TOWARDS A DESIGN THEORY FOR SOFTWARE PROJECT RISK MANAGEMENT SYSTEMS

Research-in-Progress

Mazen El-Masri
HEC Montréal
3000, Côte-Sainte-Catherine
Montréal (Québec)
Canada H3T 2A7
mazen.el-masri@hec.ca

Suzanne Rivard
HEC Montréal
3000, Côte-Sainte-Catherine
Montréal (Québec)
Canada H3T 2A7
suzanne.rivard@hec.ca

Abstract

This paper outlines the design principles for software project risk management systems. The artifact accounts for the interrelationships among factors belonging to four risk dimensions: project traits, undesirable events, management practices, and project outcomes. We conducted a case survey of 60 software projects and observed patterns of interplay among risk components. Four patterns were found: the multiplicative effect of project traits, the sequentiality effect of undesirable events, the presence of a third variable, and the tradeoff when implementing risk management practices. Those patterns were then used to derive three design principles for the IT-artifact – the association, the regulation, and the simulation principles. To validate those principles, we intend to prototype the IT-artifact and intervene in an organizational setting. If the artifact is judged to be useful, it presents a practical solution to project managers by providing a more accurate assessment of risk exposure than existing computational techniques.

Keywords: Risk management, Software project management, Risk, Case study, Case survey, Design Science, Content Analysis.

Introduction

Software project risk management has been acknowledged, both in research and practice, as playing a fundamental role in the success of software projects. The IS literature describes various practices that managers can adopt to assess project risk and mitigate negative impacts on project success. For instance, the analysis of key decisions was found to be an effective practice for mitigating requirement risk (Ropponen and Lyytinen 2000), and piloting, prototyping, and staging were found to be effective practices for mitigating risks such as technical complexity and project size (Benaroch et al. 2006).

Software project risk management is often described as comprising two sequential steps: risk assessment and risk control (Boehm 1991). In the first step, risk factors are identified, analyzed, and prioritized. In the second step, activities are planned and executed and project risk is monitored. This sequential process is repeated over the course of a project. In the past decade, however, IS researchers have underscored the intricate and dynamic nature of software project risk (e.g., Alter and Sherer 2004; Schmidt et al. 2001), suggesting that risk assessment and control activities are intertwined. According to Alter and Sherer (2004), there seem to be adverse interactions between initial and emergent risk, management practices, and project objectives. While no extant theoretical framework exists to explain such interrelationships, evidence from case studies shows that different components of software project risk, undesirable events, and management practices interact over time, causing a dynamic snowball effect that diverts software projects from achieving their objectives (see Sicotte and Paré 2010; Warkentin et al. 2009).

In practice, software providers offer tools designed to assist managers and other involved project participants in risk assessment and planning. These tools often provide basic functionalities that support the risk management process standardized by the Project Management Institute (PMI 2004). For example, RiskyProject Professional (http://www.intaver.com/) and @RISK (http://www.palisade.com/) offer generic functionalities that create risk registers, assign risk to tasks and resources, generate a probability-impact matrix, and perform Monte Carlo simulations. However, the existing tools do not account for interactions between risk components. For instance, the probability-impact matrix feature in @RISK treats risk factors as independent of one another in evaluations of project risk exposure.

We address this issue in two ways. First, we distinguish the various patterns in which components of risk interrelate by conducting a case survey of sixty IT project cases studies. To analyze these cases, we adopt a middle-ground analytical approach by imposing an existing theoretical model as our coding scheme. Second, we interpret the patterns and derive a set of design principles that can govern the construction of an IT artifact for software project risk management. This work is framed by the design science perspective of Hevner et al. (2004) and Gregor and Jones (2007).

The paper first describes our analytical approach and coding scheme, along with the software project risk conceptualization that we adopted. It then presents the patterns of interrelationships among the risk components that we identified during case analysis. These patterns are then examined in order to derive the design principles of the software project risk management artifact. The paper then briefly discusses the design principle's evaluation method and future work and offers some concluding remarks.

Identifying Patterns of Interrelationships between Risk Components

A case survey method was judged to be a suitable strategy for identifying patterns of interrelationships between risk components. Case surveys are useful as a research method when a coding scheme is applied to transform the rich data found in multiple case studies into quantitative data (Newig and Fritsch 2009). The advantage of this method lies in its ability to generalize through the aggregation of a large number of case studies (Larsson 1993). It is most appropriate when the literature contains a sufficient number of relevant case studies (Yin and Heald 1975) and when the phenomenon under study is complex and dynamic (Larsson 1993).

The cases were identified by searching the IS literature for articles that include descriptions of software project cases. ABI/INFORM Global, Business Source Complete, and IEEE Xplor were the databases searched. We also searched prominent trade publications such as IEEE Spectrum, Computerworld, and CIO Magazine. The inclusion criteria were: (1) that the study had a software project as the unit of analysis and (2) that the article provided enough information to describe the project's context and how it was conducted. We stopped analyzing new cases once we reached theoretical saturation. In total, 60 cases

were analyzed (list available upon request).

To provide a basis for our analysis of the cases, we assessed the existing conceptualizations of software project risk in the IS literature. Three main conceptualizations were found. The first conceptualization views risk as a set of factors – project traits or events – that pose a threat to the success of software projects (e.g., Jiang et al. 2007; Na et al. 2007). The second conceptualization views risk in terms of risk exposure, defined as the probability of occurrence of undesirable events and their consequences on project outcomes (e.g., Barki et al. 2001; Boehm and Bhuta 2008). The last conceptualization views risk as negative variations in project outcomes as a result of management's failure to respond to threats (e.g., Benaroch et al. 2006; Clemons 1995). One model of software project risk (see Figure 1) takes into consideration all three conceptualizations; it is comprised of four dimensions: project traits, undesirable events, management practices, and project outcomes (El-Masri and Rivard 2010). According to the authors, software project risk is the probability that project traits (e.g., project size) will lead to undesirable events (e.g., effort underestimation) that have negative consequences on project outcomes (e.g., schedule overrun). Management practices (e.g., staging or adding contingency time) are applied to mitigate project traits or dissociate undesirable events from adverse project outcomes (ibid).

We therefore adopted El-Masri and Rivard (2010) model for our coding scheme and its dimensions as our coding categories. For each category, we compiled a list of items found in empirical studies in the relevant IS literature. The literature was identified from a review of 24 years of IS research (1989-2012). Pertaining to the risk traits and undesirable events categories, we identified 98 risk factors from 28 empirical articles and compiled them into a single list. We then asked project managers to classify the risk factors under either the risk traits or undesirable events category. In order to ensure the content validity of the risk dimensions and attain an acceptable inter-judge reliability, three rounds of card sorting were conducted. We also met with two managers to discuss inter-judge disagreements. The meetings helped us to revise the wordings of ambiguous items. A Fleiss's Kappa of 0.91 was eventually achieved which demonstrates a high inter-judge agreement. With regards to the risk management practices, we examined the articles that empirically investigated these practices and derived a set of 39 items. We also scrutinised the seminal literature on IS project success and derived a list of 11 success criteria which constituted our expected outcomes category. Accordingly, our initial descriptive codes included 58 codes for project traits, 30 codes for undesirable events, 39 codes for management practices, and 11 codes for project outcomes. We nevertheless remained open to the emergence of new codes (Miles and Huberman 1994).

We then coded the cases according to the described coding scheme. We searched each case description for codes that match the items belonging to one of the four categories of the coding scheme. Once codes are found we determined whether they are related. If two or more codes were judged to be related, we indicated the nature of the interrelationship and recorded the evidence. The different scenarios of interrelationships between the codes were clustered in matrices as suggested by Miles and Huberman (1994, p. 57) in order to facilitate the analysis and identification of patterns (see examples in table 1). Those scenarios were iteratively compared and contrasted so as to distinguish the patterns. The analysis resulted in four distinct patterns that are discussed in the subsequent section.

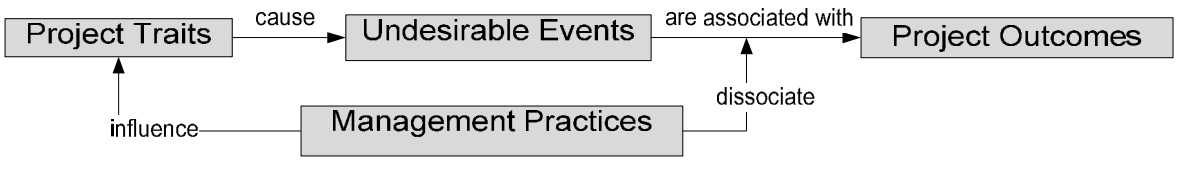

Figure 1. Software Project Risk Management Model Adopted from El-Masri and Rivard (2010)

A Dynamic Model of Software Project Risk

The majority of the pattern scenarios of risk component associations that we identified typify the relationships represented in Figure 1. However, our analysis also revealed four new patterns of associations: (1) the multiplicative effect of project traits, (2) the sequentiality effect of undesirable events, (3) the presence of a third variable, and (4) the tradeoff when selecting risk management practices. The model presented in Figure 2 further develops the existing model as proposed by El-Masri and Rivard (2010) by incorporating the interrelationships found between risk components.

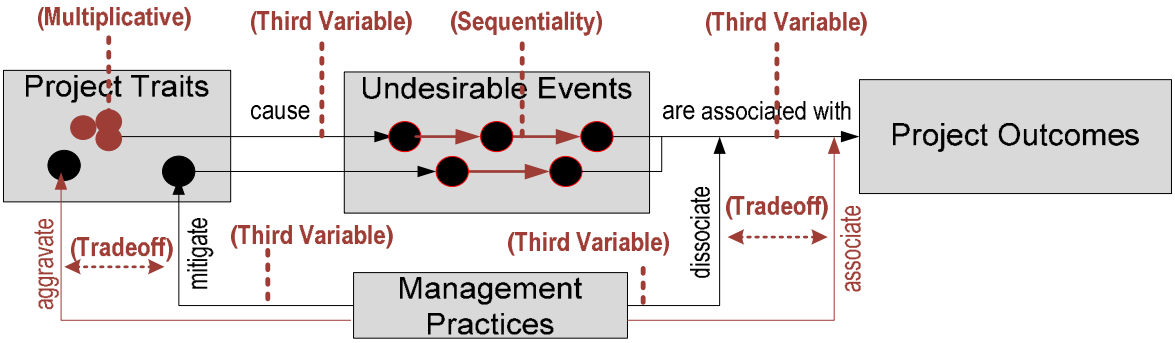

Figure 2. A Dynamic Model of Software Project Risk Management

Pattern #1: Multiplicative Effect of Project Traits

This pattern refers to situations in which the presence of two or more project traits significantly amplified the probability of occurrence or the severity of undesirable events. . Fourteen scenarios were identified under this pattern. Our analytical process can be understood through the excerpt below from the case describing the failure of the multibillion-dollar Taurus project (Drummond 1996). Four codes (in square brackets) are identified: (1) lack of requirement clarity (project trait), (2) changes in requirements (undesirable event), (3) unanticipated rework (undesirable event), and (4) interfaces with other systems (project trait). The associations between the codes were then interpreted and displayed in matrices (see table 1). The following excerpt reveals how the multiplicative effect of two project traits – the lack of requirement clarity and the number of interfaces with other systems – significantly amplified the amount of unanticipated rework:

> *The critical underlying assumption, however, was that [(1) the requirements were clear]. No sooner had [(2) one modification been agreed in one quarter than another party demanded something]... So they (the technical team) were constantly saying, 'OK well we will have [(3) to recode and redo] to cope with that.' Then [(4) software manufacturers] would say, 'Ah but then you have [(3) to make this change and you have to make that change] and you have [(4) to interface these things]...'* (Drummond 1996, p. 351).

Table 1. Sample of the Scenarios Collected on the Multiplicative Effect of Project Traits

Association No.	Code	Association	Code
1	(1) lack of requirement clarity (T)	causal	(2) requirement change (E)
2	(4) interfaces with other systems (T)	causal	N/A
3	(2) requirement change (E)	leading to	(3) unanticipated rework (E)
4	(2) lack of requirement clarity (T) and (4) interfacing with other systems (T)	significantly amplified	(3) unanticipated rework (E)

T: Project trait; E: Undesirable event.

In other scenarios, the multiplicative effect of project traits was easily recognizable. For example, in one of the cases, the multiplicative effect of two project traits – the (1) "lack of IT staff experience" and the (2) "lack of stability of software requirements" – led to (3) slow progress (Wastell 1996):

> *They (the IS staff) argued that [(3) the slow progress] was due to [(1) inexperience] with the method and the CASE tool, compounded by growth in [(2) the breadth and depth of requirements]* (Wastell 1996, p. 26).

The pattern of project traits having a multiplicative effect sheds some light on the intricate nature of risk. To evaluate project risk, the actors involved in risk assessment subjectively evaluate the probability of the occurrence of undesirable events. To this end, they assess a set $\{t_n\}$ of project traits that could influence their occurrence (Barki et al. 1993). However, if the multiplicative effect of project traits is taken into account, this assessment could entail $2^n - 1$ permutations instead of only n permutations. In other words, a software project with only five project risk traits requires the consideration of 31 (2^5 - 1) different trait combinations which exerts a substantial cognitive load on the actors evaluating risk.

Pattern #2: Sequentiality Effect of Undesirable Events

This pattern represents scenarios where one undesirable event influences the occurrence of another. In total, 161 scenarios of the interrelationships found were between undesirable events. For instance, a scenario identified in the PRISM project (Pan et al. 2008) shows that (1) management changes (an undesirable event) led to both (2) implementation problems and (3) conflicts (undesirable events).

> *[(1) The change of project manager] has contributed to several [(2) implementation problems]. No proper handover was made during this change of project leadership. Also, due to different styles of working, there were [(3) conflicts] between the new manager and some team members* (Pan et al. 2008, p. 263).

The risk assessment approaches that are prevalent in the IS literature do not account for the sequentiality effect of undesirable events. Existing approaches consider undesirable events as independent from one another. Risk exposure is often determined – using Equation 1 below – by multiplying the probability of occurrence of undesirable events by their associated losses (Boehm 1991). For example, in Han and Huang's (2007) study, managers were asked to assess, using a five-point Likert scale, the probability of occurence of a list of undesirable events and their impacts on different project outcomes (see table 2). Their assessment assumes that the probabilities of occurrence of undesirable events are independent from one another.

$$Static\ risk\ exposure\ =\ \sum_{i=1}^{n} Prob\ (UE_i)\ x\ Loss\ (UE_i)\ \ \{UE_i = i^{th}\ undesirable\ outcome\} \ldots\ldots\ldots\ (1)$$

Table 2. Sample of the Probability and Impact Measures Collected from Han and Huang's Study (2007)

Undesirable Event	Probability	Impact (tech)	Impact (cost)	Impact (schedule)	Impact (team)	Risk Exposure
Requirement changes	3.13	3.12	3.30	3.27	2.67	38.68
Management changes	2.18	2.37	2.42	2.39	2.28	20.62
Conflict between stakeholders	2.35	2.27	2.40	2.44	2.06	21.55
Project Risk Exposure						**80.85**

The fact that a significant number of scenarios were found suggesting the sequentiality effect of undesirable events highlights the need to look at the possibility of a dynamic interplay between events as they unfold. Hence, a measure of risk exposure that reflects to the sequentiality effect should include conditional probabilities between undesirable events, which can be represented by the following equation:

$$Dynamic\ risk\ exposure = \sum_{i=1}^{n} Prob(UE_i)\ x\ Loss\ (UE_i)\ (+)\ \sum_{i=1}^{n}\sum_{j=i+1}^{n} Prob\ (UE_i|UE_j)\ x\ Loss\ (UE_i)\ \ldots(2)$$

Consequently, in order to determine probability values for a single undesirable event, actors involved in risk assessment are obliged to not only assess their project's traits, but also variations in the undesirable event's probability value given the occurrence of other events. This assessment involves (n)×(n – 1)

permutations of undesirable events, and may be beyond their capacity.

Pattern#3: The Presence of a Third Variable

This pattern represents scenarios in which the relationship between two factors that contribute to risk is conditional on the existence of a third factor. Twelve scenarios show a conditional relationship between factors belonging to the different risk dimensions. An example of this pattern is exhibited in a case described by Kirsch (2004):

> *When [(1) communication with European counterparts became quite tense] as the project moved into development, stakeholders added [(2) formal evaluation mechanisms]. These mechanisms served to encourage stakeholders to set aside their individual differences…which ultimately moved the project forward. As the Shipments project moved closer to installing the first increments, additional formal measurement and evaluation mechanisms were utilized. {Because} [(3) tasks were structured and well defined], stakeholders applied the existing control mechanisms to implementation, including scheduled meetings, progress reports, project management software, and specific success criteria* (Kirsch 2004, p. 387).

Three codes were identified in the above excerpt: (1) strains in communication, (2) formal control mechanisms, and (3) well-defined tasks. The interpreted text reveals that formal control mechanisms (management practices) were implemented in order to diffuse the strains in communication arising between stakeholders (an undesirable event) and prevent them from affecting outcomes. The success of the implemented mechanisms was conditional on the *presence of a third variable*: well-defined project tasks (a project trait). Scenarios depicting this pattern were displayed in a matrix like table 3.

In another scenario from Cone (2002), formal planning – a practice that is positively associated with project performance (see Deephouse et al. 1996) – did not improve project performance. This was due to the fact that the time invested in formal planning was wasted due to changes in requirements. Thus, formal planning is related to project performance under the condition that requirements are stable.

> *The project was [probably too immense] and unmanageable to begin with… IBM tried to use ADA [to enforce discipline on the project] by making developers outline a design in high-level code, then fill in the blanks. But this was no match for an environment of where the FAA [kept changing its requirements]* (Cone 2002, p. 1).

Table 3. Sample of the Scenarios Collected Demonstrating the Presence of a Third Variable

Project	Code	Code	Conditional Code (3rd)
Shipment	Formal control mechanisms (P)	Strains in communication (E)	Well-defined tasks (T)
AAS	Formal planning (P)	Project size (T)	Stable requirements (T)

T: Project trait; E: Undesirable Event; P: Management practice.

This pattern highlights the complex multi-factor interplay between the various risk components. Actors involved in risk assessment not only need to evaluate risk as it evolves dynamically by analyzing the relationships among its components, they must also account for the presence of certain conditions in order for those relationships to occur.

Pattern #4: The Tradeoff When Selecting Risk Management Practices

This pattern represents scenarios in which the tradeoff between positive and negative consequences is – or should be – appraised prior to the implementation of risk management practices. The failure to balance both types of consequences may result in ill-informed decisions and lead to adverse outcomes. Thirteen such scenarios were found. Those scenarios demonstrate that, even though the implementation of risk management practices mitigated certain project traits or reduced the impact of undesirable events, they also increased the implication of other project traits or triggered new undesirable events. One of the recurring scenarios involved: (1) end-users' involvement; a practice that project managers implement (2) to clarify requirements and (3) ensure end-users' acceptance of the system. When involving end-users, the project managers evaluated – or failed to evaluate – the tradeoff between the probability that end users

would reject the system and the probability that (4) new requirements would emerge as a result of their involvement. An excerpt from a case described by Sillince and Mouakket (1997) demonstrates this tradeoff scenario:

> *After [(1) meeting the secretary-administrators several times] and getting more [(2) knowledge of the system and the flow of information], the two analysts realized that the system could not be developed without taking account of their requirements… That meant [(4) additional new requirements]… The analysts were not prepared for this change… but acknowledged the fact that the system needed to be expanded to include some requirements of the end-users if they wanted the system to be practical and to be [(3) accepted by the end-users]* (Sillince and Mouakket 1997, p. 376).

Another scenario in a case described by Cone (2002) shows how managers failed to assess the tradeoff between system quality and system acceptance due to end-user involvement.

> *…by giving them the opportunity to customize the [display colors], the design introduced the possibility that they might accidentally make planes appear invisible against the background* (Cone 2002, p. 2).

This pattern underscores the value of the different practices available to managers. Benaroch et al. (2006) assert that the value of a project's risk exposure should be offset by the options available to managers. However, due consideration must be given to both the positive and negative impacts of those options.

Design Principles for Software Project Risk Management Systems

The four patterns of interrelationships among risk components show that software project risk is multifaceted and dynamic. This requires that the actors involved in risk assessment take into account the interactions and conditions of interactions between risk components; forecast how future possible events trigger one another and amplify risk; and evaluate the negative and positive impacts of available mitigation practices. This problem-solving activity may be beyond a project risk manager's capacity, resulting in miscalculations and lack of foresight. For example, if the actors involved in risk assessment have to flawlessly evaluate project risk exposure based on ten project traits and their multiplicative effects then 1023 ($2^{10} - 1$) distinct combinations of project traits should be considered. This issue can be addressed with a risk management system design that intuitively considers those interactions during project risk assessment. Accordingly, we propose the design science paradigm as an alternative approach. We believe that project risk managers can be better supported using an IT artifact tailored to the specificities and intricacy of software projects.

Design science is a problem-solving paradigm used to provide practical solutions to real-world problems, and design theories are prescriptive in nature (Hevner et al. 2004; Walls et al. 1992). However, design theories must be justified by normative and descriptive theories – known as kernel theories – and integrated "into design paths intended to produce more effective information systems" (Walls et al. 1992). The first step towards building our design theory is to translate our findings on the four patterns of risk factor interrelationships into design principles. To stay faithful to the precepts of design science research, we were guided by the rubrics of Gregor and Jones (2007) on design theory construction. Accordingly, the overarching purpose of our research is to provide a practical solution that the actors involved in risk assessment can use to evaluate software project risk. Project failures are often attributed to management's inability to forecast temporal interactions between risk factors and accurately assess project risk exposure (Charette 2005). Hence our design principles must address those interactions so that the IT artifact can make a more accurate forecast of the project's exposure to risk. We propose three preliminary design principles in keeping with the identified risk factor interaction patterns.

The Association Principle

Upon determining the probability of occurrence of undesirable events, the IT artifact should allow for possible associations between project traits as well as between undesirable events. As mentioned above, the extant literature acknowledges four types of associations (El-Masri and Rivard 2010). First, a project trait is associated with one or more undesirable events. This association is required to determine the probability of occurrence of undesirable events. For example, the lack of clarity of software requirement (a

project trait) can lead to events such as requirement changes and scope creep. Second, an undesirable event is associated with one or more project outcomes. This association is necessary to identify the anticipated adverse impacts on project outcomes. For example, requirement changes can result in schedule and budget overruns. Third, management practices are associated with project traits. These associations are required to determine the decrease in the probability of occurrence of undesirable events. For instance, additional efforts to analyze client and end-user needs can clarify software requirements thereby reducing the probability or requirement changes and scope creep. Fourth, management practices are associated with undesirable events. These associations are necessary to determine the reduction in the impacts of undesirable events on project outcomes. For instance, scaling down the software requirements reduces the adverse impact of requirement changes on the project's budget and schedule. The risk factor association principle prescribes two novel ways in which risk factors ought to be associated and that the IT artifact should allow for:

1) Associations between project traits: When two or more project traits are simultaneously associated with at least one undesirable event. This is required in order to reassess the probability of occurrence of an undesirable event, given that the project possesses certain configurations of project traits. For instance, one of the scenarios identified in the case survey shows that the lack of requirement clarity together with the anticipated level of interfacing with other systems significantly amplified the amount of unexpected rework required (Drummond 1996, p. 351).
2) Associations between undesirable events: When undesirable events are associated with other undesirable events. This type of association is required in order to reassess the probability of occurrence of undesirable events, given that the probability of occurrence of other associated undesirable events is greater than zero. For example, a scenario identified in the case survey shows that changes in the project's management structure led to conflicts between stakeholders as well as implementation problems (Pan et al. 2008, p. 263).

The Regulation Principle

The IT artifact should be able to regulate the interplay between factors belonging to the different risk dimensions. There are two types of regulations; conditional and tradeoff. Conditional regulations refer to the IT artifact's ability to regulate associations between factors by attributing conditions to their interactions. In other words, the association between two factors is conditional on a third factor. For instance, a scenario from the case described by Cone (2002, p. 1) shows that formal planning can mitigate project size given the condition that the software requirements are stable. On the other hand, "tradeoff regulations" refers to the artifact's ability to regulate the anticipated gain generated by planned management practices by balancing their positive and negative impacts on project traits or project outcomes. For example, the case described by Sillince and Mouakket (1997, p. 376) shows that, involving the various end-user groups requires the analysis of the tradeoff between their level of satisfaction and the permitted increased in the number of software requirements.

The Simulation Principle

The IT artifact should be able to simulate possible project risk scenarios in order to: (1) evaluate project risk exposure, (2) prioritize project traits according to their impact on project risk exposure, and (3) suggest which management practices can help reduce project risk exposure. Prior to performing simulation analysis, the project manager should select and rank the project's expected outcomes. For instance, the project manager could specify that respecting the budget, satisfying end-user needs, and producing high quality software are the most important outcomes respectively. Afterwards, the actor involved in risk assessment selects, from a list of traits, those that are deemed risky and rate their significance (e.g., project size, staff expertise). Risk assessment is achieved by running a simulation that uses the rated project traits as well as the ranked expected outcomes as inputs. To this end, the IT artifact makes use of a database of past projects. This database should contain previously identified associations between the factors belonging to the different risk components. The results from the case survey we conducted can serve as the initial data source. Once the simulation has been performed, the project manager can choose the suitable management practices proposed by the artifact, evaluate their cost and benefits, and rerun the simulation. The simulation can be rerun with different management practices until an optimal solution is attained.

Current State and Future Direction

Theory testing in design science is justified through its usefulness in practice (Goldkuhl 2004; March and Smith 1995). Hence, in order to test the theory, we are developing a prototype consistent with the specified principle and will evaluate it in an organizational setting. An action design research (ADR) approach will be followed, in which we will execute iterative build, intervene, and evaluate (BIE) cycles (see Sein et al. 2011). ADR is a research method that combines the creation and evaluation of prescriptive design knowledge of design research with theory generation of action research (ibid). We chose an IT-dominant BIE process, since the locus of the innovation is the IS community and not the organizational intervention. Cycles subsequent to the initial BIE cycles will be based on a joint practitioner-researcher evaluation of the artifact in its use setting.

Conclusion

Managing software project risk is a complex decision-making process. It requires the capacity to interpret project information using prior experience in order to predict the possibility that future events will occur and how they can affect the outcomes. While recent assertions in IS research draw attention to the adverse impact of the interplay among risk factors, there is no extant theoretical framework to explain these interrelationships. As a result, existing project risk management software lacks functionalities that consider such interplay. This leads to risk being misrepresented and inadequately managed.

In terms of research, our study advances knowledge on software project risk and represents a first step toward the development of a design theory for software project risk management. The case survey helped unearth four patterns of interactions among risk components, which we used to define design principles for a software project risk management system. In terms of practice, we expect that an IT artifact reflecting these design principles will provide a more accurate assessment of risk exposure than existing computational techniques.

References

Alter, S., and Sherer, S., A. 2004. "A General, but Readily Adaptable Model of Information System Risk," *Communications of the Association for Information Systems* (14:1), pp. 1-28.

Barki, H., Rivard, S., and Talbot, J. 1993. "Toward an Assessment of Software Development Risk " *Journal of Management Information Systems* (10:2), pp. 203-225.

Barki, H., Rivard, S., and Talbot, J. 2001. "An Integrative Contingency Model of Software Project Risk Management " *Journal of Management Information Systems* (17:4), pp. 37-69.

Benaroch, M., Lichtenstein, Y., and Robinson, K. 2006. "Real Options in Information Technology Risk Management: An Empirical Validation of Risk-Option Relationships," *MIS Quarterly* (30:4), pp. 827-864.

Boehm, B. 1991. "Software Risk Management: Principles and Practices," *IEEE Software* (8:1), pp. 32–41.

Boehm, B., and Bhuta, J. 2008. "Balancing Opportunities and Risks in Component-Based Software Development," *IEEE Software* (25:6), pp. 56-63.

Charette, R.N. 2005. "Why Software Fails," *IEEE Spectrum* (42:9), pp. 42–49.

Clemons, E.K. 1995. "Using Scenario Analysis to Manage the Strategic Risks of Reengineering," *Sloan Management Review* (36:4), pp. 61-71.

Cone, E. 2002. "The Ugly History of Tool Development at the Faa." *Baseline Magazine* Retrieved April 23, 2012, from http://www.baselinemag.com/article2/0,1540,794112,00.asp

Deephouse, C., Mukhopadhyay, T., Goldenson, D.R., and Kellner, M.I. 1996. "Software Processes and Project Performance," *Journal of Management Information Systems* (12:3), pp. 187-205.

Drummond, H. 1996. "The Politics of Risk: Trials and Tribulations of the Taurus Project," *Journal of Information Technology* (11:4), pp. 347-347-357.

El-Masri, M., and Rivard, S. 2010. "Specifying the Software Project Risk Construct," in: *Proceedings of the Sixteenth Americas Conference on Information Systems*. Lima, Peru.

Goldkuhl, G. 2004. "Design Theories in Information Systems: A Need for Multi-Grounding," *JITTA : Journal of Information Technology Theory and Application* (6:2), p. 59.

Gregor, S., and Jones, D. 2007. "The Anatomy of a Design Theory.," *Journal of the Association for Information Systems* (8:5), pp. 312–335.

Han, W.M., and Huang, S.J. 2007. "An Empirical Analysis of Risk Components and Performance on Software Projects.," *Journal of Systems and Software* (18:1), pp. 42-50.

Hevner, A.R., March, S.T., Park, J., and Ram, S. 2004. "Design Science in Information Systems Research.," *MIS Quarterly* (28:1), pp. 75–105.

Jiang, J.J., Klein, G., Beck, P., and Wang, T.G.E. 2007. "Lack of Skill Risks to Organizational Technology Learning and Software Project Performance," *Information Resources Management Journal* (20:3), pp. 32-45.

Kirsch, L.J. 2004. "Deploying Common Systems Globally: The Dynamics of Control," *Information Systems Research* (15:4), pp. 374-374-395.

Larsson, R. 1993. "Case Survey Methodology: Quantitative Analysis of Patterns across Case Studies," *Academy of Management Journal* (36:6), p. 1515.

March, S.T., and Smith, G.F. 1995. "Design and Natural Science Research on Information Technology," *Decision Support Systems* (15:4), pp. 251-266.

Miles, M.B., and Huberman, A.M. 1994. *Qualitative Data Analysis: An Expanded Sourcebook*, (2nd ed.). Newbury Park, CA: Sage Publications.

Na, K., Simpson, J.T., Li, X., Singh, T., and Kim, K. 2007. "Software Development Risk and Project Performance Measurement: Evidence in Korea," *The Journal of Systems and Software* (80:4), pp. 596-605.

Newig, J., and Fritsch, O. 2009. "The Case Survey Method and Applications in Political Science." Rochester, Rochester: p. n/a.

Pan, G., Hackney, R., and Pan, S.L. 2008. "Information Systems Implementation Failure: Insights from Prism," *International Journal of Information Management* (28:4), pp. 259-269.

PMI, S.c. 2004. *A Guide to Project Management (Pmbok)*. (2004 ed.). Newton square, PA, USA: Project Management Institute.

Ropponen, J., and Lyytinen, K. 2000. "Components of Software Development Risk: How to Address Them? A Project Manager Survey," *IEEE Transactions on Software Engineering* (26:2), pp. 98-111.

Schmidt, R., Lyytinen, K., Keil, M., and Cule, P. 2001. "Identifying Software Project Risks: An International Delphi Study," *Journal of Management Information Systems* (17:4), pp. 5-36.

Sein, M.K., Henfridsson, O., Purao, S., Rossi, M., and Lindgren, R. 2011. "Action Design Research," *MIS Quarterly* (35:1), p. 37.

Sicotte, C., and Paré, G. 2010. "Success in Health Information Exchange Projects: Solving the Implementation Puzzle," *Social Science & Medicine* (70:1), pp. 1159-1165.

Sillince, J.A.A., and Mouakket, S. 1997. "Varieties of Political Process During Systems Development," *Information Systems Research* (8:4), pp. 368-368-397.

Walls, J.G., Widemeyer, G.R., and ElSawi, O.A. 1992. "Building an Information System Design Theory for Vigilant Eis," *Information Systems Research* (3:1), pp. 36–59.

Warkentin, M., Moore, R.S., Bekkering, E., and Johnston, A.C. 2009. "Analysis of Systems Development Project Risks: An Integrative Framework," *Database for Advances in Information Systems* (40:2), pp. 8-27.

Wastell, D.G. 1996. "The Fetish of Technique: Methodology as a Social Defence," *Information Systems Journal* (6:1), pp. 25-40.

Yin, R.K., and Heald, K.A. 1975. "Using the Case Survey Method to Analyze Policy Studies," *Administrative Science Quarterly* (20:3), September, 1975, pp. 371-381.