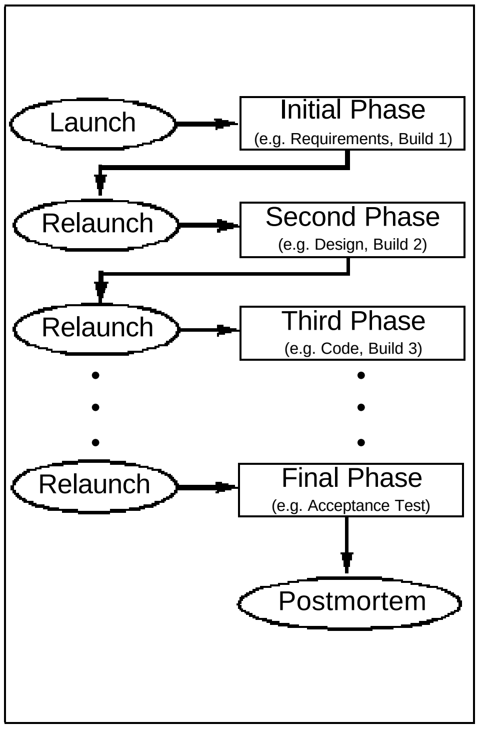
Managing Risk with TSP

by David R. Webb
Hill Air Force Base

One of the most important aspects of applying the Team Software Process (TSP^SM) to software projects of any size is the increased success of identifying, tracking, and mitigating risk. The Mission Planning Software Section of the Software Engineering Division of Hill Air Force Base (TISHD), has found the TSP's simple strategy for identifying, tracking, and handling risks to be extremely effective. In fact, many common software project risks are managed purely by adopting the TSP.

Dozens of books address the concept of software risk management and, it seems, there are even more software tools than books on this topic. Risk management tools can be as simple as a list of risks brainstormed during the start of a project and reviewed occasionally. They can also be as complex as a 100-page Risk Management Plan with risks and their associated prioritization, likelihood, impacts and mitigation strategies, along with a Web-based risk browser to track the plan. Still, the basic approach for all of these methods is the same: risks are identified early in the project, planned for, monitored, and handled. TSP takes a middle-of-the-road approach to risk management, doing what makes sense for the project with as little paperwork and tool upkeep as possible. Although the approach was originally designed for teams of fewer than 20 people, the principles can be applied to much larger groups, with equally effective results.

Figure 1. *Risks are identified at each TSP launch.*



Identification

The TSP handles a project the way you eat an elephant—one bite at a time. The TSP team estimates projects in a top-down approach, using overall size and average team productivity to determine overall schedule. This schedule is broken into manageable phases and the phase currently being worked is thoroughly estimated and tracked using a bottom-up approach wherein each engineer estimates his or her own schedule using individual data. Each time a phase begins, whether at the start of the project or at the transition from one phase to the next, there is a project launch (Figure 1). At these launches, the tasks for the current phase are thoroughly defined and each task is estimated using the rigorous methods of the Personal Software Process (PSP^SM). These estimates are used to produce a detailed *next phase* earned value plan, against which the project will be tracked and managed. Project goals, quality criteria and risks are also identified during the launches.

A portion of each launch is dedicated to brainstorming risks the project may face. These sessions can last from a dozen minutes to a few hours, depending upon the size of the project and the team's knowledge and maturity. The risks that are identified are serious problems that may occur during the life cycle of the project, not just a list of all maladies that are possible. For example, it makes little sense to manage the risk of your software being destroyed by a bomb or abducted by aliens unless, of course, you work for Special Agents Fox Mulder and Dana Scully. Barring that circumstance, most projects make a list of all the real-life problems that can be foreseen. Some common risks identified during these meetings include a lack of proper documentation, a development environment that may not support the size or type of program being developed, an impossible schedule or inadequate computer, office, or personnel resources. Each risk is assigned a likelihood of occurrence, a severity if it does occur, and a person responsible to monitor the risk. The TSP team assigns each member a role, such as Design Manager, Planning Manager, Implementation Manager, Customer Interface Manager, Quality Manager, Process Manager, Support Manager, Test Manager, or Team Leader. Typically, the team member with the appropriate role is assigned to monitor a risk. For example, a risk involving negotiations with the customer would be assigned to the Customer Interface Manager. This information is documented so that it can be regularly referenced.

Review and Mitigation

The TSP requires a weekly status meeting where team progress is compared to the team plan in terms of earned value and quality. If there are deviations from the plan, the reasons for these deviations can be determined and actions taken to bring the team's performance in line with the plan. It is also during these weekly meetings that the team reviews the risks brainstormed during the launch. The team removes risks from the list that no longer pose a threat, while the assigned engineers report on those that are still potential problems. If the mitigation strategy for a risk has failed and the risk has occurred, or is likely to occur soon, the risk is renamed an "issue" and immediate action is taken to address it. The risk list subsequently becomes a living, breathing document that changes size and shape each week. It also becomes a *used* document that helps the team focus on risks that need to be addressed *when* they need to be addressed.

Risks That Are Not

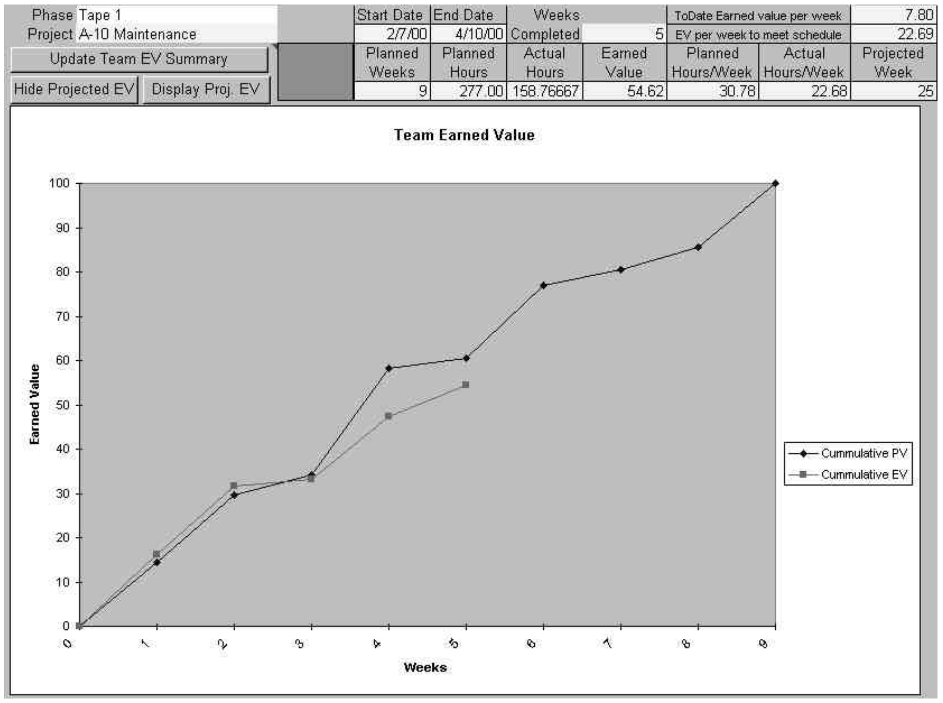
Three of the most common risks to any project are schedule overruns, requirements creep, and quality problems.

A project properly using the TSP already has the tools to handle these risks.

A TSP team determines its own schedule and coordinates it with management, marketing, and the customer, as appropriate. While outside influences may have strong impacts on the delivery date of any piece of software, the TSP team knows its productivity rates, has a rigorous estimating process, and can confidently tell management how much can be accomplished within a given time frame. The TSP launch is not successfully concluded until the team and management agree upon a list of requirements and a schedule that is satisfactory to both parties. Once this realistic schedule has been determined, it is used as the basis for measuring personal and team-earned value and is tracked daily at the personal level and weekly at the team level (Figure 2). Any deviations from the plan are identified early in the project and are dealt with by negotiating with management and the customer. The TSP virtually eliminates schedule risks.

The TSP also requires replanning, or at least updating, a project when the basic assumptions of the plan change. This means that *when* (not if) the requirements change during the course of the project, the team renegotiates schedule, delivered functionality and, if appropriate, cost. This becomes the new plan that the team tracks and the requirements creep risk is effectively dealt with, if not completely eliminated. Another great thing about this technique is that it ensures management and the customer are involved every step of the way so that no one is surprised by the project's performance, least of all those who are anticipating the product.

Finally, problems with quality can, over time, be virtually erased using the TSP. Since the quality methods used by the TSP are based upon the strict quality processes of the Personal Software Process, individual engineers perform their own extensive reviews of both detailed design and code prior to exhaustive team inspections. Defect densities at personal reviews, team inspections, compile and unit test, are used as yardsticks to determine if the finished code is of high enough quality to be passed on to integration and system test, or if the code should be pulled back and reinspected or rewritten. This ensures the quality of the code, but not always the quality of the requirements upon which the code is based. Often, requirement problems are uncovered during acceptance testing. When such defects are discovered, the TSP team adds them to team and personal review checklists to ensure such problems are never allowed to pass through the process again. An experienced TSP team can, therefore, eliminate virtually all quality risks, particularly expensive defects found during qualification and acceptance testing.



Phase Tape 1
Project A-10 Maintenance

Start Date	End Date	Weeks Completed		ToDate Earned value per week	7.80
2/7/00	4/10/00		5	EV per week to meet schedule	22.69

Planned Weeks	Planned Hours	Actual Hours	Earned Value	Planned Hours/Week	Actual Hours/Week	Projected Week
9	277.00	158.76667	54.62	30.78	22.68	25

Figure 2. *TSP Team Earned Value for TISHD A-10 AWE*

Some Examples in TISHD

TISHD has trained nearly 20 engineers in the PSP and has launched three separate mission planning projects using TSP version 0.3. These projects are an Air Tasking Order parser named TaskView [1], an A-10 Aircraft/Weapons/Electronics (AWE) software program, and an F-16 Block 30 AWE program. Of these three projects, TaskView and A-10 AWE have been using the TSP long enough for us to draw some conclusions about the usefulness of the TSP and the success rate of using the TSP risk management strategy.

TaskView

The TaskView project was the first TISHD group to pilot test the TSP. Just prior to the initial launch, the TaskView customer decided to participate in an Air Force Expeditionary Force Experiment (EFX). This new goal required the TaskView 3.0 product to be delivered one month earlier than originally planned. The team added this risk to its risk list and assigned it to the Planning Manager. With this in mind, the team adjusted the plan to meet the new schedule.

As work progressed, the team-earned value projected that TaskView 3.0 would be delivered more than a month earlier than anticipated, even with the new schedule. At this point, the first risk was closed out and another risk—that of being too early and losing revenue—was added to the list and assigned to the Customer Interface Manger. The customer was approached with the option of receiving the product early and getting a refund, or adding new capability to TaskView 3.0. The customer was delighted with this information and chose to keep the current level of funding and add in new capabilities to the software. Even with the new functionality, TaskView 3.0 was delivered well within time to participate in the EFX experiment.

TaskView has also experienced a significant reduction in the risks associated with defects, as a result of adopting the TSP. TaskView has had three major releases since TISHD started working on the project in 1997. TISHD has added new capability and robustness to each release, at times rewriting major portions of the code to do so. Compared to data from similar projects completed in the past (using the TISHD CMM Level 5 organizational process), the TaskView projects have seen a substantial decrease in defects and test time (see Figure 3 and Figure 4).

One interesting outcome of the defect data analysis was the increase in defect density experienced by the TaskView 3.1 project. Although the defect density found during Customer Acceptance Testing was steadily decreasing, defects found in earlier test phases increased. This was of some concern to the team, until it began to filter the list by defect priority (Figure 4). Once that was done, it was obvious that the TaskView team, as it had grown more confident in the use of the TSP, had begun to record more development defects than ever before; remember, TSP teams count every defect found in every development and test phase, including compile. However, despite this increase in defect recording, high-priority defects became nonexistent using the TSP. This does not mean that no issues were discovered during customer acceptance testing, but the issues dealt almost exclusively with the addition of new requirements and limitations of the operational environment, and were not defects in the delivered code. Note also that TaskView 3.1S (a special project developed in support of another mission planning tool) had zero high-priority defects at *every* test phase.

As most software project managers are well aware, the greatest risk to any project's schedule is the risk of finding defects during test, especially final or customer acceptance testing. The causes of these defects are often difficult to trace and fix and can cause significant slips in schedule. In order to eliminate this risk, test time needs to be reduced and become more consistent. Although TISHD was already seeing very low test days/thousand lines of code rates using its Level 5 process, the adaptation of the TSP reduced the test time further and made the variation much smaller (see Figure 5).

Figure 3. *TISHD Total Defect / Non-TSP Projects vs TaskView*

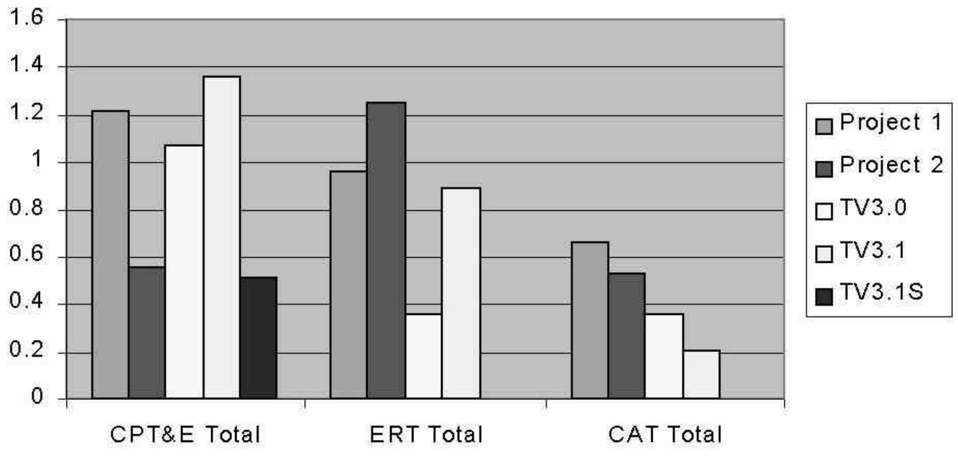
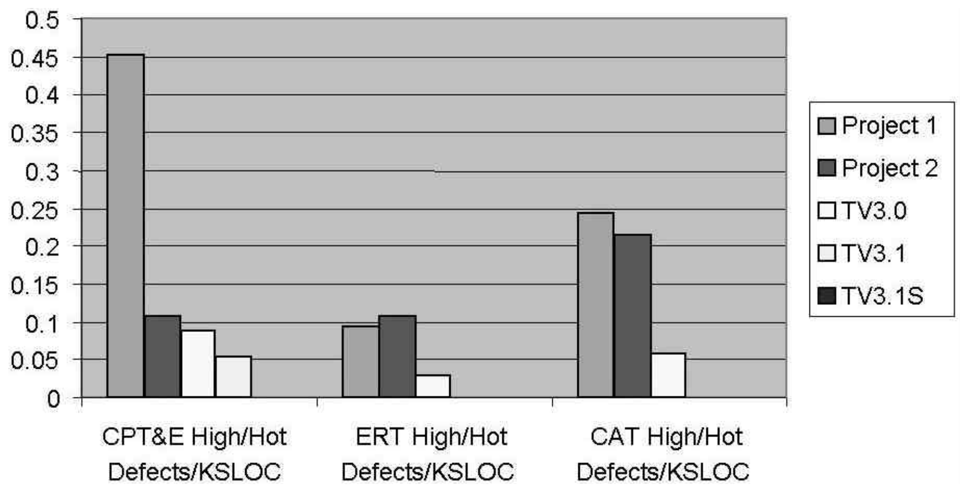


Figure 4. *TISHD High Priority Defects / Non-TSP Projects vs TaskView*



A-10 AWE

While the TaskView project was still evaluating the effectiveness of the TSP, the A-10 AWE team decided to use some of the concepts (planning, tracking, weekly updates) *without* employing the rigorous techniques of the Personal Software Process. Each A-10 AWE engineer was provided a spreadsheet for each code change he or she was working. These spreadsheets covered the estimate of size (lines of code or LOC) and time (days) as well as the actuals for LOC and time. Time was measured at distinct milestones, such as inspections, unit test, and code check-in. An earned value plan was created from the estimates provided by the engineers and used to refine the schedule. Although the engineers were not required to be PSP trained, all any engineer had to do, after estimating, was to check a box on the tracking spreadsheet once a milestone was reached. The spreadsheet would calculate how long the tasks took and export that data to the earned value tracking tool.

Sounds like a good plan, right? It did not work very well.

Estimates were often wildly inaccurate. Tracking was not consistent. Entire new capabilities would move from 0 percent complete to 100 percent complete overnight. All of these problems gave the team a false impression of the team-earned value. The earned value was, therefore, not trusted and soon ignored by most of the team members. The team reverted to the higher-level tracking process used by non-TSP projects in TISHD, which were sufficient to prevent the team from missing schedule. (Note that TISHD is part of a SW-CMM® Level 5 organization, and typically meets cost and schedule estimates anyway.) However, any advantage of using the TSP-like process disappeared.

The one thing that *did* work, was risk identification and tracking, the process that we copied directly from the Team Software Process.

For example, the A-10 AWE team determined that a required piece of core software, developed by a third-party vendor, might not be released in time to meet schedule. This risk was assigned a high likelihood and a high impact. A mitigation plan of reverting to an earlier release of the core was determined and an engineer was assigned to track the status

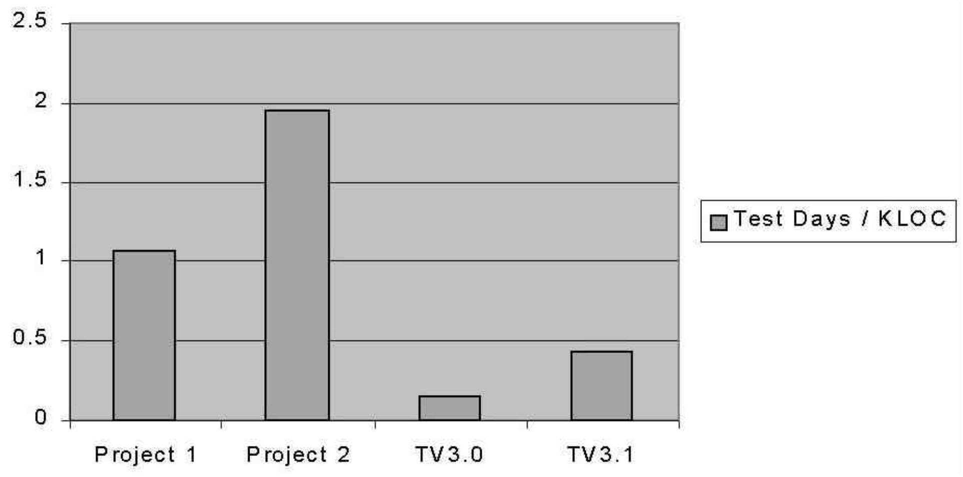


Figure 5. *TISHD Test Duration / Non-TPS vs TaskView*

of the core software. As it turned out, the third-party software did slip its schedule by several months, which would have, in turn, caused our software to slip its release date had we not planned for this risk early in the program. Due to early risk identification, planning and tracking, the A-10 AWE was able to mitigate this risk and revert to the earlier version of the core software.

One risk that was *not* identified was the hazard of using the TSP-like process, instead of the TSP. During project post-mortem, it was determined that the reason the modified process did not work as well as a traditional TSP team was that the engineers were not PSP trained and did not understand how the data they were collecting was being used. At that point, we determined to use TSP on the next A-10 AWE project and immediately scheduled a PSP course for those engineers.

The results in earned value tracking alone were astounding (see Figure 3). Code was accurately estimated and tracked; it was *very* easy to see how close to our schedule we were running. TISHD learned an important lesson: TSP does not work well without the proper data, and that data is almost impossible to gather without the rigors of the PSP. That is one risk TISHD has completely eliminated.

Conclusion

While there are many tools for software risk management, TISHD has found that utilizing the planning, tracking, and defect prevention techniques of the Team Software Process is a simple and effective way to identify, track, and mitigate most software project risks. In TISHD we have learned that, over time, TSP teams become experts at risk mitigation and management; they also become very good at writing code that is nearly free of defects, and that TSP is a risk mitigation strategy *any* software project should strive to adopt.

Reference

1. Webb, David and Humphrey, Watts S. Using the TSP on the TaskView Project, CROSSTALK, February 1999, pp. 3-10.

About the Author

David R. Webb is a Technical Program Manager for the Mission Planning Software section at Hill Air Force Base, Utah, and a part-time visiting scientist for the Software Engineering Institute (SEI). He is a member of the Software Division of the Technology and Industrial Support Directorate (TIS), which was assessed a CMM® Level 5 organization in July 1998. He has 12 years of technical and program management experience with software in the Air Force. Webb also has spent five years as a software test engineer, two years as a software system design engineer, and three years as a member of TIS's full-time Software Engineering Process Group (SEPG). He is a SEI-certified PSP instructor. He received a bachelor's degree in electrical and computer engineering at Brigham Young University.

OO-ALC/TISHD
6137 Wardleigh Road
Hill Air Force Base, Utah 84056
Voice: 801-775-2916 DSN 775-2916
E-mail: david.webb@hill.af.mil

Coming Events

June 4-7
9th Biennial IEEE
http://cefc2k.aln.fiu.edu

June 4-11
22nd International Conference on Software Engineering
www.ul.ie/~icse2000

June 5-7
2000 IEEE International Interconnect Technology Conference
www.his.com/~iitc

June 10-14
ISCA2000: 27th International Symposium on Computer Architecture
www.cs.rochester.edu/meetings/ICSA2K

June 18-22
ICC 2000—IEEE International Conference on Communications
www.icc00.org/

July 11-13
5th Annual Conference on Innovations and Technology in computer Science Education
www.cs.helsinki.fi/events/iticse

July 16-18
7th IEEE Workshop on Computers in Power Electronics
www.conted.vt.edu/compel.htm

July 16-19
Congress on Evolutionary Computation
http://pcgipseca.cee.hw.ac.uk/cec2000

August 6-11
6th Annual International Conference on Mobile Computing and Networking
www.research.telcordia.com/mobicom2000

August 7-8
IEEE Workshop on Memory Technology Design and Testing
http://pcgipseca.cee.hw.ac.uk/cec2000

August 17-19
Designing Interactive Systems (DIS)

September 10-12
Collaborative Virtual Environments (CVE)

September 10-14
Very Large Databases (VLD)
Visit www.acm.org/events for information on VLD, DIS, & CVE 2000.

April 29-May 3, 2001
STC 2001: The Premiere Department of Defense Software Technology Conference
www.stc-online.org