How to Prevent Surprises in Software Projects

Peter Kulik
KLCI, Inc.
August, 1998

1. Abstract

Software projects often experience "surprises" – tasks not completed when expected, resources available late, unexpected problems in integration testing, etc. Schedule delays and cost overruns, the bane of software organizations worldwide, can be directly attributed to these causes. While all surprises cannot be avoided, many can be predicted ahead of time. This paper describes proven practices that can be used to create an "early warning system" – allowing software organizations to anticipate problems before they occur, to help prevent schedule delays and cost overruns.

2. Introduction

In a 1994 study of more than 8000 Information Technology projects, only 16% completed on time, within budget, and to specification [1]. All projects encounter "surprises" – situations where events do not transpire as planned or as expected. For software projects, surprises can cause even the best project plans to become quickly obsolete.

Research has shown that the majority of causes of schedule slips and cost overruns are either related to project planning and monitoring, or can be anticipated by management [2]. For example, some surprises which cause project plans to become out-of-date can include:

- Tasks completed late
- Activities not started on time
- Test systems in use by another project
- Unexpected problems upon first integration of software modules
- A technology with less functionality than promised

Due to the low rate of project success, it is important for software organizations to effectively manage and prevent surprises that can cause schedule delays, cost overruns, and functional compromises. A recent study that I completed showed providing "early warning of problems" was on the top of the improvement "wish list" for software organizations [3].

The importance of planning to prevent surprises has been well understood, literally for centuries. Consider the following axiom of Sun Tzu:

> *Now the general who wins a battle makes many calculations in his temple ere the battle is fought. The general who loses a battle makes but few calculations beforehand. Thus do many calculations lead to victory, and few calculations to defeat. [4]*

The low success rate of software projects does not mean that planning is futile. A recent study of project managers emphasized the strong positive impact of "project execution planning" on project success [5]. Planning by itself is not sufficient, however; effective monitoring of project execution can provide advance warning of problems - if software organizations know where and how to look.

This paper discusses a proven "early warning system" to detect surprises - before they impact a project. With advance warning, software managers can proactively take action to prevent problems – such as lining up alternative technologies or reallocating resources to problem areas. My "early warning system" for software projects includes the following:

- GANTT Schedules
- Software Metrics
- Software Risk Management

The practices discussed are based on more than 150 surveys and interviews of software organizations conducted in late 1997 and early 1998, plus industry research and my own experience managing software projects and conducting risk assessments.

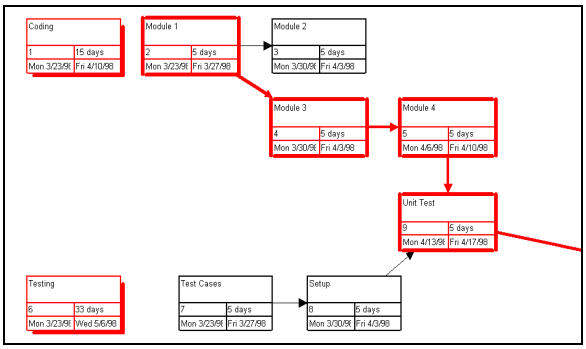

Figure 1 - PERT Schedule

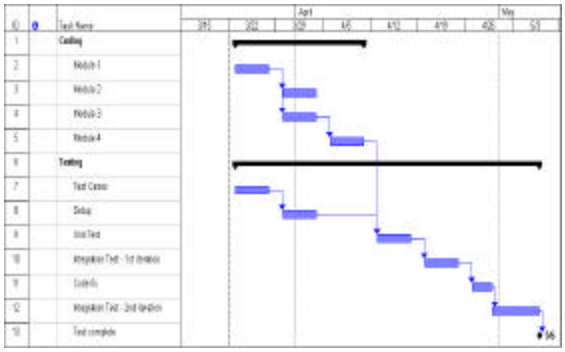

Figure 2 - GANTT Schedule

3. Tutorial

3.1. GANTT Schedules

Work Breakdown Structures and PERT schedules (see Figure 1) are important planning tools for projects of all types [6][7]. PERT schedules help project managers view relationships between tasks and project activity flow. However, in my experience, GANTT schedules are superior for project monitoring during execution.

GANTT schedules (see Figure 2) allow project managers to view projects in the context of time – critical for monitoring project activities to stay "on-schedule". GANTT schedules constructed and monitored according to the guidelines in Table 1 can be a powerful "early-warning" tool.

• Task granularity should be one week or less
• Update and review schedules weekly
• Monitor the critical path daily
• Ensure the schedule reflects the reality of project execution

Table 1 – Project Schedule Guidelines

The degree of early warning that can be achieved from a project schedule is determined by the maximum duration of tasks in the schedule. For example, if a month-long task slips in the first week of activity, the slip may not become visible until the end of the month. However, if this same month-long task is broken up into four weeklong tasks, a slip in the first week will certainly be visible at the end of the first week – early enough for a project manager to take action and possibly prevent slips in the overall project. Setting the granularity of tasks to one week or less therefore provides visibility where little existed before.

To achieve early warning, experience has shown that GANTT schedules must be tracked regularly. One successful project manager with whom we spoke credited his project's on-time completion to "intense focus" on schedules – and the few tasks that did slip were directly attributed to the lack of attention these tasks received in schedule reviews.

Schedule review should follow the guidelines in Table 2, and focus on validating the following:

- Tasks that should have completed or should have started have actually done so.
- Tasks that should soon complete or should soon start are on-track to do so.

When issues are identified, the "early-warning" from regular schedule review enables team members to work together to prevent schedule delays – for example reallocating resources or switching the order of tasks to compensate for interdependencies. An additional benefit of regular schedule reviews is regular reaffirmation of team members' commitment to the project schedule.

The duration of the critical path is the duration of a project. Projects that have developed detailed schedules according to the guidelines in Table 1 can readily identify the specific tasks that make up the project's critical path (sometimes the assistance of simulation tools can be useful to identify critical path tasks [8]). If these critical path tasks slip – the whole project slips. For most projects, weekly review of critical path tasks is too seldom – any surprises affecting the critical path need to be identified immediately for the software manager to have a

• Focus on a two week window centered at the review date
• Highlight those tasks that are:
✓ Scheduled to finish in this window
✓ Scheduled to start in this window
• Use a team approach to schedule review to monitor task dependencies

Table 2 – Schedule Review Guidelines

http://www.klci.com/accelerate
888-664-0484 (+1-937-433-5502)

chance to prevent schedule slips. As long as management attention does not detract from executing critical path tasks, daily monitoring of the critical path can enable software managers to prevent slips in the overall schedule.

For example, testing is typically on the critical path for a software project. Conceptually, this is easy to understand, but it is not actionable. Identifying the specific testing tasks that are on the critical path will enable the software manager to focus her attention appropriately. For example, one project which I analyzed had allowed test system setup and user documentation to unnecessarily fall onto the critical path - meaning that delays setting up test systems or writing user documentation would slip project completion! Understanding the specific tasks on the critical path provides early warning of potential problems such as these – and once understood, the software manager can easily take action to prevent them.

Finally, experience has shown that the above guidelines require schedules which "reflect the reality of project execution" – the scheduled tasks, duration,

1.	Lines of code
2.	Scheduled tasks completed on time
3.	Scheduled tasks completed late
4.	Test coverage
5.	Resource adequacy
6.	Fault density
7.	System operational availability
8.	Time to market
9.	Schedule, quality, and cost tradeoffs

Table 3 - Most Commonly-Measured Metrics

sequence, and dependencies reflect the way that the project is actually being executed. It is not unusual to make changes to the schedule as a project progresses; changes should be made as often as needed so that the project can be monitored effectively. Otherwise, software managers can suffer from GIGO (Garbage In, Garbage Out) in their efforts to manage project schedules.

3.2. Software Metrics

Table 3 shows the most common metrics usage found in my recent study of software metrics practices [3]. Of these metrics, several can be a key part of an early-warning system for software development – and provide warning of high post-release support costs.

Lines of Code –the amount of software to be developed is a leading indicator to the project schedule and resources needed. To be applied, however, an organization must understand the relationship between the variables – how the amount of software to be developed influences project schedule and resources. One relatively easy way to accomplish this is through "rules of thumb" [9] which associate these variables. For example, knowing that an average project can deliver 350 lines of code per "programmer-month" can help a manager easily put bounds around project schedule and resources needed based on an estimate of lines of code. By tracking metrics over time, a software manager can develop his own "rules of thumb" based on his organization's capability. While more "precise" measures than lines of code do exist, most software managers seem to have found lines of code sufficiently accurate for their purposes.

Test coverage and fault density – by setting targets and measuring test coverage, software managers can ensure comprehensive testing of typical operation, boundary conditions, and exceptions. The use of test coverage tools can reduce fault density earlier in the project schedule, preventing problems in later testing activities. Conversely, high fault density measured early in testing activities can provide advance warning of quality problems that could delay test phase completion.

Schedule, Quality, and Cost tradeoffs – the triple constraints of project management are nowhere more evident than in software development. Understanding a team's or an organization's tradeoff preferences for these constraints can provide a leading indicator to project results. For example, an organization that is readily willing to sacrifice quality to meet schedule objectives is more likely to meet their schedules with poorly functioning software. A team that is just the opposite - quality driven to the exclusion of all other factors - may never complete their projects while striving for ever-increasing levels of quality. As these extreme examples show, balance is important to meet the triple constraints of schedule, quality, and cost. Identification of imbalances can provide early warning of project problems during execution.

Code Complexity – although not mentioned as one of the most commonly measured metrics, Code Complexity can be an excellent leading indicator for software bugs, difficulties in integration, and high support costs. Studies have shown that software modules with cyclomatic complexity greater than 10 have higher defect density than less complex modules [9]. Identifying these modules, and re-coding if possible, can prevent problems later in project execution and deployment.

http://www.klci.com/accelerate
888-664-0484 (+1-937-433-5502)

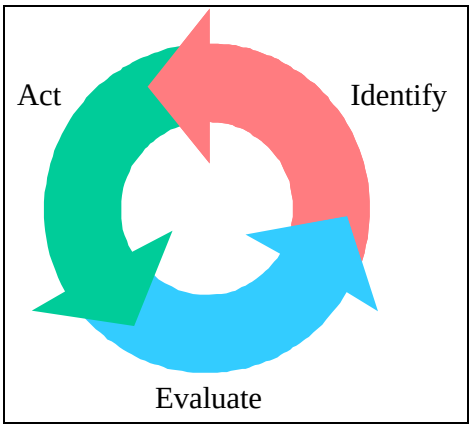

Figure 3 - Risk Management Activities

3.3. Software Risk Management

According to the Software Engineering Institute, "Today's risks are tomorrow's problems". The benefits of Software Risk Management have been understood since at least the late 1980s [10]. According to Tim Lister, "Risk management tailors a project to fit the risk, unlike the Capability Maturity Model, which tailors a project to fit the process… 90% of managing risk is recognizing it" [11]. Fundamentally, risk management is composed of three steps - Identify, Act, and Evaluate (see figure 3).

Identify

To manage risks, software organizations must first be aware of them. Typically, risk identification is accomplished one of the following ways:

- Team brainstorming - leverages the varied perspectives of team members to identify risks. This approach has proven useful, but is subject to organizational "blind spots" - risks may be missed because team members will not be aware of certain issues [12].
- Formal risk assessment - accomplished by an independent team of experienced personnel, who evaluate a project with the cooperation of a project team. This approach has the advantage of leveraging a broad, multi-organizational perspective of senior personnel. A recent trend in companies such as AT&T, Hewlett-Packard, IBM, and NCR is the addition of the risk assessment function to Project Offices [13].

  Risk assessment tools - offer a structured approach and broad perspective for risk identification, at significantly less cost than formal risk assessment. The Project Self-Assessment Kit from KLCI Inc. (http://www.klci.com/accelerate) is one such tool. This tool also measures overall project risk, as shown in Figure 4.

Traditional Risk Assessment	Self-Assessment
• Can take weeks or months to complete	• Can complete in hours and get nearly immediate results
• Requires special expertise to conduct	• Any project or technical manager can conduct
• Cost-effective for very large projects only (usually >$US 5M R&D budget	• Cost-effective for all medium and small projects (>$US 500K R&D Budget)
• In-depth analysis of cause-and-effect	• First-level analysis to highlight areas needing particular focus

- Simulation and quantitative methods - given project schedules built according to the guidelines in Table 1, simulation or other quantitative methods can be helpful to identify specific tasks that are at risk [14].

Act

Risk identification by itself will have little impact on a project. "The goals of risk management… are to identify project risks and develop strategies that either significantly reduce them or take steps to avoid them altogether." [15]. The key to risk management is developing strategies

1. Complete risk identification during project planning - or as soon as possible for projects which are past this phase
2. Prioritize risks to focus on those risks with the most potential impact on the project. In my experience, an A-B-C prioritization is effective and time-efficient.
3. Use a team approach to brainstorm proactive actions to mitigate top-priority risks
4. Put these actions on the project schedule, and track them weekly with other tasks
5. Evaluate risk management actions at major project milestones (e.g. design complete, start of integration test)

Table 4 - Recommended Risk Management Guidelines

http://www.klci.com/accelerate
888-664-0484 (+1-937-433-5502)

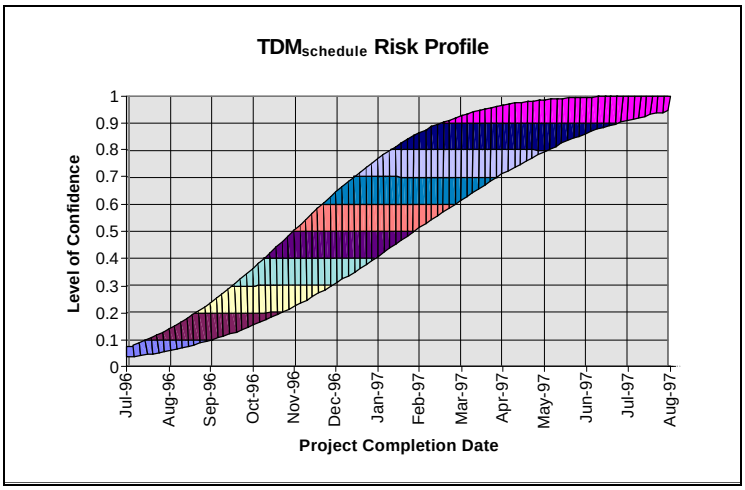

Figure 4 - Statistical Risk Profile

and taking proactive action to mitigate risks. To achieve this, we recommend the guidelines shown in Table 4.

Evaluate

Finally, because projects change as they progress, re-assessment of project risks measures the impact of risk mitigation actions and identifies risks that could not be evident earlier in the project. This step helps software organizations maintain their early-warning system - and prevent surprises - throughout project execution. In my experience, re-evaluation at major project milestones is usually sufficient - as long as the other guidelines in Table 4 are being followed.

For example, one project manager with whom we spoke used team brainstorming to identify project risks. This team put proactive actions in place to mitigate risks, in conjunction with detailed project schedules and regular review. The result - "we really didn't have any surprises", and the project was finished on time, on budget.

4. Summary and Conclusions

By predicting surprises before they occur, many can be prevented - preventing schedule delays and cost overruns. While not all surprises can be avoided, it is possible to construct an "early warning" system using proven tools and practices, including:

- GANTT Schedules
- Software Metrics
- Software Risk Management

By implementing these practices and appropriate tools - as part of their existing software development processes - software managers can anticipate problems and take action to prevent impact to their projects.

5. References

[1] Standish Group International, "Chaos, Charting the Seas of Information Technology", 1994.

[2] van Genuchten, Michiel, “Why is Software Late? An Empirical Study of Reasons for Delay in Software Development”, *IEEE Transactions on Software Engineering*, Vol. 17, No. 6, June 1991.

[3] Kulik, Peter J., “Software Metrics Best Practices”, Software Q/A Magazine, Vol.5 No.2, April/May 1998.

[4] Tzu, Sun, The Art of War, translated from the Chinese by Lionel Giles, M.A. (1910)

[5] Zimmerer, Thomas W., and Yasin, Mahmoud M., "A Leadership Profile of American Project Managers", Project Management Journal, Volume 29, Number 1, March 1998.

[6] Charette, Wilfred, and Halverson, Walter S., "Tools of Project Management", published in The Implementation of Project Management: The Professional's Handbook, Addison-Wesley Publishing Company, Thirteenth printing, 1996.

[7] Kiewel, Brad, "Measuring Progress in Software Development", PM Network, January 1998.

[8] Gump, Alan, "Scheduling High-Tech Projects", PM Network, July 1997.

[9] Grady, Robert B., "Practical Rules of Thumb for Software Managers", *Hewlett-Packard Software Engineering Productivity Conference Proceedings*, August 1990, pages 647-651.

[10] Boehm, Barry, Software Risk Management, IEEE Computer Society, 1989.

[11] Interview with Tim Lister, IEEE Software, March/April 1997.

[12] Kulik, Peter J., “Team-Based Risk Management for Software Development”, AT&T GIS Journal, December 1994.

[13] Frame, J. Davidson, "Risk Assessment Groups: Key component of Project Offices", PM Network, March 1998

[14] Weiler, Chris, “Risk-Based Scheduling and Analysis”, PM Network, February 1998

[15] Wideman, R. Max, Project and Program Risk Management, a Guide to Managing Project Risks and Opportunities, Project Management Institute, 1992.

About the Author: Peter Kulik is Managing Partner of KLCI, Inc. He can be reached via e-mail at pkulik@klci.com, or phone at 888-664-0484 (+1-937-435-5502).