ELSEVIER

Project risk management: lessons learned from software development environment

Y.H. Kwak [a,*], J. Stoddard [b]

[a] *Project Management Program, Department of Management Science, Monroe Hall 403, School of Business and Public Management, The George Washington University, Washington, DC 20052, USA*
[b] *Agilent Technologies, 2679 Monument Drive, Santa Rosa, CA 95407, USA*

Abstract

The challenges and realities in applying effective software risk management processes are difficult, in particular integrating the risk management processes into software development organizations. However, the benefits of implementing effective risk management tools and techniques in software development project are equally great. Current perceptions and emerging trends of various software risk management practices are reviewed and risks specific to software development projects are identified. Implementing effective risk management process will succeed by changing the organizational culture. This paper addresses lessons learned from implementing project risk management practices in software development environment.
© 2003 Elsevier Science Ltd. All rights reserved.

Keywords: Risk management; Software development; Project management; Technology management; Lessons learned; Organization

1. Software project environment

Project failures are the result of the multiplicity of risks inherent in software project environment. Software development projects are collections of larger programs with many interactions and dependencies. It involves a creation of something that has never been done before although the development processes are similar among other projects. As a result, software development projects have a dismal track-record of cost and schedule overruns and quality and usability problems. Jiang and Klein (1999) find different types of risks will affect budget, user satisfactions, and system performance. Other studies indicate that 15 to 35% of all software projects are cancelled outright, and the remaining projects suffer from schedule slippage, cost overruns, or failure to meet their project goals (Boehm, 1991) (Klein, 1998).

Time-to-market is the most critical factor for consumer in developing commercial software products. However the project success is difficult to predict because project scope is changed by continuous market requirements and resources are constantly being reallocated to accommodate latest market conditions. Projects for specific customers also have a large degree of uncertainty for requirements due to the customized technical attributes. As a result, software development engineers have high turnover rates among software development firms. For example, software managers in India perceived personnel turnover as their biggest source of risk (Boehm and DeMarco, 1997).

Many software projects and programs involve multiple entities such as companies, divisions, etc., that may have certain interests. There is often a feeling of disconnection between software developers and their management, each believing that the others are out of touch with reality resulting in misunderstanding and lack of trust. Research shows that 45% of all the causes of delayed software deliverables are related to organizational issues (van Genuchten, 1991).

2. Software risks and risk management perceptions

Current perceptions about risk management from majority of software project organizations contributes to the lack of project stability in addition to the inherent

---

* Corresponding author. Tel.: +1-202-994-7115; fax: +1-202-994-2736.

E-mail address: kwak@gwu.edu (Y.H. Kwak).

challenges posed by the nature of software projects. Kwak and Ibbs (2000) identified risk management as the least practiced discipline among different project management knowledge areas. Boehm and DeMarco (1997) mentioned that "our culture has evolved such that owning up to risks is often confused with defeatism". In many organizations, the tendency to 'shoot the messenger' often discourages people from bringing imminent problems to the attention of management. This attitude is the result of a misunderstanding of risk management.

Boehm (1991) identified 10 software risk items to be addressed by software development projects:

- Personnel shortfalls
- Unrealistic schedules and budgets
- Developing the wrong functions and properties
- Developing the wrong user interface
- Gold plating (adding more functionality/features than is necessary)
- Continuing stream of requirements changes
- Shortfalls in externally furnished components
- Shortfalls in externally performed tasks
- Real-time performance shortfalls
- Straining computer-science capabilities.

Jones (1998) further presented three key software risk factors and concerns of both executives and software managers.

- Risks associated with inaccurate estimating and schedule planning
- Risks associated with incorrect and optimistic status reporting
- Risks associated with external pressures, which damage software projects.

However, most software developers and project managers perceive risk management processes and activities as extra work and expense. Risk management processes are the first thing to be removed from the project activities when the project schedule slips. The free-spirited culture in many software development firms is in conflict with the amount of control often required to develop complex software systems in a disciplined way. Jones (2001) mentioned "it is a peculiarity of IT that very complex systems can be built with a very low level of control by clever, driven people". Many software development practitioners understand risk management and control as inhibiting creativity.

3. Various software development risk management processes

Despite the inherent risks associated with software development projects, there are strong indicators that these risks can be managed successfully. Research of failed software projects showed that "their problems could have been avoided or strongly reduced if there had been an explicit early concern with identifying and resolving their high-risk elements" (Boehm, 1991). Effective risk management is the most important management tool a project manger can employ to increase the likelihood of project success. Since risk management is not widely used and understood, this could be a significant competitive advantage to those that implement the risk management processes in their projects.

A large number of processes have been generated in recent years to address the need for more effective risk management. The risk management process provided in the PMBOK® (PMI, 2001) is a good overview of the typical processes, yet it is often too generic to meet the specific needs of software projects. The Software Engineering Institute (SEI) has developed the Team Software Process™ (TSP™) for the team as a whole, and the Personal Software Process™ (PSP™) for the individual during software project development (SEI, 2001). Keshlaf and Hashim (2000) have developed models for tools to aid the software risk management process. As shown in Fig. 1, it uses an eight-step process during the initial phases of the project. When any new risks are identified throughout the project, a five-step inner process is used to improve earlier estimates and judgments continuously.

'Team risk management' is a process that addresses the risks associated with multiple entities (Higuera et al., 1994). Although developed specifically for software contractual relationships, the concept is just as viable for multiple divisions or multiple projects, which is a common paradigm in most organizations. Table 1 shows the principles involved with team risk management and the requirements for effective risk management in the process.

The 'team reviews' section is the principal process that makes the team process unique from other general
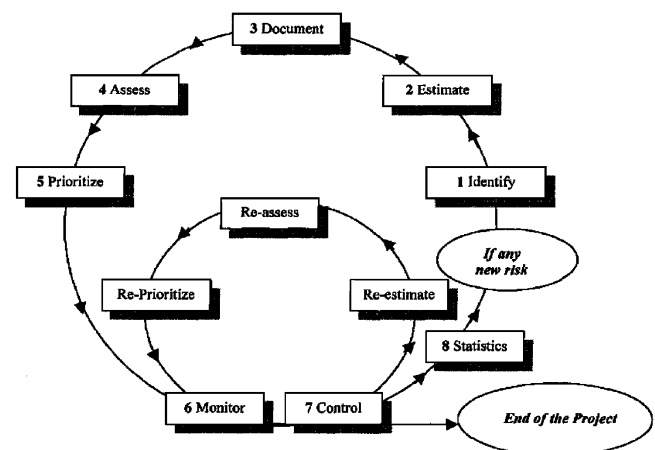

Fig. 1. 'Soft Risk' model's diagram (adapted from Keshlaf and Hashim (2000)).

Table 1
Nine principles of team risk management (adapted from Higuera et al. (1994))

Principle	Effective risk management
Shared product vision	A shared vision for success based upon commonality of purpose, shared ownership, and collective commitment.
Forward-looking search for uncertainties	Thinking toward tomorrow, anticipating potential outcomes, identifying uncertainties, and managing program resources and activities while recognizing these uncertainties.
Open communications	A free flow of information at and between all program levels though formal, informal, and impromptu communication and consensus-based processes.
Value of individual perception	The individual voice, which can bring unique knowledge and insight to the identification and management of risk.
Systems perspective	That software development is viewed within the larger systems-level definition, design, and development.
Integration into program management	That risk management is an integral and vital part of program management.
Proactive strategies	Proactive strategies that involve planning and executing program activities based on anticipating future events.
Systematic and adaptable methodology	A systematic approach that is adaptable to the program's infrastructure and culture.
Outline and continuous processes	A continuous vigilance characterized by routine risk identification and management activities throughout all phases of the life cycle of the program.

processes. The objective of team risk management is to share the risk responsibility and burden to effectively lower the risk of all entities involved. Fig. 2 shows the interaction of the team risk management processes (Higuera et al., 1994).

Yacoub and Ammar (2002) described a heuristic risk assessment methodology that uses dynamic metrics obtained from Unified Modeling Language (UML) specifications to determine the most risky components of the software architecture. It is mathematical analysis models derived from the UML diagram, and enables more attention to be placed in the areas of the system with highest risk.

Even the simplest mathematical models contain at least a rating scheme allows managers to quantify and prioritize the risks during the process. A concept used by many of these rating schemes is risk exposure

(Risk exposure = Probability (Loss) × Size (Loss)). Boehm and DeMarco (1997) showed that the probability component of risk exposure for employee turnover (one of the highest risk elements of most software projects) can be reduced by:

- Empowering performers
- Teambuilding
- Establishing significant incentive bonuses for successful project completion
- Recognizing outstanding efforts
- Structuring career paths around an organizations product lines.

Furthermore, the size component of risk exposure can be reduced by:

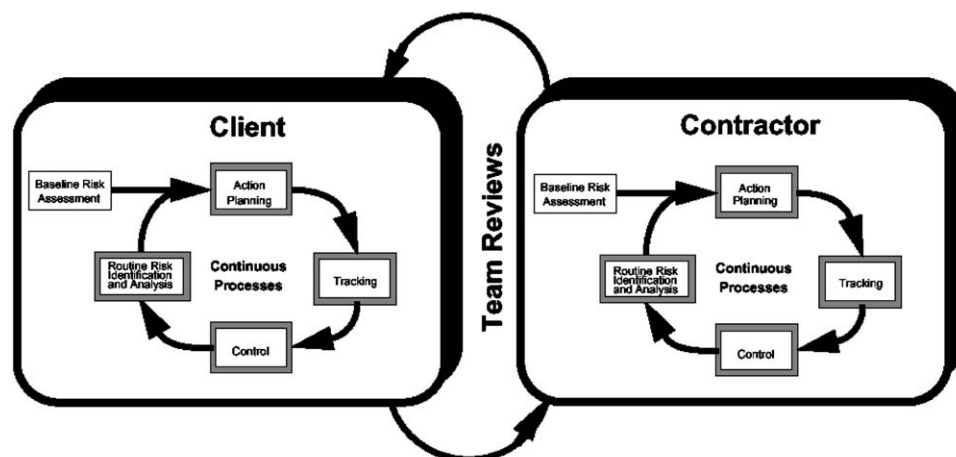- Implementing software inspections to reduce defects

Fig. 2. Concurrent team risk management process (adapted from Higuera et al. (1994)).

and spread knowledge of product components among various people

- Using egoless programming
- Clean room techniques
- Modular software architectures
- Encapsulation
- Good configuration management.

Although the foregoing strategies help to overcome the risk exposure to many risks in developing software, this is simply called good programming. Good programming methods reduce risk as well as have a positive affect on other project facets such as development time, team morale, and product quality.

4. Practical implementation and its results

The major obstacle to overcome regarding people is human nature, which encompasses many facets. Jiang et al. (2001) proposed a model that relates sources of risk to strategies and success. The result indicated that strategies involving behavioral aspects tend to be more influential in risk reduction than are those aimed at technical risks. Jiang et al. (2000) further revealed that effective project teams reduce technical risks involved in system development. Schmidt et al. (2001) developed an authoritative list of common risk factors and deployed a rigorous data collection method called a 'ranking-type' Delphi survey to produce a rank-order list of risk factors.

The practical implementation of the foregoing processes is one of the most challenging aspects of risk management. Conrow (2000) observed that "one of the biggest problems with many risk management processes is that one or more process steps is either missing, weakly implemented, or out of order". Conrow's assessment is meaningful; however, it is not very helpful until the root causes are identified for these results. It is helpful to identify the fundamental elements of any software project which are people, and their environment in finding the root causes.

4.1. Process vs. project obstacle

First, it is important to understand the differences between project and process. Table 2 compares between project and process management. Project risk management must be a natural part of the software development process to be most effective. Hall (1998) noted that, "There is no 'risk season' or a separate team to perform risk management. Risk management is integrated by distribution into routine project activities". Each team member must use the risk management process as a tool during project execution although a project itself is not a process. Ideally, project team members will be able to adapt risk management practices to their particular pro-

Table 2
Comparison between project management and process management

Project management	Process management
Project has fixed duration	Processes are continuous and endless
Resources are allocated to the project	Resources are allocated to functional group
Performance is measured for the project	Performance is measured by operation
Project is one of a kind, and unique	Work is replicated and repeated

ject environment if team members are well trained. Recent literatures emphasize the importance of educating risk management tools and techniques to software development community (Boehm and Port, 2001; Fuller et al., 2002). Project team members will benefit by learning and practicing their risk management skills to execute their project activities.

4.2. Human nature obstacle

Project team members may have the necessary skills to employ a risk management process; however, this does not guarantee that the team will use it during the project lifecycle. Gemmer (1997) described "Effective risk management requires obtaining functional behavior (as opposed to dysfunctional behavior), not just following a process or having diverse sources of information." Furthermore, he observed "behavior is due primarily to the organization's history, structure, processes, and reward system." Table 3 shows the difference between observed behavior and the ideal behavioral goal. For example, the natural tendency of software developers is to withhold technical information since information is a source of power. However, if these developers are rewarded and held accountable for sharing knowledge, then this tendency can be overcome.

4.3. The magnitude of the challenge

Even in organizations with the best processes, skills, and organizations that motivate team members towards effective risk management, the uncertainties resulting from the sheer magnitude of software project complexities can make managing risk a daunting task, because of the imperfections of human judgment. Hall (2001) noted that "In all these procedures and processes, there stands out one major source of error—the subjective nature of the risk assessments and assigning of probability of occurrence to specific risks within a program or project". Hall (2001) proposed a cyclic monitoring process to obtain better assessment accuracy throughout the project. Repetitive, continuous improvement is critical to achieving long-term success in managing risk.

Table 3
Cultural norms (dysfunctional) vs. effective risk management behavior (adapted from Gemmer (1997))

Dysfunctional behavior (observed)	Functional behavior (goal)
View each person's decision-making capability as invariant	Manage risk as an asset
View uncertainty as a negative	Treat decision making as a skill
Don't ask for risk information	Create a pull for risk information
Don't bring forward risks or problems without solutions	Seek diversity in perspectives and information sources
Ignore the soft stuff	Minimize uncertainty in time, control, and information
Be risk averse	Recognize and minimize bias in perceiving risk
Make decisions based on emotion, rather than logic	Plan for multiple futures
Make commitments without determining the probability of success	Be proactive
Be reactive	Make timely, well-informed decisions and commitments
Reward heroes	Reward those who identify and manage risks early, even if the risks become problems

Effective risk management practices in software environments are not developed instantly. A collaborative development effort between the Department of Defense, an external contractor, and the Software Engineering Institute (SEI) to transition SEI's Team Risk Management process into practice showed that the initial transition period spanned more than 18 months (Gluch et al., 1996). In most cases it will take even longer to develop a culture that embraces effective risk management.

5. Lessons learned

Roberts (2001) suggested the following lessons learned for project risk management.

- The greatest risk driver is often overlooked.
- Inappropriate attention may be given to one risk driver over others.
- Often a risk driver will impact all facets of risk (cost, schedule, technical, etc.) and the integrated result will be improperly estimated.
- Often risks are managed by lists which are ranked by subjective qualitative measures resulting in excessive expenditure of risk management resources.
- Often risks are managed by lists which are ranked by subjective qualitative measures resulting in excessive expenditure of risk management resources.
- Risk identification is the most critical step in risk management, yet often is poorly done.
- 'Faster, better, cheaper', as well as any other competitive initiative, exacerbates risk.
- All projects must include sufficient resources in the project's planning activity to adequately provide for training in the risk management process.
- A strong position must be taken by project management to enforce participation in the process.
- Validation of the tools and input data needs to be done early in the life cycle of the projects. The project manager needs to conduct this validation.
- Expect the risk management process to evolve with the project and its ever-maturing needs.
- The indirect benefits of integrated, quantitative risk analysis due to the demand for quality project management data are as valuable as the direct benefits.

The authors suggest the followings as lessons learned for implementing project risk management tools and practices to software development projects.

- A project manager does not always have the time to implement a formal process into the system. The project manager must be able to train team members 'on the fly' when the need arises.
- This is where a project management can add value to a project. Anyone can pay lots of money to hire consultants and take training classes, but usually only bureaucratic institutions like the federal government can afford it.
- Senior management holds the key to establishing an organization that encourages 'functional' risk management behavior.
- A documented process does not guarantee the process will be followed.
- As the size and complexity of the project increases, the effort for risk management increase exponentially.
- During project 'crunch time', the tendency is to focus solely on short-term objectives while neglecting long-term risks. These actions cause problems and cost more than anticipated and put the organization into a reactive mode that is difficult to reverse.
- People must be able to evaluate themselves and be motivated to act on their evaluations to change their behaviors.
- Real changes must occur in both management of the organization and behavior of individuals before risk management will improve. There are often 'feel good'

exercises where project evaluations or retrospectives are made, but no real changes occur.

- The people that are actually doing the development work on the project team must be empowered as well as have knowledge and motivation to change practices. Throwing a manager or a 'focus group' at a problem is not effective in solving the issues of poor risk management practices, unless they are empowered to affect organizational changes and train project team members in better risk practices.

6. Conclusions

Although software risk management is a daunting task, organizations that implement effective processes proved to be successful, while those that fail in this effort will be unsuccessful. The nature of software projects creates many risks that must be managed diligently to avoid the common drawback of many projects. The perceptions and attitudes towards risk management activities compound difficult challenges for implementing a risk management strategy.

Formal risk management process is recommended to manage complex issues associated with software development projects. Many risk management processes have been created to aid organizations, but integrating the processes into organizations was not successful. The theoretical aspects of the process must be reconciled with the practical challenges of the organization to implement risk management successfully. Effective risk management process will succeed by changing the organizational culture to motivate the individual. Cultural changes require time and repetition before they are firmly embedded into the organization.

References

Boehm, B.W., 1991. Software risk management principles and practices. IEEE Software 8 (1), 32–41.

Boehm, B.W., DeMarco, T., 1997. Software risk management. IEEE Software 14 (3), 17–19.

Boehm, B., Port, D., 2001. Educating software engineering students to manage risk. In: Proceedings of the 23rd International Conference on Software Engineering,, pp. 591–600.

Conrow, E.H., 2000. Effective risk management: some keys to success. American Institute of Aeronautics and Astronautics.

Fuller, A., Croll, P., Di, L., 2002. A new approach to teaching software risk management with case studies. In: Proceedings of the 15th conference on Software Engineering Education and Training,, pp. 215–222.

Gemmer, A., 1997. Risk management: moving beyond process. Computer 30 (5), 33–43.

Gluch, D.P., Dorofee, A.J., Hubard, E.A., Travalent, J.J., 1996. A col-laboration in implementing team risk management. In: Technical Report, CMU/SEI-95-TR-016. Software Engineering Institute.

Hall, D.C., 2001. Development program risk management: a case study. In: 2001 INCOSE Proceedings of a Symposium on Risk Management,, 3.

Hall, E.M., 1998. Managing Risk: Methods for Software Systems Development. Addison-Wesley.

Higuera, R.P., Gluch, D.P., Dorofee, A.J., Murphy, R.L., Walker, J.A., Williams, R.C., 1994. An introduction to team risk management. In: Special Report, CMU/SEI-94-SR-1. Software Engineering Institute.

Jiang, J.J., Klein, G., 1999. Risks to different aspects of system success. Information and Management 36 (5), 263–272.

Jiang, J.J., Klein, G., Discenza, R., 2001. Information system success as impacted by risks and development strategies. IEEE Transactions on Engineering Management 48 (1), 46–55.

Jiang, J.J., Klein, G., Means, T.L., 2000. Project risk impact on software development team performance. Project Management Journal 31 (4), 19–26.

Jones, C., 1998. Minimizing the risks of software development. Cutter IT Journal 11 (6), 13–21.

Jones, G.F., 2001. What is different about it risks. In: 2001 INCOSE Proceedings of a Symposium on Risk Management,, 3.

Keshlaf, A.A., Hashim, K., 2000. A model and prototype tool to manage software risks. In: Proceedings of First Asia–Pacific Conference on Quality Software,, pp. 297–305.

Klein, S.A., 1998. Putting methodology in perspective from a project risk viewpoint. In: IEEE Power Engineering Society 1999 Winter Meeting, vol. 1,, pp. 362–365.

Kwak, Y.H., Ibbs, C.W., 2000. Calculating project management's return on investment. Project Management Journal 31 (2), 38–47.

PMI, 2001. A Guide to the Project Management Body of Knowledge (PMBOK® Guide) 2000 Edition, Project Management Institute Publications.

Roberts, B.B., 2001. Lessons-learned: round 2. In: 2001 INCOSE Proceedings of a Symposium on Risk Management,, 3.

Schmidt, R., Lyytinen, K., Keil, M., Cule, P., 2001. Identifying software project risks: an international Delphi study. Journal of Management Information Systems 17 (4), 5–36.

SEI, 2001. Carnegie Mellon SEI Annual Report FY2001.

van Genuchten, M., 1991. Why is software late? An empirical study of reasons for delay in software development. IEEE Transactions on Software Engineering 17 (6), 582–590.

Yacoub, S.M., Ammar, H.H., 2002. A methodology for architecture-level reliability risk analysis. IEEE Transactions on Software Engineering 28 (6), 529–547.

Young Hoon Kwak is currently assistant professor of project management at the management science department at The George Washington University (GWU). He has received his M.S. and Ph.D. in engineering and project management at the University of California at Berkeley. Before joining GWU, he has spent a year at MIT as a post-doctoral scholar. Dr. Kwak has numerous publications in both academic and practitioner journals related to project management, risk management, and technology management. He has also consulted for Fortune 500 companies for assessing the maturity level of project management. For more information, visit his homepage at http://home.gwu.edu/~kwak.

Jared Stoddard is currently product manager for consumer and wireless solutions test division at Agilent Technologies. He received his B.S. in Electrical Engineering from Brigham Young University and M.S. in Project Management from The George Washington University. Jared has written and presented various technical papers in wireless semiconductor test technology. His most recent research effort has been is in the area of 'product generation excellence' at Agilent, including the project management of both software and hardware systems development.