be based on contingencies and risk analysis. There are known risk factors that need to be considered before undertaking a risky information systems development project. These include the stability, experience, and quality of the development group, the role of information systems in current and future decision-support and corporate services, recent major information systems fiascoes, or a new information systems management team (McFarlan 1981). In addition to determining whether to undertake a risk project, risk analysis can be used to define project management strategies, such as the use of formal planning and control or the integration of the project team into the users' environment.

The third stream of literature is concerned with general software development frameworks (Boehm 1988; 1989). The experience is that systems developers may enter a project with a fuzzy understanding of the situation, and this understanding is often subject to several revisions in the course of the project (Mathiassen and Stage 1992). The essential limitation of non-prototyping systems development frameworks is their static nature. This is illustrated clearly by the results of an experiment that was conducted in 1982 (Boehm, et al. 1984). It applied specifying and prototyping as two specialized and separate approaches, and the results of this experiment were used to compare their relative strengths and weaknesses.

These results have led to the formulation of Boehm's (1988) Spiral Model, which combines the use of specifications with user interface or design-driven prototypes. The Spiral Model is a constructive attempt to employ risk analysis as a means to determine the relevance of either a specification or a prototype. This analysis is conducted regularly since the relevance of a specification versus a prototype is assumed to change when the risks of a project change. The Spiral Model has also been a source of inspiration for other approaches, such as a contingency model that determines a proper mix of prototyping techniques and more structured approaches based on uncertainty and complexity (Saarinen and Vepsalainen 1993). The definition of these factors originate from theoretical work (Mathiassen and Stage 1992) that has been further explored in more recent empirical studies (Mathiassen, et al. 1995).

The second and third stream both embody an approach to risk analysis that is basically different from the conventional approach. These streams of thought recognize the importance of the interpretive, subjective contribution of the designer in estimating the costs and probabilities. Perhaps more importantly, the contribution of the designer lies in the identification of *potential* sources of risks. The designer must find the basic set of possible risks in a certain situation. The third stream has the disadvantage of fully integrating risk analysis in an essentially specification-based method. Furthermore, the Spiral Model allows only user interface design and design-driven prototypes. There is no consideration for evolutionary prototyping. We believe a broader view of the use of risk analysis will yield a qualitative technique for successful management of many forms of prototyping in a variety of organizational settings.

Overview

Risk analysis techniques can support the management of prototype development by providing a framework for determining priorities, resources, and activities during the course of an evolutionary prototyping project.

The interplay between the two overall activities of Figure 1 illustrates the essence of prototyping project management using risk analysis. The control of the project is influenced by an interplay between risk analysis cycles and prototyping cycles in which different resolution strategies are used. A risk analysis cycle evaluates the current situation of the project and determines relevant resolution strategies. These strategies may then be used as a basis for a prototyping cycle in which a new version of the prototype is developed and evaluated. The experience gained through the prototyping cycle then forms the foundation for the next risk analysis cycle.

The process of risk analysis, being illustrated as the overall activity on the left hand side of Figure 1, is divided into four activities. These activities are: (1) define potential risk factors, (2) evaluate potential risk factors and specify consequences, (3) assign priorities in order to identify essential high-risk issues, and (4) suggest and select resolution strategies to the most urgent risk factors. These strategies form the basis of the next cycle of prototype development. Thus, in time, a new prototyping cycle appears after activity 4 in Figure 1. The experiences gained through this prototyping cycle and the suggested resolution strategies then become the foundation of the next cycle of risk analysis.

This risk resolution interplay is fluid and dynamic in practice and also provides a collaborative mechanism that draws the participants toward a consensus about project priorities. One possible consequence of this collaboration is that development activities become more cohesive and better directed, particularly with respect to the most critical project problems. This directed group cohesion is notable in each of the activities described below.

Define risks

Risk analysis in prototyping essentially enables collaborative expression of a subjective evaluation of the situation. With this in mind, risk analysis should begin with an unstructured, brain-storming group session with the objective of formulating the initial risk inventory. Extensive, universal lists should be avoided at this early stage, since they can interfere with the creative process. However, once the team has exhausted their intuition, such checklists are helpful in structuring the discovery of a final formulation of risks that is very complete.

We originally developed a taxonomy from an analysis of existing contingency approaches and surveys of typical development project problems. This taxonomy emphasizes potential risk areas of a situation. The taxonomy was validated in the action research project described in the following two sections. The taxonomy divides the characteristics of the situation into the elements that are shown in the first column of Table 1.

Systems Developers: The characteristics of the systems developers relate to their knowledge about the application and problem domains, their ability to make a complete and consistent design specification, their experi-

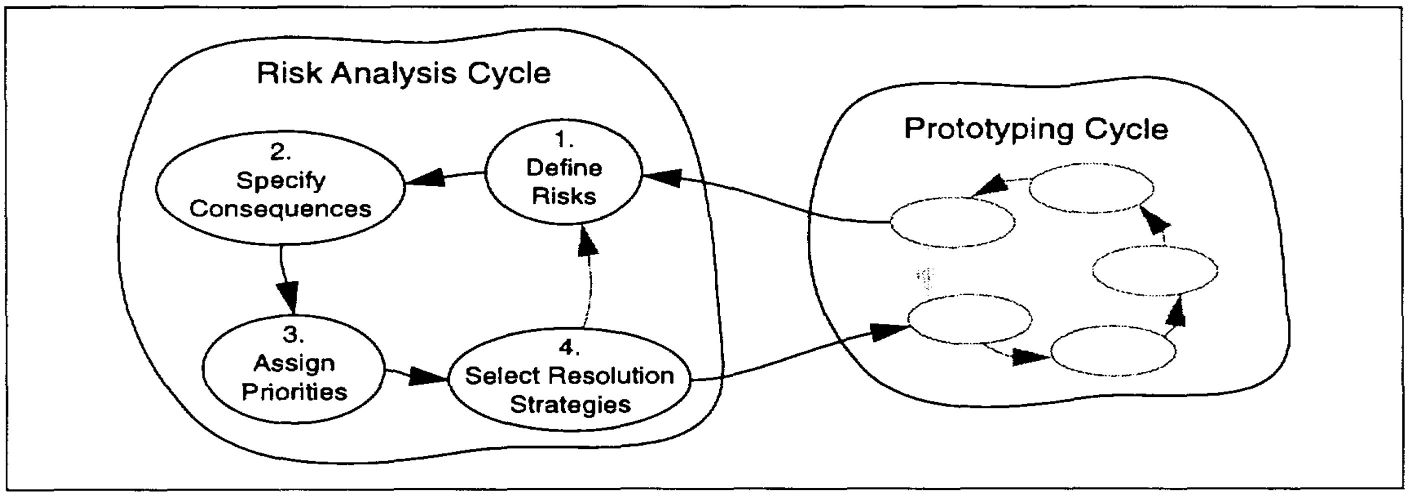

Figure 1. The Use of Risk Analysis in Evolutionary Prototyping

Table 1. Categories of Risk Analysis Factors

Define Risks	Specify Consequences	Select Resolution Strategy
Systems developers	Deteriorated social relations	Improve setting
Users	Process problems	Develop pilot prototype
Application domain	Product defects	Provide different options
Problem domain	Exceeded resource limits	Limit prototype size
Computer system		
Development environment		

ences with specification of requirements in cooperation with prospective users, and their ability to implement the requirements with the available technical environment.

Users: Characteristics of the users include their ability to describe the application and problem domains in a logical and structured fashion and to specify requirements in cooperation with the systems developers, their experience with systems development and prototyping, their understanding of design specifications and the available computer technology, and their ability to review the proposed design specifications of the computer system.

Application Domain: The application domain is the total set of work tasks in the user organization that will be supported by the computer system. Characteristics of this domain that can promote risk include lack of clarity and consistency of the organizational tasks and boundaries. Severe risks arise if the boundaries of the application domain are unclear, when there are broadly diverse, complex, or unstructured tasks, and when these tasks are continuously shifting in response to a turbulent organizational environment.

Problem Domain: The problem domain is the object of the work tasks in the application domain. It reflects the mission of the users' work. Risks may be imposed if the problem domain includes many complex objects with complex relationships, if it includes many complex occurrences of events, if the boundaries of the problem domain are not clearly defined, and if these boundaries are continuously changing due to environmental turbulence.

Computer System: Risks can arise from the characteristics of the computer system being developed. These characteristics include situations where ambiguous and inconsistent computer systems requirements exist, where the computer system entails a database with interactive, transaction processing and reporting, where there are specific computer systems performance and network data communication requirements, and where the computer system to be developed is partly incompatible with the development environment.

Development Environment: Finally, risks can arise from the technical environment. Examples include unreliability in the target computing machinery or systems software, unreliability in the development computing machinery or software, insufficient or ambiguous documentation of the development environment, and linkages to externally controlled technologies like standardized internet clients or servers.

Specify consequences

Once the risk inventory is considered complete, the project group should specify the consequences of each risk. The purpose of this is to consider what undesirable situation will result if the risk occurs. This evaluation will prepare the team to rate probability and severity for each risk item.

The distinction between risk and consequence is subjective. This is because these two events are subjectively selected from a causal network of events. For example, an unreliable computer may cause extra work tasks that may cause excessive use of resources. The group must subjectively decide whether the extra work tasks are a risk or a consequence or an unimportant link between the "real" risk and the "real" consequence. Risks and consequences that are expressed in general terms are typically harder to separate as opposed to more specific and detailed formulations. The section on risk analysis in action includes an example of a risk that was initially expressed in very general terms. A later evaluation showed that it was necessary to split this risk into two risks that were more detailed. Compared to the original risk, it was much easier to assign consequences to the new and more detailed risks.

The consequence of any particular risk is usually unique to the project setting, making a useful taxonomy difficult. The action research project, described in the following section, predicted or experienced consequences within the generalized categories that are shown in the second column of Table 1.

Deteriorated Social Relations: These include relations within the project group as well as relations between the project group, the users, and management. An example of this is a lack of cooperation from the users.

Process Problems: These problems typically arise when parts of the project plan prove impossible. This includes a variety of problems spanning from designs that cannot be implemented to unrealistic expectations on the part of one of the involved groups.

Product Defects: These consequences relate directly to design and implementation of the computer system being developed. While this category of consequences emphasizes the quality of the final system, it also regards intermediate products. Also, this category is not just limited to early and final versions of the prototypes, but also considers the related documents and specifications.

Exceeded Resource Limits: Limits that might be exceeded include time, manpower, and costs. Many risks can ultimately trigger this consequence by delaying progress and causing late deliveries. Where the risk formulation is not very specific, this category of consequences may be overemphasized. Analysts should be cautious when approaching this category of consequences, asking whether the risk description is too general. Specific risk terms will help identify the intervening consequences that lead to overtaxed resources. Where overtaxed resources are the only consequence, analysts must be specific as to exactly how limits will be exceeded.

Assign priorities

The definition of risks and the specification of consequences produce a list of potential, undesirable occurrences. The high-risk issues have to be selected from this list. There are two approaches to the ranking of potential risks:

1. Negotiation of compound ranks

2. Combination of individual rankings

The first approach is a three-step process used collectively in a project meeting. First, each risk factor is ranked on a severity scale from 0 to 5. A 0 value is relatively low and implies that the consequence of the risk is assumed to be without significant influence on the success of the project. A 5 value indicates relatively high severeness in the sense that the consequence of the risk factor is expected to be fatal to the success of the project. Second, each risk factor is ranked on a probability scale of 0 to 5. A 0 value is relatively low and indicates that the consequence of the risk is unlikely to occur. A 5 value indicates relatively high probability of the consequence of the risk in question. Third, the severity rank and probability rank of each risk factor are multiplied, giving a compound risk rank. High-risk issues will get a high product because they are very severe and/or very likely to occur.

The second approach is based on individual rankings. This approach is appropriate when there are too many participants in the risk analysis to allow quick negotiation of scalar values. Each member of the project group makes a ranking independently of the others. The ranking is a simple sequential ordering, expressed as a number between 1 and the total number of potential risks. A low number means that a risk is considered more important than one with a higher number. The rankings made by each member of the group are then added to yield a compound ranking of the risks.

Once the risk factors have received their compound rank using one of these priority assignment approaches, the discussion of resolution strategies can be simplified. Resolution strategies are selected for those risk factors that surface as high-risk issues. It is usually relevant to select resolution strategies for the two-four risks that have the highest ranks.

Select resolution strategies

Like risk consequences, many risk resolution strategies are likely to be unique to a given organizational setting. In our action research project, we used the four classes of resolution strategies that are shown in the third column of Table 1. Below, these resolution strategies are related to the risks and consequences discussed above.

Improve Setting

When risks arise from developers or users, these often represent dysfunctional aspects of the social, organizational, and technical setting of the prototyping project. For example, users may be unwilling to devote resources to the project. Another problem can relate to the unwillingness of users to make decisions or make commitments to systems requirements. There are two major strategies for resolving such risks.

First, the prototype project managers may want to promote action by organizational management that provides the project with access to resources and otherwise strengthens commitments to the project. This strategy is appropriate for organizational risks regarding users or the problem domain. This strategy corrects dysfunctional aspects of the social and organizational setting of the prototyping project.

Second, the users can be trained in the prototyping process. This strategy involves using the prototyping team to educate users in their benefits and responsibilities in the prototyping process. This can improve user understanding and promote user-designer cooperation in the project. Like organizational action, this can improve problems with user willingness and competence to participate in the prototyping process along with increasing collaboration among the social groups.

Risks that arise from computer system or development environment defects might be addressed with an early strategy of testing the environment with small prototypes. Later, it might be necessary to develop software that removes defects or to buy additional software products to improve the technical platform.

Develop Pilot Prototype

Development of a pilot prototype effectively limits the initial project planning horizon to a single prototype development cycle.[1] This initial prototype becomes the basis of prototyping management, and is only tangentially important in providing initial analysis of user requirements.

The purpose of this single prototyping cycle is to provide initial experience on which to base project management expectations. These are expectations about the performance of the development technologies, and they indicate potential application domain boundaries. They are also about the performance of the development team. In this way, this strategy addresses risks that arise from the systems developers, the computer system, and the development environment.

A pilot prototype is particularly useful with untried or undependable development technologies. Untried technologies include cases where the prototype development environment is new or cases where such an environment has been assembled especially for a particular project. Undependable technologies include cases where the development environment is known to be defective, but for practical reasons still must be used for developing the prototype.

This strategy is also effective for cases in which the development team is unfamiliar with the development technologies or lacks proper training or competence in the use of those technologies. In such cases, the project manager could not effectively predict the reliability, exact performance, and capabilities of the prototype development environment without some experience. To a certain extent, this strategy is also appropriate for cases where the development team is generally unfamiliar with prototyping.

Provide Different Options

This strategy involves parallel development of multiple, competing prototypes. Each prototype will demonstrate an alternative information architecture or technological base in the context of the users' information problems.

Competing prototypes can also demonstrate alternative ways in which functional and interface components might support user tasks. This permits the developers to investigate how various database and interface components could represent the problem domain. These prototypes will likely be shallow mock-ups that will result in the highest degree of user understanding of the architecture.

Most of these prototypes will be expendable, and the project manager should not overly develop any of the prototypes' functionality. The purpose of these prototypes is to address user, application domain, and problem domain risks by demonstrating to the users the range or domain of possible solutions.

Limit Prototype Size

This is a critical strategy connected with pilot prototypes or parallel prototypes where a high degree of uncertainty increases the likelihood that prototyping work will be discarded. But even in cases where single, more linear prototypes are being developed, each prototype is limited in certain ways compared to the final product. Thus, we have to decide which parts of a prototype should be developed and which should be ignored during a particular prototyping cycle.

The strategy for limiting the prototype is couched in terms of a "horizontal" or "vertical" slice (Budde, et al. 1992; Yourdon 1982), even where the design is not hierarchical. For example, Figure 2 shows the division of a prototype into a hierarchical set of various related components, e.g., menus, data-entry screens, functions, and reports. Figure 2(a) illustrates a horizontal slice of this prototype with shaded boxes. Such a prototype would be functional throughout the entire prototype design, but only to a limited extent in each functional area. It typically includes top-level menus and the key screens. Horizontal slice strategies help to address application domain risks by clarifying the systems boundaries and the variety of tasks that are required in the system.

Figure 2(b) illustrates a vertical slice of a prototype with shaded boxes. Such a prototype would only be completely functional in one area. It typically includes one screen and all of the underlying functionality. Vertical slice strategies help to resolve problem-domain risks by fully developing complex objects and relationships, and clearly distinguish the boundaries of the problem that can be addressed by the system. Vertical slice strategies also help with application domain problems by demonstrating systems capabilities regarding complex or unstructured tasks.

This concept of horizontal and vertical slices operates differently depending on the design technology. For example, in the style of a Coad and Yourdon (1991) object-oriented prototype, a horizontal slice refers to objects in the human-interaction component, while a ver-

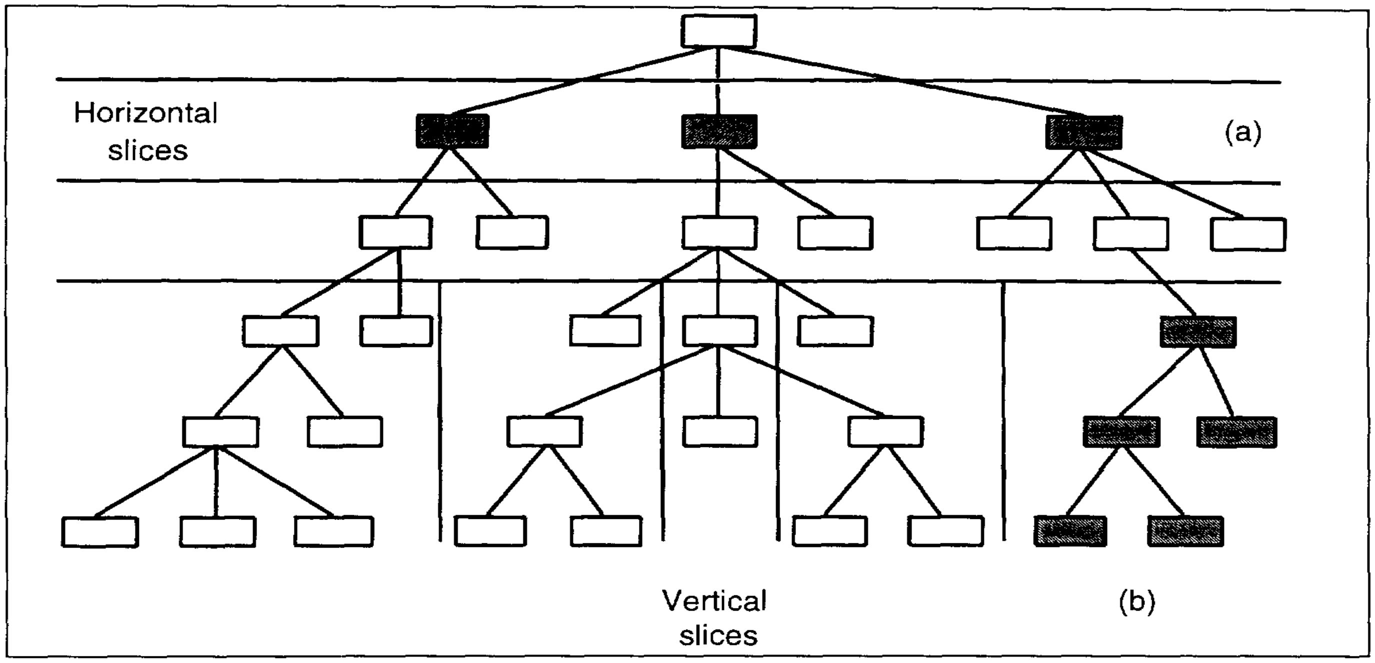

Figure 2. Horizontal and Vertical Slices

tical slice refers to one set of subject-related objects in the problem-domain component.

In our action research project, we did not discover any direct mapping or heuristic for connecting together any individual risk, consequence, or resolution categories (see Table 1). However, at least one such connected checklist exists in the software engineering literature (Boehm 1988, p. 70). Although this checklist operates at a lower level of abstraction than the categories that arose in our project, it may also prove useful in discovering risks and selecting resolution strategies.

Once risk analysis has been used to help in planning the immediate prototyping cycle, the prototype can be constructed and evaluated in collaboration with the users. The techniques used in the prototyping cycle can be selected according to the setting as described in the prototyping literature. For example, we have used the prototype development and evaluation techniques from Connell and Schafer (1989) and Mathiassen, et al. (1991) in different projects.

Risk analysis iteration

After each user review of a prototype, the team may plan to reconstruct it. During this planning, the team re-evaluates the risk analysis. As part of this iteration the team may delete or add items to the risk inventory. More frequently, the team simply will adjust the probability and severity figures. This is illustrated in a detailed way in a subsection below that discusses the evolution of the risk factors.

A high level of overall risk in a re-evaluation suggests collective uncertainty or disagreement about the design specification. This indicates that the team should plan prototype reconstruction and additional review. In such cases, the team should reconsider resolution strategies for all severe risks that remain on the list, in addition to any new risks.

This iterative use of risk analysis should improve the quality of the results. For example, risks that were unnoticed in the earlier iterations may appear more obvious later in the project. But failures may be made in any of the four risk analysis activities described in this section. We may define irrelevant risks, specify unrealistic consequences, assign inappropriate priorities, or select impractical resolution strategies. Any of these failures might occur, and iterating the risk analysis does not seem to protect against them.

Research Method

The risk analysis approach presented above was refined and validated using action research.[2] Action research assumes that complex social processes cannot be reduced for meaningful study. A complex social process is best studied by introducing changes into that process and observing their effects. The theory underlying the changes is validated by the extent to which these changes successfully solve specific problems in the setting. This is distinct from other methods such as laboratory experiments or case studies in which the validity arises from replicability or exhaustive elimination of alternative explanations.[3,4] Further details about the role of action research in information systems is discussed in Galliers and Land (1987) and Baskerville (1993).

The nature of action research is to focus narrowly on the details of a single organizational setting. Likewise, this article focuses on one such project that was conducted by both of the authors. This project is large enough to illustrate the use of our prototyping management approach, yet small enough for an adequate description to fit within the space of a single article. But this project does not represent the only experience with this approach. There have been four other action research projects using the same approach for managing the prototyping process. See the Appendix for a description of the distinct organizational setting and interventions for each of these projects. These projects developed prototypes for: a project management tool, which involved only one of the authors and only as a system user; a network administration system, which involved the construction of much larger prototypes; a job locator service system, which was developed in a very different cultural and educational setting; and a commercial laundry management system, which is an example of a prototyping process that broke down.

To illustrate the action research project, a context of a non-profit organization that provides counseling for individuals, groups, and families was chosen. In total, there are 68 centers spread across the United States. The counseling center was founded in 1983. It is headed by a counselor and staffed with two additional counselors, a secretary, and a part-time clerical aide. A subsidiary office, located in a nearby community, was established in 1989. The Center also plans to open an office in another nearby city in the near future. The Center is funded through client payments and donations. Eighty percent of the funding comes from clients paying for counseling. A limited amount of this is paid by insurance companies, since some health policies cover counseling. The remaining 20 percent of the funding comes from donations and grants paid by individuals, clubs, and churches. Some of these funds are set up exclusively for certain groups of clients such as women-in-transition, people with low income, or young adults.

The next section describes the relevant details of the action research project to illustrate the use of the prototyping management approach. Further details about the process and the products produced can be found in Baskerville and Stage (1991).

Risk Analysis in Action

The research infrastructure was clarified in a contract in which we agreed to deliver an undefined operational system. In return, the host organization agreed to support the action research and the use of an experimental systems development management technique. The collaborative team consisted of the authors plus a counselor and a secretary from the center.

The team focused on the problem of administration of clients and counseling sessions. All record-keeping was being handled manually. Client data were registered on 3x5 index cards and referenced both by client name and client number. Two labor-intensive statistical reports included a monthly status report for the Board of the Center and an annual status report for the Samaritan Institute in Denver. The monthly status reports describe the number and composition of clients, the number of sessions conducted by each counselor, and the income for that month. The annual reports describe the status of the center and summarize the main financial figures.

The collaborative team decided to create an evolutionary prototype of a computer-based client administration and reporting system. Within this infrastructure, the team studied the use of risk analysis as a means of controlling the prototype development process.

Initial risk analysis

Prior to the first meeting of the collaborative team, we spent about three hours creating the initial risk analysis. As detailed earlier, risk management in prototyping requires four steps. Below, these steps are reviewed in the context of the experiment, and the results are summarized in Table 2.

The initial risk analysis began with a brainstorming process, which allowed us to identify

Table 2. Initial Risk Factors

No.	Risk Factor	Consequences	Severity	Probability	Product
1	The users only want advice and training on existing tools such as Lotus, OneWrite+, etc.	The users will not cooperate seriously in the design activity.	4	2	8
2	The technological resources available in the application domain are very limited.	The design cannot be implemented.	5	1	5
3	The users will not understand what we are doing.	The users become alienated.	3	3	9
4	The computer system is larger than expected.	We are unable to implement it within reasonable resource limits.	2	5	10
5	We have only limited knowledge about the application domain.	The functions and interface of the computer system are irrelevant.	4	3	12
6	We have only limited knowledge about the problem domain.	The database is wrong.	5	1	5
7	We have uneven experience with the technical environment.	There are cooperation problems internally.	4	1	4
8	We do not know the users.	The expectations on our part are unrealistic.	4	3	12
9	The technical environment is unreliable.	The process and product becomes bigger.	2	4	8
10	The time limit is very short.	Nothing relevant is developed by the end of the experiment.	5	4	20

four risk factors we considered to be essential and specific to the project. Having exhausted our independently perceived risks, we considered the taxonomy described earlier and selected six additional factors that were relevant. The consequence for each of these 10 risks was then specified.

The negotiation approach described earlier was used in assigning priorities. Each risk factor was ranked on a severity scale from 0 to 5. After much constructive discussion over the assignment of these figures, each risk factor was also ranked on a probability scale of 0 to 5. Then the severity rank and probability rank of each risk factor were multiplied, giving a compound risk rank.

Once the risk factors received their compound rank, the analysis was simplified, and resolution strategies for the risk factors with the highest product were discussed. In this experiment, we chose risk factors that exhibited compound risk ranks greater than or equal to 12. For each of these important risks, resolution strategies were established and are detailed in Table 3.

This analysis led to the following resolution strategies regarding the prototyping project: First, we should present a number of alternative designs of a computer system to the users by building and evaluating various prototypes; second, we should make a clear commitment to the users; and third we should define and regulate the project by means of very close and specific baselines.

The first prototype

After the first risk analysis, the collaborative team held its first meeting. The first part of this meeting served to make the commitments explicit. The second part focussed on the work of the users. The agenda of the latter part was clearly inspired by the questions and uncertainties that surfaced during the first risk analysis.

To a large extent, the meeting resolved risk factor number 8. Thus, two major risks remained. A closer analysis indicated that the resolution strategies to risk factor numbers 5 and 10 were contradictory. To solve this problem, we decided to develop only one prototype, leaving less attention to risk factor number 5. This decision was based on three arguments. First, risk factor number 10 had obtained a higher product than number 5, i.e. 20 compared to 12. Second, the requirements expressed by the users indicated a small system. Third, the development environment available would force us to make the user interface quite simple. Thus, it would be impossible to develop basically different alternatives.

The prototype was developed with a database software package that included a screen editor, a report generator, and a relational database. This first version was based on a data model with five entities, and it included nine screens and one report. It was evaluated by the collaborative team in a three-hour session, and it uncovered a need for fundamental

Table 3. Initial Resolution Strategies

No.	Risk Factor	Consequences	Resolution Strategy
5	We have only limited knowledge about the application domain.	The functions and interface of the computer system are irrelevant.	Show different alternatives to the users by building and evaluating various prototypes.
8	We do not know the users.	The expectations on our part are unrealistic.	Make a clear commitment.
10	The time limit is very short.	Nothing relevant is developed by the end of the experiment.	Define and regulate the project by means of close and specific baselines.

changes in the data model and the interface design. New functions were also required.

Evolution of the risk factors

After the evaluation of the first prototype, approximately two hours was dedicated to revising the risk analysis. Each of the original risk factors was reconsidered, especially for potential additions or changes to severity or probability rankings.

One major change that occurred was the division of factor number 3 into two separate factors because of two clearly delineated consequences. As before, we analyzed these 11 factors, adjusting the severity and probability rankings according to our experience with the prototype. The results of this analysis are presented in Table 4. In the table, figures that have been revised are marked with either a "+", indicating a higher value or "–", indicating a lower value as compared to the first risk analysis.

According to the revised risk analysis, it was necessary to discuss resolution strategies for the two risk factors that still remained with a compound ranking greater than or equal to 12. These strategies are detailed in Table 5.

This analysis led to two resolution strategies regarding the prototyping project: First, we should design the whole computer-based client management system and convey this design to the users by building and evaluating a prototype with a full user interface and one fully developed function. For example, this meant that all menus and screens were present in the system, but only one branch of the menu tree was operationally functioning from root to leaf. Second, we should continue to define and regulate the project by means of very close and specific baselines.

The second prototype

In the second prototyping cycle, there were no contradictory resolution strategies. This prototype was redesigned according to the experiences gained through the first evaluation. The data model included eight entities, and the prototype comprised 16 screens and five reports.

The evaluation of this version showed that it was close to completion. A few minor bugs appeared, minimal changes to the screens and reports were specified, and two simple functions were added. The users were generally enthusiastic about the prospects reflected by this prototype.

Discussion of Results

Most of the published prototyping management techniques essentially limit the range of projects in which prototyping can be successfully employed. They are merely avoidance mechanisms. The risk analysis approach proposed in this article addresses each of the major managerial problem areas without limiting the range of projects and thus preserves the benefits of prototyping for potentially many more types of projects. Risk factor analysis helps collaborative teams in three important ways: (1) to identify user-developer miscommunication and disagreements that lead to lost collaboration; (2) to point out difficulties in maintaining the project scope in proper proportion to such elements as development technology, problem domain, and application domain; and (3) to evaluate prototype project progress, keeping project management closely aware of the problems that remain before the prototype can be properly delivered into production. Project managers can use a framework of strategies in developing ideal solutions for each severe risk that arises during the progress of the project.

The prototyping project concluded successfully after the evaluation of the second prototype. A third prototype, tested and fully operational and delivered on schedule, passed user acceptance testing and was in service for over a year without further changes. The experience indicates that risk analysis can provide a useful tool to help managers control prototyping projects. It forced the team to consider

Table 4. Revised Risk Factors

No.	Risk Factor	Consequences	Severity	Probability	Product
1	The users only want advice and training on existing tools such as Lotus, OneWrite+, etc.	The users will not cooperate seriously in the design activity.	4	1–	4–
2	The technological resources available in the application domain are very limited.	The design cannot be implemented.	5	1	5
3a	The users will not understand what we are doing.	The users do not know what product they will receive.	4+	3	12+
3b	The users will not understand what we are doing.	The users do not understand their role in the development process.	2–	3	6–
4	The computer system is larger than expected.	We are unable to implement it within reasonable resource limits.	2	4–	8–
5	We have only limited knowledge about the application domain.	The functions and interface of the computer system are irrelevant.	4	2–	8–
6	We have only limited knowledge about the problem domain.	The database is wrong.	5	1	5
7	We have uneven experience with the technical environment.	Cooperation problems internally.	3–	2+	6+
8	We do not know the users.	The expectations on our part are unrealistic.	4	1–	4–
9	The technical environment is unreliable.	The process and product becomes bigger.	2	2–	4–
10	The time limit is very short.	Nothing relevant is developed by the end of the experiment.	5	3–	15–

Table 5. Revised Resolution Strategies

No.	Risk Factor	Consequences	Resolution Strategy
3a	The users will not understand what we are doing.	The users do not know what product they will receive.	Build a prototype with a fully developed interface and one vertical slice of fully developed functionality.
10	The time limit is very short.	Nothing relevant is developed by the end of the experiment.	Define and regulate the project by means of close and specific baselines.

potentially disastrous risks and to closely manage their resolutions. As a result, the project did not suffer major disruptions from any of the managed risks.

Problems in risk analysis iteration

The iterative use of risk analysis was not without its problems. One such problem occurred when a highly expected risk failed to develop in the first cycle. The database software used to construct the prototype was a new release, and earlier there had been reports of corrupted database files. Hence, risk factor nine — "The technical environment is unreliable" — was included. Yet the software performed flawlessly during construction of the first prototype, and given this experience, this risk factor was downgraded in probability from a moderately high "4" to a moderately low "2."

The second prototype contained one full vertical slice of functionality and was considerably more complex than the first. As a result, our activity with the database software increased following the risk analysis revision. During this intense usage, where many unused features of the application generator were involved, the database software corrupted a table, which in turn led to further problems with the application generator. Known "work-arounds" enabled us to avoid any major losses and confirmed the moderately low severity rank assigned to this risk ("2").

It is interesting, however, that the team was inclined to reduce the probability rank of a known risk factor while the conditions that lead to such a risk became less favorable. The two combined facts — no problems had yet been experienced with known faulty software and the usage would increase dramatically — should have led to an *increase* in probability rank. Because the team preferred to rely on its own experience with the product and reduce the factor, a "guardian angel" mentality was revealed that is clearly not recommended. Prototyping project managers should regard the reduction of any probability factors with great caution.

Prototyping process breakdowns

The action research projects referred to above have indicated two kinds of prototyping process breakdowns that relate to the risk analysis. The first has been discussed in detail in the preceding subsection. The second category occurred in a project where the participants identified a critical risk factor but selected an irrelevant resolution strategy. The situation described by this risk arose, and no prototype was delivered (see the Appendix for a description of the Job Locator Service project).

In both kinds of breakdowns, the usefulness of the risk analysis technique originates from its value as a means to understand the nature of the failure. The risk analysis documentation shows how the management of the prototyping processes broke down. Without this documentation, the failures might have been blamed on other factors. To this degree, the risk analysis technique succeeds as a helpful management tool, even though the project outcome may be failure.

Limitations of the Study

From the action research study its not exactly clear what other difficulties managers might face when implementing this approach. Issues that arise include scalability, expertise, and uncertainty in the host environment. In addition, the nature of action research imposes further limitations on an implementation of the approach.

The approach suggested in this article has been used in other action research projects involving medium- and large-scale projects; various combinations of action research, systems development, and risk analysis expertise; and uncertainty in the organizational environments (see the Appendix). This additional work does not indicate that these factors impose problems on the implementation of the approach.

One key limitation of this study is a natural outcome of the action research method used to validate the risk management technique. Action research focuses user attention on a particular problem area and brings external resources to bear on this problem. The Hawthorne effect (Roethlisberger and Dickson 1939) is an intrinsic component of action research, and this means that the routinized use of risk analysis for prototyping management remains undetermined.

Conclusion

Prototyping is a development approach with potentially powerful effects on the quality of information systems analysis and design. Yet major problems have prevented many development organizations from achieving these benefits. Our main finding is that the use of a more effective management approach would facilitate control of prototype development by providing practical mechanisms for defining expectations, assigning resources, signaling pitfalls, and measuring progress. Our evidence suggests that the proposed risk-based project management technique will extend the promised benefits of prototyping to a wider range of the systems projects.

The risk analysis approach presented in this paper includes a practical framework for subjectively planning various prototype strategies and controlling project progress. The framework addresses the analysis and evaluation of risks as well as a taxonomy for evaluating consequences and selecting resolution strategies. The taxonomy is specific to our study but may be a useful supplement in other situations and organizations. Many existing techniques for prototyping management are inextricably linked to a complete development method. This limits their usefulness as well as the extent to which prototyping is applied. The proposed risk analysis approach does not imply the same dependence.

This research opens several avenues for future research. These avenues should extend our understanding of the risk analysis approach. Further action research would be useful for validating the techniques in different kinds of organizational settings and extend our knowledge about different categories of risk factors, consequences, and resolution strategies. This research could also improve the techniques for evaluating consequences, severity, and probability. Furthermore, a practical prototyping management technique enables future researchers to consider the impacts of both risk analysis and prototyping on the broader business agenda. McFarlan's (1981) work successfully applied risk analysis at the portfolio level of information systems project management, while the above action research applied the technique successfully within certain types of projects. Future research might reveal that risk analysis is a critical success factor for managing uncertainty at all levels of the information systems organizations of the future. Other future research might address the possibility of applying prototyping to a broader range of systems development projects, including those risky projects that are essential to business processes, internal work flow, and external client-server and internetworked business partner collaboration.

Acknowledgements

The research behind this article has been partially financed by the Danish Natural Science Research Council, Programme No. 11-0061. We received valuable comments from Ivan Aaen, Heinz K. Klein, Lars Mathiassen, Peter A. Nielsen, Carsten Sørensen, and several anonymous reviewers (particularly one reviewer who made extensive helpful comments). Finally, we wish to thank the Samaritan Counseling Center in Endicott, New York and, especially, Doug Beattie and Kay Hollister who participated in the action research project.

Endnotes

1 This lowering of planning and control emphasis is similarly suggested by McFarlan (1981) in situations of low project structure, i.e. prototyping, accompanied by high (unfamiliar) technology.

2 Action research uses change as a means to study complex social processes. The researcher intervenes in the

research setting and studies the effect of the intervention. Action research has its foundations in the post-positivist tradition of the social sciences (Hult and Lennung 1980; Susman and Evered 1978). Lewin (1951) pioneered action research in his study of field theory, and Checkland (1981) introduced the method to the information systems research community in developing his Soft Systems Methodology.

Although most methodologies suggest four to six stages, Blum (1955) explains the essence of action research as a two-stage process. First, the *diagnostic stage* involves a collaborative analysis of the social situation by the researcher and the subjects of the research. Hypotheses are formulated concerning the nature of the research domain. Second, the *therapeutic stage* involves collaborative change experiments. In this stage, changes are introduced, and the effects are studied, contributing both to the practical concerns of an immediate problem situation and to the goals of social science (Rapoport 1970).

3 The philosophical footings of action research rise from hermeneutics, existentialism, and phenomenology (Susman and Evered 1978). As a scientific method, action research is clearly post-positivist and is evaluated on different criteria than the positivist standards of falsifiability, repeatability, and generalization (Gummesson 1988).

There are four essential criteria for evaluating action research (Susman and Evered 1978). First, the researchers must intervene into the subject under study. The success of the research depends on engagement rather than detachment. Data are collected with participant observation, and this develops the empathy, the values exchange, and the role reversals that make researchers' knowledge really useful and accepted by the subjects.

Second, the project must be collaborative, and the subjects must be dynamically involved in determining the directions of the project. The researcher's role is not one of prediction in a passive world, but one of making things happen in an interactive world.

Third, the knowledge goals should be interpretive and framed as "understanding" rather than "explanation." Although the theoretical constructs under empirical testing are complex and multivariate, they gain scientific usefulness as the conceptual "point of departure" for intervention in other settings; i.e. the theory must be interpreted and adapted in order to achieve construct validity in each new organizational setting.

Fourth, the action research project must yield a solution to the immediate problem situation. Action research develops learning from experience, which should be disseminated within the organization and published to the scientific community. This learning can lead to further action and major positive effects in diverse organizational settings.

4 The knowledge produced by the counseling center project is valid under the criteria of action research. First, the researchers, acting as the prototype developers, clearly intervened in the subject under study, their knowledge was accepted by the subjects and the data collection method was participant observation. Second, the project was collaborative and the subjects were dynamically involved in determining the directions of the project. Both the users and the developers were aware of the practical and research aspects of the project and the concepts and actions developed interactively. Third, the project aimed at developing deeper understanding of the prototype project management process. The researchers bypassed explanatory themes and experimental rigor in favor of situational interpretations of the behavior of human organizations in relation to the concepts. Fourth, the action research project led to a solution to the immediate problem situation, thus validating its learning from experience.

References

Alavi, M. "An Assessment of the Prototyping Approach to Information Systems Development," *Communications of the ACM* (27:6), June 1984, pp. 556–563.

Badenhorst, K. and Eloff, J. "Computer Security Methodology: Risk Analysis and Project Definition," *Computers & Security* (9), June 1990, pp. 339–346.

Bang, S., Efsen, S., Hundborg, P., Janum, H., and Schultz, C. *Development of Quality Software*, Master's thesis, Institute for Electronic Systems, Aalborg University, Aalborg, Denmark, 1991.

Baskerville, R.L. "Risk Analysis as a Source of Professional Knowledge," *Computers & Security* (10:8), December 1991, pp. 749–764.

Baskerville, R.L. "Semantic Database Prototypes," *Journal of Information Systems* (3:2), 1993, pp. 119–144.

Baskerville, R.L. and Stage, J. *Developing the Prototype Approach in Rapid Systems Modelling*, technical report, R 91-35, Institute for Electronic Systems, Aalborg University, Aalborg, Denmark, September 1991.

Blum, F. "Action Research – A Scientific Approach?" *Philosophy of Science* (22), January 1955, pp. 1–7.

Boar, B.H. *Application Prototyping: A Requirements Definition Strategy for the 80s*, John Wiley and Sons, New York, NY, 1984.

Boehm, B.W. "A Spiral Model of Software Development and Enhancement," *Computer* (21:5), May 1988, pp. 61–72.

Boehm, B.W. *Software Risk Management,* IEEE Computer Society Press, Washington, DC, 1989.

Boehm, B., Gray, T., and Seewaldt, T. "Prototyping versus Specifying: A Multiproject Experiment," *IEEE Trans. Software Eng.* (SE-10:3), May 1984, pp. 290–303.

Brooks, F.P. "No Silver Bullet. Essence and Accidents of Software Engineering," *Computer* (20:4), April 1987, pp. 10–19.

Budde, R., Kuhlenkamp, K., Mathiassen, L., and Züllighoven, H. (eds.). *Approaches to Prototyping*, Springer-Verlag, Berlin, 1984.

Budde, R., Kautz, K., Kuhlenkamp, K., and Züllighoven, H. "What is Prototyping?" *Information Technology & People* (6:2–3), 1992, pp. 89–95.

Burns, R. and Dennis, A. "Selecting the Appropriate Application Development Methodology," *Data Base,* Fall 1985, pp. 19–23.

Checkland, P. *Systems Thinking, Systems Practice,* John Wiley and Sons, Chichester, 1981.

Coad, P. and Yourdon, E. *Object-Oriented Design*, Prentice-Hall, Englewood Cliffs, NJ, 1991.

Connell, J.L. and Schafer, L.B. *Structured Rapid Prototyping,* Yourdon Press, Englewood Cliffs, NJ, 1989.

Courtney, R. "Security Risk Assessment in Electronic Data Processing," *AFIPS Conference Proceedings NCC* (46), 1977, pp. 97–104.

Davis, G.B. "Strategies for Information Requirement Determination," *IBM Systems Journal* (21:1), 1982, pp. 4–30.

Duke, E., Brumbaugh, R., and Disbrow, J. "A Rapid Prototyping Facility for Flight Research in Advanced Systems Concepts," *Computer* (22:5), May 1989, pp. 61–66.

Ehn, P. *Work-Oriented Design of Computer Artifacts*, Arbetslivscentrum, Stockholm, 1988.

Fisher, R. *Information Systems Security*, Prentice-Hall, Englewood Cliffs, NJ, 1984.

Fitzgerald, J. "Edp Risk Analysis for Contingency Planning," *EDP Audit Control and Security Newsletter* (I), August 1978, pp. 6–8.

Galliers, R. and Land, F. "Choosing Appropriate Information Systems Research Methodologies," *Communications of the ACM* (30:11), November 1987, pp. 900–902.

Grønbæk, K. "Supporting Active User Involvement in Prototyping," *Scandinavian Journal of Information Systems* (2), 1990, pp. 3–24.

Gummesson, E. *Qualitative Methods in Management Research*, Chartwell-Bratt, Lund, Sweden, 1988.

Gupta, R., Cheng, W., Gupta, R., Hardonag, I., and Breuer, M. "An Object-Oriented VLSI CAD Framework: A Case Study in Rapid Prototyping," *Computer* (22:5), May 1989, pp. 28–37.

Hult, M. and Lennung S.-O. "Towards a Definition of Action Research: A Note and Bibliography," *Journal of Management Studies* (17), May 1980, pp. 241–250.

Iivari, J. and Karjalainen M. "Impact of Prototyping on User Information Satisfaction During the IS Specification Phase," *Information & Management* (17), August 1989, pp. 31–45.

Lantz, K.E. *The Prototyping Methodology*, Prentice-Hall, Englewood Cliffs, NJ, 1988.

Larsen, T., Liebmann, S., Millum, C., Solberg, H., and Tolstrup, F. *Spiralmodellen i en praktisk systemudvikling (The Spiral Model Used in a Practical System Development Project)*, Master's thesis, Institute for Electronic Systems, Aalborg University, Aalborg, Denmark, 1990.

Lewin, K. *Field Theory in Social Science*, Harper & Bros, New York, 1951.

Luqi. "Software Evolution Through Rapid Prototyping," *Computer* (22:5), May 1989, pp. 13–27.

Mackenzie, K. "Virtual Positions and Power," *Mangement Science* (32:5), May 1986, pp. 622–643.

Martin, J. *Information Engineering, Book II*, Prentice-Hall, Englewood Cliffs, NJ, 1990.

Mason, R. and Carey, T. "Prototyping Interactive Information Systems," *Communications of the ACM* (26:5), May 1983, pp. 347–354.

Mathiassen, L. and Stage, J. "The Principle of Limited Reduction in Software Design," *Information Technology & People* (6:2–3), 1992, pp. 171–185.

Mathiassen, L., Munk-Madsen, A., Nielsen, P.A., and Stage, J. "Rapid Systems Modelling: The Soul of a New Methodology," in *Proceedings of the Fourteenth Information Systems Research Seminar in Scandinavia*, O. Forsgren (ed.), University of Umeå, Institute for Information Processing, Umeå, Sweden, 1991, pp. 421–441.

Mathiassen, L., Seewaldt, T., and Stage, J. "Prototyping and Specifying: Principles and Practices of a Mixed Approach," *Scandinavian Journal of Information Systems* (7:1), 1995, pp. 55–72.

McFarlan, F.W. "Portfolio Approach to Information Systems," *Harvard Business Review* (59:5), September–October 1981, pp. 142–150.

National Bureau of Standards. *Guideline for Automatic Data Processing Risk Analysis*, Federal Information Processing Standards Publication FIPS 65, Washington, DC, August 1979.

Naumann, J. and Jenkins, A. "Prototyping: The New Paradigm for Systems Development," *MIS Quarterly* (6:3), September 1982, pp. 29–44.

Parker, D. *Computer Security Management*, Reston, Reston, VA, 1981.

Parnas, D. "Software Aspects of Strategic Defense Systems," *Communications of the ACM* (28:12), December 1985, pp. 1326–1335.

Rapoport, R. "Three Dilemmas of Action Research," *Human Relations* (23), 1970, pp. 499–513.

Roethlisberger, F. and Dickson, W. *Management and The Worker*. Harvard University Press, Cambridge, MA, 1939.

Saarinen, T. and Vepsäläinen, A. "Managing the Risks of Information Systems Implementation," *European Journal of Information Systems* (2:4), 1993, pp. 283–295.

Saltmarsh, T. and Browne, P. "Data Processing – Risk Assessment," in *Advances in Computer Security Management*, Vol. 2., M. Wofsey (ed.), J. Wiley, Chichester, 1983, pp. 93–116.

Susman, G. and Evered, R. "An Assessment of the Scientific Merits of Action Research," *Administrative Science Quarterly* (23), December 1978, pp. 582–603.

Vonk, R. *Prototyping. The Effective Use of CASE Technology*, Prentice-Hall, New York, 1990.

Yoo, Y. "An Investigation of Group Development Process in 'Virtual' Project Team Environments," in *Proceedings of the Sixteenth Annual International Conference on Information Systems*, Amsterdam, December 10-13, 1995, p. 346.

Yourdon, E. *Managing the System Life Cycle*, Yourdon Inc., New York, 1982.

Zelkowitz, M. (ed.). *ACM SIGSOFT Workshop on Rapid Prototyping*, Columbia, MD, April 19–21, 1982.

About the Authors

Richard L. Baskerville holds a B.S. degree in management from the University of Maryland, an M.Sc. in analysis, design, and management of information systems, and a Ph.D. in systems analysis from The London School of Economics. Before beginning his doctoral studies he experienced 10 years of practice as a systems engineer, chiefly working with decision support and telecommunications systems in universities and government agencies. He is currently associate professor at the School of Management, State University of New York at Binghamton. His research areas are security of information systems and design methods. His credentials include the CCP from The Institute for Certification of Computer Professionals, and the C.Eng. from the British Engineering Council. He is a member of the ACM, the BCS, and IFIP Working Group 8.2.

Jan Stage received the M.Sc. degree in computer science from Aalborg University, Denmark, in 1984 and the Ph.D. degree in informatics from the University of Oslo, Norway, in 1989. He is currently associate professor in software engineering and information systems at Aalborg University, Department of Computer Science. His research interests include theoretical and methodological aspects of systems development, especially development and evaluation of new methods and techniques for analysis and design. He is also teaching analysis and design courses for software professionals. He is a member of IFIP Working Group 8.2.

Appendix

The counseling center project was a professional project conducted under rigorous action research guidelines. Additionally, there were four other action research projects carried out by graduate students at Aalborg University in Denmark and Binghamton University in the USA. The distinctive aspects of these studies and experiences are highlighted below. The projects are mentioned here because they provide further insight into the factors that do not seem to influence the validity of our approach. Factors like project size, developer culture, and even project success do not seem to limit the usefulness of the approach.

Project Management Tool

This project was conducted at Aalborg University to develop an interactive tool for the COCOMO software development method. The project involved five graduate students. Unlike the counseling center project, this project only involved one of the authors and only as a user. Over a three-month period, they spent 1484 person-hours (nine person-months) developing two prototypes with 4000 and 5500 lines respectively of Modula-2 code combined with graphical interfaces developed in two different tools. The project is documented in Larsen, et al. (1990). The team made the following observations regarding risk analysis:

1. The definition of resolution strategies relied partly on their experience and intuition, and partly on their evaluation of specific alternatives.

2. Risk analysis proved most relevant in problem-solving situations, as opposed to routine or problem-definition situations.

3. Risk resolution activities often changed their conception of the specified risks since they became able to describe them more precisely.

Network Administration System

The purpose of this project was to develop a network administration system for the computer center at the Computer Science Department at Aalborg University. The administration system was developed at a turbulent point in time because the department was changing from a small number of centralized computers to a fully networked architecture based on a considerably larger number of workstations. The project group consisted of five graduate students, and the users were the computer center manager and his employees. The project is documented in Bang, et al. (1991).

This project produced a distinctively large system compared to typical evolutionary prototyping projects such as the counseling center or the project management tool. The size of the system required 15,000 lines of code, programmed in C and Perl. In addition, the team developed a graphical user interface in XView. There were 800 pages of documentation. Five participating students spent 2568 hours (16 person-months) on the project. Out of this, management accounted for 406 hours, including 110 hours spent specifically on risk analysis. Concerning risk analysis, they made the following observations:

1. The resources dedicated to risk analysis were a good investment. The analysis required only a minimum of effort, and major risks were identified and resolved early.

2. The development team achieved a good understanding of the project at an early point because they had to examine the whole project and not just the next activities.

3. The technique forced them to focus on the highest risk items as opposed to postponing the most difficult tasks.

4. The resolution of a risk often made that risk develop into several new risks that were each of a more specific nature.

5. The most difficult part of risk analysis was the process of estimating the consequences of a risk.

6. The use of the technique oriented the team toward risk assessment. Informal risk assessment became a natural part of their general approach to problem solving.

Commercial Laundry Service

This project was conducted at Binghamton University. Its purpose was to develop an interactive customer database system for a commercial laundry and diaper service. The team of four graduate students developed one semantic database prototype and two application prototypes using R:Base running under a Novell network. The project is distinguished by a different cultural setting; unlike the Aalborg projects this team involved American business students instead of Danish computer science students. This team conducted two risk analysis sessions and identified their major risks as "unfamiliar technology" and "not enough time." Their resolution strategies included an early decision to limit the prototype scope to exclude supplier and transaction functions initially requested by the users. The team developed and successfully delivered their prototype system to their customer on schedule.

Job Locator Service

This project, also at Binghamton University, developed an interactive system to track students seeking jobs and prospective employers seeking university student applicants. The systems requirements outlined needs to maintain applicant and employer profiles, track employment events, and provide a statistical analysis of the activities of the job locator work group. The development team consisted of four graduate students and developed one semantic database prototype in a Novell network using R:Base and two application prototypes using REXX and SQL/DS on an IBM mainframe computer.

This project is distinguished by its different outcome. The team did not meet its minimum goals, and the prototype system was only partially working at the end of the project. The prototype and documentation had to be handed over to a professional computer services staff for further development, where it was essentially shelved because of lack of project funds.

Team diaries and documentation clearly indicate why the project failed. The team risk analysis highlighted the most serious risk as "may run out of time before completing the prototype." Rather than selecting a reasonable resolution strategy (such as limiting the scope or monitoring the progress), the team resolved to "work harder and faster." But the team had firm boundaries on the time they could dedicate to the project, and "working faster" did not succeed as a strategy. The team inevitably ran out of time and could only demonstrate partially operating prototypes to the users prior to packaging the documentation and delivering an inoperative and incomplete system to their customer.