What is Software Risk Management?

(And why should I care?)

Peter Kulik, **KLCI, Inc.**

1st Edition, October 1996
2nd Edition, June 2000

Risks are schedule delays and cost overruns waiting to happen.

As industry practices have improved, recognition of Software Risk Management has grown dramatically. Proven results from Software Risk Management include higher productivity, more consistent attainment of customer commitments, and improved business results.

The practice of Software Risk Management includes both top-down and bottom-up perspectives. Projects that implement Software Risk Management become simpler and more focused. Further, a variety of tools are available for easier implementation.

This white paper presents an overview of Software Risk Management practices and tools.

According to Edward Yourdon, high-productivity software development groups are as much as *600 times* more productive than low-productivity groups [1]. Key factors driving productivity improvements in the most productive groups have been adoption of advanced tools and software management processes. Low productivity groups, on the other hand, continue to repeat software development mistakes of the past.

Why should you care about risk management? *Risks are schedule delays and cost overruns waiting to happen.* Failure to manage project risks makes a business less competitive, by causing *unnecessary* quality, schedule, and functionality tradeoffs – and cost overruns. If you are not managing project risks, your organization will most likely be relegated to sub-par productivity and above-average rates of project failure.

Software Risk Management is a key component of the advanced software processes adopted by the highest productivity software groups. It involves assessing overall project risk and identifying, prioritizing, and proactively managing specific risks. Risk management practices block schedule delays and cost overruns before they can impact a project. A number of very good publications on the subject are available [2, 3, 4, 5]. In addition, the Software Engineering Institute (SEI) holds annual conferences devoted to Software Risk Management [6].

Projects using Software Risk Management to manage their risks have realized benefits including:

- Increased programmer productivity
- Fewer “surprises”
- Better attainment of customer commitments

If your organization has projects larger than 10 to 15 people – and you are not implementing Software Risk Management – you are missing opportunities to improve your productivity and bottom-line business results.

Figure 1

Definition of Software Risk Management

As shown in Figure 1, Software Risk Management includes the following aspects:

- Identify and Analyze
- Mitigate and Control
- Early Warning System

Identifying and analyzing risks includes top-down and bottom-up aspects. Top-Down Risk Management measures overall project risk, while Bottom-Up Risk Management identifies the specific risks that drive the overall project. An example of a top-down risk profile is shown in Figure 2 [7].

Mitigating and controlling risks involves proactive actions to block risks before they impact a project. Best practice organizations control risk mitigation by including specific tasks right in their project schedule.

An "early warning system" allows new risks to be identified and risk mitigation actions put in place for these risks [8].

Why Software Risk Management?

At first glance, Software Risk Management might appear to just add complexity to an already complex undertaking. In reality, however, the activities listed above make software projects *less complex*:

- Identifying and prioritizing risks enables project managers and project staff to *focus* on the areas with the most impact to their project.
- Appropriate risk mitigation actions reduce overall project risk – which actually *accelerates* project completion.
- Projects that finish sooner cost less, plus risk mitigation actions can further reduce project cost.
- Projects using Software Risk Management experience *fewer "surprises"*, since they have identified (and in many cases, eliminated) root causes of surprises before they can occur.

In sum, Software Risk Management helps projects secure their customer commitments. Further, managers and project staff utilizing Software Risk Management have a better overall understanding of their project and make better business decisions.

Identify and Analyze - Top-Down

This step provides a top-down perspective on project risk, and determines an overall risk framework for a project. Models such as Figure 2 enable informed decisions about schedule commitments and contingency. Example tools to develop a risk framework include:

1. $\text{TDM}_{\text{schedule}}$ metric [6]
2. SLIM from QSM [http://www.qsm.com]
3. Project schedule simulation [2]

An example of the $\text{TDM}_{\text{schedule}}$ metric is shown in Figure 2. In this example, a schedule commitment of 12-Feburary would be only 10% % likely to complete on time. A commitment of 18-June, however, would be 90% likely to be successfully met. Given this risk profile, what customer commitment would you be willing to make on this project?

Top-down risk estimates can also reflect the impact of risk management actions; as risk mitigation actions

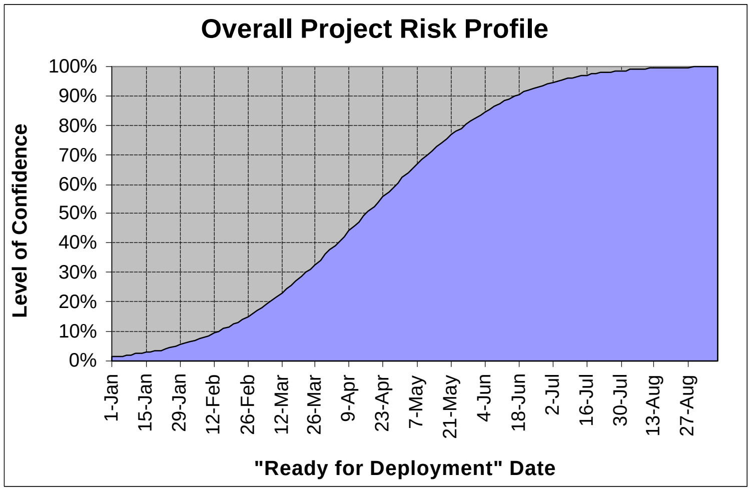

Figure 2

Tools for Risk Management

- ✓ **KLCI's Project Self-Assessment Kit**
  - **Applies to all projects with 7 or more team members.**
  - **Includes:**
    - **Top-Down Schedule Profile**
    - **Risk Identification**
    - **Customized Mitigation Actions**
- ✓ **Risk Assessment**
  - **Detailed Assessment offered by a variety of independent vendors**
    - **Generally applies to projects of greater than US $750,000 total budget**
- ✓ **Checklists**
  - **SEI Taxonomy of Risks [6]**
  - **Others [3, 4, 5]**

Figure 3

reduce project risk, the risk profile will shift *up* – reflecting greater confidence in achieving earlier completion dates!

The SLIM tool from QSM [http://www.qsm.com] uses historical data on literally hundreds of software projects as an aid to estimation and management. Results include risk-weighted profiles for schedule and effort to complete a particular project.

Simulating the project schedule can be accomplished using a tool such Risk+ from Program Management Solutions, Inc. Based on probability distributions for individual tasks, simulation will construct a statistical model of project risk [2]. For simulation to provide useful results the project schedule must be accurate and complete; missing tasks, underscoped or overscoped tasks, and missed or invalid task or resource dependencies will result in GIGO (Garbage In, Garbage Out) results.

Identify and Analyze - Bottom-Up

After the top-down perspective has been developed, the underlying reasons for the risk profile need to be determined. This is accomplished by identifying and prioritizing individual risks for the project.

Individual risks can be identified using a variety of approaches:

- Reviewing published lists of project risk sources
- Evaluating requirements specifications, project plans and schedules, etc.
- Surveying project staff
- Brainstorming

Risks can be prioritized through a number of methods. Some projects have used A-B-C or High-Medium-Low lists. Others have estimated individual risk likelihood of occurring and potential schedule impact to produce an overall rating for each risk. By definition, risks that may impact the project's critical path or critical chain should always have highest priority.

Risk identification and prioritization can often be completed more quickly and comprehensively with the help of a facilitator skilled in Software Risk Management practices. This strategy can also provide training for organizations and project staff not familiar with Software Risk Management practices.

Mitigate and Control

Risk identification and prioritization is only useful if actions are defined and executed to mitigate risks. Aggressive, proactive risk mitigation action for top priority risks is essential to achieve the benefits of Software Risk Management.

Risk mitigation actions are defined individually for project risks. In some cases, immediate action will be called for; in others, future consideration will be more appropriate. For example, user interface requirements may be a risk on a particular project. Developing prototypes and using iterative development can mitigate this risk. Another risk might be establishing host communication for the test environment. This risk could be re-evaluated during the later phases of development, at which point a preliminary test environment can be constructed.

Planning for risk mitigation actions should not be confused with contingency planning. Risk mitigation actions are implemented proactively, to *prevent* a risk from impacting a project. Contingency plans are executed *after* a risk impacts a project. For most software project risks, contingency planning is best executed reactively; the selection of good alternative actions will change as a project evolves.

Early Warning System

Software Risk Management is an integral part of project execution. As a project proceeds, some risks will be eliminated, but some new risks may also occur. Some risk mitigation actions will work well, but some may not work and new action will need to be taken. As the project proceeds, priorities will change and new risk management planning will need to be undertaken.

For example, a host communication protocol risk may be eliminated, but end-user system capacity may become a new risk for initial deployment. Prototypes may solidify user interfaces, but testing with unskilled operators may not work and alternative strategies need to be implemented. Setting up software development environments may be a high priority risk early in a project, but testing environments will become much more important as the project proceeds.

Monitoring project risks can be accomplished through the following mechanisms:

- Scheduling risk mitigation tasks and review milestones on the project schedule
- Holding formal project risk management review meetings
- Conducting regular anonymous surveys of project staff
- Collecting key software metrics that give insight into aspects of project progress.

The project schedule is an excellent "early-warning" tool for risk management. By scheduling explicit risk mitigation tasks, their progress and effectiveness can be reviewed on a timely basis at project status meetings. Further, schedule milestones can serve as a reminder of formal project risk management review meetings, which provide a forum for evaluation of project risks and the effectiveness of risk mitigation actions.

At project risk management review meetings, new risks can be identified, all project risks can be re-prioritized, and new risk mitigation actions can be planned. This review should take a holistic approach to a project, considering all core and supporting areas for effective project execution.

Regular, anonymous surveys provide a mechanism to gather feedback from all project staff, efficiently allowing them to provide input on current project risks. In large projects, surveys are often the only practical way to gather input from all staff. After consolidation, survey results can be used as a basis for evaluating risk mitigation actions and planning new actions to address risks.

Finally, well-chosen software metrics can be a "leading-indicator" to future project problems. For example, defect discovery rates can be used to highlight test procedure changes needed early in the test cycle.

Risk Assessment

Risk Assessments are used regularly by experienced software development organizations to optimize project execution. Risk assessment methodologies include SEI's Taxonomy-Based Risk Identification, PMI's Risk Management Practices, and **KLCI**'s Detailed Risk Assessment^{SM}. Risk assessments are typically performed by outside agents expert in Software Risk Management.

Risk assessments generally focus on top-down and bottom-up identification of project strengths and risks. The results are used to develop an actionable framework of risk mitigation actions based on assessor experience and individual project characteristics.

Risk assessment is most effective for relatively experienced software development organizations. An organization for which a project is being assessed needs to have sufficient project management infrastructure to be able to take action based on the results. The organization also needs to have a commitment to improving their project execution effectiveness.

Why Should I Care?

Project Risks are schedule delays and cost overruns waiting to happen. Software Risk Management includes structured techniques to block these "surprises" before they occur.

Because of proven results, Software Risk Management has become widely implemented in software organizations with the highest relative levels of productivity in the industry.

Further, Software Risk Management is no longer cost prohibitive. Several excellent, affordable tools enable smaller projects and software organizations to implement risk management with little up-front investment. Risk Management practices can be implemented at any point in a project – and will often have immediate, positive payback in increased productivity.

References

1. Yourdon, Edward, Rise & Resurrection of the American Programmer, Yourdon Press, 1996.
2. Hewlett, David T., “Project Schedule Risk Analysis: Monte Carlo Simulation or PERT”, PM Network, February 2000.
4. Karolak, Dale, Software Engineering Risk Management, IEEE Computer Society Press, 1996.
5. Kulik, Peter, “Team-Based Risk Management in Software Development”, AT&T GIS Journal, December 1994.
6. Boehm, Barry, Software Risk Management, IEEE Computer Society, 1989.
7. Kulik, Peter, “How to Prevent Surprises in Software Projects”, August 1998. Available for download at http://www.klci.com.
8. For information about the Software Engineering Institute (SEI) conference on Software Risk or Risk Taxonomy, visit the SEI website at http://www.sei.cmu.edu, or contact SEI Customer Relations at 412-268-5000.
9. Kulik, Peter, “Team-Driven Schedule Metrics”, March 1996. Available for download at http://www.klci.com.

***Peter Kulik** is Managing Partner of **KLCI, Inc.** With more than 14 years experience in all aspects of software development, he holds an MS in Engineering Management with the thesis “Practical Quantitative Methods for Software Development Process Management”, a Certificate in Economics and Finance, and a BS in Electrical Engineering. He can be reached via e-mail at pkulik@klci.com.*

***KLCI, Inc.** helps software development organizations implement risk management, software metrics, and other process improvement initiatives. With innovative tools including the **Project Self-Assessment Kit**, **KLCI** applies apply proven practices in an action-oriented framework. Services enable clients to improve their productivity and meet customer commitments on software projects 10 to 100 people. **KLCI** can be contacted at 888-664-0484 (Toll Free, US/Canada) or +1-937-433-5502, or on the World Wide Web at http://www.klci.com.*