Functionality Risk in Information Systems Development: An Empirical Investigation

Amrit Tiwana and Mark Keil, *Member, IEEE*

***Abstract*—Functionality risk is defined as the risk that a completed system will not meet its users' needs. Understanding functionality risk is central to successful information systems development (ISD), yet much of the research in this area has focused on identification of risk factors and development of conceptual frameworks. Little is known about how various functionality risk factors collectively influence managers' perceptions about the risk that a project will fail. As organizations become increasingly reliant on software, it is evermore important to understand functionality risk in ISD. In this paper , we develop an integrative model of functionality risk to explain the *relative* importance of six salient functionality risk factors that have been consistently identified in the prior ISD literature as being important: 1) related technical knowledge; 2) customer involvement; 3) requirements volatility; 4) development methodology fit; 5) formal project management practices; 6) project complexity. The model is tested empirically with 60 highly experienced MIS Directors in sixty organizations. In addition to providing empirical support for the proposed model, the *relative* importance of each of the key functionality risk factors is empirically assessed. Development methodology fit, customer involvement, and use of formal project management practices emerged as the top three functionality risk factors. Additional finer-grained analyses show that high methodology fit lowers several other sources of risk. Implications for research and practice are discussed.**

***Index Terms*—Conjoint, functionality risk, implementation failure, information integration theory, information systems development, knowledge integration, knowledge transformation, project management, software project risk.**

I. Introduction

SOFTWARE that fails to deliver its promised functionality can have devastating consequences. The opening of Denver International Airport was delayed over a year because of software that did not do what it was intended to do; software problems wrecked a $500 million European satellite; functionality problems with the infamous London's stock exchange TAURUS system cost taxpayers £1.6 billion; and elsewhere it destroyed a NASA Mars mission [50]. While the operational consequences of dysfunctional software are not always so dramatic, the financial consequences are, in the aggregate, quite dramatic: Nearly 40% of the $2.5 trillion spent in the United States on IT during 1997–2001 is estimated to be gambled on such underperforming projects [8]. While some projects are judged to be outright failures, more often, they fall into a gray area somewhere between complete success and abject failure. While there are many different kinds of risks that can affect information systems development (ISD) projects, our focus here is on *functionality risk*, which has received little attention in prior research. Functionality risk is defined as the risk that the completed system will not meet its users' needs [15].

Our emphasis on functionality risk stems from two observations about how modern organizations depend on information systems and how they acquire them. First, organizations are increasingly reliant on novel IS applications to innovate in their business processes and activities. Examples of such applications include: enterprise resource planning, supply chain management, reverse logistics, and knowledge management. In such applications, even a technically sound software application might be worthless unless it provides the intended business functionality. Second, organizations increasingly outsource development to external organizations (e.g., consulting firms and off-shore developers) that must understand the user group's task domain in order to deliver appropriate functionality. Therefore, functionality risk is a pervasive and increasingly critical type of project risk to which ISD projects are exposed.

Some have argued that the high incidence of IT project failure may be partly the result of managers initiating risky projects without an adequate assessment or understanding of the risks involved [48]. While several studies have developed a variety of tools such as checklists, frameworks, and prescriptions for assessing ISD project risk [5], [18], [37], [59], there has been little or no attempt to understand the *relative* influence of various drivers of functionality risk and how they shape overall risk perceptions. In this study, we seek to establish the relative influence of six prominent functionality risk factors (derived from a synthesis of the prior ISD risk literature) in shaping managerial perceptions of overall project risk.

We draw on a foundational premise in the requirements elicitation literature: Effectively delivering the appropriate functionality can be viewed as a "knowledge transformation" problem in which user needs are transformed and embodied into software code during the ISD process [11], [21], [34].

We draw on prior theory on risk perceptions, particularly building on the notion of information integration emphasized in this body of work (e.g., [2], [46], and [67]), which we relate to risk factors identified in the ISD risk literature. We introduce and test a proposed model using empirical data collected from MIS Directors[1] in 60 large organizations, applying conjoint analysis in which both individual- and aggregate-level analyses are conducted.

Manuscript received July 1, 2004; revised December 1, 2004, May 1, 2005, and Sepember 1, 2005. Review of this manuscript was arranged by Department Editor, R. Sabherwal.

A. Tiwana is with the Iowa State University, College of Business, Ames, IA 50011-1350 USA (e-mail: tiwana@iastate.edu)

M. Keil is with the Georgia State University, Robinson College of Business, Atlanta, GA 30302 USA (e-mail: mkeil@gsu.edu).

Digital Object Identifier 10.1109/TEM.2006.878099

[1]MIS directors are senior IT managers who are responsible for managing IT applications development and infrastructure maintenance in corporate environments.

The primary contribution of this study is *simultaneously* assessing the relative importance of consistently identified functionality risk factors and how they shape managerial perceptions of overall project risk. The findings offer some surprising insights into where managers believe the key functionality risks lie. We also introduce conjoint analysis, a technique that has not previously been used in IS research, but which we believe offers a promising approach for assessing a wide range of tradeoffs that are inherent to ISD project decision-making. Together, these contributions hold significant implications for both ISD research and practice.

The remainder of the paper is organized as follows. The next section develops the hypotheses. The third section describes the methodology and data collection. The fourth section describes the analysis and results. The paper concludes with the implications of our findings for research and practice.

II. Theory and Hypothesis Development

A foundational premise behind effective ISD recognized in the requirements elicitation literature is that the design of the application accurately and completely embody user needs [11], [21], [34]. Following Byrd *et al.* [11], we define this process of transferring and transforming problem domain and problem solving expertise from the relevant knowledge sources (end-users, programmers, and project stakeholders) into software code as *knowledge transformation*. Knowledge transformation involves accurately representing and translating knowledge of the problem domain into the functional requirements of the software and related design artifacts (such as conceptual design and formal specifications) [11], [34], [58], [65]. Much of this knowledge is idiosyncratic to the project's context and is tacitly held by specialized stakeholders (e.g., their constraints, requirements, and viewpoints) [3], [12], [34]. Thus developing a system that truly reflects user needs requires accurately determining what a system should accomplish and then building a system that does it. Successful knowledge transformation of user requirements into software, therefore, increases the likelihood that a resulting system will satisfy users' actual needs [34], [66]. Following this logic, it should be possible to explain some of the variance in managerial perceptions about overall project risk through an examination of factors that potentially impede such knowledge transformation during the ISD process (i.e., functionality risk factors).

In our conceptual development, we build on prior theory on risk perceptions that links prior knowledge, communication, and involvement to risk perception [5], [46], [48], [67]. We also draw on prior research in marketing that establishes the moderating role of knowledge and involvement on perceived risk associated with intangibles such as software applications [46]. Prior work on risk in marketing has also shown that high involvement of customers in the purchase process lowers their perceived risk [46].

A central theoretical idea in this body of risk literature is the notion of information integration in forming risk perceptions [2], [67]. Individual managers' judgments about the level of perceived risk are based on heuristics and systematic integration of information about a variety of characteristics of a given project—a central tenet of the heuristic-systematic information processing model of risk [67]. In integrating such information into a holistic assessment of perceived risk, individuals apply cognitive models through which they weigh each attribute that can potentially influence risk perceptions. Using these heuristic cues, they integrate information about these project attributes into an overall assessment about a project's risk perception. Addressing our research question, therefore, requires inferring such cognitive models that are used by IS managers as they integrate the available information into a holistic risk assessment for a given project.

We draw on this set of theoretical ideas to explore six functionality risk factors that are consistently mentioned in the ISD risk literature that we expect to influence perceptions of overall project risk. The relative importance of these six factors is assessed in a subsequent empirical test of the nomological network of relationships summarized in Fig. 1.[2] Table I summarizes these six risk factors and representative studies in which each has been examined. These risk factors are: 1) related technical knowledge; 2) customer involvement; 3) volatility of requirements; 4) project complexity; 5) fit between development methodology and type of project; 6) use of formal project management practices. A notable pattern in the prior studies represented in Table I is that no previous study has examined all of these risk factors in a single study, leaving a gap in our understanding about their relative importance (which this study addresses). We next discuss each in detail.

A. Related Technical Knowledge

Technical knowledge is clearly one type of knowledge required to successfully develop a software-based system. Both the introduction of new technology on a project and a lack of required knowledge in the project personnel are recognized in the ISD literature as important risk factors [9], [39]. The broader risk literature suggests that prior related experience lowers the perceived risk in undertaking an activity [46]. Prior experience in similar projects helps develop a cognitive representation of a largely intangible investment such as software, which reduces the level of perceived risk associated with developing it [46]. In contrast, an unclear mental representation of the project creates uncertainty and anxiety owing to its intangible characteristics, increasing perceived risk. As an organization gains familiarity with a given set of hardware, operating systems, domains, and programming languages, it becomes aware of the relevant technical constraints and the problems that can surface [53]. Experience—schemas, operational blueprints, abstract problem-solving approaches, and cognitive frameworks—gained in prior projects can then guide problem solving in future projects.[3] This means that prior problem-solving patterns and approaches can be repurposed for similar projects in the future—even when they

[2]We used two criteria for selecting these six functionality risk factors from the larger universe of risk factors: 1) they should have consistently been identified in prior research on software project risk (i.e., at least three studies); 2) they should influence the knowledge transformation process identified in the requirements elicitation literature [11], [58]. We do not claim that these six functionality risk factors are the only such factors that influence project risk.

[3]Some of this knowledge is readily accessible because it may be captured in artifacts (such as documents, contracts, project plans, requirements, software code, specifications) or embedded in models (design rationales, patterns, heuristics, and estimation models).

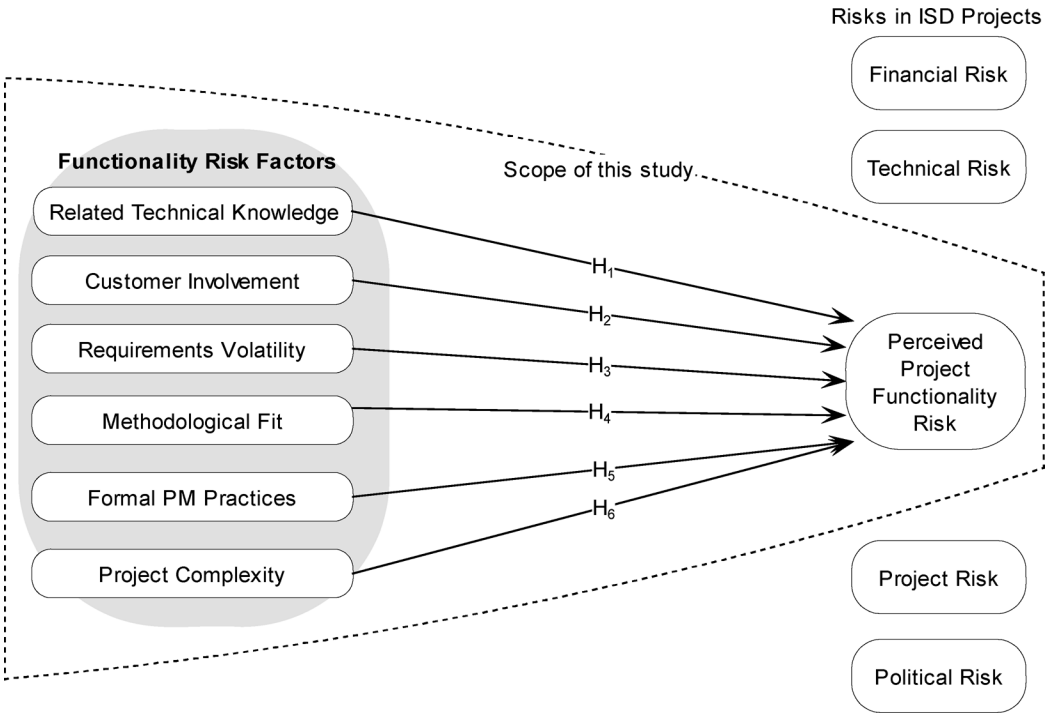

Fig. 1. A nomological network of relationships between various risk factors and perceived functionality risk.

TABLE I
FUNCTIONALITY RISK FACTORS IN THE SOFTWARE PROJECT LITERATURE

Study	Fit	Customer involvement	Formal PM practices	Related technical knowledge	Project complexity	Requirements volatility	Source of data and evidence provided
[59]	●	●	●		●		International survey of IT project managers
[22]	●			●	●	●	Case study
[39]		●		●		●	International survey
[49]	●	●		●		●	29 software projects
[1]				●	●	●	Experiment with 33 graduate students
[6]	●	●	●	●	●		120 software projects in 75 Canadian organizations
[17]		●				●	17 large software development projects
[19]	●	●		●	●	●	"Several" projects
[20]		●	●	●	●	●	Historical project analysis
[23]				●			608 firms adopting object-oriented methods
[27]					●	●	34 application software projects
[41]		●					82 software projects in Hong Kong & Australia
[52]				●	●		Electronic filing projects in 103 tax firms
[53]		●	●	●	●		Three case studies
[55]		●				●	64 software projects
[57]		●		●	●	●	Survey of 83 project managers
[60]		●	●				Experiment with 72 graduate students.
[70]	●			●			Experiment with six subjects

cannot be directly reused without adaptation [46], [58]. Organizations' familiarity with related technologies, hardware, software, and programming languages also reduces the likelihood that unexpected technical problems will surface during development. Therefore, the more related know-how, experience, and skills that an organization possesses, the less new knowledge is required to successfully implement the required functionality [23]. This leads to our first hypothesis.

Hypothesis 1: The higher the related technical knowledge in the domain of a project, the lower is the managerial perception of overall project risk.

B. Customer Involvement

System requirements—the basis for design—are derived from expressed or inferred end-user (or customer) needs. Although end-users might clearly understand their own needs for

a system, the development team often finds it difficult to understand and capture knowledge about these needs as they tend to be somewhat "sticky" (i.e., not easily transferable through formal requirements) [62]. Prior work suggests that higher levels of customer involvement lowers the level of perceived risk associated with developing products that are highly intangible [46]. Interpreted in the ISD context, the accuracy with which user requirements are understood by the development team determines how precisely they can be translated into requirements specifications which are ultimately reflected in the software code that is produced. In the ISD literature, lack of adequate customer involvement and misunderstanding customer requirements have been identified as key factors affecting the success of software projects [11], [33], [38], [39], [43], [57]. Similarly, extensive customer involvement has been shown to enhance project outcomes [31], [36]. User involvement through mechanisms such as customer walkthroughs, prototypes, use cases, pilot implementations, and user reviews ensures consistency between the designers' view and the customers' view of the system and provides an active mechanism for feedback on the evolving design. Such feedback serves to validate the developers' interpretations of customer needs and their translation into specifications and subsequent implementation in software code. Therefore, high levels of expected customer involvement reduce the risk that the delivered system will be incongruent with their needs, lowering the level of perceived risk associated with pursuing a project. This leads to our second hypothesis.

Hypothesis 2: The higher the level of customer involvement in a project, the lower is the managerial perception of overall project risk.

C. Requirements Volatility

Requirements volatility is defined as the extent to which the initial requirements change over the trajectory of a project. While no prior studies have explicitly examined the link between requirements volatility and risk, prior research has suggested that requirements volatility is an important software project risk factor [39], [57]. Changing user or business requirements makes it more challenging to accurately capture customer needs during the development process and can lead to errors in the design decisions and scope assessments of software projects [1]. The importance of changing or volatile requirements in formulating project risk assessments is also recognized in various classic software project management frameworks [53], [55], as well as subsequent activities such as software maintenance [4]. Requirements volatility might also arise from the project team's failure to elicit the true requirements from the system's users. Therefore, higher requirements volatility hinders requirements knowledge transformation, heightening the risk that the delivered system will not provide the functionality required by the end-users. While it is difficult to accurately assess whether requirements will be volatile at the outset of a project, it is the *expectation* of volatility that influences perceived risk at the outset of a project. This leads to the third hypothesis.

Hypothesis 3: The higher the level of requirements volatility, the higher is the managerial perception of overall project risk.

D. Development Methodology Fit

A common thread that runs across all ISD methodologies is that they formally encapsulate *some* approach for embodying customer needs in the functionality provided by a system.[4] Although the merits of different methodologies are fiercely debated by practitioners [35], it is not our intention to contribute to that debate. The issue of methodology fit has however been alluded to in some classic works on software project risk [10], [19] as well as another recent large-scale international field study [59]. The larger body of work on software project risk management also recognizes the holistic notion of fit [6], although it does not directly explore on fit as alignment between the development method and characteristics of ISD projects. Development methodology fit is defined as the degree to which the ISD methodology is appropriate for the characteristics of the given project [13], [26], [41]. Lack of fit exists when the chosen methodology is inappropriate for the characteristics of the project such as schedule, innovativeness, and complexity.

Universal, project-independent methodologies are characterized as being weak approaches to development and make the assumption that one development methodology can serve the same purpose in a variety of project types and problem domains [7], [26]. Hickey and Davis [34] caution that one methodology or technique cannot possibly be sufficient or appropriate for all conditions and that a methodology should be chosen based on the specific situation. For example, structured programming approaches, object-oriented methodologies, and agile development methods all can serve the same purpose but to different degrees of effectiveness in a given project. Agile methodologies for example, work best in projects that involve two to eight member teams that are collocated, have milestones no further than one month apart, use experienced developers, and depend on fully automated testing procedures [16]. Similarly, "black-box" structured development methodologies work best in non-novel projects but under-perform in conceptually novel projects [63]. Therefore, the methodology that is used in a given project must fit the characteristics of the project.

There is growing consensus in the software development community that it is not the technique per se that matters, but the *fit* between the technique and the specific project context in which it is used [35]. Prior research has also suggested that the risk exposure faced by a project is influenced by the fit between the nature of the project and the methodology that is selected to execute it [6]. We, therefore, posit that the higher the degree of fit between the chosen methodology and the characteristics of a project, the lower the level of perceived risk associated with pursuing it. This leads to our fourth hypothesis.

Hypothesis 4: The higher the level of development methodology fit, the lower is the managerial perception of overall project risk.

E. Formal Project Management Practices

Formal project management is defined as reliance on formal plans, schedules, and budgets to ensure the efficient and timely execution of a project [6], [53]. The importance of formal

[4]For example, agile or lightweight methodologies encourage extensive user involvement, others such as object-orientation suggest decomposition of projects into subprojects, others such as incremental implementation and waterfall methods recommend phased development , yet others recommend parallel development of subsystems.

project management practices as a mechanism for enhancing software development performance is well recognized in the ISD literature [24], [25], [30]. Formal project management practices introduce behavioral directives and well-defined patterns for interaction among project team members and thereby reduce the need for extensive communication among the project constituents to coordinate the joint application of their knowledge [28]. A clear structure for sequencing project activities also provides mechanisms to monitor progress and to spot discrepancies [14], [42], [53], [71]. Therefore, use of formal project management practices should increase the likelihood that the delivered system will meet its intended functionality needs, thereby lowering perceived functionality risk. This leads to our fifth hypothesis.

Hypothesis 5: The higher the use of formal project management practices in the development process, the lower is the managerial perception of overall project risk.

F. Project Complexity

Complexity refers to the interdependent organizational and technical elements that constitute a system [45]. Project complexity in this conceptualization involves both technical and organizational elements. For example, the number of other systems with which an application interacts, the number of interfaces to other applications, the complexity of its operations [27], [72], and the amount of data that it must process [4] drive technical complexity. Organizational complexity is driven by such factors as the number of user departments affected by the application as well as the number of outside contractors involved in the project. As the complexity from both organizational and technical sources increases, so does the number of communication paths among subsystems and various stakeholders, thereby raising interdependencies and coordination challenges [12], [34]. When a project is faced with such challenges, it is more difficult to develop system functionality that fully satisfies the users' needs. Therefore, we expect that managers will perceive higher risk in pursuing projects with greater perceived complexity. This leads to our final hypothesis.

Hypothesis 6: The higher the overall complexity of the project, the higher is the managerial perception of overall project risk.

III. Research Design and Data Collection

An empirical investigation in which IT managers were recruited to respond to a series of hypothetical scenarios was used to collect the data for this study and to test the proposed model. A conjoint research design was used, which we first describe.

A. Research Design

1) An Overview of the Conjoint Research Approach: Conjoint analysis is a multi-attribute judgment analysis technique based on Information Integration Theory [2] that involves posteriori decomposition of the respondent's decision process [47]. The decision process examined in this study is managerial assessment of overall project risk. There are three central elements in a conjoint research design: attributes, part-worth and overall utilities, and conjoint profiles. An *attribute* refers to a decision criterion that respondents might use to evaluate the dependent variable (a project's overall risk). The overall value assigned to the dependent variable is referred to as its *overall utility*. The contribution of each attribute towards the formation of the overall utility of a project is called its *part-worth utility*. Different combinations of attribute levels are called treatments or *conjoint profiles*.

The technique requires respondents to make a series of judgments about a dependent variable based on a set of attributes from which the underlying structure of their cognitive system can be analyzed and decomposed. A series of conjoint profiles with different combinations of attribute levels are presented to each respondent, and the respondent provides an assessment of the dependent variable for each project profile. The six risk factors in the model were the attributes in the study and managerial perceptions of overall project risk was the dependent variable. Thus assessing the dependent variable for each project profile involves integration of information about various project attributes by the respondents. The underlying structure of the respondents' cognitive models regarding the influence of various functionality risk factors on overall perceptions of project risk (i.e., the value that the respondents place on each attribute in formulating a holistic assessment) can be statistically inferred from these judgments by analyzing the responses at the individual and aggregate levels [47].

Multiple regression is used to decompose the assessments into its underlying structure and the corresponding beta coefficients. To identify attributes that are statistically significant at the aggregate level, the regression coefficient for each attribute is averaged across individuals and the sign of this coefficient indicates the direction of the relationship between the attribute and the dependent variable [69]. A Z-statistic aggregates the T-statistics derived from the individual-level analysis for each attribute to assess the statistical significance of the attribute in predicting the dependent variable. Such analysis at both the individual respondent level and aggregate level increases the predictive ability of the model [54]. These significance tests are supplemented with Hays' [32] Omega Squared that reflects the variance explained by each attribute and is used to assess the relative importance of each attribute [47, p. 67]. In sum, beta values indicate the directionality and ω^2 indicates the relative importance of each attribute. This approach allows assessments of the contributions of each of the six functionality risk factors to perceptions of overall project risk. The conjoint approach, thus, allows both individual and aggregate-level analyses while explicitly accounting for the non-independence between the assessments of multiple profiles by a single individual. In that sense, it is comparable to repeated measures experimental design.

2) Factors Guiding the Choice of the Conjoint Research Design: In order to assess the simultaneous effects of the six functionality risk factors in our model, our research design had to satisfy four criteria. These criteria were best satisfied by the conjoint research design that was used for the study. First, the

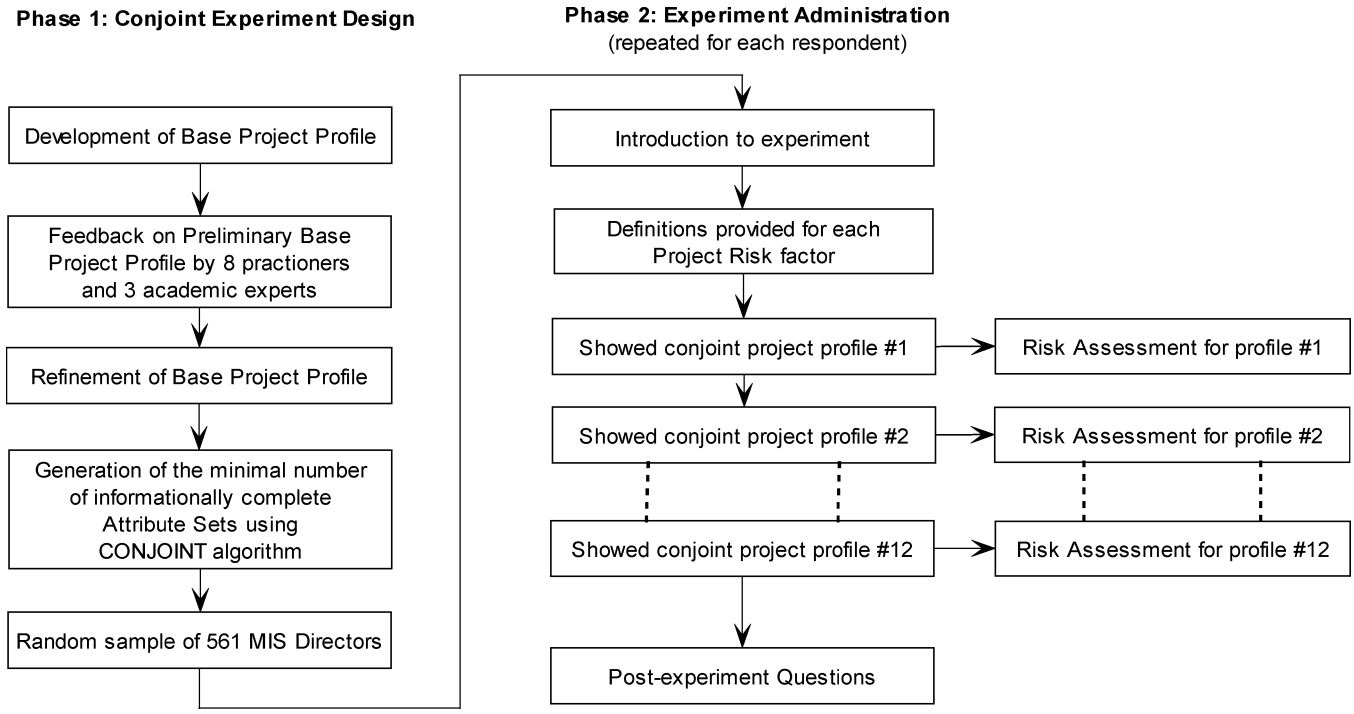

Fig. 2. An overview of the research methodology.

empirical design had to allow the respondents to assess project risk by *simultaneously* considering all six functionality risk factors. Here, our use of the conjoint approach is a unique contribution in itself because prior studies on software risk have not assessed the effects of multiple risk factors that might *simultaneously* be present. Second, the approach had to provide high levels of external validity (which comes from relying on highly experienced IT professionals) while allowing high levels of control over factors extraneous to the model. It is important to note that although the project profiles themselves were hypothetical, the assessments of these project profiles were performed by practitioners with extensive field experience. The conjoint project profile evaluations used in this study mirror actual decision making activities in the respondent pool: MIS Directors frequently choose among multiple possible projects that their organization can pursue. The conjoint approach, thus, provides the control of a laboratory experiment with the external validity of surveys by drawing on the experience of real-world subjects. Third, different organizations can use different risk mitigation strategies. The study's design had to separate these confounding interventions by capturing the risk assessments for a comparable pool of projects at the same point in the development lifecycle across all participating organizations. Fourth, the research design had to be immune to two key types of bias associated with traditional field-based studies: 1) retrospection bias that comes from recalling completed projects; 2) bias towards giving socially desirable responses such as not admitting project failures. By using hypothetical project profiles on which IT managers pass judgments, the conjoint approach overcomes both these biases by estimating inferred importance rather than relying on stated importance [29]. Fig. 2 provides an overview of the key phases of the study, each of which is discussed in detail in the following sections.

B. Measures and Conjoint Profile Development (Phase 1)

The first phase of the study involved development of the base conjoint project profile for the empirical investigation in Phase 2. A fractional factorial design was used for the study. The primary benefit of using a fractional factorial design is that it protects against sources of variation that are not estimated (e.g., other factors that might be important but excluded from the model), are confounded with what is estimated, and are likely to produce the most bias in the parameters that are estimated [47]. The number of treatments for each respondent increases greatly as the number of attributes considered increases; the most feasible conjoint designs, therefore, rely on a fractional factorial design [47], [56]. Since our objective was to assess the relative importance of the risk factors, a fractional factorial design was informationally more efficient than other types of conjoint designs [47] because it reduced the number of attribute combinations into a manageable set of profiles that had to be assessed by each participant [29]. The CONJOINT algorithm implemented in SPSS 11.5 was used to generate the appropriate number of project profiles (i.e., a number that most efficiently generated the maximum information) to be presented to each respondent [44]. This algorithm produced 12 project profiles that had to be evaluated by each participant (see Table II).

A pretest with eight IT executives and three academic domain experts confirmed the face validity of the attributes, the clarity with which they were defined, their levels, and whether the combinations presented in the profiles were realistic. Feedback from

TABLE II
MEANS AND STANDARD DEVIATIONS FOR EVALUATIONS OF EACH PROJECT PROFILE

Conjoint project profile	Project Attributes						Functionality Risk	
	Related technical knowledge	Customer involvement	Requirements volatility	Project complexity	Fit	Formal project mgmt	Mean	Std. Dev.
1	Low	Low	High	Low	High	High	3.9	1.955
2	High	Low	Low	High	High	High	3.44	1.731
3	High	Low	Low	High	Low	Low	5.33	1.946
4	High	High	High	High	Low	High	4.39	1.845
5	High	Low	High	Low	High	Low	4.02	1.644
6	Low	High	Low	Low	Low	High	3.83	1.693
7	Low	Low	High	High	Low	High	5.98	1.639
8	Low	Low	Low	Low	Low	Low	6.27	1.753
9	High	High	Low	Low	High	High	2.41	1.580
10	Low	High	Low	High	High	Low	3.87	1.696
11	High	High	High	Low	Low	Low	4.47	1.459
12	Low	High	High	High	High	Low	4.48	1.845

this panel was also used to ensure that the attribute definitions were presented succinctly and unambiguously.

Each respondent was presented a series of 12 hypothetical software projects (conjoint profiles) described in terms of six attributes, each with two levels (high/low): 1) related technical knowledge; 2) customer involvement; 3) volatility of requirements; 4) project complexity; 5) fit between development methodology and type of project; 6) use of formal project management practices. Overall project risk perception was assessed for each project using a measure previously developed by Keil *et al.* [40]. A nine-point scale with bipolar anchors "I think this project will succeed" and "I think this project will fail" was used. Use of such semantic differential anchors is recommended in the conjoint methodological literature [47]. To control for individual level factors, we measured the respondents' confidence in their project risk assessments (an 11-point semantic differential scale), prior risk assessment experience (measured as number of prior projects for which the respondent had participated in risk assessment) and number of years of IT experience [67].

C. Data Collection (Phase 2)

A mail survey, which is appropriate when the objective of a conjoint study is hypothesis testing [47], was used for data collection. The *Dun and Bradstreet* executive directory was used to identify a random sample of senior IT managers with the job title of MIS director in U.S. organizations. We contacted the potential respondents via a mailed cover letter in which we explained the purpose of the study, provided a URL for the on-line instrument, and offered two movie tickets and a copy of the findings as a token incentive.[5] On the survey site, the respondents were provided with instructions, the conjoint project profiles, and a postevaluation questionnaire. After reading the instructions, the respondents were sequentially presented 12 project profiles—each with different combinations of attribute levels—which they evaluated by rating the dependent variable, overall perceived project risk, on a 9-point scale. Only the levels of the six functionality risk factors were manipulated across the 12 project profiles (the Appendix shows a sample project profile along with the attribute definitions provided to the respondents at the outset of the process). These profiles are shown in Table II and a sample profile is shown in the Appendix . Following Louviere's [47, p. 62] recommendation, the conjoint profile data were coded as -1 for low and 1 for high. Following the project evaluations, the respondents completed a postevaluation questionnaire in which they answered demographic questions (IT experience and prior project risk assessment experience) and provided an assessment of their confidence about the evaluations they had just completed. We received 61 responses from MIS directors in 61 organizations and discarded one set of responses because of substantial missing data. This represents a response rate of 11.5% (61/531), which is favorable considering the length of the instrument (13 screens). These responses provided data on overall project risk assessments for 12 conjoint project profiles each from 60 managers (720 total observations) in 60 organizations.[6] The sample size at the individual level is, therefore, 60, for which there are 720 observations.

IV. RESULTS AND DISCUSSION

A. Descriptive Statistics and Respondent Demographics

The participants in the study were MIS Directors with extensive IT experience, who were also actively involved in decision-making related to pursuing new software projects. On average, the participants in our study had 15.1 years of IT experience and had previously been directly involved in a decision

[5]Of the 569 executives initially contacted, 35 requests were returned as undeliverable because of incorrect addresses or the contacted executive having left the job. Four other respondents declined to participate citing reasons such as company policy and lack of time.

[6]No statistically significant differences were found between the early and late respondents, which suggest that our findings are not threatened by non-response bias.

TABLE III
RESULTS

Variables {Presented in order of importance}	Expected Sign	β (Z-statistic)	Omega squared (ω^2)	$Z(\omega^2)$
Risk Factors		*Direction*	*Relative Importance*	
Methodological fit (H_4)	-	-.303***(-9.147)	.092***	74.031
Customer involvement (H_2)	-	-.194***(-5.850)	.038***	31.584
Formal PM practices (H_5)	-	-.172***(-5.193)	.030***	25.749
Related technical knowledge (H_1)	-	-.150***(-4.535)	.023***	20.161
Project complexity (H_6)	+	.106**(3.199)	.011***	10.160
Requirements volatility (H_3)	+	.081**(2.452)	.007***	6.010
Control variables				
Confidence in evaluations	-	-.130***(-3.935)	.019	
Prior risk assessment experience	-	-.023(-.629)		
IT experience	-	-.059(-1.598)		
Adjusted R^2	**21.2%**			

***$p \leq 0.001$; **$p \leq 0.01$; 1-tailed T-tests

making role in 49 software projects. On average, the organizations in our sample had 372 employees and $58.3 million in annual revenues. A wide variety of industries such as construction, manufacturing, chemicals, IT services, financial and insurance services, medical services, and publishing were represented in our sample. The means and standard deviations for each conjoint profile are summarized in Table II.

B. Analyses

Hypothesis Tests: Multiple regression analyses were conducted using SPSS 11.5 to analyze the data. Following the recommended procedures native to conjoint analysis [29], [47], both individual and aggregate level analyses were conducted as described in Section III-A1. To recap, beta coefficients indicate the directionality, Z-statistics indicate the statistical significance, and ω^2 indicates the relative importance of each attribute. Table III summarizes the aggregated regression coefficient for each risk factor, the corresponding Z-statistic, its ω^2 explained variance value, and the significance of the change in explained variance from adding the factor to the model (beyond the variance explained by the control variables). The variables are listed in the order of their relative importance, as suggested by our analysis. At the aggregate level of analysis, the Z-statistics are statistically significant for each of the six predictors and the signs (indicated by the β coefficients) are in the hypothesized directions. The effects of related technical knowledge (Hypothesis 1), customer involvement (Hypothesis 2), fit between methodology and type of project (Hypothesis 4), and use of formal project management practices (Hypothesis 5) on perceptions of overall project risk were negative and statistically significant. As expected, requirements volatility (Hypothesis 3) and project complexity (Hypothesis 6) had positive and statistically significant relationships with overall project risk. Thus, Hypotheses 1 through 6 are supported.

Relative Importance of Functionality Risk Factors: The magnitude of the ω^2 values for each risk factor indicates its relative importance in shaping managerial perceptions of overall project risk. The results indicate that on average, the fit between the type of project and the chosen methodology is the most important predictor of software project risk ($\omega^2 = 9.2\%$). The second most important factor is the level of customer involvement ($\omega^2 = 3.8\%$). The third most important predictor of risk perception is the use of formal project management practices ($\omega^2 = 3.0\%$). Related technical knowledge ($\omega^2 = 2.3\%$), project complexity ($\omega^2 = 1.1\%$), and requirements volatility ($\omega^2 = 0.7\%$) are next, in that order. The model explained 21.2% of the variance in project risk perception, of which the control variables explained 1.9%.[7] As indicated by the significance tests in the last column ($Z(\omega^2)$), the change in variance explained by the addition of each risk factor to the model was significant, although roughly half of the explained variance arises from a single variable: Fit between the methodology and the project.

C. Discussion

The predictors of software project risk in their order of importance were (see Table III) methodological fit, customer involvement, use of formal project management practices, related technical knowledge, project complexity, and requirements volatility as discussed next.

[7]For control variables, we assessed the variance explained by individual level factors such as IT experience of the respondent, prior project risk assessment experience, and confidence in their risk assessments. The postevaluation self-assessment of respondents' confidence in their own evaluations was negatively and significantly related to risk perception ($\beta = -0.130$, T-value $= -3.93$, $p < 0.0001$), but only explained 1.6% additional variance above and beyond the 21.7% variance that the six factors explained. These values suggest that managers who are more confident about their own judgments of project risk are likely to perceive less functionality risk in a project than managers who are less confident about their ability to assess risk. One explanation for this may be that some managers are more susceptible to cognitive biases such as "overconfidence" and "illusion of control" and that these biases affect their perceptions of the risk associated with a project [52]. Further analyses of individual-level control variables revealed the individual's IT experience (measured in years) was negatively associated but statistically non-significant ($\beta = -0.059$, NS), as was the respondent's prior risk assessment experience (measured as number of prior projects for which s/he actively participated in risk assessments) ($\beta = -0.023$, NS). Further, the degree of confidence that subjects expressed was independent of their years of experience or the number of project initiation decisions in which they had previously participated. These results suggest that certain software project managers may develop a level of confidence that is unwarranted and which may affect their perceptions of project risk. These observations are consistent with Keil et al.'s [39] suggestion that software project failure follows from unrealistically downward-biased risk perceptions among software project managers.

Methodological Fit: The results of this study shed new light on the debate among the proponents of various ISD methodologies. The "one perfect methodology" debate assumes that one method is superior to others. Our results indicate that establishing the right fit between the characteristics of a project and the chosen methodology is the single most important predictor of overall project risk. Devoid of the project context, the debate over which methodology is correct or incorrect is meaningless. A large and highly significant path coefficient of over -0.3 and a ω^2 value of 0.092 suggests that software projects that attempt to utilize ill-fitting methodologies are perceived as being significantly riskier. Recent research has called for "advice on how and when to use methodologies [26]." The results obtained here underscores the importance of answering these "how" and "when" questions, as methodology fit is an important predictor of perceived project risk. In terms of future research, this finding raises the question of whether such perceptions are warranted and, if they are, how software project managers can do a better job of achieving good fit to reduce this source of risk. The issue of fit between project type and development methodology has not received a great deal of attention in the ISD literature and our findings point to the need for additional research. This finding confirms the importance is fit that is holistically recognized in the broader software project risk management literature [6], [7], and refines that conceptualization with a focus on fit as alignment between the development method and project characteristics. It is important to note that we did not examine how fit can be assessed but whether a manager perceived that a methodology fit the project at its outset. A taxonomy of existing methodologies and the characteristics of projects where they work best would be a useful first step in developing the idea of fit further. In developing more detailed models to assess fit, three considerations appear pertinent: 1) characteristics of the problem domain; 2) characteristics of the solution domain; 3) characteristics of the project [34].

Customer Involvement: Next, our results emphasize the importance of customer involvement, the second most important predictor of managerial risk perception. Higher levels of customer involvement can reduce a project's exposure to risk—a finding that is consistent with prior work on IS project management [11], [31], [36].

Formal Project Management Practices: The third most important predictor of risk was use of formal project management practices. This finding suggests that traditional project management practices could go a long way toward reducing the risk associated with IT projects, which are frequently managed in a more informal way. It also supports the assertion that formal project management practices enhance software development performance that has previously been reported in the larger body of work in software development [24], [25], [30], [53], [71].

Related Technical Knowledge: The results also show that the more dissimilar a new project is from those previously undertaken in an organization, the higher the managerial perception of overall project risk. Projects that draw on existing technical knowledge gained from similar undertakings in the past are considered less risky. This result is consistent with prior work in IS [23], [53], [63] and reference discipline literature such as that in marketing [46].

TABLE IV
DIFFERENCES IN OVERALL PROJECT RISK PERCEPTIONS ACROSS HIGH AND LOW METHODOLOGY FIT SUBGROUPS

Subgroup	Mean	SD	**One-way ANOVA Test** F-value for across-group differences ***p ≤ 0.001; 1-tailed T-test
Low fit	4.91	1.75	72.477***
High fit	3.81	1.72	

Project Complexity: Complexity was also a significant predictor of perceived risk but its path coefficient of about 0.11 and a ω^2 of 0.011 suggests that it carries lesser weight that the aforementioned factors. While demonstrating the importance of complexity in predicting risk, these data suggest that IT managers tend to weigh it lower than the other risk factors in their assessments. This result is consistent with prior work [4], [27], [34].

Requirements Volatility: Surprisingly, requirements volatility was the least important of the six predictors of risk examined in this study. Although requirements volatility is frequently an implicit or latent factor in studies of software development, we believe that this is the first study to explicitly examine its relationship to functionality risk. A significant and positive path coefficient of about 0.08 suggests that, although it significantly figures into their assessments of risk, its relative importance is low (as indicated by an ω^2 value of 0.007). Although the significance of this factor is in agreement with the emphasis on requirements volatility in prior work on software project management and software maintenance [4], [53], [55], the relatively low weight ascribed by managers to this factor is somewhat surprising. It is possible that managers view complexity and requirements volatility as a given in most projects. If this is the case, they may not have viewed a project with low complexity or low requirements volatility to be within the realm of their control and this might explain why these variables explain comparatively less variance than the other variables in the model. On the other hand, it is also possible that other conditions such as good project management practices will compensate for such factors as complexity and requirements volatility.

D. Post-Hoc Analyses of Methodological Fit

Since development methodology fit emerged as the most significant functionality risk factor, we conducted additional post hoc analysis to gain further insights into this result. We created two subgroups in the dataset for low and high fit and compared the mean risk overall risk perception in a one-way ANOVA (Table IV).

The results show that the difference in overall project risk across the subgroups is statistically significant.

Next, we performed a subgroup analysis by running separate regressions for high and low fit projects. The results of the subgroup analyses are shown in Table V.

All risk factors are significant in the low fit subgroup. However, the last column shows that three of the five risk factors that were significant in the low fit subgroup are *not* significant in the high fit subgroup. This suggests that a good fit between the project and the development methodology can compensate for other functionality risk factors. Specifically, it appears when methodological fit is high, managers do not need to worry as

TABLE V
SUBGROUP ANALYSES FOR LOW AND HIGH DEVELOPMENT METHODOLOGY FIT

Variables	Methodological Fit	
	Low Fit	High Fit
	β(Z-statistic)	β(Z-statistic)
Customer involvement	**-.458***(-6.93)**	-.108(-1.53)
Formal PM practices	**-.457***(-6.89)**	**-.136*(-1.93)**
Related technical knowledge	**-.448***(-6.75)**	-.086(-1.22)
Project complexity	**.349***(5.88)**	**.134*(1.90)**
Requirements volatility	**.151**(2.55)**	.141(1.63)

$^{***}p < 0.001$; $^{*}p < 0.05$; 1-tailed T-test
Significant results are shown in **bold**.

much about the risks associated with customer involvement, related technical knowledge, and requirements volatility. For example, it is plausible that customer involvement by itself becomes less important because use of an appropriate development methodology might already provide an *appropriate* level of interaction with customers in the requirements and development process (e.g., prototyping and agile development).

It is also interesting to note that two other functionality risk factors *remain* significant when development methodology fit is high: 1) formal project management practices; 2) project complexity. This result suggests that even when methodology fit is high, managers need to ensure that formal project management practices are followed and that project complexity is kept to a manageable level. Fig. 3 illustrates how the relationship between overall risk perception with the use of formal project management approaches [Fig. 3(a)] and project complexity [Fig. 3(b)] changes across projects where methodology fit is low versus high.

E. Limitations

Before moving on to implications of the research, we discuss a number of limitations associated with the study. First, this study was based on six of the risk factors that have been consistently identified in the software project risk literature and focused principally on functionality risk. This was a deliberate choice driven by the pragmatics of keeping the number of project profiles to a manageable number (including more factors would have greatly increased the number of project profiles presented to each respondent). Limiting our focus to functionality risk factors may explain why we were able to explain only 21% of the variance in overall project risk perceptions. Consideration of other types of risk factors (such as political risk and financial risk) remain as open avenues for future research and their inclusion would likely improve the amount of variance in overall project risk perception that can be explained by such a model. The influence of the risk factors is also likely to evolve over the course of the project (e.g., customer involvement might lower risk at the outset but customer requests for new features and functionality in the late stages might increase risk); future research should explore such dynamics of functionality risk. Still, the fact that functionality risk factors explain 21% of the variance in overall project risk perception indicates that this type of risk is quite significant and cannot be ignored.

Second, the profiles evaluated by the respondents represented hypothetical scenarios rather than actual projects. While this approach provides excellent control and reduces the threats of social response and retrospection bias, it does not allow for the testing of relationships between functionality risk and project outcomes. Future research should compare these findings with those from post-hoc evaluations of completed projects in order to examine the influence of these risk factors on project outcomes. The findings offered in the present study lay a rigorous foundation for such future research.

Third, the assessments in the study were made at the outset of a project. Clearly, risk is dynamic because project managers can employ risk mitigation strategies to control the risks that are identified at the outset of a project. Future research should examine how the levels of the six risk factors change over the course of development and the risk mitigation strategies that facilitate such changes. In this respect, the present study lays the foundation for future longitudinal research on software project risk management.

Fourth, the dependent variable was assessed using a single-item semantic differential scale, as is established practice in conjoint studies [47], [61], [68]. However, this must be viewed as a limitation, as single item measures cannot be assessed for reliability. Finally, the study focused on in-house application development projects and relied solely on MIS Directors' risk perceptions. Future research should examine different types of projects (e.g., outsourced projects) and the perceptions of other project stakeholders, including customers. Such comparisons will be particularly insightful since many organizations are moving towards customizing commercial off-the-shelf (COTS) software instead of developing similar systems in-house. The extent to which the functionality risk factors investigated here would apply to COTS-based projects is an open question, but one that can be tested empirically. Since implementing COTS software often requires extensive customization, this would suggest that such projects are exposed to functionality risks that are similar to those associated with in-house software development [71]. Related technical knowledge is necessary for accomplishing successful integration with other interdependent systems that might already be in place in the organization. Customer involvement is necessary to ensure that the appropriate package is chosen. Requirements volatility can still be an issue as business needs may change during the software acquisition process. Project complexity can still be quite high if the COTS software is complex with a multitude of settings that must be customized for the organization at hand, or if the project involves many different functional units or departments within the organization. Formal project management practices still apply to COTS software, as these implementations can still be quite tricky. Of the six functionality risks investigated here, the only one that vanishes from the client's perspective in a pure COTS-based project is development methodology fit (and even this risk factor may come into play if a significant amount of customization is required). This is not to say that development methodology fit is no longer important, but merely that in a COTS-based project,

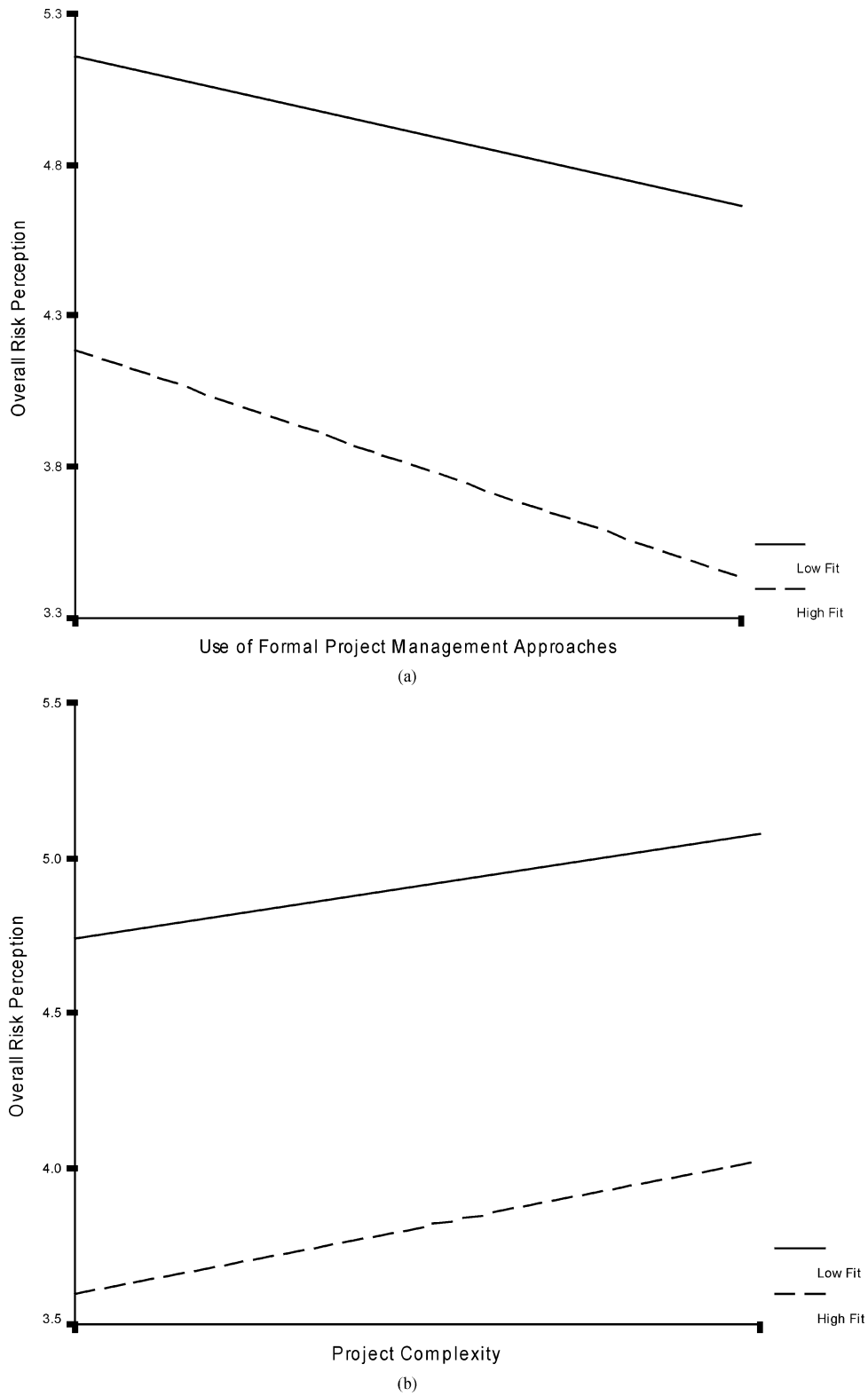

Fig. 3. A comparison of the relationship between overall risk perception and (a) use of formal project management practices and (b) project complexity across projects with high and low methodology fit. The solid lines represent low fit and the dotted lines represent high fit.

this functionality risk is largely (if not entirely) shifted to the vendor. Given the importance of development methodology fit on overall project risk perception, this suggests that acquiring COTS software might be one way to reduce an organization's exposure to functionality risk, provided that the COTS software does not require significant customization.

V. Implications for Research and Practice

Software development involves five types of risks: 1) financial risk; 2) technical risk; 3) project risk; 4) functionality risk; 5) political risk [15]. In this paper, we focused on functionality risk. This focus was motivated by the growing business necessity for developing systems that provide the desired functionality in innovative application problem domains. This requires that organizations be able to get a better handle on functionality risk in ISD projects. Drawing on the knowledge transformation logic in the requirements elicitation literature, we focused on six functionality risk factors that are frequently identified in the ISD literature.

The key research implications of this study are: 1) functionality risk factors can explain a significant fraction of overall project risk; 2) different functionality risk factors exert different levels of influence in terms of shaping overall project risk perception. These findings have research implications for how managers arrive at an overall assessment of project risk and the relative weights that each functionality risk driver carries in such evaluations. Our results suggest that managers ascribe different levels of importance to different functionality risk factors.

An intriguing finding is the high importance ascribed by respondents to development methodology fit. Unlike other factors in the model that have consistently emerged as being important in prior software risk studies, this factor, while mentioned in the literature [59], had not been examined empirically. The results of this study underscore just how important methodological fit can be. Our subgroup analysis showed that when methodological fit is high, risks that might otherwise be significant, such as customer involvement, related technical knowledge, and requirements volatility cease to explain variance in perceptions of overall risk. This suggests that choosing the appropriate development methodology can play an important role in reducing the exposure that a project may face from these other functional risk factors. Therefore, as the use of new ISD methodologies becomes more accepted, the issue of methodological fit and frameworks for assessing it represents an important area for future research.

It is also illustrative to consider where MIS Directors believe functionality risks are concentrated and how controllable this risks are. The top three predictors of risk—methodological fit, customer involvement, and formal PM practices—are factors over which project managers have some degree of control. This is consistent with Keil *et al.*'s [39] observations that managers will clearly distinguish between risk factors that are perceived as being within their control versus those outside their control. These findings are also consistent with March and Shapira's [51] distinction between risk taking (where skill or information can reduce uncertainty) versus gambling behavior (where the odds are exogenously determined and perceived as being uncontrollable) among managers. Together, these results suggest that in assessing project risk managers put the heaviest weight on the factors that they perceive as being controllable most heavily.

Our results also suggest that MIS Directors assess software project risk holistically, i.e., they use their assessments of various functionality risk factors to arrive at an aggregate project risk assessment. While prior studies on software project risk have measured the importance of specific risk factors across projects, this study has introduced a new way of measuring the effects of the simultaneous presence of various risk factors in a single project.

Finally, from a methodological perspective, this study is novel in that it illustrates how the conjoint methodology, which has been so successfully employed in marketing research, can be employed to study ISD phenomena. A major advantage of this approach is that it affords the precision and control of an experiment but taps into the experience base of seasoned MIS professionals. The conjoint research methodology, which has rarely ever been used to date in IS research, is promising for studying ISD phenomena because it allows delineation of the phenomenon of interest while avoiding social desirability and retrospection biases that plague traditional retrospective project reviews and surveys.

In terms of implications for practice, this research suggests that formal project risk assessments, even if they focus only on the six functionality risk factors in our model, would go a long way toward helping managers to get a better handle on software project risk [64].

VI. Conclusion

This study sought to assess the relatively importance of six prominent functionality risk factors for ISD by simultaneously considering them in a single model. Although a large body of work exists on ISD risk, no prior study has focused on the relative importance of functionality risk factors in shaping overall project risk perception. We tested the model using data collected from IT managers in 60 organizations. The results showed that—in the order of their *relative* influence—the key drivers of project risk were: choice of an appropriate development methodology, customer involvement, use of formal project management practices, similarity to previous projects, project complexity, and requirements volatility. The study used the conjoint methodology that combines the realism of field-based practitioner surveys with the high level of control normally associated with an experimental design, which has not been used in prior MIS research. While this study represents a significant step in our understanding about ISD functionality risk, ample work remains to be done to develop pragmatic tools to help managers reduce the frequency and severity of functionally handicapped information systems. We hope that the conjoint research approach illustrated in this paper can be used as a complement to the more prevalent qualitative and survey methods.

Appendix
Conjoint Instrument, Sample Project Profile, Attribute Descriptions, and Levels

The respondents were sequentially presented 12 conjoint profiles and asked to provide a risk assessment on a 9-point semantic differential scale following each profile based on the information provided and their experience and knowledge. Data on the individual-level control variables were collected after the conjoint profile assessments. The definitions were provided at the outset for each project attribute (see Table VI).

1) Sample Conjoint Profile (1 of 12): (see Table VII).

TABLE VI

Project attribute	Description
Related technical knowledge	The extent to which the project is technically similar to other IT projects completed in the organization.
Customer involvement	The extent to which the project's customers are involved throughout the project, including involvement via customer walkthroughs, prototypes, use cases, pilot implementations, and user reviews.
Requirements volatility	The extent to which a project's business requirements change over the course of the project.
Development methodology fit	Fit refers to the lack of misfit between the software development methodology that is chosen for a project and the tasks that must be accomplished for successfully completing a project of this type.
Use of formal project management practices	The extent to which the project relies on formal plans, schedules, and budgets for sequencing project activities, monitoring progress, and to spot discrepancies during the development process.
Project complexity	Project complexity refers to the technical and organizational complexity of the project. Technical complexity is driven by the number of other systems with which an application interacts, the number of interfaces to other applications, the complexity of the system's operations, and the amount of data that it must process. Organizational complexity is driven by the number of user departments that are affected by the application as well as the number of outside contractors involved in the project.

TABLE VII

Compared to other IT projects in your organization, for this project…	
Similarity to previous projects is…	Low
Customer involvement is…	Low
Requirements volatility is…	High
Project complexity is…	High
Fit between software development methodology & type of project is...	High
Use of formal project management practices is…	Low

ACKNOWLEDGMENT

The authors are grateful to A. Bush, A. Bharadwaj, and B. Ramesh for inputs at the initial stages of this research. They would also like to acknowledge the constructive inputs they received from the Department Editor, R. Sabherwal, and the three anonymous reviewers.

REFERENCES

[1] T. Abdel-Hamid, K. Sengupta, and D. Ronan, "Software project control: An experimental investigation of judgment with fallible information," *IEEE Trans. Softw. Eng.*, vol. 19, no. 6, pp. 603–612, Jun. 1993.
[2] N. Anderson, *Foundations of Information Integration Theory*. New York: Academic, 1981.
[3] L. Argote, B. McEvily, and R. Reagans, "Managing knowledge in organizations: An integrative framework and review of emerging themes," *Manag. Sci.*, vol. 49, no. 4, pp. 571–582, 2003.
[4] R. D. Banker and S. A. Slaughter, "The moderating effects of structure on volatility and complexity in software enhancement," *Inf. Syst. Res.*, vol. 11, no. 3, pp. 219–240, 2000.
[5] H. Barki, S. Rivard, and J. Talbot, "Toward an assessment of software development risk," *J. Manag. Inf. Syst.*, vol. 10, no. 2, pp. 203–225, 1993.
[6] ——, "An integrative contingency model of software project risk management," *J. Manag. Inf. Syst.*, vol. 17, no. 4, pp. 37–69, 2001.
[7] R. Baskerville and J. Stage, "Controlling prototype development through risk analysis," *MIS Quart.*, vol. 20, no. 4, pp. 481–504, 1996.
[8] C. Benko and W. McFarlan, *Connecting the Dots: Aligning Projects with Objectives in Unpredictable Times*. Boston, MA: Harvard Bus. Sch. Press, 2003.
[9] B. Boehm and R. Ross, "Theory-W software project management: Principles and examples," *IEEE Trans. Softw. Eng.*, vol. 15, no. 7, pp. 902–916, Jul. 1989.
[10] B. W. Boehm, "Software risk management: Principles and practices," *IEEE Softw. Mag.*, vol. 8, no. 1, pp. 32–41, Jan. 1991.
[11] T. Byrd, K. Cossick, and R. Zmud, "A synthesis of research on requirements analysis and knowledge acquisition techniques," *MIS Quart.*, vol. 16, no. 1, pp. 117–138, 1992.
[12] P. Carlile and E. Rebentisch, "Into the black box: The knowledge transformation cycle," *Manag. Sci.*, vol. 49, no. 9, pp. 1180–1195, 2003.
[13] P. D. Chatzoglou, "Use of methodologies: An empirical analysis of their impact on the economics of the development process," *Eur. J. Inf. Syst.*, vol. 6, no. 4, pp. 256–270, 1997.
[14] V. Choudhury and R. Sabherwal, "Portfolios of control in outsourced software development projects," *Inf. Syst. Res.*, vol. 14, no. 3, pp. 291–314, 2003.
[15] E. K. Clemons, "Evaluation of strategic investments in information technology," *Commun. ACM*, vol. 34, no. 1, pp. 22–36, 1991.
[16] A. Cockburn, *Agile Software Development*. Reading, MA: Addison Wesley, 2002.
[17] B. Curtis, H. Krasner, and N. Iscoe, "A field study of the software design process for large systems," *Commun. ACM*, vol. 31, no. 11, pp. 1268–1287, 1988.
[18] H. Drummond, "The Politics of risk: Trials and tribulations of the Taurus project," *J. Inf. Technol.*, vol. 11, pp. 347–357, 1996.
[19] R. Fairley, "Risk management for software projects," *IEEE Softw. Mag.*, vol. 11, no. 3, pp. 57–67, May 1994.
[20] R. Fairley and M. Willshire, "Why the Vasa sank: 10 problems and some antidotes for software projects," *IEEE Softw. Mag.*, vol. 20, no. 2, pp. 18–25, Mar.-Apr. 2003.
[21] S. Faraj and L. Sproull, "Coordinating expertise in software development teams," *Manag. Sci.*, vol. 46, no. 12, pp. 1554–1568, 2000.
[22] R. Fichman and S. Moses, "An incremental process for software implementation," *Sloan Manag. Rev.*, pp. 39–52, 1999.
[23] R. G. Fichman and C. Kemerer, "The assimilation of software process innovations: An organizational learning perspective," *Manag. Sci.*, vol. 43, no. 10, pp. 1345–1363, 1997.
[24] B. Fitzgerald, "Formalized systems development methodologies: A critical perspective," *Inf. Syst. J.*, vol. 6, no. 1, pp. 3–23, 1996.
[25] M. Fraser, K. Kumar, and V. Vaishnavi, "Informal and formal requirements specification languages: Bridging the gap," *IEEE Trans. Softw. Eng.*, vol. 17, no. 5, pp. 454–466, May 1991.
[26] R. Glass, "Matching methodology to problem domain," *Commun. ACM*, vol. 47, no. 5, pp. 19–21, 2004.
[27] A. Gopal, T. Mukhopadhyay, and M. Krishnan, "The role of software processes and communication in offshore software development," *Commun. ACM*, vol. 45, no. 4, pp. 193–200, 2002.
[28] R. Grant, "Prospering in dynamically-competitive environments: Organizational capability as knowledge integration," *Organization Sci.*, vol. 7, no. 4, pp. 375–387, 1996.
[29] P. E. Green, K. Helsen, and B. Shandler, "Conjoint internal validity under alternative profile presentations," *J. Consumer Res.*, vol. 15, no. 3, pp. 392–397, 1988.

[30] P. J. Guinan, J. G. Cooprider, and S. Faraj, "Enabling software development team performance during requirements definition: A behavioral versus technical approach," *Inf. Syst. Res.*, vol. 9, no. 2, pp. 101–125, 1998.
[31] J. Hartwick and H. Barki, "Explaining the role of user participation in information systems development," *Manag. Sci.*, vol. 40, no. 4, pp. 440–465, 1994.
[32] W. Hays, *Statistics*. New York: Holt, Rinehart, and Winston, 1973.
[33] J. Herbsleb and A. Mockus, "An empirical study of speed and communication in globally distributed software development," *IEEE Trans. Softw. Eng.*, vol. 29, no. 6, pp. 481–494, Jun. 2003.
[34] A. Hickey and A. Davis, "A unified model of requirements elicitation," *J. Manag. Inf. Syst.*, vol. 20, no. 4, pp. 65–84, 2004.
[35] J. Highsmith, "The great methodologies debate part I: opening statement," *Cutter IT J.*, vol. 14, no. 12, pp. 2–4, 2001.
[36] B. Ives and M. M. Olson, "User involvement and MIS success: A review of research," *Manag. Sci.*, vol. 30, no. 5, pp. 586–603, 1984.
[37] J. Jiang, G. Klein, and J. Balloun, "Systems analysis attitudes toward information systems development," *Inf. Resources Manag. J.*, vol. 11, no. 4, pp. 5–10, 1998.
[38] M. Keil and E. Carmel, "Customer-developer links in software development," *Commun. ACM*, vol. 38, no. 5, pp. 33–44, 1995.
[39] M. Keil, P. E. Cule, K. Lyytinen, and R. C. Schmidt, "A framework for identifying software project risks," *Commun. ACM*, vol. 41, no. 11, pp. 76–83, 1998.
[40] M. Keil, B. C. Y. Tan, K. K. Wei, T. Saarinen, V. Tuunainen, and A. Wassenaar, "A cross-cultural study on escalation of commitment behavior in software projects," *MIS Quart.*, vol. 24, no. 2, pp. 299–325, 2000.
[41] M. Khalifa and J. Verner, "Drivers of software development method usage," *IEEE Trans. Eng. Manag.*, vol. 47, no. 3, pp. 360–369, Aug. 2000.
[42] L. J. Kirsch, "The management of complex tasks in organizations: Controlling the systems development process," *Organization Sci.*, vol. 7, no. 1, pp. 1–21, 1996.
[43] R. Kraut and L. Streeter, "Coordination in software development," *Commun. ACM*, vol. 38, no. 3, pp. 69–81, 1995.
[44] W. Kuhfeld, R. Tobias, and M. Garratt, "Efficient experimental-design with marketing-research applications," *J. Marketing Res.*, vol. 31, no. 4, pp. 545–557, 1994.
[45] R. Langlois, "Modularity in technology and organization," *J. Econ. Behav. Organization*, vol. 49, 2002.
[46] M. Laroche, J. Bergeron, and C. Goutaland, "How intangibility affects perceived risk: The moderating role of knowledge and involvement," *J. Services Marketing*, vol. 17, no. 2, pp. 122–140, 2003.
[47] J. Louviere, *Analyzing Decision Making: Metric Conjoint Analysis*. Beverly Hills, CA: Sage, 1988.
[48] K. Lyytinen, L. Mathiassen, and J. Ropponen, "Attention shaping and software risk–A categorical analysis of four classical risk management approaches," *Inf. Syst. Res.*, vol. 9, no. 3, pp. 233–255, 1998.
[49] A. MacCormack, R. Verganti, and M. Iansiti, "Developing products on internet time: The anatomy of a flexible development process," *Manag. Sci.*, vol. 47, no. 1, pp. 133–150, 2001.
[50] C. Mann, "Why is software so bad," *MIT Technol. Rev.*, pp. 33–38, Jul./Aug. 2002.
[51] J. G. March and Z. Shapira, "Managerial perspectives on risk and risk taking," *Manag. Sci.*, vol. 33, no. 11, pp. 1404–1418, 1987.
[52] L. Martins and A. Kambil, "Looking back and thinking ahead: Effects of prior success on managers' interpretations of new information technologies," *Acad. Manag. J.*, vol. 42, no. 6, pp. 652–661, 1999.
[53] F. W. McFarlan, "Portfolio approach to information systems," *Harvard Bus. Rev.*, vol. 59, no. 5, pp. 17–25, 1981.
[54] W. Moore, "Levels of aggregation in conjoint analysis: An empirical comparasion," *J. Marketing Res.*, vol. 17, pp. 516–523, 1980.
[55] S. Nidumolu, "The effect of coordination and uncertainty on software project performance – Residual performance risk as an intervening variable," *Inf. Syst. Res.*, vol. 6, no. 3, pp. 191–219, 1995.
[56] R. Priem, "An application of metric conjoint analysis for the evaluation of top managers' individual strategic decision making processes: A research note," *Strategic Manag. J.*, vol. 13, pp. 143–151, 1992.
[57] J. Ropponen and K. Lyytinen, "Components of software development risk: How to address them? A project manager survey," *IEEE Trans. Softw. Eng.*, vol. 26, no. 2, pp. 98–111, Feb. 2000.
[58] I. Rus and M. Lindvall, "Knowledge management in software engineering," *IEEE Softw. Mag.*, vol. 19, no. 3, pp. 26–38, May-Jun. 2002.
[59] R. Schmidt, K. Lyytinen, M. Keil, and P. Cule, "Identifying software project risks: An international delphi study," *J. Manag. Inf. Syst.*, vol. 14, no. 4, pp. 5–36, 2001.
[60] K. Sengupta and T. Abdel-Hamid, "The impact of unreliable information on the management of software projects: A dynamic decision perspective," *IEEE Trans. Syst., Man, Cybern.*, vol. 26, no. 2, pp. 177–189, Mar. 1996.
[61] D. A. Shepherd, "Venture capitalists' assessment of new venture survival," *Manag. Sci.*, vol. 45, no. 5, pp. 621–630, 1999.
[62] G. Szulanski, "Exploring internal stickiness: Impediments to the transfer of best practice within the firm," *Strategic Manag. J.*, vol. 17, no. 2, pp. 27–43, 1996.
[63] A. Tiwana, "Beyond the black-box: Knowledge overlaps in software outsourcing," *IEEE Softw. Mag.*, vol. 21, no. 5, pp. 51–58, Sep.-Oct. 2004.
[64] A. Tiwana and M. Keil, "The one minute risk assessment tool," *Commun. ACM*, vol. 47, no. 11, pp. 73–77, 2004.
[65] A. Tiwana and E. McLean, "Managing the unexpected: The tightrope to E-business project success," *Commun. ACM*, vol. 46, no. 12, pp. 345–350, 2003.
[66] A. Tiwana and E. R. McLean, "Expertise integration and creativity in information systems development," *J. Manag. Inf. Syst.*, vol. 22, no. 1, pp. 13–43, 2005.
[67] C. Trumbo, "Information processing and risk perception: An adaptation of the heuristic-symantic model," *J. Commun.*, vol. 52, no. 2, pp. 367–382, 2002.
[68] B. Tyler and H. Steensma, "Evaluating technological collaborative opportunities: A cognitive modeling perspective," *Strategic Manag. J. (Special Issue)*, vol. 16, no. , pp. 43–70, 1995.
[69] J. Vancouver and E. Morrison, "Feedback inquiry: The effect of source attributions and individual differences," *Organizational Behav. Hum. Decision Processes*, vol. 62, pp. 276–285, 1995.
[70] I. Vessey and S. Conger, "Learning to specify information requirements: The relationship between application and methodology," *J. Manag. Inf. Syst.*, vol. 10, no. 2, pp. 177–201, 1993.
[71] L. Wallace, M. Keil, and A. Rai, "How software project risk affects project performance: An investigation of the dimensions of risk and an exploratory model," *Decision Sci.*, vol. 35, no. 2, pp. 289–322, 2004.
[72] R. W. Zmud, "Management of large software development efforts," *MIS Quart.*, pp. 45–55, 1980.

Amrit Tiwana received the Ph.D. degree from the Georgia State University Robinson College of Business, Atlanta, in 2001.

He is an Assistant Professor with the Iowa State University College of Business, Ames, Iowa. He was previously on the faculty of Emory University in Atlanta. His research focuses on knowledge management. His work has appeared and is forthcoming in the *California Management Review*, the *Journal of Management Information Systems*, IEEE TRANSACTIONS ON ENGINEERING MANAGEMENT, *Decision Sciences*, *IEEE Software*, *IEEE Internet Computing Magazine*, *Communications of the ACM*, *Decision Support Systems*, *Information Systems Journal*, the *Journal of Knowledge Management*, and others.

Mark Keil (M'02) received the DBA degree from the Harvard Business School, Boston, MA, in 1991.

He is the Board of Advisors Professor of Computer Information Systems at Georgia State University's Robinson College of Business, Atlanta. His research focuses on software project management, with particular emphasis on understanding and preventing software project escalation. His research is also aimed at providing better tools for assessing software project risk and removing barriers to software use.

Dr. Keil currently serves on the editorial boards of the IEEE TRANSACTIONS ON ENGINEERING MANAGEMENT, the *Journal of Management Information Systems*, and *Decision Sciences*.