Experiences With the TSP Technology Insertion

Ray Trechter
Sandia National Laboratories

Iraj Hirmanpour
AMS, Inc.

Transitioning the Team Software Process℠ (TSP℠) methodology into an organization is not an easy or simple task. It requires significant behavior change by not only the developers, but also the major stakeholders. In this article, we share three years of experience with the TSP transition team at Sandia National Laboratories with two intertwined perspectives: that of the TSP coach, and that of the development manager supervising the TSP projects.

The mission of the Information Systems Development Center of Sandia National Laboratories in Albuquerque, N.M. is to provide software development and support services for a variety of internal customers in support of their national defense and energy-related work. The organization's software development process, called Software and Information Life Cycle (SILC), is used for all work and is meant to embody Capability Maturity Model® for Software (SW-CMM®) Level 3 processes.

In 2001, the organization decided to run a pilot Personal Software Process℠ (PSP℠)/Team Software Process℠ (TSP℠) technology insertion program by training developers in PSP and launching TSP teams. The motivation for introducing these processes was to supplement SILC with personal/team processes that encouraged process improvement for the individual developer.

The SILC process is organized in phases consisting of Planning, Analysis, Design, Implementation, and Deployment. The organization's business model requires a proposal for each project, which must be approved by both the customer and management. After approval, the SILC process is enacted throughout the life of the project. The TSP projects launched so far have, with the exception of one case, started with the analysis phase. Figure 1 shows the overlay of the TSP process on the SILC process.

Early Experiences Using TSP

Since the initial pilot in 2001, a significant number of development staff has been trained in PSP and TSP. Fifty-six developers have completed the 10-day PSP class, and 44 managers and non-developers have completed the two-day Introduction to PSP class. The TSP process requires a multi-day project launch, which produces an overall project plan and a detailed next phase or cycle plan. A relaunch occurs at the end of each phase or cycle to develop the next cycle or phase detailed plan. The first TSP launch occurred in February 2002; currently, a total of eight projects have used TSP with eight launches and five relaunches.

During the early launches with TSP, there was a lot of hesitation by teams about using TSP in addition to the mandated SILC process. A great deal of discussion centered on whether TSP could replace SILC outright since TSP already has a defined process; the argument went further on the congruity of SILC roles and TSP roles. SILC defines a set of product engineering roles such as analyst, designer, builder, and database administrator. The TSP defines roles that are a combination of product engineering roles and project/process management roles such as design manager, implementation manager, planning manager, process manager, quality manager, support manager, and test manager. Aligning these two role sets consumed a fair amount of time and team energy.

The Software Engineering Institute prototype TSP tool also became a major point of contention: the tool did not provide the user-friendly aspects developers have come to expect. Phases in the TSP tool did not align readily with SILC phases, and users were not able to change them. Even the TSP earned value system came under attack from some project leads who had attended Project Management Institute (PMI) training as different from the PMI and Department of Defense (DoD) definition of earned value. The TSP earned value system focuses on effort and does not address cost, while the PMI and DoD version of earned value track cost and budget burn rate. In addition to the information on effort expended provided by TSP, project managers have to monitor the organizations' financial management system to have an accurate picture of total project costs.

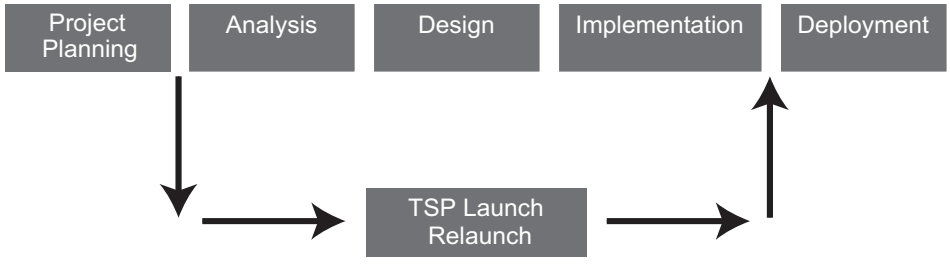
Based on these early experiences, it was not clear whether these struggles would result in the outright rejection of TSP as an organizational process. Two developments led to a more favorable view of TSP use within the organization. Management issued a policy – without specifying use of the TSP – that required all projects to report their status based on facts and data. The policy provided an addendum to the SILC glossary, defining terms such as plan and actual task hours, percent work complete, unplanned task hours, etc. This policy changed the nature of the TSP insertion from a *push* from above to a *pull* from managers and project leads that needed a way to satisfy the policy. Additionally, TSP began to be viewed by project leads as a useful way to structure and run their projects.

Overall, it has taken some time to build support for TSP among management and practitioners. However, each project provided some valuable lessons learned and also provided greater visibility into some of the organization's issues. Lessons learned from the early experiences are described here.

Lessons Learned From the Initial Experiences Using TSP

Most participants in the early TSP project saw the potential of this methodology, if some of its concerns could be addressed. The use of TSP highlighted several process improvement areas for the organization as a whole. While the launch meet-

Figure 1: *Insertion of TSP into SILC Process*



ing scripts and requirements engineering improvements were specific to TSP, project time accounting and the number of projects per developer applied to the organization as a whole.

Launch Meeting Scripts

First, everyone agreed that the TSP launch process requiring extended discussion on project issues between team management and customer, and among team members was highly beneficial. One developer commented, "The most beneficial aspect of this process was the TSP launch.

"The launch gave us the opportunity to step away from work for a few days, think about what the project required, and plan for the project accordingly." The launch scripts were altered to include SILC-mandated management reviews at the end of each launch. Stakeholders, including funding sources, end-user representatives, and project management were able to see the results of the launch and were presented with an accurate plan of what the project would accomplish in the next three to four months.

Requirements Definition

Through the use of both SILC and TSP, it was discovered that the feature-level requirements typically found in a SILC proposal did not provide enough detail for systems analysts and designers to create their TSP plans. Some early TSP launches were suspended while the team worked requirements to the point where the team had a common mental model of what was needed from the system. It was decided that the minimum knowledge of requirements and systems characteristics needed was equivalent to the information required by a concept of operations document (IEEE Standard 1362-1998). A concept of operations document is now required for TSP launches.

Time Tracking

The idea of time logging was initially not well received by the practitioners. It was not clear to them how this data would be used and some worried that management might use the data to judge their performance. Over time, teams have come to realize that the increased visibility provided by this data not only helps management, but also the teams have a better understanding of project progress.

Task Time

Task time is defined by TSP as time spent on project-related tasks. All other activities such as attending meetings, e-mailing, taking a break, etc. are not considered task time. The organization did not have the metrics to know the amount of task time available for developers. When the TSP coach suggested starting with 20 hours per week for planning purposes, one manager's reaction was, "What will we do with the other 50 percent of the time?" After a few projects, data showed that 20 hours per week was not possible due to other duties and the dynamic nature of the environment.

Multiple Projects Per Developer

Developers were divided among several projects; some developers were divided among as many as four or five projects making task time availability for the TSP project only 10 percent (two hours per week) or 20 percent (four hours per week) using the measure of 20 hours of available task time per week. Data showed that such a limited resource assignment method is inefficient. The practice has now been changed to allocate each developer to a maximum of two projects.

Moving Up to the Next Level

As mentioned earlier, last year the Information Systems Development Center mandated that projects report their status with facts and data. Guidelines for a reporting system and a glossary of terms were published to help management and practitioners understand the policy requirements. When a new manager (co-author of this article) took over the management of a department within the Development Center, one of 10 such units, he was faced with the mandate of data reporting.

The new manager made the tactical decision to make TSP available to any team that requested it. So far, three teams have selected the TSP as their method of choice to manage their data projects. The experience this time around is markedly different from our initial experiences with TSP. In what follows, we describe the policy for managing projects with data and our new experiences with TSP. We also describe the behavior of a TSP practitioner, and the TSP reporting system that provides status reports that one can act on. Finally, insights are offered on how we use the data to manage a portfolio of projects.

Table 1: *A Developer's Weekly Status Report*

Weekly Data	Plan	Actual	Plan/Actual
Project hours for this week	20.0	22.6	0.88
Project hours this cycle to date	33.0	28.0	1.18
Earned value for this week	19.2	36.3	0.53
Earned value this cycle to date	30.0	51.1	0.59
To-date hours for tasks completed	49.7	25.7	1.93

A Typical Day in a TSP Practitioner's Life

The TSP practitioners in the group noted previously exhibit very different behavior patterns from those in the past. They like the model of personal and team planning as opposed to a plan handed to them by a project lead. They take ownership of the project and not just of the tasks assigned to them.

A typical day for a TSP practitioner is partitioned into two areas: the time spent on tasks related to the TSP project and time, and the time spent on other activities. TSP practitioners keep track of project task time to the minute using a time log. Individual developers use a defined process to develop software and record time based on phases of their personal process; it is rich with personal reviews for early defect removal.

A typical TSP developer's personal process would have phases such as *design*, *design review*, *code*, *code review*, *compile*, and *test*. All these steps are measured in terms of time spent and defects injected or removed. By collecting the three basic measures of effort, defect, and size, and by recording task completion date, a host of metrics is available to the developer. These metrics help the developer manage work, and compare actual work with planned work at a personal level. An example of the type of metrics derived from the three basic core measures is the dashboard style data shown in Table 1.

The data in Table 1 is from a developer's plan in week three of an eight-week project. It shows that, so far, effort estimation has been twice that of actual effort needed for completed tasks. As a result, it shows that the project is ahead of schedule: 51 percent complete compared to the planned 30 percent complete. Clearly, this is good information to have at an early stage, as there inevitably is someone else who may be behind in his plan. During the TSP weekly meeting, one of the activities is load balancing when the data indicates the need.

The practitioner submits his/her personal plan to the planning manager (a member of the team) for consolidation on

a weekly basis. The planning manager creates a consolidated plan for the team leader to use for weekly status reporting, as well as for periodic management reporting. This consolidated plan allows the team to know the true status of a project and to take corrective actions when necessary. A similar view is available for the team to analyze the status of the project.

TSP Project Reports

The primary outcome using the TSP model is project reports. Periodic review of project deliverables and regular status reports are built into the TSP process. These reports need to be accurate, actionable, and based on verifiable project data. In essence, these status reports can be thought of as *dials* on a development manager's equivalent of a pilot's instrument panel that show the status of many projects. A continual scan of the dials should reveal unfavorable project trends and allow for their early correction.

Status reporting for a software development project – the dials – provides a quantitative answer for some typical status questions such as the following:

1. How much of the project is complete at this time?
2. Given the progress at a given point in time, when will the project finish?
3. How much time has the development team been able to devote to project tasks?
4. Has the team spent substantial time on unplanned work?

Two charts can be created that will quickly provide answers to these questions. The first chart uses a measure called *earned value*. Earned value is one way to determine how much of a project is complete, and predict when it might finish. This measure is calculated by comparing the planned hours for completed tasks to the total planned hours for a project. A couple of comments on the earned value calculation: It is important to stick to the planned hours when calculating earned value even though more *actual* hours were needed to complete the tasks involved. Also, the hours for completed tasks should only include those tasks that are 100 percent complete. In other words, a task is done or it is not done. This prevents counting tasks that are perpetually 90 percent complete and thereby overstating a project's progress.

Figure 2 shows an earned value chart for a project. Notice that the *actual* earned value, or the sum of the hours for completed tasks, is shown against the *planned* earned value. In a straightforward way, these two lines show how well the project is tracking against the original plan. Ideally, these two lines will overlay each other, thus signifying a project that is on plan. As is often the case, however, actual earned value will lag behind planned earned value. When this lag is 10 percent or less, the cause may lie with major tasks that will soon finish or project conditions that will respond to minor corrections.

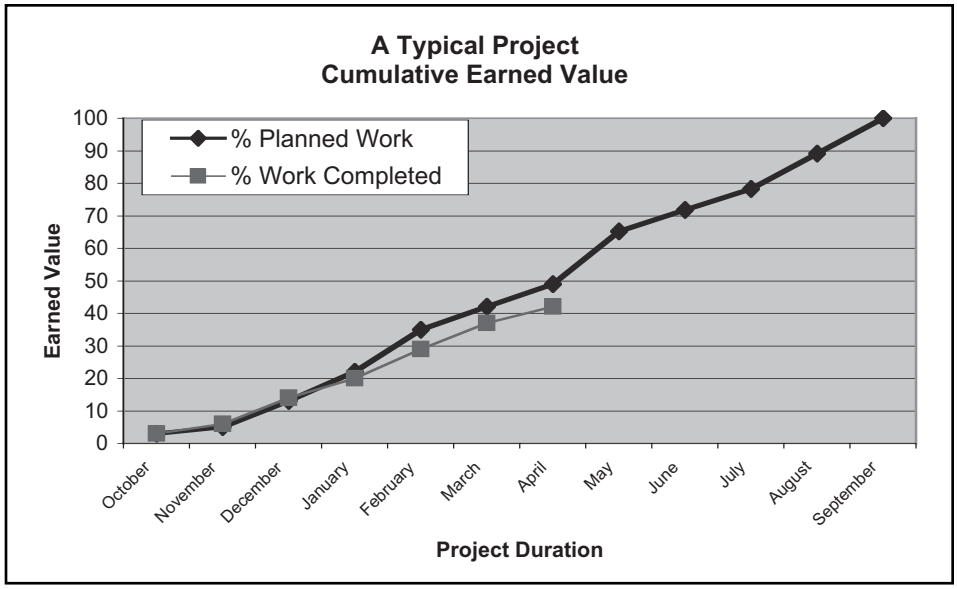


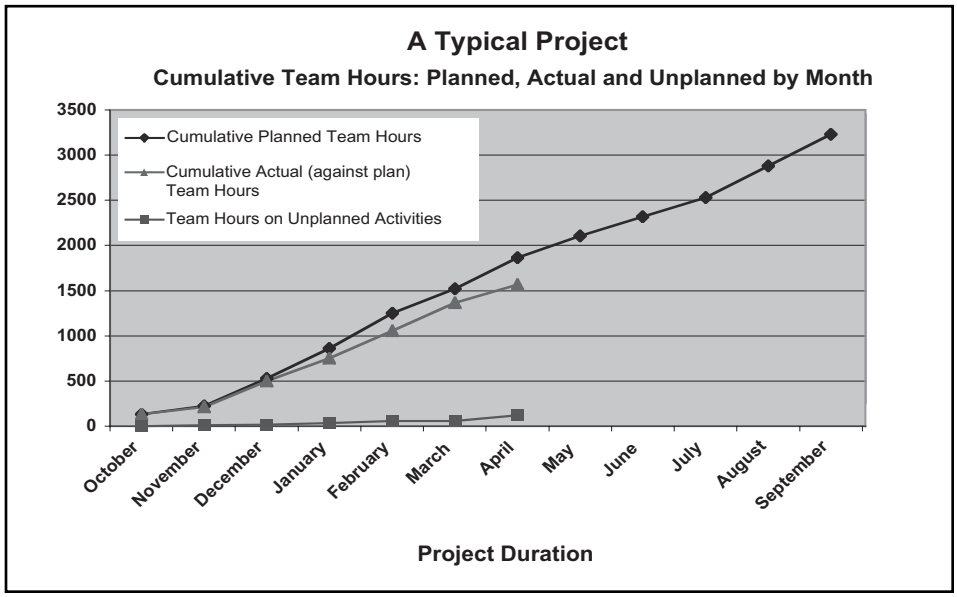
Figure 2: *Planned Versus Actual Earned Value*

When actual earned value deviates by a larger amount, stronger corrections may be needed. One of the causes of this deviation often is unplanned work. Work that was not anticipated and included in the project's planning, but was deemed necessary. A line plotting the hours for unplanned work is included in Figure 3 to track this phenomenon.

From looking at the earned value chart in Figure 2, overall project progress is easily determined. Since only completed tasks are counted, the actual earned value readily shows that the project is 40 percent complete. An expected time of completion can also be surmised from this data. Given a constant earned value rate for the project, a 10 percent variance on a 52-week plan would cause the project to finish approximately five weeks after the planned finish date. Alternatively, the development manager could attempt to recover and catch the project up by requiring overtime or assigning additional staff.

The planned and actual team hours shown in Figure 3 can often provide clues on why the project's earned value is lagging. In this case, the team's actual hours are behind what was planned; for some reason, team members were not able to devote as much time to the project as expected. This could result from an estimation error – typical software engineers can expect to put in 20 hours of a 40-hour

Figure 3: *Planned Versus Actual Hours*



week on project work. The rest go to meetings, ad-hoc assignments, etc. This 50 percent productivity factor can be lower depending on the environment. Other projects with competing priorities may have pulled team members away. Also, team hours used on unplanned activities are worth tracking and monitoring; this is the time spent on project-related work that was not anticipated or included as part of the project plan. Often these hours can be substantial, and one can see an increase in unplanned activities that are usually accompanied by a corresponding decline in actual team earned value.

Stepping back and taking a look at both charts, the development manager can draw the following conclusions. This project started to deviate from estimates as early as January. Progress appears to be hampered by the amount of time the team is able to apply to the project; if no action is taken, the project will miss its completion date. This is an early warning for the development manager to talk to the team lead and project members, ask some pointed questions, and develop a much richer picture of the project's status and obstacles.

In addition to schedule data, the TSP reports provide a rich set of quality data and an early warning system related to the quality being built. Although the previous two charts help us to track schedule progress, we also need to know that the product being built is of acceptable quality so it will not get bogged down during the test phase – a typical scenario in many software projects.

One *dial* that TSP does not provide the development manager with is projected and actual projects costs. There are a number of reasons why TSP-supplied data is not sufficient for providing financial status. TSP plans and measures activity in terms of available hours, and even though the typical software engineer has 20 hours available per week, he or she must be paid for the full week. There is also no provision for tracking common project expenses such as training and purchases. A scheme for loading available hours to accommodate these costs seems possible but has not been attempted in the projects in which either author has been involved. That said, TSP's performance and schedule have proven very useful at Sandia, and the use of the organization's financial system to help provide the overall project status has not been burdensome.

Managing Project Portfolio

The *secondary outcome using the TSP model is the ability to manage a portfolio of projects.* As mentioned before, Sandia's Information Systems Development Center has many projects at any given time; currently only three projects are using the TSP. However, development managers must report status on all projects in their portfolio to their superiors and funding sponsors. Understandably, these stakeholders want visible and objective measures of progress for the projects involved. Again, the desire is to stay abreast of the development portfolio and intervene should the development manager need assistance keeping projects on track. Whether that report is a scorecard or takes another format, accurate data that reflects the true state of affairs is needed.

The difference between two types of projects (TSP vs. non-TSP) is like day and night when one tries to prepare scorecard reports. The TSP projects have all the data collected as part of their process; with non-TSP projects, the project lead must work much harder to collect time worked, the numbers are often less accurate, and it is harder to determine the true status of the project. After having experienced both reporting systems, undoubtedly the TSP satisfies management by data requirements without any additional effort. As mentioned earlier, the developers readily see the value of data collection from their personal work processes, and when meeting weekly as a team.

Summary and Conclusion

We do not intend to leave the impression that all issues are resolved, and that TSP is being used seamlessly in this organization. The size issue raised during the first launch has not gone away. The organization uses function points to measure project size by personnel outside of the project. The TSP teams are not completely trained to use function points for their size estimates and are resistant to using lines of code as their measure. Developers reason that since they are working in a multi-tier architecture environment using multiple languages, using lines of code does not make sense for their application environment. The collection of defects at the personal level is just getting started, and defect-counting standards for design have been created. Developers have been hesitant to record defects, fearing that this information may fall into the wrong hands and be used for evaluation purposes. Not all teams have chosen to use TSP to respond to the management-by-data mandate.

On the other hand, software projects using the TSP have experienced a number of successes. The three TSP projects exhibit good project control and tend to need only minor corrections because problems are detected early. TSP meeting scripts and the guidance of the TSP launch coach have been an excellent way to support new development project leaders. The format of TSP team meetings and use of its roles has increased the teams' sense of ownership of the work and process. We look forward to improved quality and performance as metrics are collected from all members of the development team and used to improve team and personal processes. As mentioned earlier, the rigorous timekeeping of these TSP projects provides excellent project visibility. Since this data comes directly from those doing the work, status reports derived from this information are fairly objective and can provide good insight into the progress of a project.◆

About the Authors

Ray Trechter, Certified Software Development Professional, is a software development manager at Sandia National Laboratories. In addition to managing software development projects, Trechter has worked in the areas of software architecture, software process improvement, and as a developer of distributed systems.

Sandia National Laboratories
MS 0661
Albuquerque, NM 87185
E-mail: ratrech@sandia.gov

Iraj Hirmanpour is principal of AMS, a software process improvement firm and a Software Engineering Institute Personal Software Process℠ (PSP℠) and Team Software Process℠ (TSP℠) transition partner. Hirmanpour is a PSP/TSP consultant and trainer on the transition of technology to organizations.

AMS, Inc.
421 Seventh ST NE
Atlanta, GA 30308
Phone: (404) 394-2028
E-mail: ihirman@earthlink.net