The Team Software Process: Breaking the SPI Barrier

Alan Willett for the Boston, SPIN
November 16th, 2004

This material is approved for public release. Distribution is limited by the Software Engineering Institute to attendees.

Sponsored by the U.S. Department of Defense
© 2004 by Carnegie Mellon University
Trademarks and Service Marks

The following are service marks of Carnegie Mellon University.
• Capability Maturity Model Integration SM
• CMMi SM
• Team Software Process SM
• TSP SM
• Personal Software Process SM
• PSP SM

The following are registered trademarks of Carnegie Mellon University
• Capability Maturity Model®
• CMM ®
Overview

This talk is intended primarily for people who work on improving the software process.

My objective is to provide an overview of the TSP and illustrate it with examples from my personal experiences with process, teams, and organizations.

The presentation is organized as follows:
• The software “problem”
• The solution of “SPI” and its associated barriers
• An overview of the PSP
• An overview of the TSP
• The introduction strategy
Typical Industry Schedule Performance

Reported by the Standish Group

Of the cancelled projects, average schedule error was 222%
Every Business is a Software Business

Software is now a critical component of nearly all products and services.

Software costs and schedules now dominate many business plans.

Software quality is also a critical concern.
  • cancelled projects
  • costly rework
  • safety issues
Current Software Practice

Software is the only modern technology that ignores quality until test. Typically, software engineers
• do not participate in planning their work
• are rushed through requirements and design
• do the design while coding

These practices introduce volumes of defects.
• Experienced engineers inject a defect every 7 to 10 lines of code.
• For even moderate-sized systems, this is thousands of defects.
• Most of these defects must be found in test.
• This usually takes about half of the development schedule.
Software Business Needs

1. Better/predictable cost and schedule management
2. Effective quality management
3. Cycle time or time to market reduction
The Answer is “SPI”

Why Software Process Improvement?

• Doing the same thing over and over again gives the same results.

• The process must be changed to get different results.
Process Improvement Pays Off - 1

<table>
<thead>
<tr>
<th>Level</th>
<th>Defects/KLOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level 1</td>
<td>7.5</td>
</tr>
<tr>
<td>Level 2</td>
<td>6.24</td>
</tr>
<tr>
<td>Level 3</td>
<td>4.73</td>
</tr>
<tr>
<td>Level 4</td>
<td>2.28</td>
</tr>
<tr>
<td>Level 5</td>
<td>1.05</td>
</tr>
</tbody>
</table>
Process Improvement Pays Off - 2

Schedule Deviation Individual Value Control Chart - Commercial Systems

Date of Project Start

- Individual Data Points
- Mean
- Upper Natural Process Limit
- Lower Natural Process Limit
- One Standard Deviation

[Source: AIS]
But there are SPI BARRIERS

Software project members often view SPI work as a tax.

Software project members don’t readily adopt the work done for the projects by the Software Engineering Process Group (SEPG) or the organizational “process action teams”.

There is often a lack of consensus on how to do SPI.

The SEPG often find it difficult to role model project management with SPI.

Planning SPI is especially difficult.

Process improvements take a long time.
## Time Between CMM Levels

### Average Historical Times Between Levels (CMM)

<table>
<thead>
<tr>
<th>From Level to Level</th>
<th>Average Time to Move Up (Months)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level 1 to 2</td>
<td>22</td>
</tr>
<tr>
<td>Level 2 to 3</td>
<td>21</td>
</tr>
<tr>
<td>Level 3 to 4</td>
<td>25</td>
</tr>
<tr>
<td>Level 4 to 5</td>
<td>13</td>
</tr>
</tbody>
</table>

Data source: September 2003 SEI Process Maturity Profile of the Software Community
(At this time, the SEI has insufficient data to publish similar CMMI data.)
TSP breaks the SPI Barrier

Organizations have achieved high CMM maturity rating 4 times faster than the historical average using TSP. A NAVAIR organization went from 2 to 4 in 16 months as opposed to the average of 46 months.

Many TSP projects achieve CMM Level 5 cost, schedule, and quality performance on their first project.

Team members like it; turn-over has been near 0.

Predictable costs and schedules, within +/- 10%.

Productivity has nearly doubled for some teams.

Late defects and testing schedules are reduced by 4 to 5 times.

Release product quality better than .1 defects/KLOC.
TSP Pays Off - 1

Defects/KLOC

Level 1: 7.5
Level 2: 6.24
Level 3: 4.73
Level 4: 2.28
Level 5: 1.05
TSP: 0.06
TSP Pays Off - 2

Schedule Deviation Individual Value Control Chart - Commercial Systems

Date of Project Start

- Individual Data Points
- Mean
- Upper Natural Process Limit
- Lower Natural Process Limit
- One Standard Deviation

[Source: AIS]
The Team Software Process

The TSP is a process framework for building and guiding engineering teams that develop software.

A typical TSP team consists of 3 to 15 members, not necessarily all software engineers.

Larger projects are organized into several TSP teams operating with TSPm, a multi-team extension to TSP.

If you are familiar with the SEI CMM, think of TSP as an instance of a Level 5 process for a team.

Team members must exercise process discipline, so PSP is a prerequisite for TSP.
The Personal Software Process

The PSP is a process designed for individual use, based on scaled-down industrial software practice.

With PSP, engineers
• are process users and owners
• routinely estimate and plan their work
• gather data for tracking and improvement
• manage quality at every step of the process

If you are familiar with the SEI CMM, think of PSP as an instance of a Level 5 process for an individual.
Building High-Performance Teams

The TSP strategy is to improve performance from the bottom up.

This strategy starts with PSP training.

Quality work starts at the bottom.

Team Member Skills
- Process discipline
- Performance measures
- Estimating & planning skills
- Quality management skills

Team Building
- Goal setting
- Role assignment
- Tailored team process
- Detailed and balanced plans

Team Management
- Team communication
- Team coordination
- Project tracking
- Risk analysis

PSPTSP
TSP, PSP, and the CMM

CMMs - Build organizational capability

TSP - Builds quality products on cost and schedule

PSP - Builds individual skill and discipline
<table>
<thead>
<tr>
<th>Level</th>
<th>Focus</th>
<th>Key Process Areas (KPA)</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>Optimizing</td>
<td>Continuous process improvement</td>
</tr>
<tr>
<td></td>
<td></td>
<td>✓ Defect prevention</td>
</tr>
<tr>
<td></td>
<td></td>
<td>✓ Technology change management</td>
</tr>
<tr>
<td></td>
<td></td>
<td>✓ Process change management</td>
</tr>
<tr>
<td>4</td>
<td>Managed</td>
<td>Product and process quality</td>
</tr>
<tr>
<td></td>
<td></td>
<td>✓ Quantitative process management</td>
</tr>
<tr>
<td></td>
<td></td>
<td>✓ Software quality management</td>
</tr>
<tr>
<td>3</td>
<td>Defined</td>
<td>Engineering process</td>
</tr>
<tr>
<td></td>
<td></td>
<td>✓ Organization process focus</td>
</tr>
<tr>
<td></td>
<td></td>
<td>✓ Organization process definition</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Training program</td>
</tr>
<tr>
<td></td>
<td></td>
<td>✓ Integrated software management</td>
</tr>
<tr>
<td></td>
<td></td>
<td>✓ Software product engineering</td>
</tr>
<tr>
<td></td>
<td></td>
<td>✓ Intergroup coordination</td>
</tr>
<tr>
<td></td>
<td></td>
<td>✓ Peer reviews</td>
</tr>
<tr>
<td>2</td>
<td>Repeatable</td>
<td>Project management</td>
</tr>
<tr>
<td></td>
<td></td>
<td>✓ Requirements management</td>
</tr>
<tr>
<td></td>
<td></td>
<td>✓ Software project planning</td>
</tr>
<tr>
<td></td>
<td></td>
<td>✓ Software project tracking</td>
</tr>
<tr>
<td></td>
<td></td>
<td>✓ Software quality assurance</td>
</tr>
<tr>
<td></td>
<td></td>
<td>✓ Software configuration management</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Software subcontract management</td>
</tr>
</tbody>
</table>

✓ CMM Key Process Area addressed at the project level when using PSP and TSP
PSP Overview
The PSP Strategy - 1

Software is currently taught as an individual skill.
• Engineers focus on producing programs that run.
• The emphasis is on individual performance.

For development teams, teamwork skills are needed.
• planning and coordinating the work
• overall product quality
• total project productivity
The PSP Strategy- 2

Personal Software Process (PSP) training teaches software professionals
• what sound engineering methods are
• how to use these methods in their work

With the Team Software Process (TSP), professionals use the methods they learned in PSP training. They
• follow a defined and measured process
• plan, track, and manage their personal work
• gather time, size and defect data
• use these data to consistently produce quality products
Sound Engineering Practices

Until they try them, most programmers do not believe that sound engineering practices will be helpful.
- They won’t try the practices without evidence.
- They can’t get evidence without trying the practices.

In the PSP course, engineers use the PSP to write 10 programs.

They can see from their personal data that sound engineering practices work for them.
The PSP Course

PSP0
• Current process
• Basic measures

PSP1
• Size estimating
• Test report

PSP1.1
• Task planning
• Schedule planning

PSP2
• Code reviews
• Design reviews

PSP2.1
Design templates

PSP3
Cyclic development

Team Software Process
• Teambuilding
• Risk management
• Project planning and tracking

Introduces process discipline and measurement
Introduces estimating and planning
Introduces quality management and design
My Experience with PSP
My Experience with PSP

I was frustrated with speed and effectiveness of my organization’s process improvement.

I decided to take the Personal Software Process.

Although it had been about 10 years since I had written software programs, I was excited to take the course.
What doing the PSP course did for me

Size Estimating Error

Time Estimating Error

Defects Removed in Unit Test

Failure Cost of Quality

Productivity vs. Yield
But speed and overall defects for the first 9 programs...

I could have explained these parts of my performance by not having programmed in 10 years…but I was not satisfied. I set aggressive goals for the 10th program.
My Goals for Program 10

• I have a level 5 personal process.
• I will look for improvements that if applied to 1a-9a would product targets results of:
  – <80 Total Defects/KLOC
  – <5 Unit Test Defects/ KLOC
  – 2x Delivery
My Goal

*Faster Delivery times of Beautiful, High Quality Software*

- Question: What types of actions would help achieve this goal?
  - Less Defects /KLOC (Defect Prevention)
    - Examine defect logs, and design preventive measures
    - more beautiful designs (design class? java class? A Mentor?)
    - work at my mental peaks (not late nights, not by waking at 4:30 AM)
    - more contiguous time -- don’t juggle so many things
  - Higher Yields = Less Defects in test = less time
    - Better Inspections
      » Examine Defect logs, and insert checks for biggest hitters
  - Higher Productivity (by doing the above -- I can’t think of any way to just work faster? Perhaps design classes and java classes could help me work faster.)
  - Smaller Overall Code Count (better design) THUS Less Time
What would the quality and delivery of 9A look like if I had prevented 60% of the defects?

- From 57 to 22 defects injected.
- 188 Total Defects/KLOC to 75 Total Defects/KLOC
- 20 Test Defects / KLOC to 8 Test Defects / KLOC.
  - And at a only 73% yield rate, there is room for improvement to something respectable here.
- Delivery from 1563 minutes (25 Hours!) to 1362 minutes (22.6 hours). A 201 minute improvement.
Can 60% defect prevention be achieved?

- An analysis of the defect log shows YES. Here are the simple steps to take.
  - Print out the top 10 defects I make in design and read it before I write each design.
  - Fully use the design templates. I have not been doing that.
  - Print out the top 10 defects I make in code. Read it each time before I code.
  - Augment design with some loop templates - proper terminations, notes on counters.
  - Spend time learning about casting of different types - trying things out till I have it cold.
  - Add some coding templates to address some types of commenting problems.

- I know that the 40% is a conservative minimum of the defects I will prevent. I believe that 60% is possible with the other actions I have planned.
Will reduced size yield speed?

- Looking at program 6a.
  - 205 LOC. I examined this and explored different designs. I found one that I am confident I could have written the program with 50 less lines of code.
  - At my productivity for 6A of 13 LOC, this would be a 230 minute reduction to 711 minutes (after defect prevention).
  - The combined actions are a great improvement. Still not 2x, but I believe this has potential.
# My Results for Program 10

<table>
<thead>
<tr>
<th></th>
<th>Average first 9 Programs</th>
<th>Program 10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time Estimation Error:</td>
<td>38%</td>
<td>1%</td>
</tr>
<tr>
<td>Total # Defects/ KLOC:</td>
<td>231</td>
<td>77</td>
</tr>
<tr>
<td>Defects/KLOC in Unit Test:</td>
<td>42</td>
<td>9</td>
</tr>
<tr>
<td>LOC/ Hour:</td>
<td>10</td>
<td>16</td>
</tr>
</tbody>
</table>
Overall PSP Results
Effort Estimation Accuracy Trend

Program Number

Estimated Minutes - Actual Minutes / Estimated Minutes

Mean Time Misestimation
PSP Level Average
Design Quality

Time Invested Per (New and Changed) Line of Code

Program Number

Mean Minutes Spent Per LOC

- Design
- Code
- Compile
- Test
Product Quality

Defects Per KLOC Removed in Compile and Test

Program Number

Mean Compile + Test

PSP Level Mean Comp + Test

Mean Number of Defects Per KLOC
Take the PSP!
Overview of TSP
A Software Product is a Team Effort

Software products are made from software modules or components.

These modules are designed, built, integrated, and tested by a team of software engineers.

The team’s skills, spirit, discipline, and commitment govern the results.
Successful Team Practices

When engineering teams work well, they generally have successful projects.

To be successful, teams need
- clear goals
- established roles
- defined processes
- agreed-upon plans

Without these, teams will rarely be successful.
Building Self-Directed Teams

Self-directed teams must be built.

This requires a teambuilding process.

- **Team Building**
  - Goal setting
  - Role assignment
  - Tailored team process
  - Detailed and balanced plans

- **Team Member Skills**
  - Process discipline
  - Performance measures
  - Estimating & planning skills
  - Quality management skills

- **Team Management**
  - Team communication
  - Team coordination
  - Project tracking
  - Risk analysis

© 2004 by Carnegie Mellon University

October 2004

Breaking the SPI Barrier - 45
The TSP Launch

In the four-day TSP teambuilding process, teams
• establish their own goals and roles
• define their own strategies and processes
• produce their product development plans
• defend their plans to management

TSP teams
• own their own processes and plans
• understand why their goals are important
• have voluntarily committed to their goals
• believe that their commitments are achievable

All team members participate in the launch.
The TSP Launch Process

Day 1
1. Establish product and business goals
2. Assign roles and define team goals
3. Produce development strategy and process

Day 2
4. Build overall and near-term plans
5. Develop the quality plan
6. Build individual and consolidated plans

Day 3
7. Conduct risk assessment
8. Prepare management briefing and launch report

Day 4
9. Hold management review
Launch postmortem

A qualified TSP team coach guides the team through a defined process to develop its plan and to negotiate that plan with management.
Establish Product and Business Goals

The first step in the launch is for the team to understand what they are being asked to do.

In meeting 1, management and marketing meet with the team to tell the team:
- what they want the team to develop
  - essential features
  - nice-to-have features
- when the product is needed
- what resources the team has
- what flexibility the team has
- why this job is important
- how management will measure success
Establish Team Goals

In launch meeting 2, the team sets its goals and organizes itself.

The team starts with the goals defined by management.
   • reviews and refines management’s goals
   • adds team-specific goals
Organize the Team -1

Self-directed teams manage their own work.

This responsibility must be shared.

Traditional project management responsibilities are distributed among eight team roles.

- customer interface manager
- design manager
- implementation manager
- test manager

- planning manager
- quality manager
- process manager
- support manager
Organize the Team -2

Each team member assumes the responsibility for at least one team management role.

Defined roles
• ensure that key areas are covered
• give each team member a responsibility in running the team
• build a sense of membership and ownership among the team members
• free the team leader to focus on the role of coaching and leading the team
Define the Work and the Approach

In meeting 3, the team accomplishes three important prerequisites to building the team plan.

- identifies all of the "work" the team needs to do
- identifies the "build strategy" the team will use to develop the software
- identifies or defines the "processes" the team will follow to do the work
In launch meeting 4, the team creates the overall plan.

- the estimated sizes of the products to be produced (how big is the job)
- the tasks needed to do the work (with effort estimates)
  - next-phase tasks, detailed to the work step level
  - later phases at a high level
- the estimated team hours available each week for the work
- an initial schedule for the project

The team may consider alternative plans if management goals are not met.
Build a Quality Plan

In launch meeting 5, the team builds a quality plan that estimates the
- number of defects that will be injected in each phase
- number of defects that will be removed in each phase
- quality (defect density) of the final product
- quality (process quality index) of the development process

The team ensures that the plan meets the quality goals.
Build Individual Work Plans

In launch meeting 6, the team allocates tasks to the team members, who build individual work plans for their tasks for the next phase.

In building their plans, the engineers
• refine size and effort estimates using their own data and processes
• break tasks to the granularity of around 10 task hours per task or less
• estimate their own available task hours for each week
• create an earned-value plan

The team reviews individual plans and balances workloads across all team members.
Build a Consolidated Team Plan

The individual team-member plans for the next project phase are consolidated into a single team plan that contains a

• task list of all individual tasks with
  - estimated hours
  - planned date
  - planned value

• schedule with
  - planned team hours per week
  - planned team earned value per week

The team uses this plan to guide and track its work during the next project phase.
Identify Risks and Mitigation Strategies

In launch meeting 7, the team identifies and assesses risks to the team meeting its plan.

The risks are documented in the team’s plan.

Risks are assigned to team members to track during the project.

The team identifies mitigation and recovery strategies for high-priority risks.
The TSP Launch Products

Business needs
Management goals
Product requirements

What?
- Team goals
- Conceptual design
- Planned products
- Size estimates

How?
- Team strategy
- Team defined process

When?
- Task plan
- Schedule Plan
- Earned-value Plan

Who?
- Team roles
- Task plans
- Earned-value Plan

How well?
- Quality plan

What if?
- Risk evaluation
- Alternate plans
Prepare Management Briefing

In launch meeting 8, the team prepares a presentation on the plan for management.

If the team’s plan does not meet management’s goals, the team generates alternate plans to come closer to or meet management’s goals by

• adding more resources
• delivering a reduced-functionality initial version, followed later by a full-functionality version(s)
Management Review of the Plan

In launch meeting 9, the team presents its plan (and alternatives, if any) to management and answers questions.

Management determines whether
- the plan is based on sound estimates, historical data, and a quality process
- the team believes in their plan
- the plan or an alternative meets the business need

Management
- approves the plan or an alternative
- asks the team to look at a different alternative
After the TSP Launch:

TSP Teams in Motion
Managing Self-Directed Teams

Capitalizing on team potential is management’s responsibility.

Team Member Skills
- Process discipline
- Performance measures
- Estimating & planning skills
- Quality management skills

PSP

Team Building
- Goal setting
- Role assignment
- Tailored team process
- Detailed balanced plans

Team Management
- Team communication
- Team coordination
- Project tracking
- Risk analysis

TSP

Senior Management
- Team support
- Team discipline
- Program visibility

Team Performance

© 2004 by Carnegie Mellon University
“Quality without numbers is just talk.”

TSP uses four base measures.
  • size
  • time
  • defects injected and removed
  • task-completion date

Both plan and actual data are gathered and recorded.

Actual data are used for tracking to the plan.

The personal and team data are archived to provide a repository of historical data for future use.
The TSP measurement framework consolidates individual data into a team perspective.

Other views of the data are available: by product or part, by phase, by task, by week, etc.

Many other measures are derived from this base.
EXAMPLES OF TSP TEAMS IN MOTION
Fly the Plan - 2

<table>
<thead>
<tr>
<th></th>
<th>Plan</th>
<th>Actual</th>
<th>Plan/Actual</th>
</tr>
</thead>
<tbody>
<tr>
<td>Schedule hours this cycle to date</td>
<td>265.2</td>
<td>273.4</td>
<td>0.97</td>
</tr>
<tr>
<td>Earned value this cycle to date</td>
<td>69.4</td>
<td>52.6</td>
<td>1.32</td>
</tr>
<tr>
<td>To-date hours for tasks completed</td>
<td>199.7</td>
<td>259.9</td>
<td>0.77</td>
</tr>
</tbody>
</table>

Work hours on target

Completed Tasks show severe underestimates

Detail tasks log shows that majority of problem is in Unit Test

Defect Fix Time by type shows that majority of problem is legacy system defects
The team upon understanding the problem undertook what it could do.

- Inspections of legacy code
- Looked at ways to increase effort hours

The team worked with the management and customer to
- increase the project team size

The team delivered in time for the customer a defect free product.
A gelled team = speed

What happens when a teammate is falling behind?

In one organization, this occurred before TSP
  • team members complained to management with no effective results
  • the project delivered late a product the customer rejected

In this organization's first TSP project, with many of the same people, it was occurring again.
  • some members of the team met and decided to approach the behind team member
  • they offered help and he readily accepted
  • through mentorship and reallocating work the team delivered on time a high quality product
Inspections = Speed

It took our Code Peer Review 7000 total effort minutes to find & fix 280 defects. (this was our most time consuming inspection)

It took 7100 total effort minutes to find and fix 5 system test defects.
Quality = Speed

The quickest this organization completed system test was about 30% of the overall schedule.

Their first TSP project
- had 7400 lines of code
- 3 documentation defects in system test
- 0 code or design related defects
- Completed system test in 3 days, less then 4% of overall schedule
Introduction
Strategy
For More Information

Visit the Software Engineering Institute web site at www.sei.cmu.edu

Visit the TSP web site at www.sei.cmu.edu/tsp

For a list of TSP SEI Transition Partners, visit the web site at http://www.sei.cmu.edu/tsp/partners.html

Contact SEI Customer Relations at 412-268-5800
Getting Started -1

SEI’s recommended strategy for introducing TSP involves these steps:

• Identify key areas for initial introduction.
• Hold executive seminar and transition planning session.
• Identify projects that could serve as pilots for TSP.
• Train the affected managers and engineers.
• Conduct a few (2-4) trial-use projects.
• Evaluate pilot project results.
• Train and authorize an internal TSP/PSP transition team.
• Plan for and initiate broad rollout.
Getting Started - 2

Executive seminar and planning session - 1 1/2 days.

Management training - 3 days.

Engineer training - 12 days.
  • teaches personal project management
  • engineers learn to measure and manage quality
  • required for effective team participation

Internal transition agent training:
  • PSP Instructor - 5 days.
  • TSP Launch Coach (FY01) - 5 days.
Broad Rollout Strategy

The rollout strategy follows the same steps as the initial introduction.
• Identify projects.
• Train affected managers and engineers.
• Launch projects using TSP.

The internal transition team can be expanded as needed.

The SEI or Transition Partners will provide support for the initial pilot effort
• Offer on-site training.
• Provide support for pilot projects.
• Train and mentor the internal transition team.

The internal transition team then assumes the lead for the rollout.
Summary
Quotes from an SEPG using TSP

“This was difficult but very worth it. We could not have done this without the TSP.” – S/W manager part time on SEPG

“Our management group is a team now.” – the organization’s overall manager

“I look forward to doing a TSP launch on my software project.” - S/W manager part time on SEPG

“This is the best planned, best run SPI effort I have been part of.” - SEPG member

“This is the first SPI program I have been part of that has actually helped the engineers.” – Long Time Engineer speaking to SEPG members.
Questions?
Contact Information

Alan W. Willett  
Software Engineering Institute  
Carnegie Mellon University  
Pittsburgh, PA 15213-3890

Email: awillett@sei.cmu.edu  
Phone: 607-592-7279