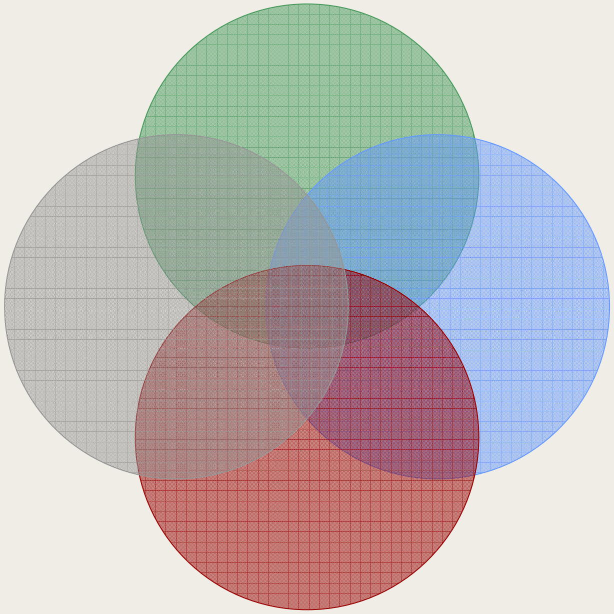
Team Software Process and Personal Software Process

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213-3890

Sponsored by the U.S. Department of Defense

Calidad = T P ^{3}



Producto

Tecnología

Proceso

Personas

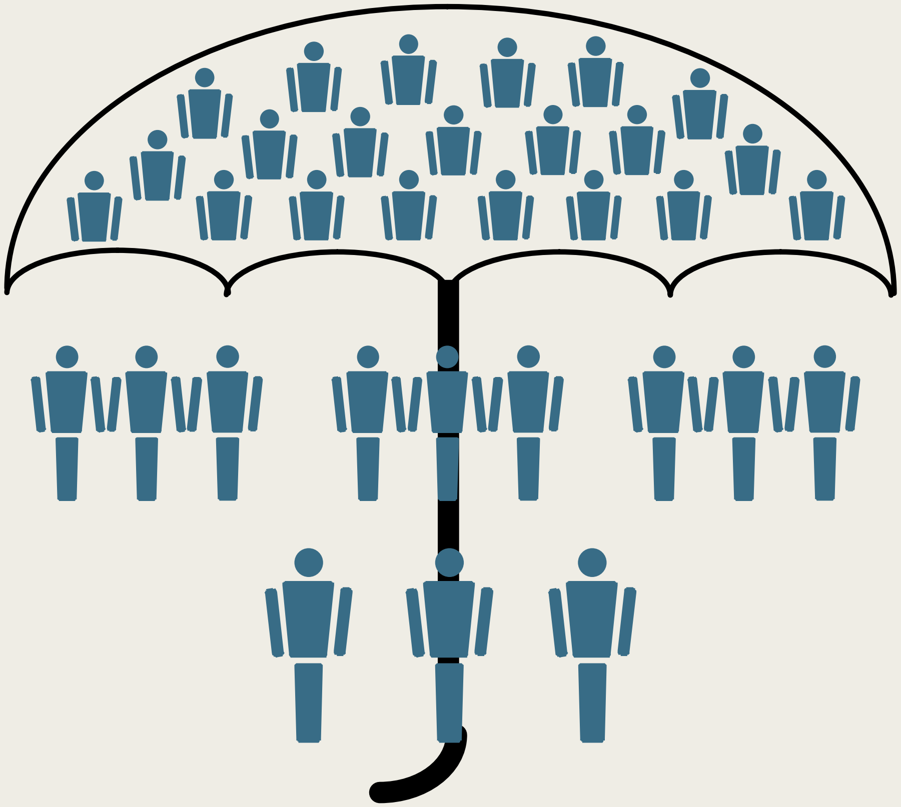
Fuente: Ing. Rafael Salazar

Improving Software Practice

CMMI – *Model of organizational capability*

TSP – *Instance of high-maturity practice for teams*

PSP – *Instance of high-maturity practice for individuals*



Software Process Quality

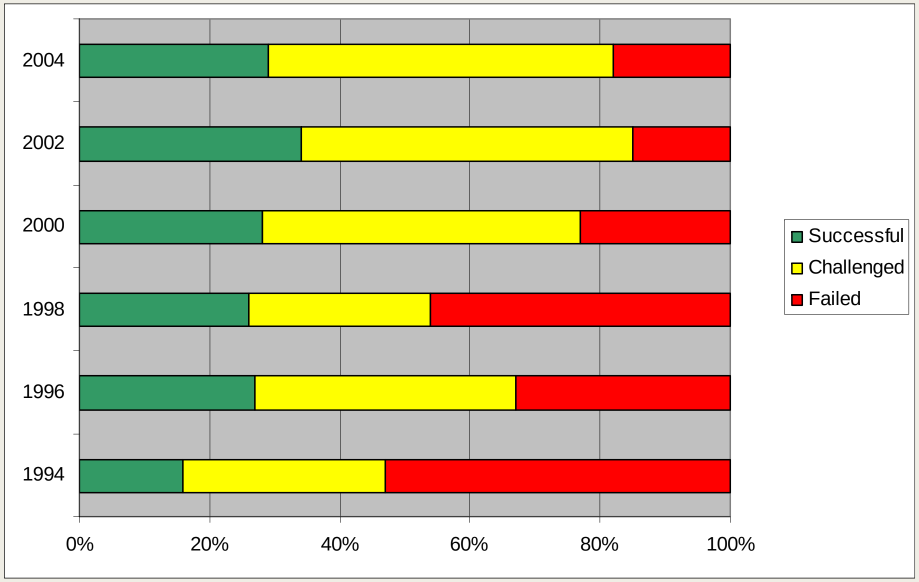
Software is the only modern technology that ignores quality until test.

Most software defects are found in or after test when defect removal is the most expensive and least effective.

This strategy results in buggy products and unnecessary rework, inflating development costs.

Software defects are also a principal cause of software security vulnerabilities.

Cost and Schedule Performance Trends



Successful projects delivered on time, on budget, with required features and functions.

Challenged projects estimated a 43% average cost overrun, time overruns of 83%, and delivered only 52% of required features and functions (in 2002).

Failed projects were cancelled prior to completion or delivered and never used.

(This chart represents over 50,000 IT projects in large, medium, and small cross-industry world-wide companies tested by The Standish Group since 1994.)

Team Software Process

The Team Software Process (TSP) is a software development process for engineering teams.

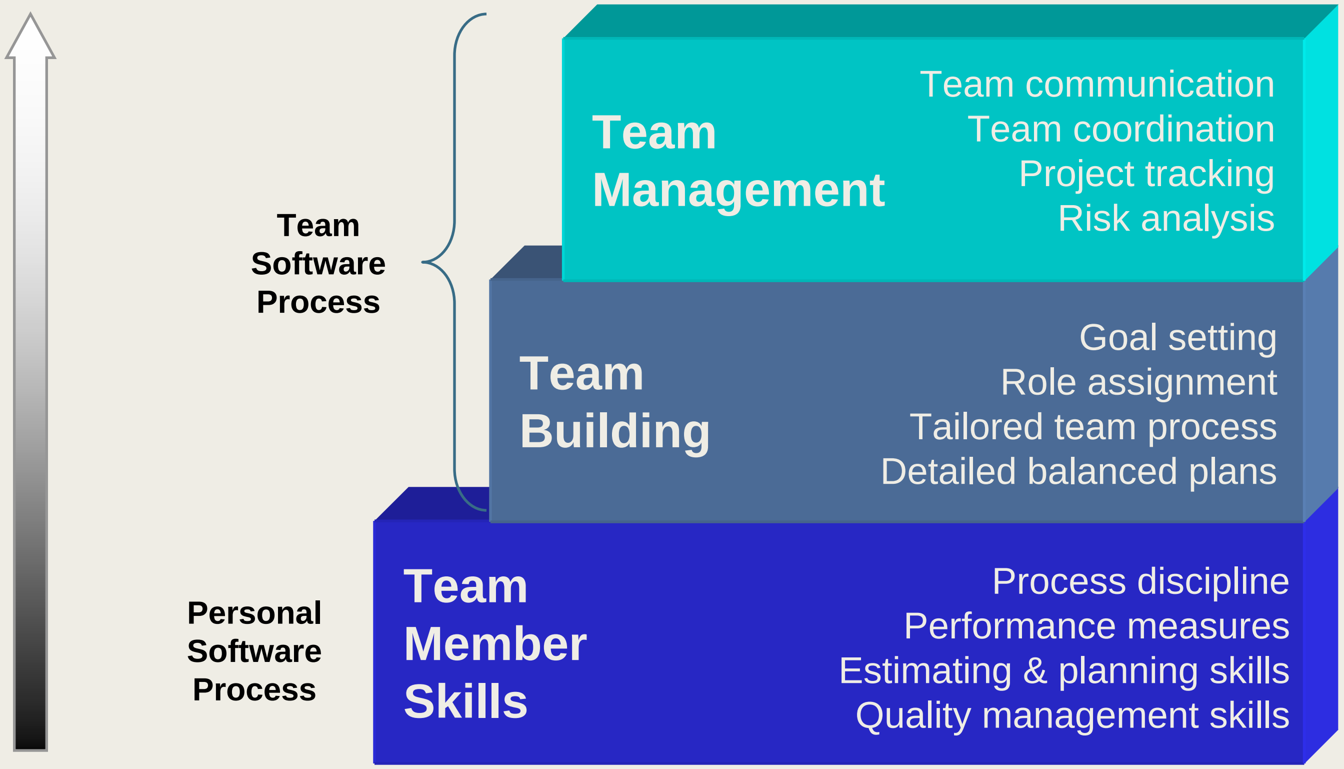
TSP is a process-based solution to common software engineering and management issues.

- cost and schedule predictability
- productivity and product quality
- process improvement

Unlike other methods, TSP

- combines discipline and agility
- improves team and individual performance
- provides immediate, measurable business benefits

Building High-Performance Teams



TSP builds high-performance teams from the bottom-up.

Carnegie Mellon
Software Engineering Institute

Pittsburgh, PA 15213-3890

Personal Software Process

Sponsored by the U.S. Department of Defense

Personal Software Process

The PSP is a process designed for individual use that applies to structured personal tasks.

With PSP, developers use defined and measured personal processes.

They gather size, time, and defect data as they work.

They use the data to

- plan and track their work
- manage the quality of the products they produce
- measurably improve performance

PSP Changes Software Practice

Software work is planned.

Plans are based on processes and estimates.

Estimates are based on historical process data.

Software work is measured and tracked.

Status is based on the data.

Software quality is also planned, estimated, tracked, and managed.

What is a Process?

A process is a defined set of steps for doing a job.

A process guides your work.

A process is usually defined for a job that is done multiple times.

A process provides a foundation for planning.

- A *process* is a template, a generic set of steps.
- A *plan* is a set of steps for a specific job, plus other things such as effort, costs, and dates.

What is a Personal Process?

When you use a process for your personal work, it is called a personal process.

It is usually developed from your personal experience.

- You may start with a proven process that was developed by someone else.
- You modify and improve the process to suit your needs.

Elements of a process

In documenting a process, it is common to specify the

- required inputs to the process
- steps of the process
- exit criteria for the process

The steps are described only with enough detail to guide the user of the process; they are not a set of training instructions.

The next slide shows an example of a process for balancing a checkbook.

Example Personal Process

Inputs required	• Checkbook register balance is current and is reconciled with previous account statement • New statement • Notepad and calculator
Verify checks phase	For each check on new statement, verify with checkbook register and mark as cleared
Update register phase	• Record ATM and EFT transactions in checkbook register, mark as cleared, and update register balance • Add interest to register balance • Subtract service charges from register balance
Reconcile register phase	• On notepad, subtract all checks not marked as cleared from the new statement balance and compare to new register balance • If the two balances don’t agree, find error in register and correct
Exit criteria	Checkbook register reconciled with new account statement

Why Define and Use a Personal Process?

Benefits of a personal process include

- consistency
- efficiency
- basis for improvement

The next slides will discuss these points.

Consistency

Using a defined personal process helps you to achieve consistent results.

- Your results are more likely to be similar each time that you use the process.
- Your work becomes more predictable.

Efficiency

Using a defined personal process helps you to be more efficient.

- It structures and guides your work.
  - orders the steps
  - avoids rework
- It keeps you focused on what needs to be done.
  - fewer restarts
  - manage interrupts

You can accomplish your work in less time.

Basis for Improvement

By gathering data on your work, you can determine which steps

- take the most time
- cause you the most trouble
- are least effective

With this information, you can identify opportunities for improving your results by making changes to your process.

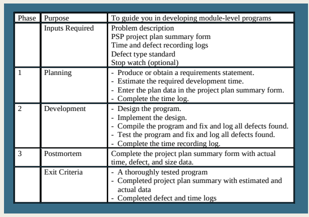
The Need for Personal Discipline

Building successful high-performance teams requires more than technical ability; team members must be committed to the concept of personal discipline.

Personal discipline means that all team members understand their own abilities and can make realistic commitments to each other and to management.

Team members develop personal discipline by learning the principles of the Personal Software Process (PSP).

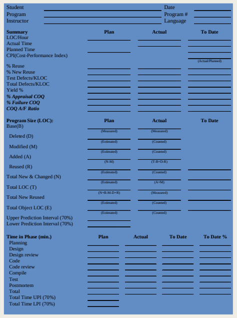
The PSP Process Elements



Phase	Purpose	To guide you in developing module-level programs
	Inputs Required	Problem description PSP project plan summary form Time and defect recording logs Defect type standard Stop watch (optional)
1	Planning	- Produce or obtain a requirements statement. - Estimate the required development time. - Enter the plan data in the project plan summary form. - Complete the time log.
2	Development	- Design the program. - Implement the design. - Compile the program and fix and log all defects found. - Test the program and fix and log all defects found. - Complete the time recording log.
3	Postmortem	Complete the project plan summary form with actual time, defect, and size data.
	Exit Criteria	- A thoroughly tested program - Completed project plan summary with estimated and actual data - Completed defect and time logs

Scripts Document the process entry criteria, phases/ steps, and exit criteria. The purpose is to guide you as you use the process.

Measures Measure the process and the product. They provide insight into how the process is working and the status of the work.



Student ______________ Date ______
Program ______________ Program # ______
Instructor ______________ Language ______

Summary	Plan	Actual	To Date
LOC/Hour			
Actual Time			
Planned Time			
CPI(Cost-Performance Index)			(Actual/Planned)
% Reuse			
% New Reuse			
Test Defects/KLOC			
Total Defects/KLOC			
Yield %			
% Appraisal COQ			
% Failure COQ			
COQ A/F Ratio			

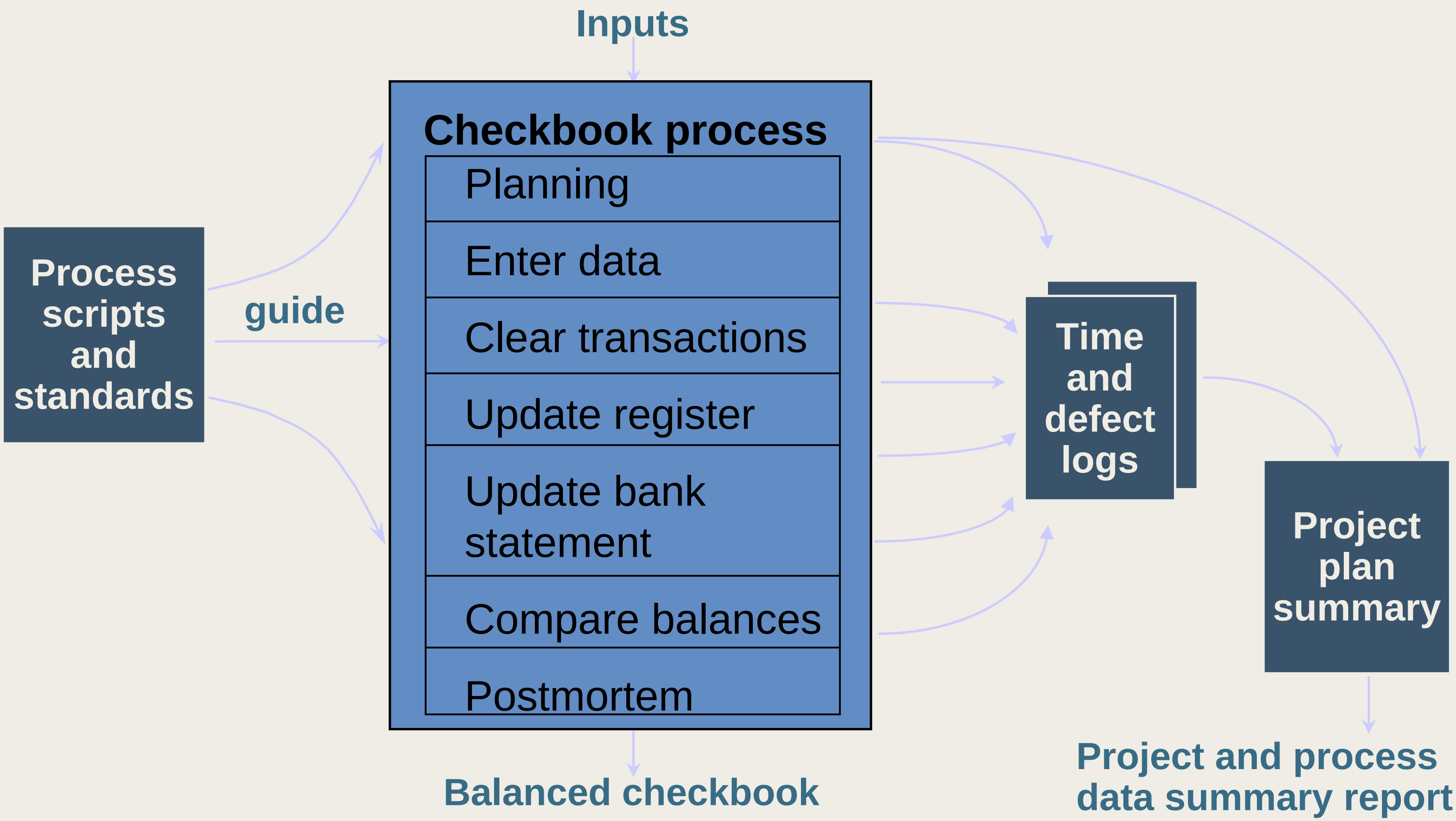
Program Size (LOC):	Plan	Actual	To Date
Base(B)	(Measured)	(Measured)	
Deleted (D)	(Estimated)	(Counted)	
Modified (M)	(Estimated)	(Counted)	
Added (A)	(N-M)	(T-B+D-R)	
Reused (R)	(Estimated)	(Counted)	
Total New & Changed (N)	(Estimated)	(A+M)	
Total LOC (T)	(N+B-M-D+R)	(Measured)	
Total New Reused	(Estimated)	(Counted)	
Total Object LOC (E)	(Estimated)	(Counted)	
Upper Prediction Interval (70%)			
Lower Prediction Interval (70%)			

Time in Phase (min.)	Plan	Actual	To Date	To Date %
Planning				
Design				
Design review				
Code				
Code review				
Compile				
Test				
Postmortem				
Total				
Total Time UPI (70%)				
Total Time LPI (70%)				

Forms Provide a convenient and consistent framework for gathering and retaining data

Standards Provide consistent definitions that guide the work and gathering of data.

Putting the Elements Together



Inputs

Process scripts and standards

guide

Checkbook process

- Planning
- Enter data
- Clear transactions
- Update register
- Update bank statement
- Compare balances
- Postmortem

Time and defect logs

Project plan summary

Balanced checkbook

Project and process data summary report

PSP Base Measures

Size

Effort

Quality

Schedule

Source: CMU/SEI-92-TR-019

Carnegie Mellon
Software Engineering Institute

Pittsburgh, PA 15213-3890

TSP/PSP Results

Sponsored by the U.S. Department of Defense

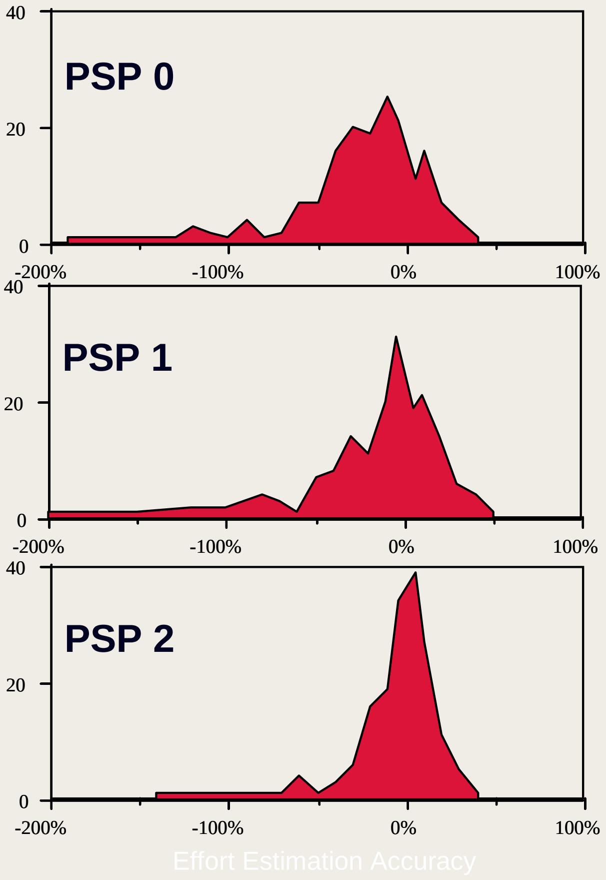
PSP Improves Performance

Estimation accuracy

- fewer underestimates
- more accurate estimates
- estimates balanced around zero

Quality

- yield improves by 2X to 3X
- fewer defects in unit test, integration test, system test
- COQ is flat or reduced

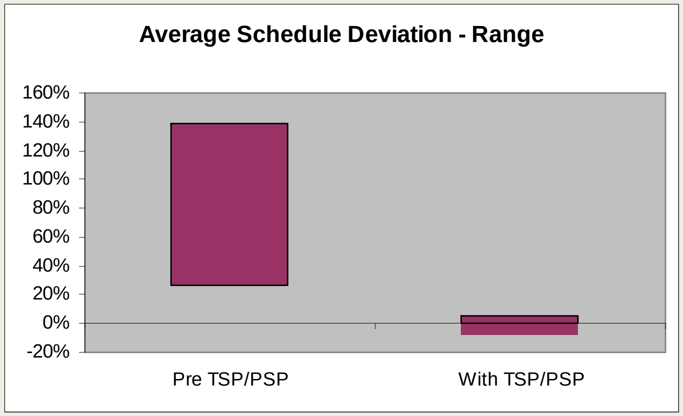


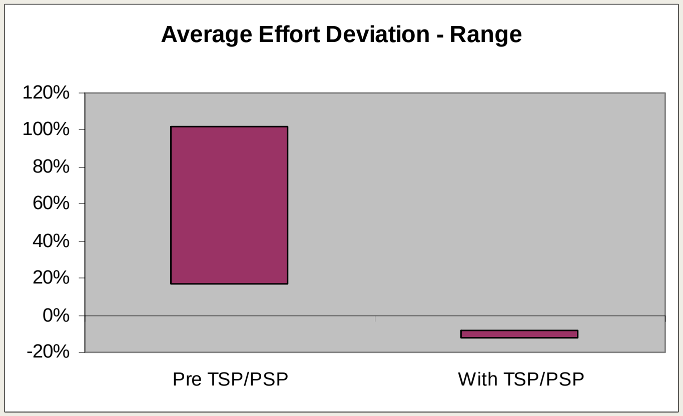
TSP Improves Predictability

Effort and schedule deviation are dramatically improved.

Schedule Performance	
Typical Industry	100%+
Study baseline	27% to 112%
TSP	< 10%

Effort/Cost Performance	
Typical Industry	100%+
Study baseline	17% to 85%
TSP	< 5%



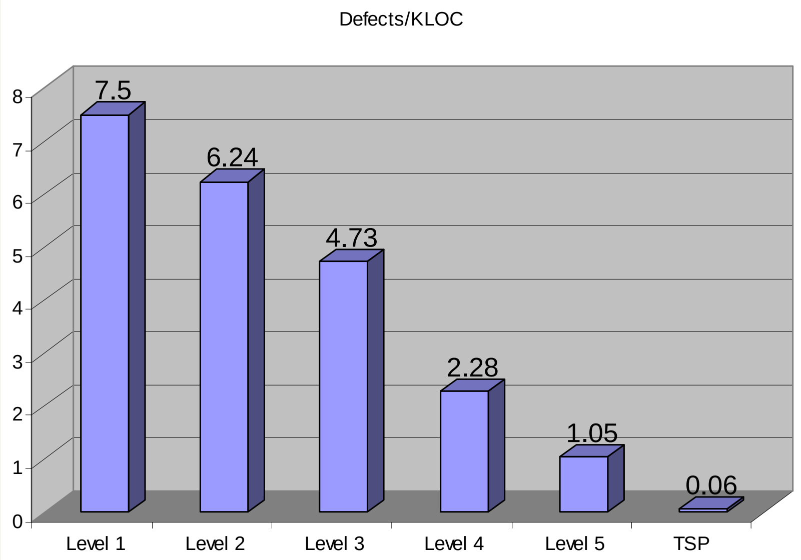


Source: CMU/SEI-2000-TR-015

TSP Improves Quality

An analysis of 20 projects in 13 organizations showed TSP teams averaged 0.06 defects per thousand lines of new or modified code.

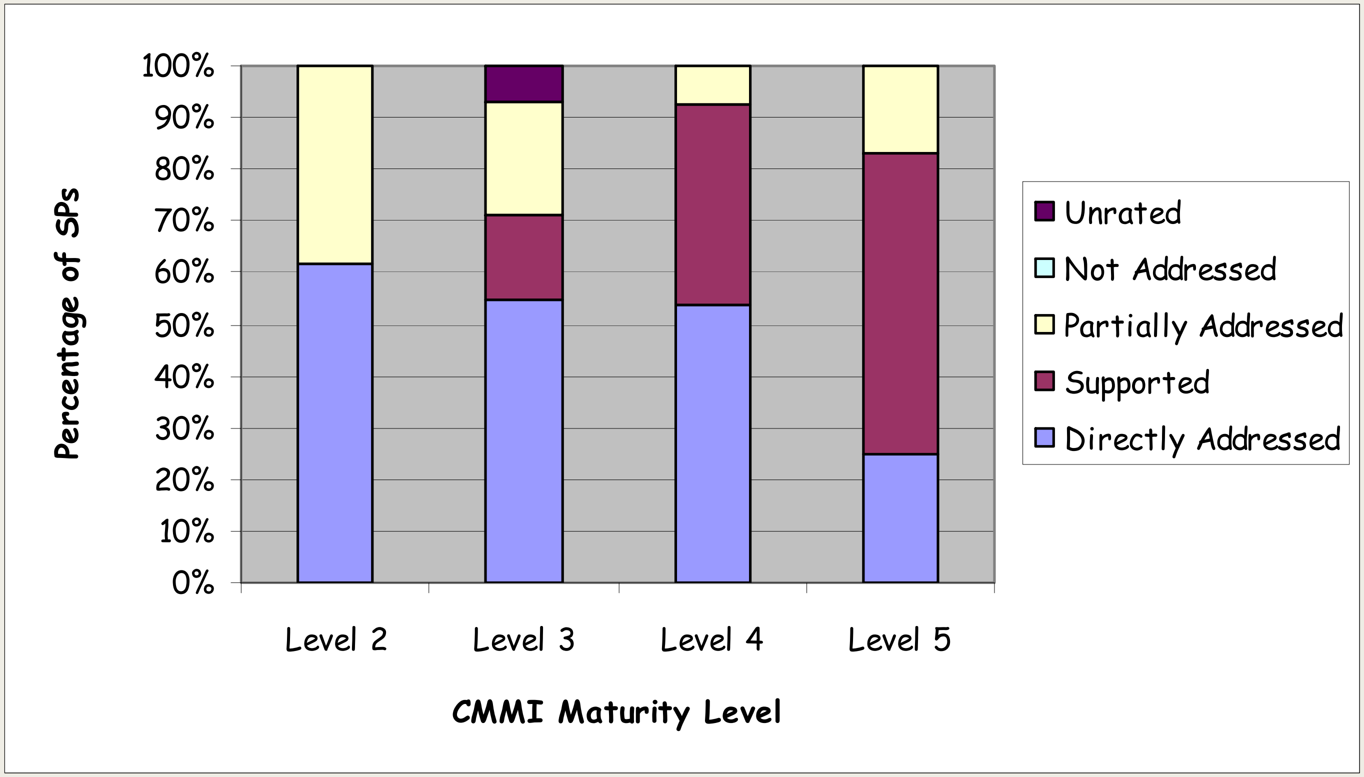
Approximately 1/3 of these projects were defect-free.



Source: CMU/SEI-2003-TR-014

Carnegie Mellon
Software Engineering Institute

TSP Accelerates CMMI Improvement



Carnegie Mellon
Software Engineering Institute

Pittsburgh, PA 15213-3890

Team Software Process

Sponsored by the U.S. Department of Defense

Working in Teams

Successful teams are both satisfying and rare.

Although many teams come close to meeting their product and business goals, they often do so at the expense of the team members.

We describe a team that works together smoothly and efficiently as being a “jelled team.”

“A jelled team is…greater than the sum of its parts… and the enjoyment people derive from the work is greater than you would expect.”

- Peopleware, DeMarco and Lister

Building Jelled Teams

Artificially-jelled teams can be built quickly through “team-building” exercises.

- retreats, seminars, workshops
- games (paintball, laser tag, etc.)
- contrived challenges (e.g., blind obstacle courses)

Team-building activities are unlikely to produce long-term team success unless they address real workplace issues.

- common understanding of roles, goals, products
- management support of team needs

The TSP is a proven-effective way of building jelled teams quickly.

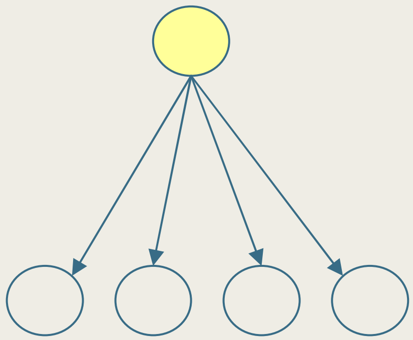
Accelerating Team-building

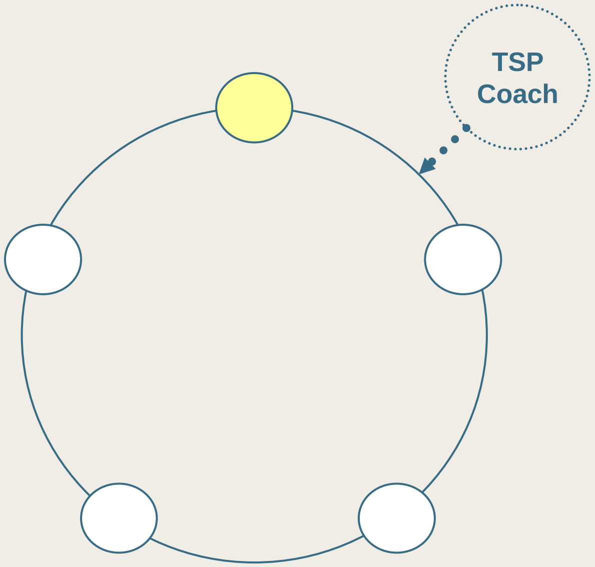
The TSP was developed to address the need for

- software engineering teams who can build quality products within cost and schedule constraints
- building teams quickly and reliably
- optimizing team performance throughout a project
- accelerating software process improvement
- making use of mature processes normal and expected

If you are familiar with the SEI Capability Maturity Model or the Capability Maturity Model Integration, think of TSP as a CMM/CMMI maturity level 5 process for teams.

TSP Teams are Self-directed





Traditional team

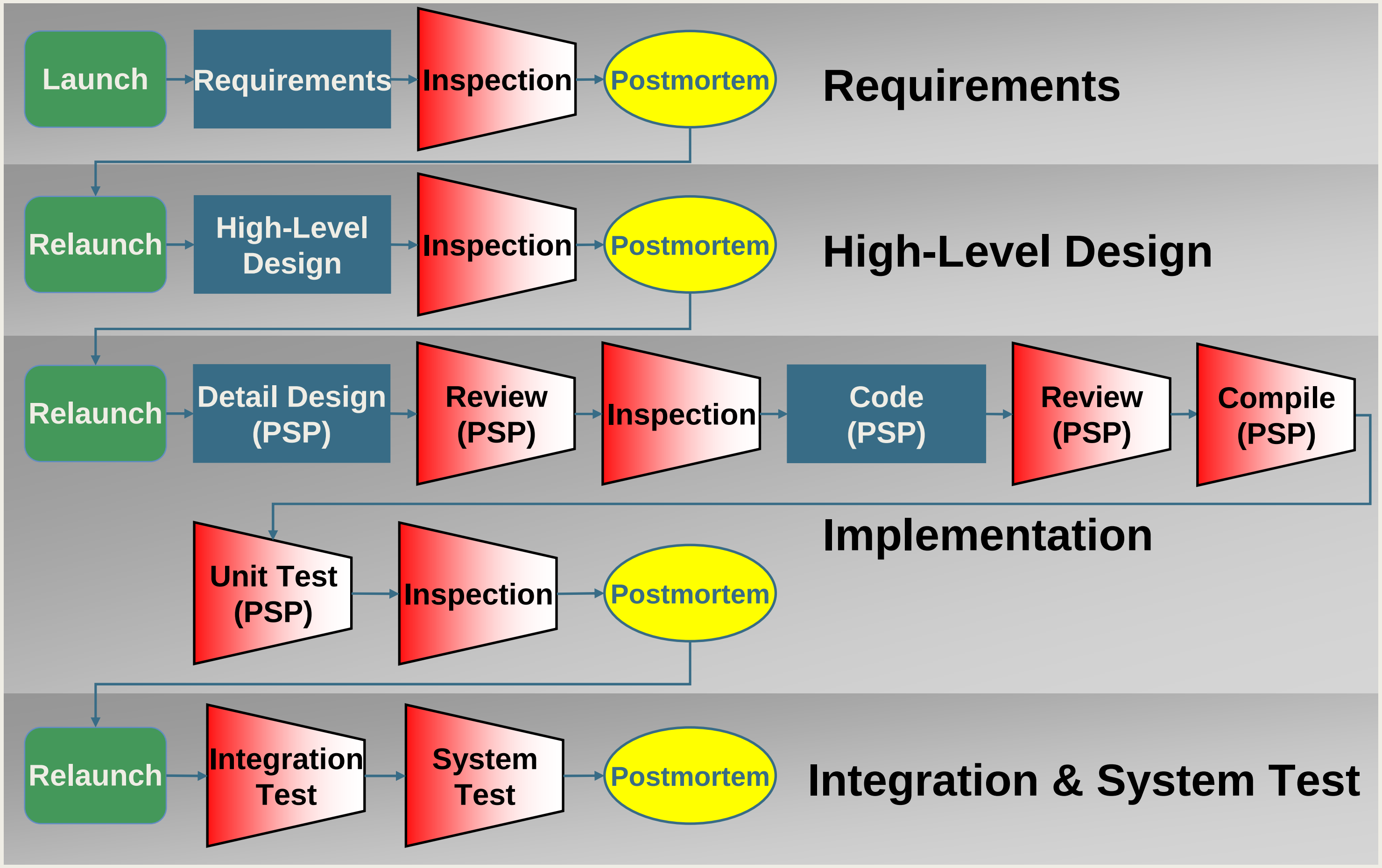
The leader plans, directs, and tracks the team’s work.

Self-directed team

The team members participate in planning, managing, and tracking their own work.

The TSP Process Elements

🟩 Planning
🟦 Development
🟥 Defect Filter
🟡 Postmortem



Launch → Requirements → Inspection → Postmortem
Requirements

Relaunch → High-Level Design → Inspection → Postmortem
High-Level Design

Relaunch → Detail Design (PSP) → Review (PSP) → Inspection → Code (PSP) → Review (PSP) → Compile (PSP)
→ Unit Test (PSP) → Inspection → Postmortem
Implementation

Relaunch → Integration Test → System Test → Postmortem
Integration & System Test

Summary

TSP gives teams a significantly greater measure of project ownership.

It makes them responsible for the outcome, but provides the tools to be successful.

The results demonstrate that this works.

- high-maturity performance
- predictable and improved cost and schedule
- near defect-free quality
- satisfied developers, managers, and customers

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

Contact SEI Customer Relations at 412-268-5800