INTERNATIONAL DESIGN CONFERENCE - DESIGN 2002
Dubrovnik, May 14 - 17, 2002.

COLLABORATIVE ENGINEERING: 'TOWARDS DESIGN GUIDELINES FOR RISK MANAGEMENT IN DISTRIBUTED SOFTWARE DEVELOPMENT'

Jürgen van Grinsven and Gert-Jan de Vreede

Keywords: collaborative engineering, software risk, distributed software development, project management, process improvement, design guidelines, organizational risk, and environmental risk

1. Introduction

Managers of software development projects usually perceive the word ‘risk’ in a negative way. There are two reasons that underlie this perception. The first is that many software projects fail because they are over budget, and the second is that their implementation is not always successful [Schmidt et al., 2001]. The high failure rates associated with software development projects indicate that organizations need to improve their ability to identify and manage the project risks. Literature shows that, in general, there are three important risk dimensions concerning software development: the technical, organizational, and environmental dimension [Ropponen, 2000; Sherer 1995]. Technical risks result from uncertainties in both tasks and procedures. Organizational risks result from poor information and decision-making, and environmental risk results from poor coordination of interpersonal actions. Over the past thirty years, researchers have made meaningful efforts to manage the technical dimension. Unfortunately, our understanding of the organizational and environmental dimension has remained fragmented and largely anecdotal [Keil et al., 1998]. On top of this, the technological, organizational and environmental landscape has changed [Willcocks et al., 1999; Sherer, 1995].

Nowadays, companies have found advantages in distributing their software development activities over geographic dispersed locations. These distributed activities require intense collaboration between groups that are scattered over those locations. These activities are subject to all the risks resulting from group dynamics, interactions, coordination and communication [Ewusi-Mensah, 1997]. While group-oriented software development has always provided a source of risk, several trends increased the importance of these risks. First, because software is developed in a distributed manner groups need different channels to communicate. Second, both inter- and intra group relations are (culturally) different e.g. they become more formal. Finally, the actions and interactions taken by distributed groups are more difficult to coordinate.

Advocates of software risk management claim that by assessing the threats to success, action can be taken to prevent them from occurring. Assessment usually consists of three activities: identifying, analyzing and prioritizing the risks. A large number of threats to success can be found in literature [see e.g., Barki, 2001; Boehm 1991]. Yet to our knowledge no systematic attempts have been made to assess the organizational and environmental risks in distributed software development activities. In this paper we report on four case studies into the organizational and environmental risk factors of distributed software development.

In this paper, we report on a multiple case study. The objective of this study was twofold. First, to identify the most important organizational and environmental risk factors in distributed software

development, as perceived by expert project managers in distributed software development. Second, to build towards design guidelines to help avert these risks from occurring.
The remainder of this paper is structured as follows: in the next section relevant literature in the area of software risk management is discussed. The research approach and design of the study is elaborated in section three. In section four, we present the results of our study and in section five we discuss them. This paper concludes with a summary of the main findings, a description of the limitations of the study, and an outline of future research.

2. Background

Building upon his experience in the 1980s, Boehm [1991] identified ten risk items to anticipate to problems that may arise during software development. These risk items, including: personnel shortfalls, unrealistic schedules and budgets, developing the wrong software functions, developing the wrong user interface, gold-plating, continuing stream of requirement changes, shortfalls in externally-performed tasks, shortfalls in externally-furnished components, real-time performance shortfalls, and straining computer science capabilities, may help to identify threats that affect cost, schedule and quality [Sherer, 1995]. But in the meantime, the technological, organizational and environmental landscape has changed. Nowadays, software development activities are distributed over several dispersed locations. As such, the risk items mentioned by Boehm may no longer be perceived the most important and need reexamination.
A couple of studies have classified risk items along several dimensions [see e.g. Sherer, 1995; Keil et al., 1998]. Sherer [1995] first classified twelve risk items into three components: development, use and maintainability. The component ‘development’ consists of three risk items: personnel, schedule/cost, and process. The component ‘use’ consists of: functionality, performance, reliability, safety, security, and financial. The component ‘maintainability’ consists of: correctability, adaptability, and portability. Then Sherer systematically investigated each component following three dimensions: the technical, organizational, and environmental dimensions. This study provides insights into typical component variations in each dimension. Furthermore it showed that the environmental risk dimension has grown in significance and that there is a challenge to develop risk management procedures for this dimension. Keil et al. [1998] present a risk categorization framework based on two dimensions: perceived level of control, and perceived relative importance of the risk. They assembled panels of experienced software project managers in different parts of the world and asked them to identify specific risk factors and then rank them in terms of their importance. One of the strengths of this approach is that it provides a higher-level framework for thinking about four distinct types of software project risk: customer mandate, scope and requirements, execution, and environment. Maybe the most interesting finding of this study is that risks thought to be most important are often not under the project manager’s direct control.
From the above, it follows that while organizational and environmental risks have been recognized, no systematic attempts have been made yet that provide a detailed analysis of these risk factors in distributed software development. Studies also show a lack of consensus about the risk factors affecting the organizational and environmental dimension. As such, we believe that the organizational and environmental risk factors in distributed software development need to be reexamined. Our goal is to expand the knowledge of the research community in this area, by providing a two dimensional framework. This framework allows us to delineate the risk factors affecting distributed software development.

3. Method

Our study used a multiple case study drawn from a large consumer electronics company. According to Yin [1994] a case study is intended to examine a contemporary phenomenon in its natural setting where the boundaries between the phenomenon and its context are unclear. Following [Benbasat et al. 1987], we decided to use a case study because (1) risks in distributed software development represent a complex phenomenon that needs to be studied in its natural setting, and (2) there is a lack of similar studies on developing design guidelines to help avert these risks from occurring in distributed software development.

We included four cases in our study. Two highly innovative cases, projects that had not been done before, and two traditional cases, where experience had been gained in similar projects. These cases were chosen to get a mix of highly innovative and traditional distributed software development projects. The focus of our research is organizational and environmental risks in distributed software development.

3.1 Data collection and Analysis

Embedded in the case studies the following strategies were used: a pilot study and expert panels. The list of risk factors, which was used in our survey, was built from existing organizational and environmental risk factors.

The twenty-two respondents from the pilot study, who represented different countries and backgrounds, selected numerous items from our list as being important to project success in distributed software development. Then, face-to-face in-depth interviews were held with the respondents to collect additional data about the reasons why certain factors were selected and others not. Each interview was conducted using a set of both open-ended and closed questions. The response rate was 100%; because the pilot study resulted in only minor changes, the data was included in our analysis.

Four expert panels, each consisting of eighteen project managers coming from six different countries represented four to nine years of experience in distributed software development. These experts narrowed the selected risk factors down by assessing the relative contribution of each risk factor to the success of the projects. Two separate panels assessed the innovative cases and the two other panels, also separated, assessed the traditional cases. The panels used a five-point Likert scale, where five represented the highest perceived contribution to project success. After this, all the experts were interviewed to elaborate on their choices. Finally, nine risk items remained: four concerning social-dynamic risk factors and five concerning system-technical risk factors. Figure 1 shows the three steps taken in this research.

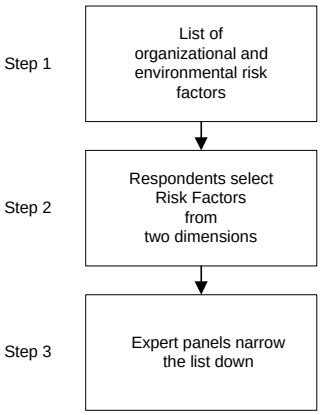

Figure 1. Method used in this study

4. Results

Having analyzed the data from our case studies we found nine risk factors that affect distributed software development. The nine risk factors are presented in table 1 in decreasing order of their perceived contribution to project success. One of the interesting findings that emerged from our analysis was that while our panelists found that four risk factors were very difficult to assess and then only subjectively, they found that the other five risk factors could be assessed more or less objectively. The four risk factors that are difficult to assess are mapped into a group called: social-dynamic risk factors and the five risk factors that can be objectively assessed are mapped into a group called: system-technical risk factors.

Table 1. Risk factors ordered by their perceived importance

	Risk factor	Perceived contribution to project success
Social-dynamic Risk factors	Lack of commitment	4.9
	Lack of trust	4.8
	Lack of leadership	4.6
	Lack of accountability	4.4
System-technical Risk factors	Poor coordination	4.3
	Lack of required abilities	4
	Inappropriate role definition	3.9
	Lack of familiarity with technology	3.6
	Lack of process	3.1

5. Discussion

In distributed software development, groups are geographically separated. Our results indicate that this changes the perceived contribution to project success of both social-dynamic, and system-technical risk factors. If we compare our results with Sherer [1995] and Keil et al. [1998] we find that several system-technical and social-dynamic risk factors are missing from their lists. These factors were likely introduced because of the distributed nature of our software development context.

5.1 Social-dynamic risk factors

Although commitment seems an obvious risk factor in software development, the reviewed literature only distinguishes between top management commitment and user commitment [see e.g. Barki, 2001; Keil et al., 1998; Ewusi-Mensah, 1997]. While, in line with this literature, our case results found support for both types of commitment, they also indicated that a strong commitment of the separated groups becomes increasingly important to project success. According to one of the experts: ‘it is more difficult to commit people who are not in the same geographic location’.

Another important risk factor in distributed software development was trust. Analysis of our case data revealed that in building trust, distributed groups often relied on each other’s technical expertise. In line with Willcocks et al. [1999] our analysis revealed that building trust cannot be done overnight and that processes have to be in place to allow the relationship to develop.

Data analysis revealed that management can provide leadership by assigning a project leader to a team, or the project teams themselves can provide leadership. As one of the experts noticed: ‘before they commit themselves they have to trust you, then you have to prove that you can deliver what you promised’. This finding is in line with Keil et al. [1998] who argue that: project managers must take reasonable steps to ensure that they have the support and commitment needed to deliver a successful project.

Our data showed that accountability was perceived to be important to project success in distributed software development. This finding is in line with Ewusi-Mensah [1997] who argues that cooperation between team members is critical to project success. Moreover, Sherer [1995] who argues that poor cooperation can prevent teams from effectively working together. By holding individuals accountable for what they are doing, project managers can prevent people to engage in social loafing and coast on the group’s effort.

5.2 System-technical risk factors

In contrast to many risk lists [see e.g. Ropponen and Lyytinen 2000; Willcocks et al., 1999] coordination appears as a separate risk factor in our list for distributed software development. We listed coordination as a separate factor because project managers indicated that in distributed software development the costs to coordinate activities increase dramatically. This is in line with Sherer [1995] who argues that a large part of the software development process is spent on coordination of

interpersonal actions.
Our data indicates that abilities are important to project success. It also indicates that in distributed software development three types of skills are required: technical expertise, problem solving, and interpersonal skills. This finding is somewhat similar to Keil et al. [1998] who argue that the required knowledge/skills involve: relationship management, trust building, and political skills. Our results furthermore indicate that technical expertise is the most important skill upon which individuals build trust. This finding can be influenced by the fact that we conducted this research in a highly technical environment.
In line with Keil et al. [1998] and Ewusi-Mensah [1997] we found that roles are important to project success, but in contrast to them we list 'roles' as a separate item on our risk list. The reason for this is that the notion of roles permits us to create repeatable processes in which tasks can be associated with roles and not to the individuals who perform them. This is especially useful in distributed software development because it can decrease coordination costs.
One of the most cited pitfalls is lack of familiarity with technology [Schmidt et al., 2001; Barki et al., 2001; Boehm, 1991]. In line with Ropponen and Lyytinen [2000] we also found that tailored project management systems have a better fit to organizational needs and were more easily accepted and used. We also found that risk managers did not have appropriate tools to tailor their risk management efforts to distributed software development.
In line with Willcocks et al. [1999] and Sherer [1995] we found that processes have to be in place to manage the interpersonal relationships involved in distributed software development. In addition to Willcocks et al. [1999] who emphasize single site relationship management techniques like co-location our results indicate that it is possible to maintain these relationships with proper information technology (IT). Our findings also indicated that IT is able to prolong the time between face-to-face contacts but cannot replace them.

6. Conclusions

6.1 Summary of the main findings

In this research we have sought answers to the question: What are the organizational and environmental risk factors concerning distributed software development? Using the data from our study, we found nine risk factors that affect distributed software development. Our analysis furthermore revealed that the proposed two-dimensional framework consists of both social-dynamic and system-technical risk factors. We found evidence that suggests that the system-technical risk factors can be more or less objectively assessed and that the social-dynamic risk factors are much more difficult to assess, and then only subjectively. Our analysis also indicated that social-dynamic risk factors are more critical to success in distributed software development than the system-technical risk factors.

6.2 Limitations of the research

There are some limitations to be taken into account when interpreting our findings. First there is a possible bias in using the data from one large multinational in consumer electronics, though our sample was representative in terms of distributed software development. Second, our study did not use a mechanism to control the influence of historical data regarding distributed software development incidents. Third, the subjectivity of the risk management experts is itself a source of risk. The generalization of our research is limited to the context in which it was undertaken.

6.3 Future research

Future research might pay attention to the design of repeatable risk management processes that can decrease coordination costs in distributed software development.

Acknowledgements

We are grateful to the people of the consumer electronics company who made it possible to do our research. Furthermore we would like to thank all the people who participated in our four case studies. Unfortunately, their names cannot be mentioned for reasons of confidentiality.

References

Barki, H., Rivard, S. ,and Talbot, J. "An Integrative Contingency Model of Software Project Risk", Journal of Management Information Systems, volume 17, 2001, (37-69)

Benbasat, I., Goldstein, D.K., and Mead, M "The case research strategy in studies of information systems", MIS Quarterly, 11, 3, (369-386)

Boehm, B.W. "Software risk management: principles and practices", IEEE Software, 1991, (32-41)

Ewusi-Mensah, K. "Critical Issues in Abandoned Information Systems Development Projects", Communications of the ACM, volume 40, 1997, (74-80)

Keil, M., Cule, P., Lyytinen, K., and Schmidt, R.,"A framework for Identifying software project Risk", Communications of the ACM, volume 41, 1998, (76-83)

Ropponen, J., and Lyytinen, K., "Components of Software Development Risk: How to Address Them? A project Manager Survey", IEEE transactions on software engineering, volume 26, 2000, (98-112)

Schmidt, R., Lyytinen, K., and Keil, M., "Identifying Software Project Risks: An International Delphi Study", Journal of Management Information Systems, volume 17, 2001, (5-36)

Sherer, S., "The Three Dimensions of Software Risk: Technical, Organizational, and Environmental", Proceedings of the 28th Annual Hawaii International Conference on System Sciences, Los Alamitos, CA: IEEE Computer Society Press, 1995, (369-378)

Willcocks, L.P., Lacity, M.C., and Kern, T.,"Risk mitigation in IT outsourcing strategy revisited: longitudinal case research at LISA", Journal of Strategic Information Systems 8, 1999, (285-314)

Yin, R. K., "Case Study Research, Design and Methods", SAGE Publications U.S.A, 1994

Drs. ing. Jürgen van Grinsven
Delft University of Technology, Faculty of Technology Policy and Management, Section Systems Engineering
Jaffalaan 5, P.O. Box 5015, 2600GA Delft, The Netherlands
Telephone: +31152783730
Telefax: +31152783429
Email: jurgeng@tbm.tudelft.nl,
url: www.tbm.tudelft.nl/webstaf/jurgeng/