

Software Risk Management: Dispatches from the Front

Robert N. Charette, ITABHI Corporation

A decade ago, when I finished my first of several books on software risk management, a good friend of mine wished me the best of success. He said I would need it since getting software developers to pay attention to risk was about as likely as getting his children to eat turnips. It has taken a while to acquire the taste, but the serving of risk management is becoming increasingly popular on software project menus.

This change in attitude towards risk management has been remarkable. As recently as 1992, Ed Yourdan asked the readers of *American Programmer* whether “risk management” should be a topic for a future issue. The most common reply he received was a perplexed, “What’s that?” Today, you want to know what’s new in the area. Pretty significant progress, by any measure.

The downside of this progress, however, is that risk management is also beginning to resemble a trendy *idea du jour*. It is hard to read a computing article without the word “risk” popping up a dozen times. To keep risk management from turning into another software engineering fruit fly, those who practice or intend to practice risk management must carefully manage expectations concerning what risk management can and cannot do. Risk management will not turn an unfriendly operating environment into a helpful one, will not change resource shortfalls into resource surpluses, nor guarantee a successful outcome every time. With this in mind, what follows are my personal observations on the current state of risk management practice, where it is now heading, and most importantly, what it

needs to do, if it is not to fall by the wayside from over-promises and over-expectations.

Needed Now, More Than Ever

Several factors account for the surge in software risk management's current popularity. The first is the recognition of the true magnitude of software's contribution to the success of an organization.¹

The last ten years in computing should rightly be termed the “age of reengineering.” Despite what you think about the efficacy of reengineering itself, it has forced organizations to ask themselves why they use software systems the way they do. I find organizations today have a much better understanding of how software contributes, or maybe better stated—should contribute—to meeting organizational objectives.

This understanding of how software and the business should align has also created a recognition that the opportunity costs of software failure, either in canceled development or unreliable operations are huge and growing. Numerous surveys continue to point out that most software projects are still either canceled, late, do not meet requirements and/or are unreliable. In 1995, for instance, the Standish Group estimated that US companies spent almost \$150b on canceled or late software developments alone.² The true costs of opportunities foregone are much higher, of course, since it is hard to determine the ultimate costs of losing market share due to software that is delivered late or lost customers due to poor software reliability and the like. The previous attitude that software overruns or cancellations were merely “the cost of doing business” is disappearing.

¹ We are using the term “software” in its widest possible sense in this article.

² Standish. *Chaos. Charting the Seas of Information Technology*. The Standish Group International, Dennis, Massachusetts, 1994.

Additionally, in case anyone forgets software's significance—or what happens when you forget to manage your risks—there are weekly articles in the mainstream press on the latest software implosion. Take, for instance, the case of Oxford Health Plans. Its billing system could not keep up with the company's growth, thereby leaving the company in the dark regarding its accounts payable or receivable. This ultimately forced Oxford last October to issue an unexpected earnings warning of a very large 3rd quarter loss. This resulted in the company's stock dropping by 63% (or \$3.4b in its market value) the largest percentage single-day loser in the history of the stock market.

Software's critical contribution to organizational success makes risk management needed now, more than ever.

Risk Management's Performance

The second factor in stimulating the demand for software risk management is its growing record of accomplishment, which in the last few years has reached a point where it could be quantitatively documented. We are starting to see project, organization, and industry data that demonstrates the value of risk management.

For example, Hughes Aircraft (now part of Raytheon Systems Corporation) used risk management on the very complex, four-and-a-half year, 750-person Peace Shield air defense system.³ The company's risk management approach, which it used aggressively as a problem-preemption strategy, helped deliver the project 10% ahead of schedule and significantly below projected cost. Peace Shield's success, which many believed was impossible at the time of its start, sparked the company to institutionalize risk management for use on its other major programs.

³ Sutherland, Chuck. "Peace Shield Risk Management." *Proceedings of the 5th SEI Conference on Software Risk Management*. Software Engineering Institute, Pittsburgh, Pennsylvania, 1997.

Rockwell Collins is a company that has been aggressively institutionalizing risk management since 1994. Collins's in-house risk management group provides a complete risk management consultancy service, from training to in-depth project support. Over 1200 managers and engineers participating in approximately 20 major programs involving 175 projects across several Collins divisions have received some formal risk management training. In a recent review, Collins determined that there is at least a 17% difference in the cost performance index (CPI) between projects that perform risk management diligently and those that don't.⁴ A study is underway to determine the impact of risk management on project schedule, but early indications are that projects that use risk management have the best schedule performance also. The perceived value of risk management is such that many of Rockwell Collins's customers and strategic partners are requesting help in starting their own risk programs.

Finally, data is now available on the value of different project management-related disciplines across different industries. A recent Project Management Institute (PMI) benchmarking study reported that while risk management is the least practiced of all project management skills (especially in the software community), it is strongly correlated with project success.⁵ In other words, benchmarking data is now indicating that if you want your projects to come in on time and within budget, applying risk management is an essential discipline to follow, regardless of industry.

Risk Knowledge Infrastructure

The third major factor for risk management's success is to those wishing to embark on implementing risk management can access an intellectual capital infrastructure

⁴ CPI is defined as the Budgeted Cost of Work Performed divided by the Actual Cost of Work Performed, or BCWP/ACWP.

of published theoretical knowledge tied to practical experience. Governmental initiatives are responsible for much of the current infrastructure being made available. For example, the Software Engineering Institute (SEI) published the results of its software risk management program in a book that provides a guiding process and a set of techniques to support continuous team risk management.⁶ Over the past several years, the UK's Government Centre for Information Systems has published a multi-volume risk management guidebook series that explains how to perform risk management at the enterprise, program, project and operational levels of an organization and across life-cycle activities. Additionally, several US government organizations, such as the Government Accounting Office (GAO), the Department of Defense (DoD), the Office of Management and Budget (OMB), have each published software-oriented risk management pamphlets.

Furthermore, where once there was only two software engineering books devoted to the subject of software risk management in 1989, now there are over a dozen in print and several more on the way. Dozen of articles devoted to risk management, such as these in *American Programmer*, have appeared in most major software periodicals over the past two years.

Along with the host of books and articles, both government and private organizations are offering a spectrum of software risk management training courses. Special interest groups focused on risk management in organizations like the International Council on Systems Engineering (INCOSE), the Society for Risk Analysis (SRA), and the PMI also exist where one can learn what is going on in the area.

⁵ Ibbs, C. Williams and Young-Hoon Kwak. *The Benefits of Project Management: Financial and Organizational Rewards to Corporations*. PMI Publications, Upper Darby, Pennsylvania, 1997.

⁶ Dorofee, Audrey et al. *Continuous Risk Management Guidebook*. Pittsburgh: Software Engineering Institute, 1996.

Finally, there is also a new wave of software risk management tools becoming available on the market. These tools range greatly in sophistication and cost, with many aimed specifically at identifying, evaluating and tracking risks that affect software projects or product developments. Additionally, many tools that support traditional risk assessment techniques such as decision trees, linear programming, and Monte Carlo simulation have been updated to support software risk management.

Not All Good News

While there are many positive signs on the risk management front, you should not be left with the impression that software risk management has become a main stream practice. As the PMI study indicates, the use of risk management on software projects is more the exception than the rule. Some reasons for avoiding risk management are personal, while some are culturally bound.

Personal aversion to performing risk management comes in various forms. For example, many software project managers already see themselves as actively managing risk, and view any formal process as an unnecessary overhead. There are other managers who believe their projects don't have any risk, by definition, because they have no choice but to succeed. Similarly, there are those who are in denial: "Risk happens to other projects, not mine." To admit risk is to admit not being in control.

Then there are the managers who know they have risks, but are willing to gamble that they will never occur on their project—a form of risk arrogance. Finally, there are the managers who honestly can't see the benefits gain from spending scarce resources looking for things that might not happen, and then spending even more resources to try to prevent things that might not happen from happening. Why not just spend the resources on the project itself, they ask?

Organizational culture can also inhibit the use of risk management.⁷ In many commercial organizations, uncertainty is viewed as bad, thereby making risk a subject not to be discussed. Further, worrying about risks before acting or making commitments is considered wimpish, while purveyors of “bad news” are seen as whiners. Crisis managers, on the other hand, are rewarded. Working hard and believing in the project is the recommended cure for potential problems in risk-averse organizations.

Government organizations are even more risk averse than commercial ones. The reason for this can be summed up in a comment I have repeatedly heard from senior government program managers around the world: “Highlighting program risk is the kiss of death.” This belief is not groundless. Often, it stems from the fact that to get their programs funded; they were required to sell it under the fiction that every one of its risks is already averted. This makes admitting risks afterwards a dangerous affair.

Then there is the issue that government-auditing groups everywhere seem to equate program risks with program problems, as well as to poor program management. For example, the GAO by law must report to the US Congress every year the government’s ten highest risk programs (which in reality are the government’s ten highest *problem* programs). These select few have the pleasure of seeing their programs minutely scrutinized by several congressional committees, as well as watching silently as their programs are pilloried in the press. This atmosphere is not conducive for convincing government program managers to begin risk management programs, especially when that information can be used against you.

Then there is the belief of many government program managers that risk management is just “excuse management” in disguise, especially when the risks are

⁷ Gemmer, Art. “Risk Management: Moving Beyond the Process.” *IEEE Computer*, May 1997, pp. 61–72.

being raised by government contractors. Government program managers hate to admit a risk exists, not only for the above reasons, but because many really believe a contractor will use it as a pretext for delivering the project late and over budget.

Two years ago, to encourage the application of risk management, the US Congress passed the Clinger-Cohen Act that among other things, made risk management mandatory on most government software programs. Given government culture, it will be interesting to see how many managers comply with more than lip service.

Overcoming the personal and cultural inhibitions to risk management is hard, especially for those who really want to within government, and the risk management community needs to find ways to address them in an effective manner. My admittedly optimistic belief is that as more projects use risk management to become successful, the cultural and personal aversions to risk management will begin to decline, at least in the commercial sector.

In government, removing as a crime punishable by program cancellation the open discussion of risk would go a long way to achieving the goals of the Clinger-Cohen Act. The GAO changing the name of its “High Risk Program” to “Severe Problem Program” would also help, especially in getting Congress to understand the difference between a risk and a problem. This assumes, however, that some of the shortcomings in today’s software risk management practice are rectified relatively soon.

Weaknesses in the Practice

There are many aspects of current software risk management practice that, if not rectified, will begin to injure risk management’s credibility and thereby keep from reaching its full potential. The greatest threat I see is the proliferation of quick-and-dirty (Q&D) approaches to performing risk management. While sometimes

these are well meaning attempts to get an organization or software project on the path to using risk management, they are more likely to push the project over a cliff.

There is an old rule of thumb that states that everything should be made as simple as possible, but no simpler. This rule seems to get broken quite regularly in respect to implementing risk management on software projects, at least given the number of risk management programs I keep untangling.

In too many software projects, risk management is approached in an over simplistic manner. Projects are told that all they have to do is just brainstorm potential problems for an hour or two, create a list of the perceived top ten risks or so, and then come up with a plan to avert each one. Follow-up to this analysis, if any, needs to occur only at major milestones or the like. While this approach can work if your project lasts two weeks or less, it is a complete waste of time for anything remotely complex.

Risk management practice is hard work when performed even to minimum standards. It takes time to understand exactly what the risks are and how they relate to the objectives, assumptions and constraints of the software project, all of which are typically vague or ill defined. Q&D approaches tend to underestimate the number and severity of the risks, especially the potentially high impact on the project of large numbers of low consequential risks.

Worse, problems are identified frequently as risks. Unthinking use of risk taxonomies as the means for identifying risks is a big culprit. An example I keep that encountering is the so-called “requirements volatility risk.” Requirement volatility as such is *not* a software project risk—it is a *problem*. Its likelihood on every software project I have ever been associated with is 1. The *risk* I need to grasp is the *specific* requirement that is subject to change, along with its implications. Q&D taxonomic-driven risk identification approaches tend to be at such a high a level of granularity that little useful knowledge for making decisions about risk is ever produced.

It takes even more time to understand the subtleties of how risks interact with one another, how the attempt to avert one risk will cause existing ones to change, or worse, create new risks. It also takes time to get the prioritization of the highest risks correct so risk aversion resources are allocated effectively. In practice, not making the present situation worse in the project is a very difficult objective to achieve. Q&D approaches emphasize the identification of risks, which is the easy part, at the expense of actions required to manage them, which is the important part.

Much effort also has to go into determining what is an acceptable level of risk, and how it needs to change over time. Without a level of acceptability defined, how can you decide what to do about a risk, or whether something is even a risk? Defining risk acceptability is one of the hardest things to do in my experience. Q&D approaches avoid the issue entirely. Further, how accurate are the estimates of a risk's likelihood, consequences, and timing and how does that accuracy affect the outcome of the analysis? I have seen few Q&D approaches that even bother with estimation issues.

None of these issues above are part of advanced risk management, but are fundamentals that should be followed. Glossing over or skipping fundamental principles is not only dangerous, but also professionally irresponsible if those who are making decisions based on the risk information produced are not informed of the underlying process limitations.

Accelerating the move towards the implementation of Q&D approaches is the desire to use risk management tools before one understands what is involved in performing comprehensive risk management. Relying on tools to do risk management begs the CASE fiasco to be relived again. Remember that no risk management is better than risk management poorly done. At least you won't be fooling yourself that you think you understand the risks when you really don't.

Future Directions in Risk Management

There are many improvements on the horizon that should make risk management practice more effective. The connection of software measurement to risk management is one that should make the control and monitoring aspects of risk management much stronger, as well as help address the issues of risk estimation and risk acceptability definition. The DoD-sponsored Practical Software Measurement (PSM) initiative is taking a lead in making this occur.⁸

I see future software risk management practice supporting all aspects of the software life cycle, from concept definition to maintenance, in an integrated fashion. Today, risk management is still rather stove-piped across the software life cycle, with certain activities, such as maintenance, given short shrift from a risk management perspective. I also see the various software risk management specialties of security, safety and operational recovery becoming more merged into a single general practice.

Seamless risk management, or the integration of software, systems and business risk management activities, something I have been advocating for over a decade, will grow tremendously in the next few years. Already, several major consulting firms over the last year, like Andersen, KPMG and Ernst & Young, have created “one-stop” risk management divisions to address the full range of an organization’s risk management’s need. Chief Risk Officers (CRO) or “crows,” will soon take their place alongside CIOs and CFOs.

More work will be done to improve risk management communication among management peers as well as between management and the software development team. This will require streamlined risk management processes that can support the individual software developer who is making risk-laden decisions daily. This

⁸ PSM. Practical Software Measurement. OUSD(A&T) and the Joint Logistics Commands Joint Group on Systems Engineering, Ver. 2.2, 27 March 1996.

will in turn require more extensive risk management training involving the software development team as a whole, as well as support tools to that emphasize risk communication more than risk assessment.

I also see risk entrepreneurialism, or how to turn an organization's risks from liabilities into assets, growing in importance.⁹ The creation of profit comes from being able to take on risk. Risk management today is aimed at reducing costs. Tomorrow it will be aimed at creating new business opportunities.

Finally, I see software risk management getting more entangled with, and possibly brought under many companies' legal departments. The infamous Y2K problem is almost certain to spawn a plague of lawsuits. The first question the lawyers are going to be asking is did your organization know about the Y2K issue, and if so, where is your risk management plan showing how you were going to prepare for it? If you have one, you might stand a chance in court. If you don't, well, it won't be pretty. I think Y2K will create the last major impetus needed for institutionalizing software risk management within organizations over the next few years.

Conclusions

Risk management is a hot bed of activities, and I have only touched on a small portion of what is happening, both good and bad. As with all new software disciplines, it is an exciting area to be involved in, with a lot of work to still be done.

However, we are approaching a crossroad in this emerging discipline. We should never forget that risk management is not an end in itself—but a means to help people make better decisions. If we are not careful about how risk management is practiced—if it is performed sloppily, if its weaknesses are not clearly expressed, if unrealistic expectations are raised—it will quickly lose its

⁹ Charette, Robert N. "On Becoming a Risk Entrepreneur." *American Programmer*. March 1995, pp. 10-15.

credibility. Then, in the near future, we can all spend our time reading articles entitled, “Risk Management: Just Another Software Engineering Fad.”

*This article first appeared in Cutter IT Journal magazine in June 1998.
Copyright © 1998 ITABHI Corporation. All rights reserved.*