

THE ONE-MINUTE RISK ASSESSMENT TOOL

An analysis of risks in software development, using data from senior IT managers, produced surprising results. Our one-minute assessment tool applies those results to assessing the risks of specific projects.



Of the \$2.5 trillion spent on IT during 1997–2001, nearly \$1 trillion was wagered on underperforming projects [3]. A large number of underperforming projects ultimately fail, costing U.S. companies more than \$75 billion each year [8]. While some events cannot be predicted or controlled, many of the risks that repeatedly plague software projects can be assessed and managed.

How do software project managers walk the fine line between calculated risks that can lead to innovative business solutions and outright gambling that can lead to career-ending projects? Existing software project risk frameworks do not provide the kind of simple, pragmatic tools needed to rapidly assess and manage risk. To date, the development of such tools has been hampered by the fact that we know precious little about the *relative* importance of the key drivers of software project risk. To remedy this, we examined 720 software project assessments by senior IT managers in 60 large companies. Some of our findings were a surprise because the top risk driver we uncovered is one that is often neglected by management, while the least influential risk driver is one

that managers often complain about the most. This article examines the relative importance of six key drivers of software project risk and introduces a one-minute risk assessment tool that can be applied to improve software practice.

Good software is hard to come by but easy to recognize: It solves the problem at hand and does what its users expected it would do. Most software project failures can be traced to one cause: The delivered solution does not fit the problem. The key to averting this is to deliver a system that embodies the relevant knowledge. Thinking of software as a product rather than a medium in which knowledge is embodied is often a prescription for failure, because it doesn't focus attention on the true risks associated

with software development [2, 4]. Instead, it is helpful to think of software development as a process of embodying technical knowledge and knowledge of customer needs into a coherent solution. Under this view, if we want to understand and manage the risks associated with software development, we must first focus on the translation of customer needs into project requirements and specifications. These specifications then guide the design, which is ultimately implemented in system functionality, features, and code.

Following this perspective, we focus on two types of knowledge that go into building any software system: technical knowledge and customer knowledge (that is, the needs that the software must address). While technical knowledge is critical for developing a system, customer knowledge is equally—if not more—important if a successful outcome is to be achieved. The problem is that most knowledge is held tacitly in the heads of developers and customers and is fragmented across the organization, making it difficult to coordinate its application. The key risk drivers that determine how well such knowledge can be embodied in the system can be organized into two categories:

- **Embedded knowledge drivers** that govern access to the relevant technical application domain, and customer knowledge. Key among these are technical similarity to previous projects, customer involvement, and stability of project requirements.
- **Execution coordination drivers**, or the mechanisms by which this knowledge is applied to a project. Key among these are use of an appropriate development methodology, formal project management practices, and coordination complexity.

The Seeds of Project Failure

As part of a multiyear research program on software project risk, we asked MIS directors in 60 organizations to make 720 separate project evaluations (see “How the Study was Conducted”). The purpose of the study was to understand the relative importance they ascribed to the software project risk drivers described earlier. The results revealed that the most critical risk driver was the choice of methodology—a result that we were not expecting—followed by customer involvement, use of formal project management practices, similarity to previous projects,

project complexity, and requirements volatility (see Figure 1).

Use of an inappropriate methodology. While the raging debate about methodologies assumes that one methodology is inherently superior, such judgments are unwarranted without consideration of the project context in which the methodology is applied [5]. It is not the chosen methodology per se that drives project risk but how well it fits a given project. All develop-

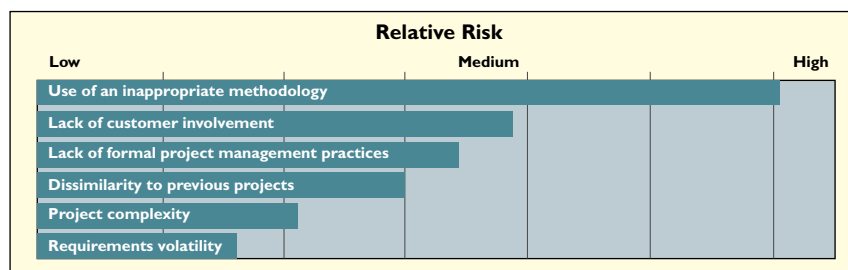


Figure 1. The relative weights associated with six key software project risk drivers.

ment methodologies—agile, lightweight, spiral, or structured—encapsulate *some* approach for embodying customer needs in the features and functionality provided by a system. For example, a methodology such as rapid prototyping relies on iteration to uncover novel or poorly understood user needs. In contrast, a structured methodology emphasizes structure over iteration and might be more useful for managing larger projects where requirements are better understood. A “one-methodology-fits-all” mentality can lead to uneducated choices that can raise rather than lower project risk. While better tools or heuristics are needed to help project managers assess project-methodology fit, it is perhaps easiest to think of fit as *lack of misfit* between the chosen methodology and what it must accomplish. A well-fitting development methodology can also help cope with other risk drivers such as project complexity and requirements volatility. This surprising finding clearly calls for more research attention in the software development community.

Lack of customer involvement. The second most important driver of project risk—customer involvement—is an inexpensive but underused form of project insurance. Project requirements are derived from expressed or inferred customer needs [1]. The accuracy with which these needs are understood determines how well the intermediate artifacts of the development process, such as requirements documents, specifications, features, and code, embody them. The problem is that customers often know more than they can tell: Their understanding of their own needs is “sticky” or difficult to articulate [9]. This creates the classic chicken-and-egg problem: A system cannot be built without requirements, and require-

Figure 2. The One-Minute Risk Assessment Tool.

ments cannot be fully elicited until a rudimentary system can be seen. Some approaches for involving customers in the development process include customer walk-throughs, prototypes, use cases, pilots, and user reviews. Feedback and ideas for refinement from involved customers can clear up misunderstandings and help align developers' and customers' visualizations of the system. Moreover, involved customers are likely to be more receptive to the project's outcomes, which lowers the risk of late-stage rejection [6].

Lack of formal project management practices. The third driver of project risk is the extent to which formal project management practices such as formal plans, schedules, budgets, and milestones are used. Some would still argue today that software development projects are fundamentally "different" from other types of projects and that formal project management practices somehow don't apply. Our findings strongly suggest otherwise, implying that formal project management practices have the power to reduce software project risk. The value of such practices lies largely in the well-defined patterns and directives that they create for coordinating interac-

tions and integrating inputs from various project constituents. Formal milestones also help in monitoring progress and spotting discrepancies throughout the project trajectory.

Dissimilarity to previous projects. The fourth driver of risk is a project's similarity to ones that have previously been completed in an organization. If a new project resembles previous projects, a company is likely to be familiar not just with the hardware, software, operating systems, programming languages, and application domain, but also with their constraints and the problems that might unexpectedly occur. Tried-and-tested heuristics, estimation models, design rationales, project plans, problem-solving approaches, and even code from previous projects can be readily repurposed [7]. The more a new project differs—technically and conceptually—from

❶ On a scale of 1-10, where 1 is low and 10 is high, how would characterize this project compared to other projects completed in your organization?

❷ Add the six weighted ratings (see the worked example).

❸ A lower overall project risk score indicates higher project risk. Range: 10 (most risky) to 100 (least risky).

❹ Use the table below as a guide for interpreting of this score.

Overall risk score →	10-28	29-46	47-64	65-82	83-100
Project risk level →	High	Moderately high	Medium	Moderately low	Low

Project Characteristic Question	❶ Rating	X	Weight	=	❷
Fit between the chosen methodology and type of project	5	X	3.0	=	15.2
Level of customer involvement	6	X	1.9	=	11.6
Use of formal project management practices	1	X	1.7	=	1.7
Similarity to previous projects	3	X	1.5	=	4.5
Project simplicity (lack of complexity)	7	X	1.1	=	7.4
Stability of project requirements	9	X	0.8	=	7.3
❸ Overall project risk score (higher score indicates lower project risk) →					48

How the Study was Conducted

We studied how project risk assessments are done in over 60 companies representing construction, manufacturing, chemicals, services (IT, financial, medical, and insurance), and publishing. We approached MIS directors in 590 companies to participate in the study, of whom 61 agreed (one response set was discarded from the analysis because of missing data). On average, these companies had revenues of approximately \$58 million and employed about 370 people. The executives had, on average, over 15 years of IT experience and had previously assessed the feasibility of 49 IT projects. The respondents for this study were therefore seasoned IT executives with considerable experience in assessing software projects.

Our analysis is based on 720 project evaluations for which these executives provided overall risk assessments. The analyses were conducted using structural equation modeling. The weights in the Risk Assessment Tool are based on the standardized regression coefficients derived from the 720 project assessments fitted to the model. Recall that these results are based on MIS managers' assessments, and the bias toward managerially controllable risk drivers might be a function of this stakeholder pool.

Further research is needed to determine whether other stakeholders, such as project managers or system users, share MIS directors' views, and incorporate other risk drivers such as organizational politics that were not captured here. The standardized regression coefficients for the six project attributes were -0.303 (use of appropriate development methodology), -0.194 (customer involvement), -0.172 (formal project management), -0.15 (similarity to previous projects), 0.11 (project complexity), and 0.08 (requirements volatility). These weights were proportionally rescaled and the project risk drivers rephrased to simplify the use and interpretation of the tool. Statistically, we have over 95% confidence that our findings are not due to chance. **C**

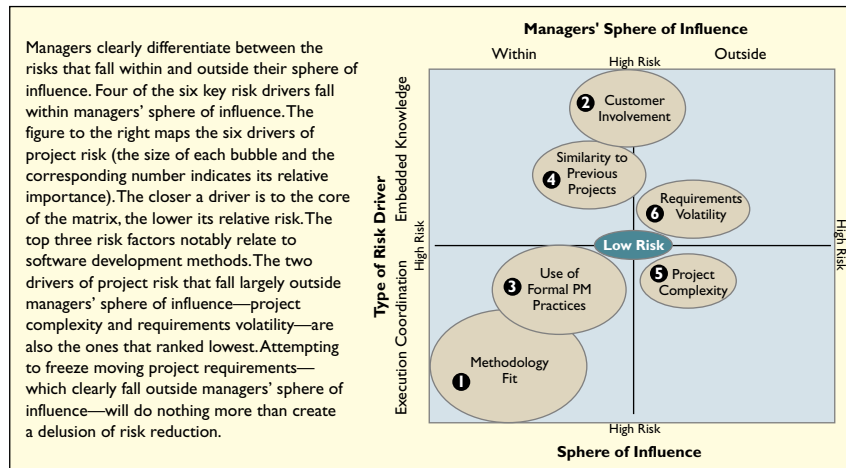
Our findings strongly suggest that formal project management practices have the power to reduce software project risk. The value of such practices lies largely in the well-defined patterns and directives that they create for coordinating interactions and integrating inputs from various project constituents.

previous projects in an organization, the greater its exposure to risk.

Project complexity. Project complexity emerged as the second lowest of the six drivers of project risk in our study. A project can involve both technical

functionality provided by a system. Building a system on volatile requirements is like attempting to build a structure on a foundation of sand. Tweaking an unfinished system to match shifting requirements requires additional programming effort and expensive reworking. Even if requirements

are correctly elicited, there is no guarantee that they will remain unchanged over the project trajectory. Both changing business environments and fickle customers can contribute to requirements volatility. A certain level of requirements volatility has come to be expected in software projects, which may explain why this factor emerged with the lowest weighting among the six drivers of project risk in our study.



and organizational complexity. Technical complexity stems, in part, from a system having to interact with a large number of other applications, interdependencies, and the complexity of the interfaces to other systems. Technical complexity raises the difficulty of estimating the resources that must be allocated to a project, exposing the project to feasibility, budget, and scheduling risks. Organizational complexity grows with the number of departments or organizations involved in and affected by the system. Organizational complexity raises coordination challenges and exposes the development process to unpredictable coordination failure. However, software project managers tend to accept complexity as a known driver of project risk over which they exercise little influence, which may explain its relatively low weighting in relation to other drivers of project risk.

Requirements volatility. Project requirements define the mapping between customer needs and the


Figure 3. Managerial Spheres of Influence

The One-Minute Risk Assessment Tool

How can a project manager use these findings to compare the riskiness of different software projects? A simple risk assessment worksheet derived from the empirical findings of our study can provide a quick-and-dirty assessment of overall project risk (see Figure 2). Inputs in the form of ratings for each risk driver question (the shaded boxes) from project managers and the relevant stakeholders for each project can be combined to provide a quick back-of-the-envelope assessment of each project's risk exposure. The score will range from 10 to 100. The lower the overall risk score for a project, the higher its risk exposure.

This one-minute risk assessment tool can be used to perform intuitive "what-if" analyses to guide managers in determining how they can proactively reduce software project risk. For example, what would happen if customer involvement were increased and for-

mal project management was minimal? Such assessments can be compared across different types of prospective projects. The tool can also be given to each of the key stakeholders in a project, allowing the project manager to bring out differences in perception or conflicting views that may create trouble later on if they are not resolved.

Only risk that is underappreciated and unmanaged has the power to surprise. While in any given situation, some risk drivers may be more controllable than others, we note with optimism that most—if not all—of the risk drivers fall within the project manager's sphere of influence (as illustrated in Figure 3). In particular, the most important drivers are ones over which managers can clearly exercise *some* control. Recognizing which risk drivers are amenable to managerial control and which are not is the key to walking the tightrope between calculated risks and outright gambling. Careful project planning and partitioning of attention can mitigate the former but rarely the latter. An understanding of the factors that drive risk and which ones can be influenced in a given situation can empower managers to have the serenity to accept the risks they cannot change, the courage to manage the risks they can control, and the wisdom to know the difference. 

REFERENCES

1. Alexander, C. *Notes on the Synthesis of Form*. Harvard University Press, Cambridge, MA, 1964.
2. Armour, P. A case for a new business model: Is software a product or a medium? *Commun. ACM* 43, 8 (Aug. 2000), 19–22.
3. Benko, C. and McFarlan, W. *Connecting the Dots: Aligning Projects with Objectives in Unpredictable Times*. Harvard Business School Press, Boston, 2003.
4. Faraj, S. and Sproull, L. Coordinating expertise in software development teams. *Manage. Sci.* 46, 12 (2000), 1554–1568.
5. Highsmith, J. The great methodologies debate part I: opening statement. *Cutter IT J.* 14, 12 (2001), 2–4.
6. Leonard-Barton, D. and Sinha, D. Developer-user interaction and user satisfaction in internal technology transfer. *Acad. Manage. J.* 36, 5 (1993), 1125–1139.
7. Rus, I. and Lindvall, M. Knowledge management in software engineering. *IEEE Softw.* (May/June 2002), 26–38.
8. Standish Group. *Chaos chronicles II*. West Yarmouth, MA, 2001.
9. Von Hippel, E. Lead users: a source of novel product concepts. *Manage. Sci.* 32, 7 (1986), 791–805.

AMRIT TIWANA (atiwana@bus.emory.edu) is an assistant professor in the Goizueta Business School at Emory University, Atlanta, GA.

MARK KEIL (mkeil@gsu.edu) is Board of Advisors Professor of computer information systems, J. Mack Robinson College of Business, Georgia State University, Atlanta.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

© 2004 ACM 0001-0782/04/1100 \$5.00
