Carnegie Mellon
Software Engineering Institute
Pittsburgh, PA 15213-3890

Basics of PSP and TSP for Systems Engineering

James McHale
Software Engineering Institute
November 2006

Sponsored by the U.S. Department of Defense

Agenda

- Why PSP and TSP for Systems Engineering?
- Things That Change, Things That Don't
- Time Logging Exercise
- The TSP Launch
- The TSP Management Framework
- TSP Quality Management

Team Software Process

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

TSP addresses common engineering and management issues (the same ones addressed by CMMI).

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

TSP

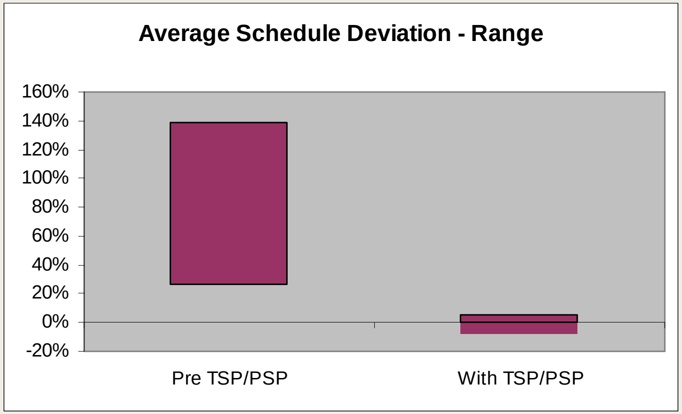
- truly empowers teams and team members
- is a complete, mature, “operational” process
- provides immediate and measurable results

Carnegie Mellon
Software Engineering Institute

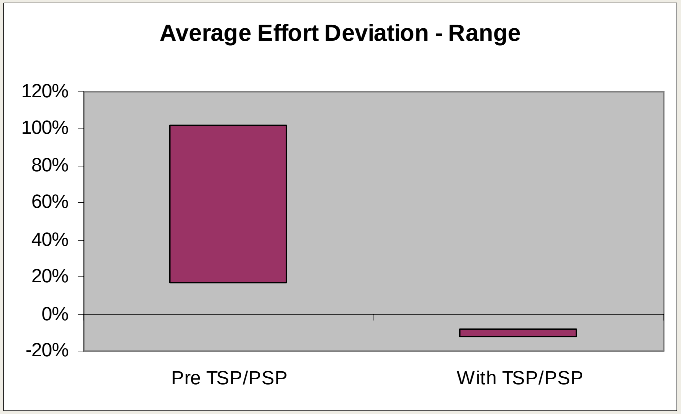
Improved Predictability

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

Improved Productivity

A nine person TSP team from the telecommunications industry developed 89,995 new LOC in 71 weeks, a 41% improvement in productivity.

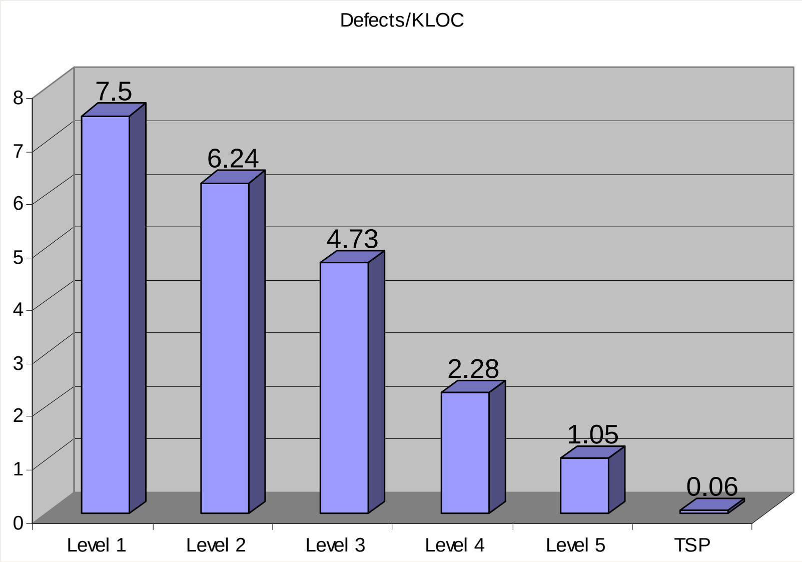
A TSP team from the commercial software industry, developing an annual update to a large “shrink-wrapped” software product, delivered 40% more functionality than initially planned.

A TSP team within the DoD, developing a new mission planning system, delivered 25% more functionality than initially planned.

Improved 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.



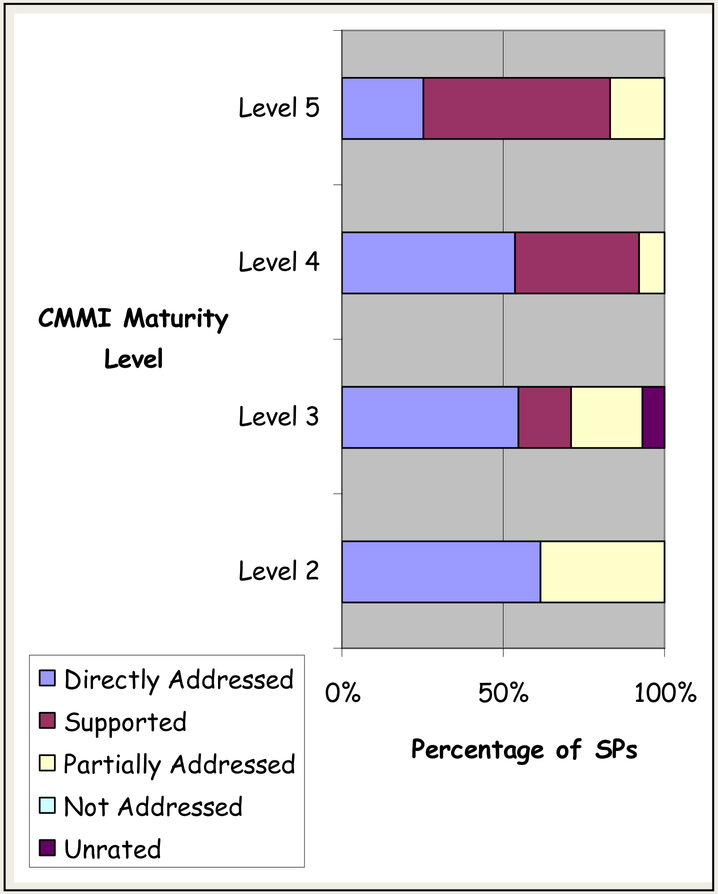
Source: CMU/SEI-2003-TR-014

Accelerated Process Improvement

TSP addresses or supports most of the capabilities expected of a project team through CMMI Level 5.

It provides either a "starting point" or a "next step".

Using TSP as a starting point, three organizations have advanced from ML1 to ML4 in less than 3 years.



TSP Results: NAVAIR AV-8B

Mar. 2000	Began current CMM-based improvement effort (now a CMMI-based effort)
Oct. 2000	Began PSP/TSP introduction sequence
Jan. 2001	First TSP team launched
May 2001	CBA-IPI: CMM level 2; 3 KPAs satisfied at level 3; level 4/5 observations on TSP
June 2001	Received draft of CMM-TSP gap analysis (levels 2 and 3 only, minus SSM and TP) to help guide improvement efforts
Feb. 2002	Received late-model gap analysis (including TP at level 3 and levels 4 and 5)
June 2002	Launched second TSP team
Sep. 2002	CBA-IPI: CMM level 4 (16 months from L2!)

See Crosstalk, Sep. 2002, "AV-8B's Experiences Using the TSP to Accelerate SW-CMM Adoption," Dr. Bill Hefley, Jeff Schwalb, and Lisa Pracchia, and Crosstalk, Jan. 2004, "The AV-8B Team Learns Synergy of EVM and TSP and Accelerates Software Process Improvement"

AV-8B CMMI "Quick Look" Profile

PA ->	RM	RD	TS	PI	VE	VAL	CM	PPQA	MA	CAR	DAR	OEI	OPD	OPF	OID	OT	OPP	PP	PMC	IPM	QPM	SAM	RSKM	IT
Specific Goal 1	U	FI	NR	S	S	S	S	S	U	U	NR	S	S	S	U	S	U	S	S	U	U	U	U	S
SP1.1	FI	FI	NR	FI	FI	FI	FI	FI	PI	PI	NR	FI	FI	FI	LI	FI	PI	FI	FI	FI	FI	FI	FI	FI
SP1.2	FI	FI	NR	FI	FI	FI	FI	FI	PI	PI	NR	FI	FI	FI	LI	FI	PI	FI	FI	FI	PI	FI	LI	FI
SP1.3	FI		FI	FI	FI	FI	FI		LI		NR	FI	FI	FI	FI	FI	FI	FI	FI	FI	PI	LI	FI	FI
SP1.4	PI								PI		NR		FI		FI	FI	LI	FI	FI	FI	FI			
SP1.5	FI										NR		FI				PI		FI	PI				
SP1.6											NR								FI					
SP1.7																			FI					
Specific Goal 2		S	NR	S	S	S	S	U	U	U		NR		U	U	U		S	S	NR	U	U	S	S
SP2.1		FI	FI	FI	FI	FI	FI	LI	LI	PI		PI		FI	FI	LI		FI	FI	NR	PI	NR	FI	FI
SP2.2		FI	FI	FI	FI	FI	FI	FI	LI	PI		NR		FI	FI	FI		FI	FI	FI	PI	LI	FI	FI
SP2.3		FI	FI		FI				PI	NR		PI		FI	LI	FI		FI	FI	FI	LI	FI		FI
SP2.4			NR						LI					LI				FI			LI	FI		FI
SP2.5																		FI						FI
SP2.6																		FI						
SP2.7																		FI						
SP2.8																								
Specific Goal 3		NR	S	S	S		S											S		S			S	
SP3.1		NR	FI	FI	FI		FI											FI		FI			FI	
SP3.2		FI	FI	FI	FI		FI											FI		FI			FI	
SP3.3		LI		FI														FI						
SP3.4		NR		FI																				
SP3.5		NR																						
Specific Goal 4																				S				
SP4.1																				FI				
SP4.2																				FI				
SP4.3																				FI				
PA ->	**RM**	**RD**	**TS**	**PI**	**VE**	**VAL**	**CM**	**PPQA**	**MA**	**CAR**	**DAR**	**OEI**	**OPD**	**OPF**	**OID**	**OT**	**OPP**	**PP**	**PMC**	**IPM**	**QPM**	**SAM**	**RSKM**	**IT**
Generic Goal 2	S						S	S	U									S	S			S		
Generic Goal 3	S	U	U	S	S	S	S	S	U	U	NR	NR	S	S	NR	S	U	S	S	S	S	S	S	S

LEGENDS

Practices

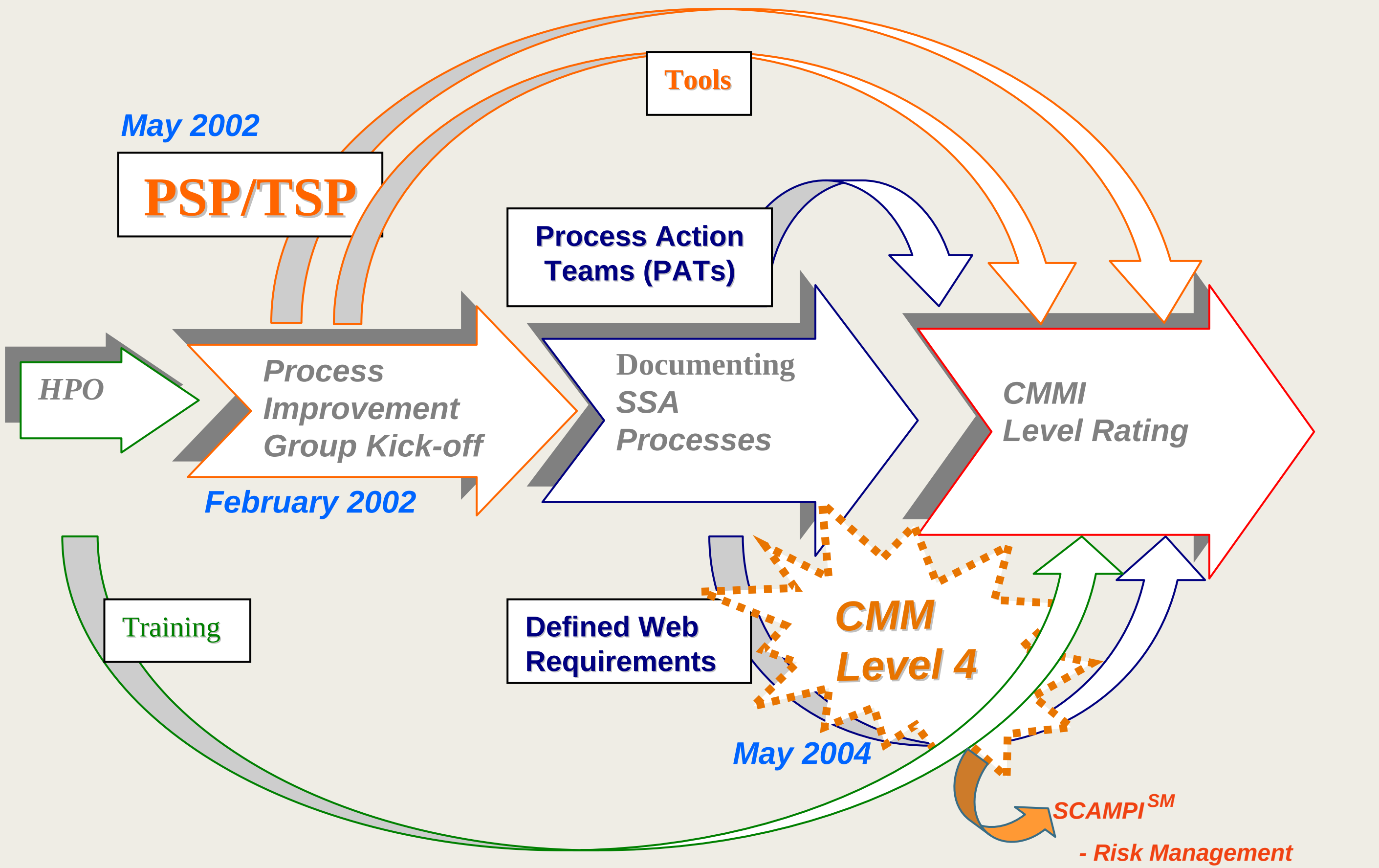
FI	Fully Implemented or Satisfied
LI	Largely Implemented
PI	Partially Implemented
NI	Not Implemented
NR	Not Rated

Goals

S	Satisfied
U	Unsatisfied (Goals)
NR	Not Rated

Source: NAVAIR

NAVAIR P3-C Journey



Source: NAVAIR

Improved Quality of Work Life

"A more disciplined process allowed me to do a better job, and allowed me to balance my job with other aspects of my life."

"This project ended up a lot less stressful than other projects."

"Promotes a less stressful environment. Can track that the project is on schedule. Fewer defects are seen positively in the organization."

"It is nice to be associated with a project that had few defects."

"I liked the level of detail that went into initial plan, and the constant awareness of the schedule. Allowed us to make adjustments as the project went on, instead of waiting for a major milestone."

"It was nice that management finally allowed the team to create the schedule."

Adoption

Organizations that are using, piloting, or preparing to pilot the TSP.

ABB
ABC Informatica
Activision
Advanced Information Services
Advanced Maturity Services, Inc.
Alan S. Koch Consultants
Ambient Consulting
AMRDEC
Boeing
Centre De Investigacion En Matamaticas
Census Bureau
CQG, Inc.
CRSIP / STSC / DRAPER
Davis Systems
DOE / Los Alamos
DOE / Naval Reactors
DPC Cirrus
Dynamics Research Corp.
EDS
Halex Associates
Heath Solutions, Inc.
Helsana
Honeywell
IBM
Intuit*
Iomega
I.Q. Inc.
KPMG
L. G. Electronics
Lockheed Martin / KAPL*
LogiCare
Los Alamos National Laboratory
M/A-Com Private Radio Systems, Inc
Magellan Navigation*
Microsoft*
Motiva
NASA Langley
NCR/Teradata
NCS Pearson
Northern Horizons
Northrop Grumman
Oracle*
Prodigia S.A. de C.V.
PS&J Consulting / Software Six Sigma
QuarkSoft
Respironics
Rockwell Collins
SAIC
Samsung SDS
Siberlink
STPP, Inc.
STSC
Trilogy
TYBRIN Corporation - Air Logistics
University of Alabama / Huntsville
University of Queensland
US Army / AMRDEC
US Navy / NAVAIR*
US Navy / NAVOCEANO*
US Navy / NAVSEA*
Xerox

***Organizations we are currently working with**

TSP for Systems Engineering

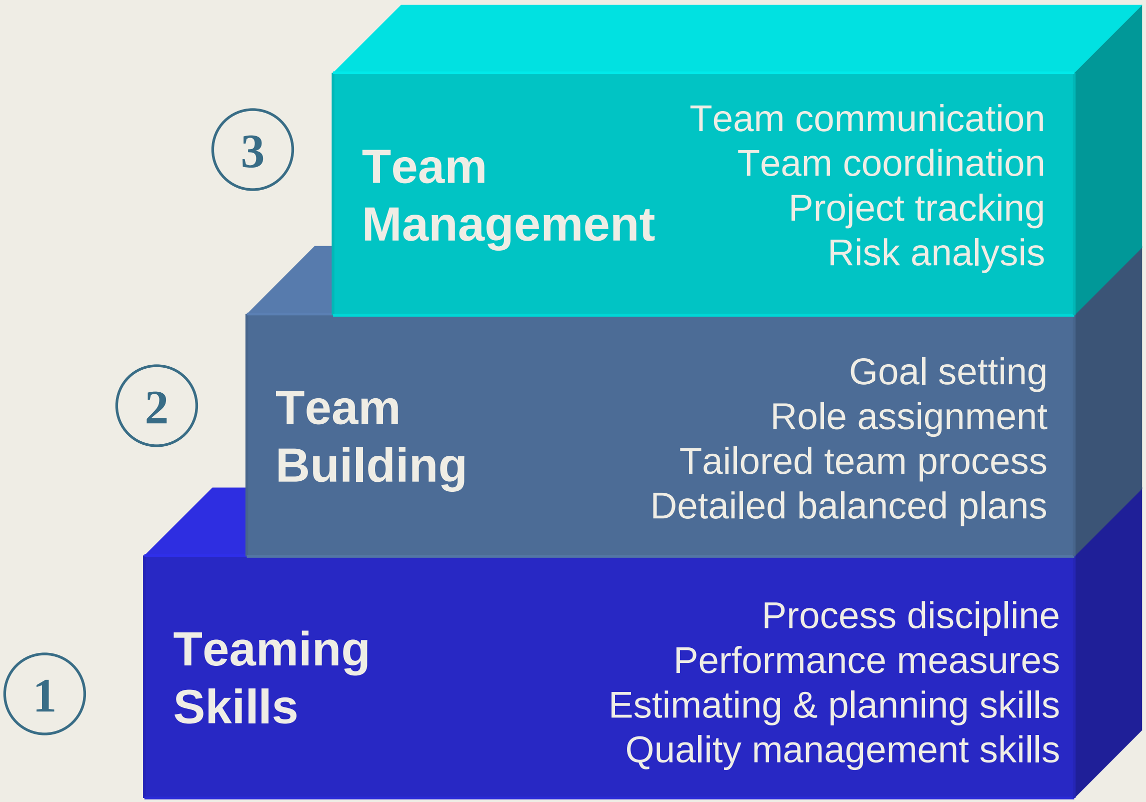
NAVAIR and other organizations have discussed the possibilities of adapting TSP for systems engineering use for several years.

Late in 2005, an effort was launched to extend TSP practice to systems engineers working in NAVAIR organizations, beginning with those that have had success using TSP for software development.

Several organizations, including at least one within NAVAIR, are forging ahead with their own TSP adaptations.

Building High-Performance Teams

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



Personal *Software* Process?

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

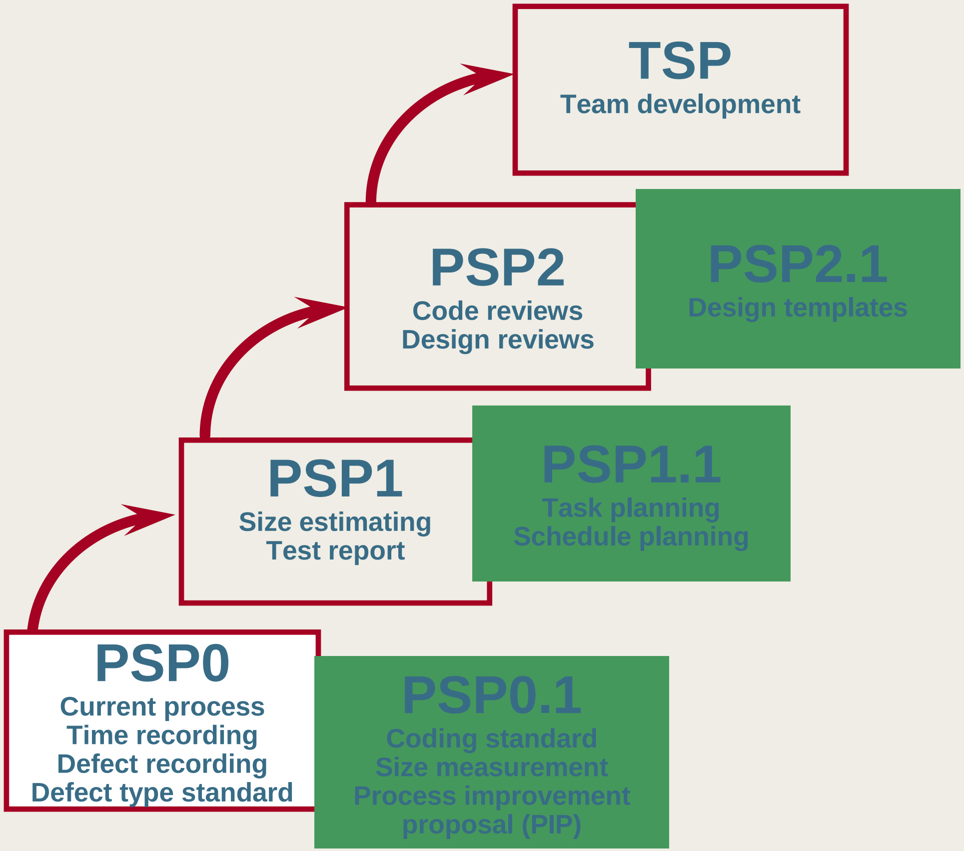
PSP builds the teaming skills required for the TSP.

With PSP, developers learn how to use a defined process and how to measure, estimate, plan, and track their work.

This leads to

- better estimating, planning, and tracking
- protection against over-commitment
- a personal commitment to quality
- personal involvement in process improvement

PSP-TSP Process Evolution



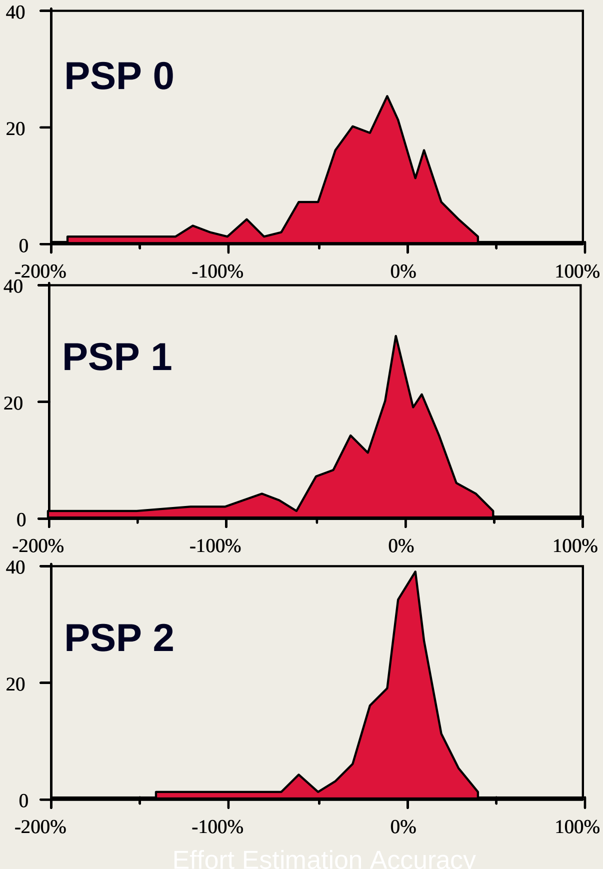
PSP Improves Performance

Estimation accuracy

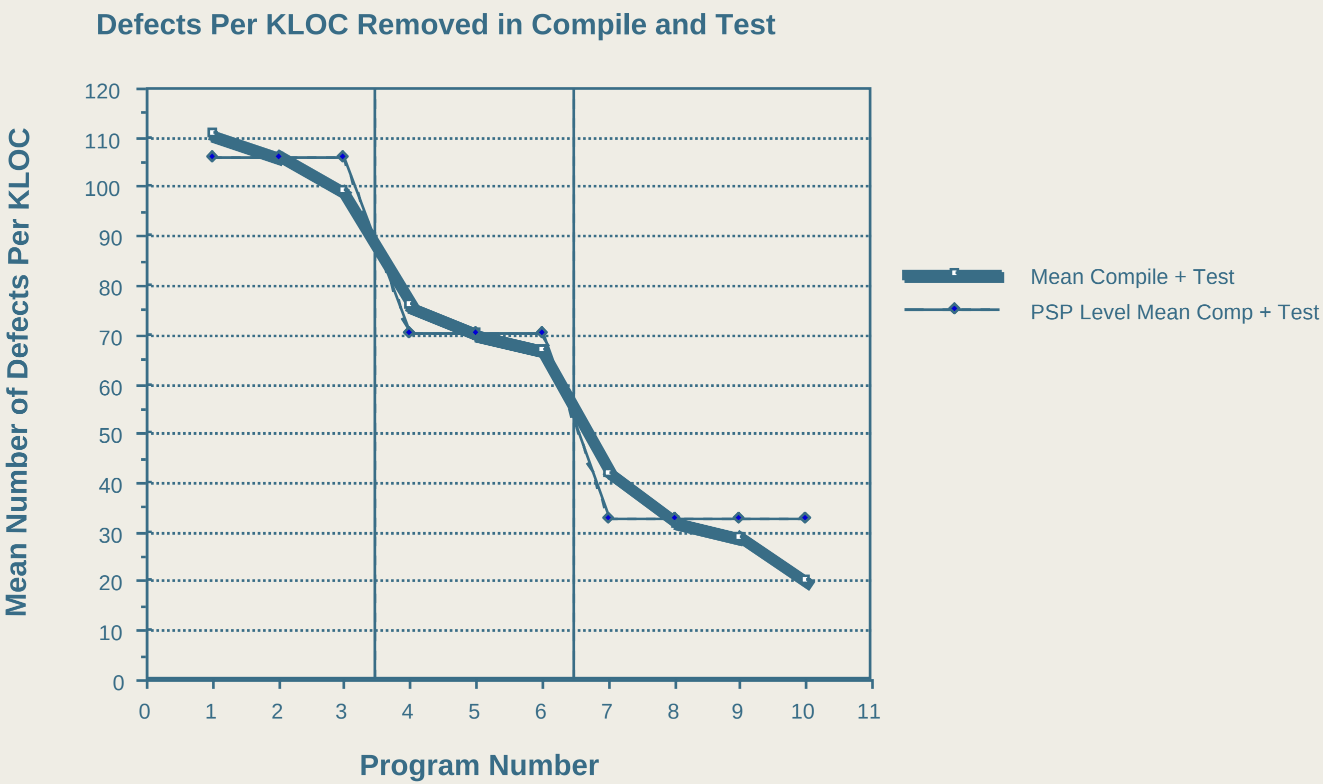
- fewer underestimates
- more accurate estimates
- estimates balanced around zero

Quality

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



PSP Quality Results



Agenda

Why PSP and TSP for Systems Engineering?

➡ Things That Change, Things That Don't

Time Logging Exercise

The TSP Launch

The TSP Management Framework

TSP Quality Management

Non-Software Disciplines

Many software-intensive projects have significant non-software components in terms of

- requirements and test
- support activities
- customer deliverables

The ways that these “other” activities are planned, staffed, and managed are reflected in organizational structure.

- separate departments for systems engineering, test, documentation, etc.
- often depends on the size of the organization and the size of the typical project
- multi-disciplinary teams
- matrixed project teams

Introduction to Personal Process

SEI teaches a two-day class, *Introduction to Personal Process*, which begins the individual quality journey by raising the issues of size measures and process and defect definitions for intellectual work other than software development.

It makes both economic and technical sense to extend the formal definitions of such work so that it may be planned and tracked with TSP methods.

NAVAIR has been a leader in adapting PSP and TSP to non-software work, and is actively engaged with SEI to formalize this work.

Process Improvement for "Others"

Applying TSP practices to other disciplines besides software engineering can be relatively straightforward.

- many teams are already doing it successfully
- based on CMM originally, which was based roughly on Crosby's five-level model of the manufacturing quality journey
- planning and tracking mechanisms are not software-specific
- *size and defect definitions (by default) are rooted in the software-specific examples from PSP training!*

In order to adapt PSP for use by other disciplines, size measures and defect definitions must be addressed.

Size Measures

For a size measure to be useful, it must be

- useful for planning
- precisely defined
- directly countable in an intermediate or final product

Defect Definitions

A defect is anything in an interim or finished product that must be changed for the product to be used as intended.

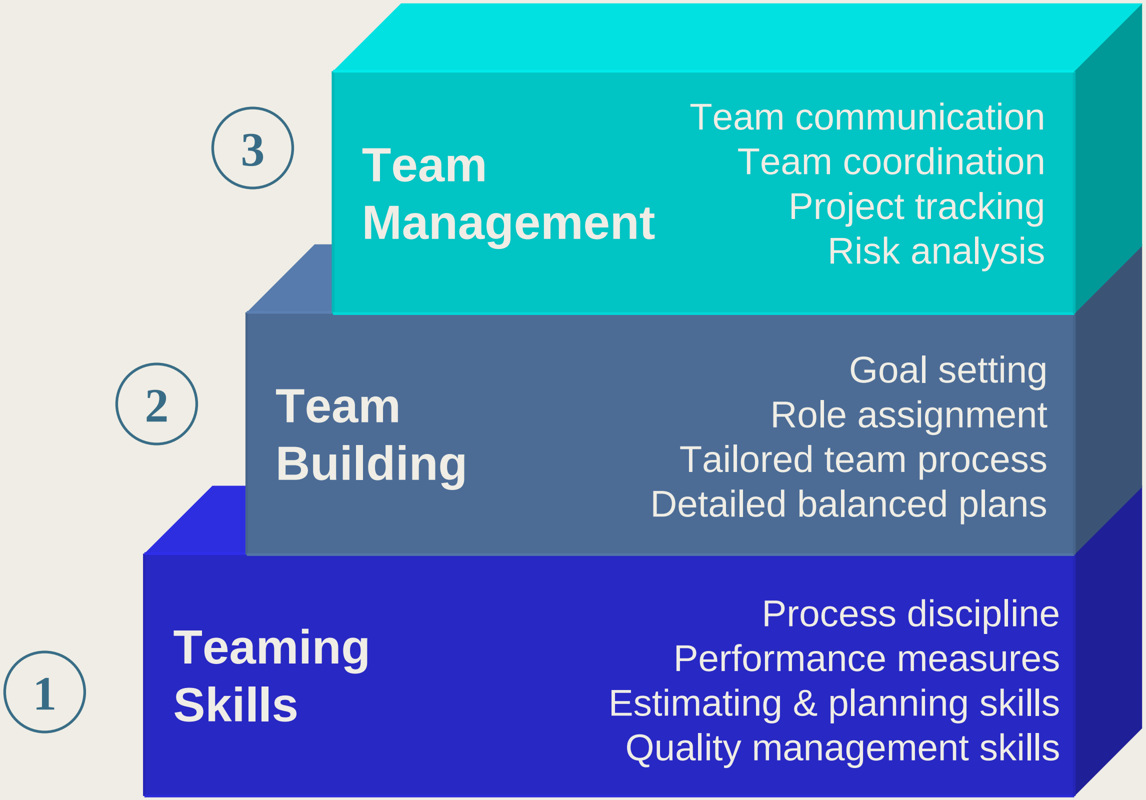
Defects in test procedures, requirements analyses, specifications, or user documentation can all adversely affect a customer's use of the delivered product.

Defect definitions must make sense to the people who must correct them.

Defect correction is sometimes called rework.

Building High-Performance Teams

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



Team Management Framework

The TSP team management framework helps the team meet their planned commitments by providing support for

- team communication and coordination
- project tracking and status reporting
- requirements management
- change management
- risk management

Team members gather data and manage their personal plans.

These data are consolidated at the team level and used by the team to manage the team’s plan.

TSP Base Measures

Size

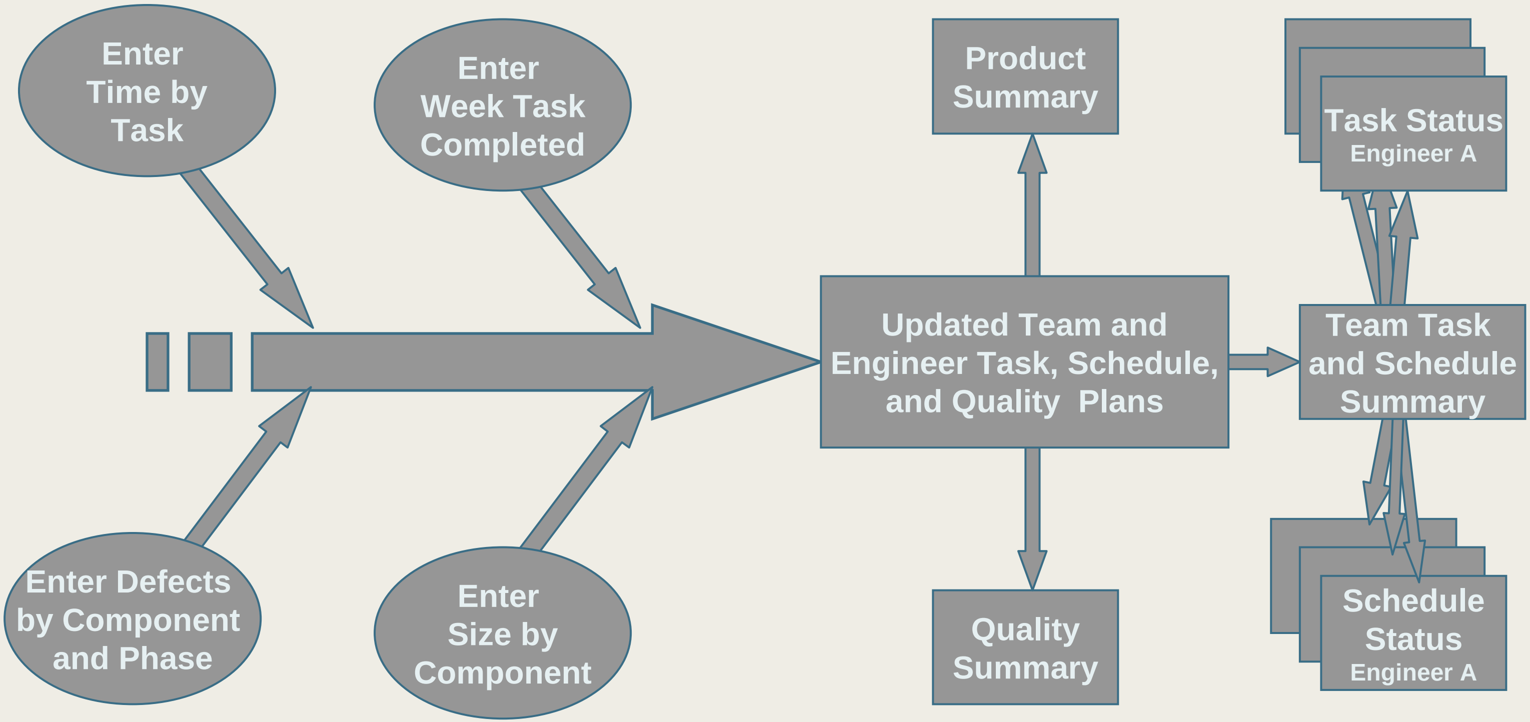
Effort

Quality

Schedule

Source: CMU/SEI-92-TR-019

TSP Project Tracking



Tracking with TSP Measures

The TSP base measures can be combined to provide a number of derived measures for managing projects.

TSP Derived Measures

Estimation accuracy (size/time)	Defect density
Prediction intervals (size/time)	Defect density by phase
Time in phase distribution	Defect removal rate by phase
Defect injection phase distribution	Defect removal leverage
Defect removal phase distribution	Review rates
Productivity	Process yield
%Reuse	Phase yield
%New Reusable	Failure cost of quality
Cost performance index	Appraisal cost of quality
Planned value	Appraisal/Failure COQ ratio
Earned value	Percent defect free
Predicted earned value	Defect removal profiles
	Quality profile
	Quality profile index

TSP Weekly Tracking

TSP teams track their status weekly using a defined process and the weekly status summary in the TSP support tool.

TSP Week Summary - Form WEEK

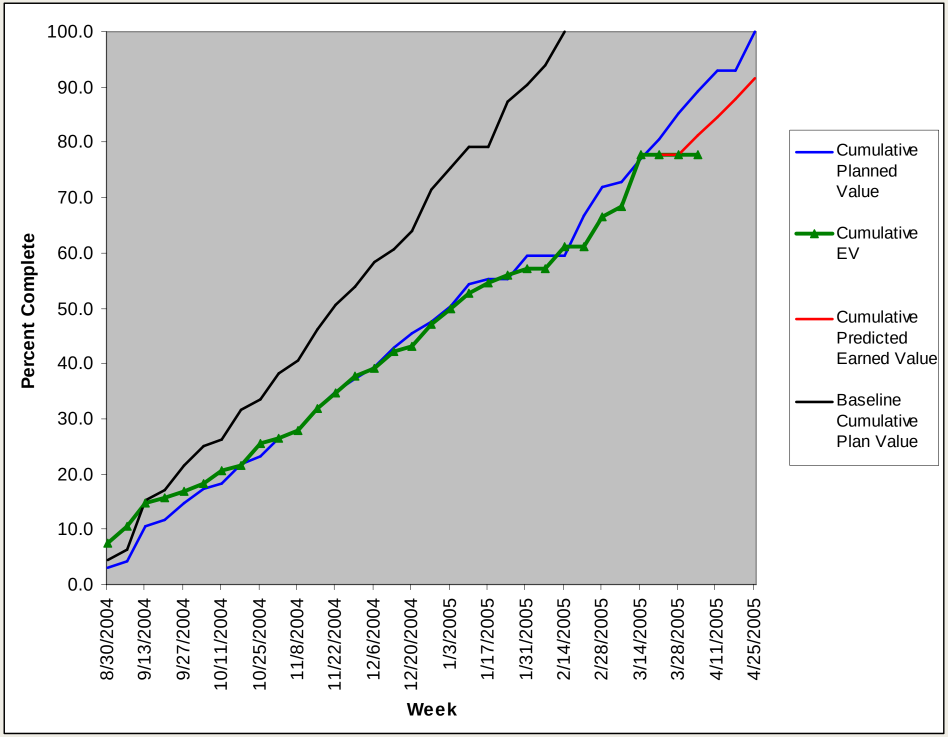
Name	Carol	Date	4/7/2003
Team	PSP Ghost		
Status for Week	15	Cycle	
Week Date	3/10/2003		

Weekly Data	Plan	Actual	Plan/ Actual
Schedule hours for this week	151.0	86.0	1.76
Schedule hours this cycle to date	1526.0	1594.8	0.96
Earned value for this week	6.9	4.2	1.64
Earned value this cycle to date	79.5	84.3	0.94
To-date hours for tasks completed	1580.7	1568.1	1.01
To-date average hours per week	101.7	106.3	0.96

Assembly	Phase	Tasks Completed or Due	Resource	Task Plan Hrs.	Task Actual Hrs.	Earned or Plan Value	Planned Week	Plan vs. Actual Hrs.
Main Form	CODEINSP	Main Form Code Inspection	SA	1.5	2.4	0.1	10	0.63
OEMMOO Delivery.aspx	UT	OEMMOO Delivery.aspx (FE-Server) U	NK	8.9	3.0	0.5	13	2.91
OEMMOO Delivery.aspx	DLDINSP	OEMMOO Delivery.aspx (FE-Client) D	NK	0.0	0.0	0.0	13	
OEMMOO Delivery.aspx	CODE	OEMMOO Delivery.aspx (FE-Client) C	NK	7.5	5.7	0.4	14	1.32
OEMMOO Delivery.aspx	CR	OEMMOO Delivery.aspx (FE-Client) C	NK	3.8	1.7	0.2	14	2.26
OEMMOO Delivery.aspx	COMPILE	OEMMOO Delivery.aspx (FE-Client) C	NK	1.3	0.9	0.1	14	1.44
OEMMOO Delivery.aspx	CODEINSP	OEMMOO Delivery.aspx (FE-Client) C	NK	0.0	0.0	0.0	14	
OEMMOO Delivery.aspx	UT	OEMMOO Delivery.aspx (FE-Client) U	NK	5.9	6.8	0.3	14	0.87
Query Object	TD	Query Object Test Development	MB	0.0	0.0	0.0	14	
Query Object	CODEINSP	Query Object Code Inspection	MB	0.0	1.2	0.0	14	0.00
Query Object	UT	Query Object Unit Test Dialogs	MB	1.1	1.7	0.1	14	0.66

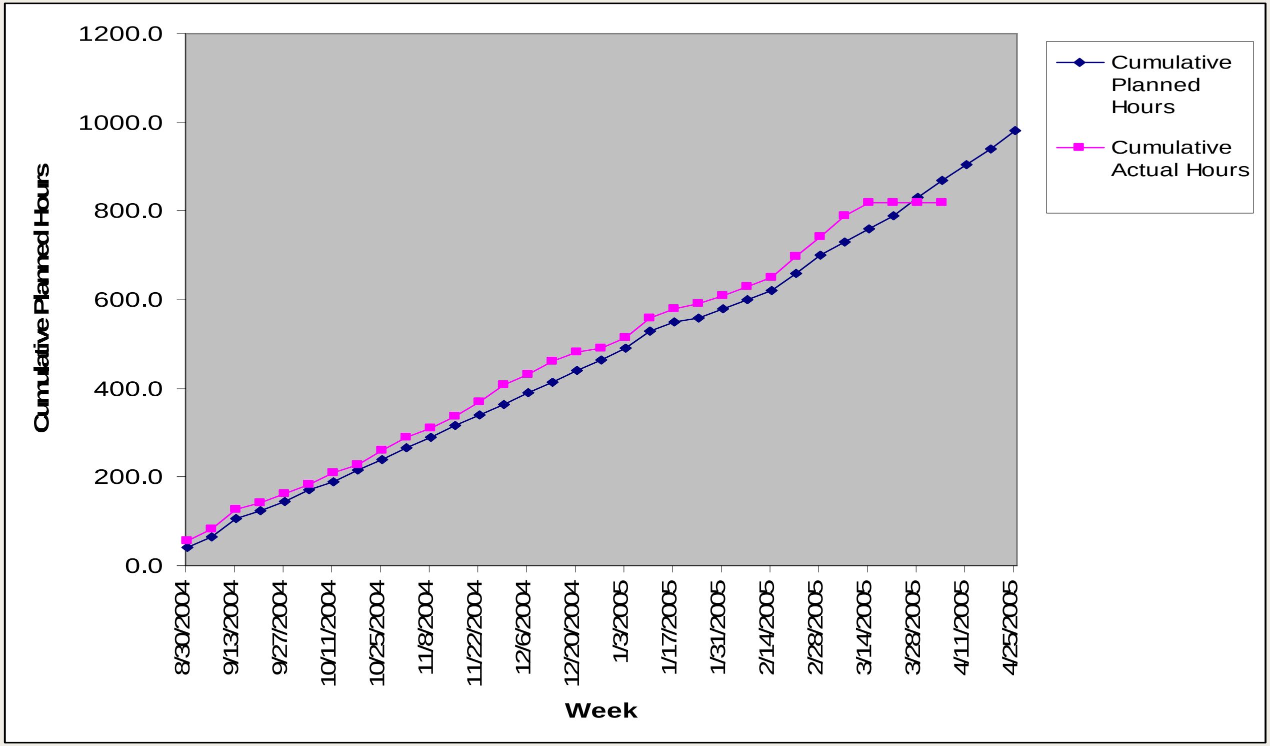
Earned Value Management

TSP teams review progress at the weekly meeting using earned value tracking provided by the TSP support tool.



Resource Management

TSP teams review resource utilization at the weekly meeting using analyses provided by the TSP support tool.

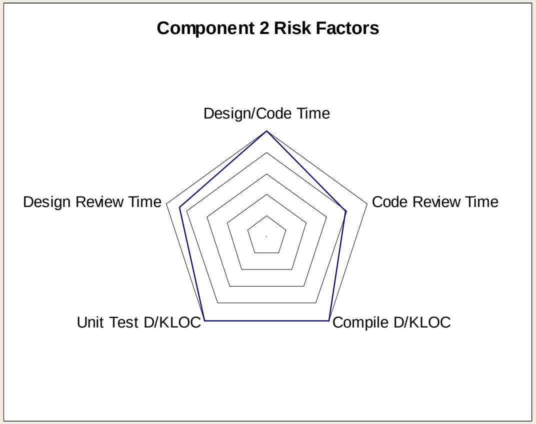


Quality Management

TSP teams use the Quality Profile as an early warning indicator of post-development defects.

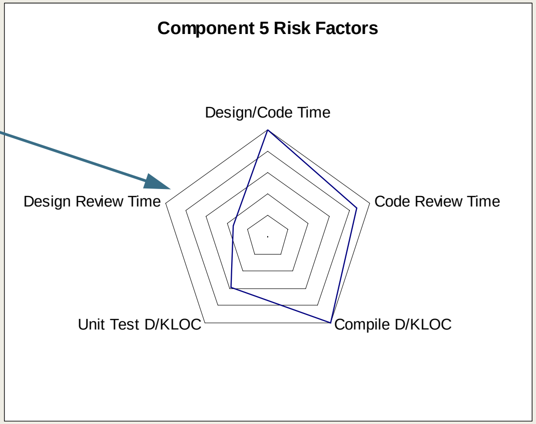
The quality profile uses five software quality benchmarks.

Satisfied criteria are plotted at the outside edge of the chart.



High quality component

Inadequate design review time results in design defects escaping to test and production.

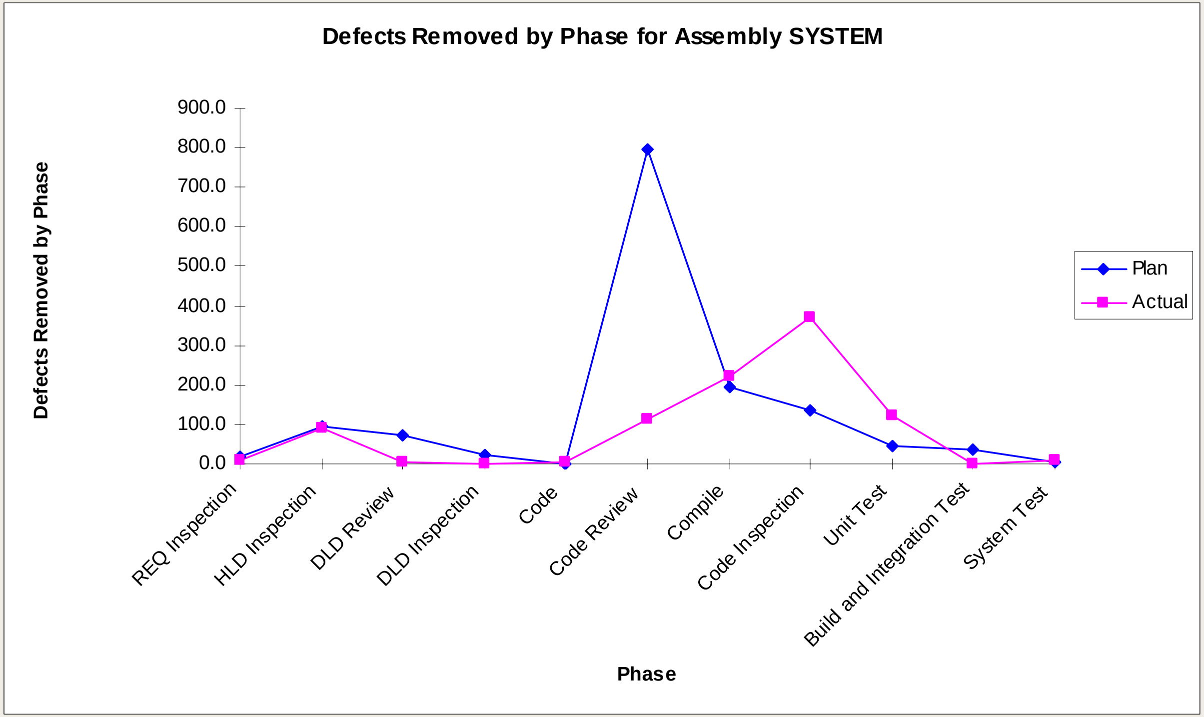


Poor quality component

Defect Removal Profile

TSP teams use the Defect Removal Profile to track

- plan and actual defects removed by phase
- early vs. late defect removal plan



Agenda

- Why PSP and TSP for Systems Engineering?
- Things That Change, Things That Don't
- ➡ Time Logging Exercise
- The TSP Launch
- The TSP Management Framework
- TSP Quality Management

Exercise Objectives

The PSP is the foundation for the TSP.

This exercise provides

- an understanding of the baseline process, PSP0
- familiarity with the basic measurement forms used in the PSP

Similar measures and forms are used in the TSP.

Basic Process Elements

A process script and basic measures

A project plan summary form

A time recording log

A defect reporting log

A defect type standard

Basic Process Measures -1

The reason to measure a process is to understand it.

- how much time is spent in various activities
- what is produced at various times
- how many defects are injected and removed, and when

With these data, engineers can better

- plan and estimate the work to be done
- evaluate the results
- improve the process for the next project

Basic Process Measures -2

To measure the process, the work is divided into defined activities called *phases*.

Each phase consists of

- the task to be done during the phase
- the entry criteria, or the items required before the work can start
- the exit criteria, or the items that must be produced by the end of the phase
- verification steps to ensure that the work is properly done

Basic Process Measures -3

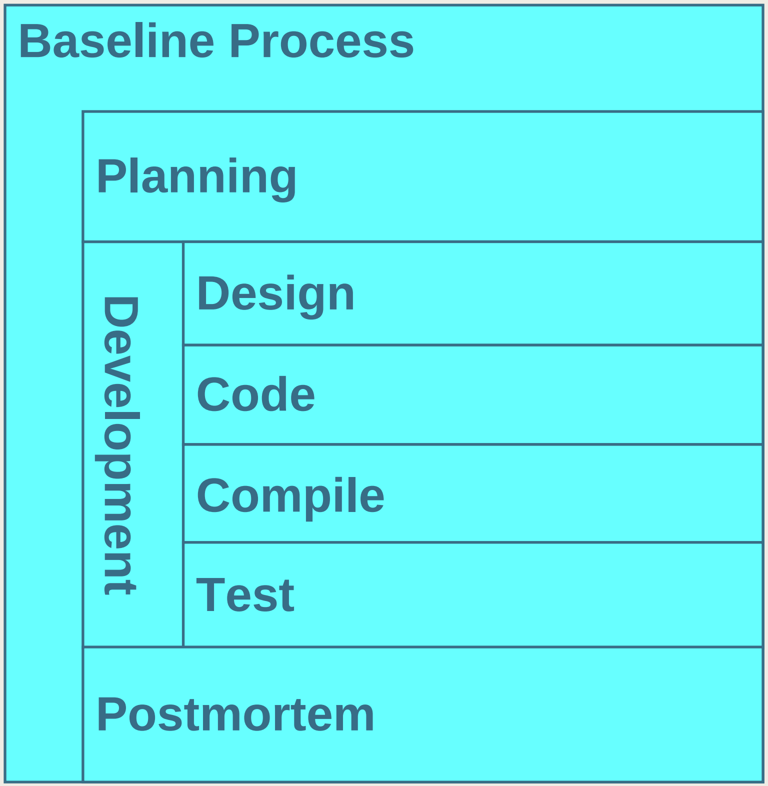
The measures for each phase are

- time spent (in minutes) in that phase
- defects injected in that phase
- defects removed in that phase

The program size is also measured, but only during the postmortem phase at the end of the project.

These measures provide the foundation for all PSP measurements, analyses, and planning.

Baseline Process Phases



A Process Script

PSP0 Process Script

Phase Number	Purpose	To guide you in developing module-level programs
	Entry Criteria	• Problem description • PSP0 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 Recording 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 form with estimated and actual data • Completed Defect and Time Recording Logs

PSP0 Project Plan Summary

The project plan summary holds project data in summary form.

- planned and actual data
- to date history
- time in phase
- defects injected
- defects removed

PSP0 Project Plan Summary

Student		Date	
Program		Program #	
Instructor		Language	

Time in Phase (min.)	**Plan**	**Actual**	**To Date**	**To Date %**
Planning				
Design				
Code				
Compile				
Test				
Postmortem				
Total				

Defects Injected	**Actual**	**To Date**	**To Date %**
Planning			
Design			
Code			
Compile			
Test			
Total Development			

Defects Removed	**Actual**	**To Date**	**To Date %**
Planning			
Design			
Code			
Compile			
Test			
Total Development			
After Development			

Time Recording Log

Engineers use the time recording log to record

- the time when they start on a project phase
- the time when they stop work on a phase
- the interruption time
- the elapsed time less interruption time
- comments

Defect Recording Log

Engineers use the defect recording log to record information about all defects found in reviews, compiling, and test.

- the defect number
- the defect type
- the phase in which it was injected
- the phase in which it was removed
- the time to find and fix the defect
- a brief description of the defect

If the defect was injected while fixing a defect, that defect's number is recorded.

Exercise Instructions -1

Read through the PSP0 process scripts (in the workbook) so that you understand the entry and exit criteria for each phase.

Read JD's scenario for program 1A and fill out the time log. The defect log and project plan summary are already filled out for you.

Refer to the instructions for each form to determine what information goes in each field.

Exercise Instructions -2

When did JD start?

When did he finish?

Was he interrupted?

What process phase is this?

Where should this information be recorded?

JD begins work on assignment 1A [8:00] **by reviewing the requirements in the assignment package, including the test requirements, to be sure he understands them. He copies the requirements to his note pad. Then, based on the data presented on past student performance and JD's feeling about his own performance, he estimates this assignment will take 3 hours and writes this on his note pad** [8:06]**.**

Results

How long did the project take? __________

How many defects were removed? __________

In what phase did JD spend the most time? __________

What percent of JD’s time was spent in compile + test? __________

Exercise Summary

The baseline personal process is simple and easy to use.

The PSP forms simplify data collection and provide a convenient reference for planning future projects.

The basic PSP time, size, and defect measures provide the data for the TSP.

HOMEWORK: For systems engineering in your organization, how would the Plan Summary change? What phases of development would you define?

Agenda

Why PSP and TSP for Systems Engineering?

Things That Change, Things That Don't

Time Logging Exercise

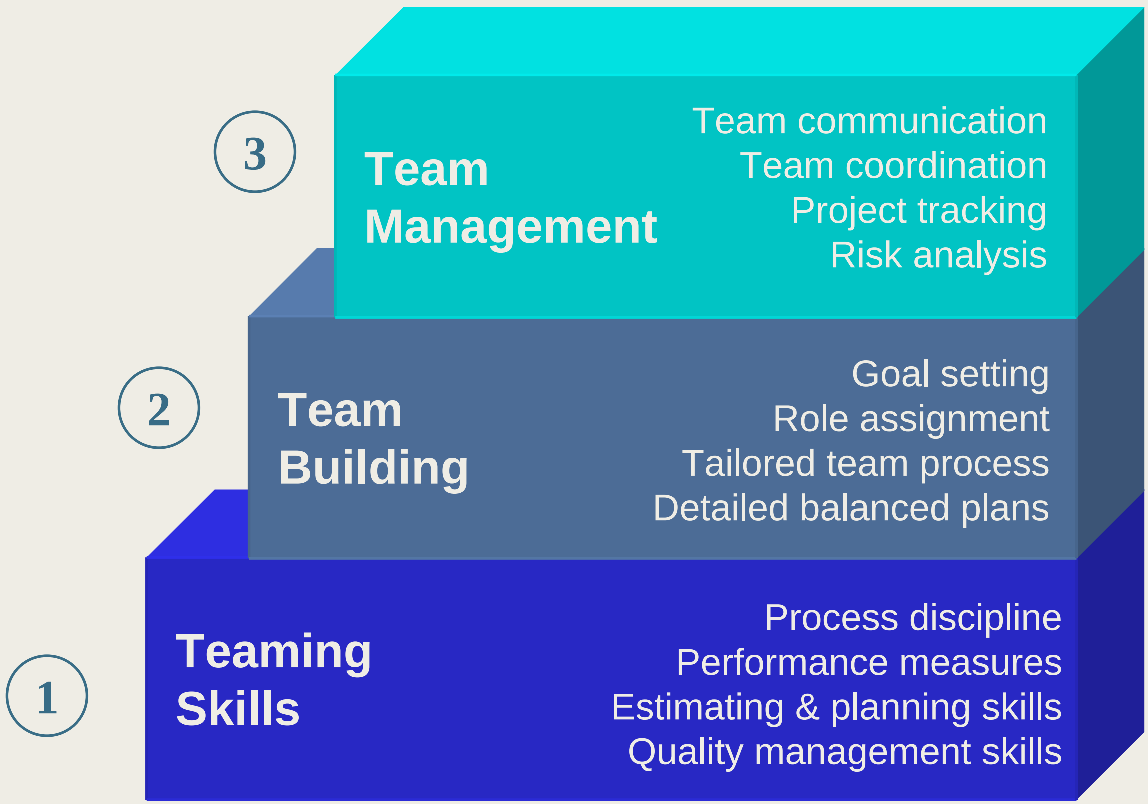
➡ The TSP Launch

The TSP Management Framework

TSP Quality Management

Building High-Performance Teams

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



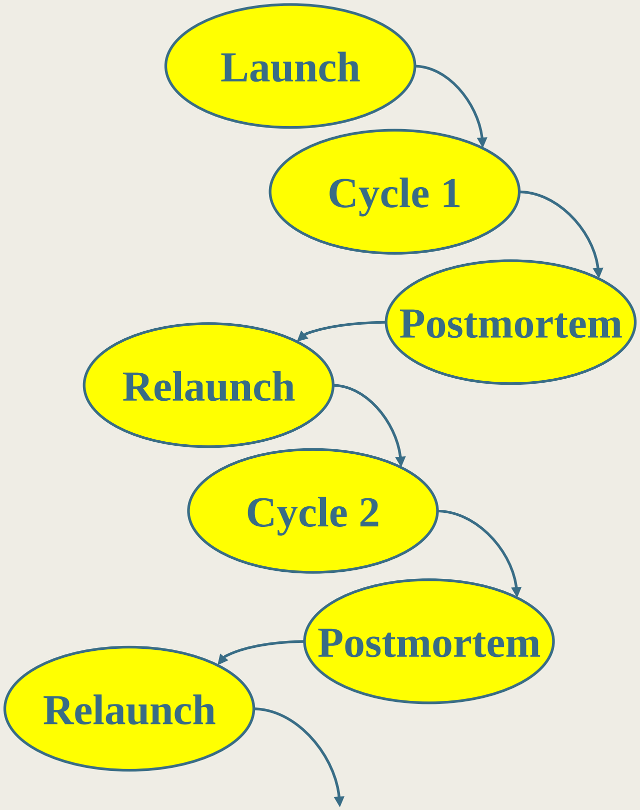
TSP Structure and Flow -1

In the TSP, each major project cycle or phase begins with a Launch.

The Launch is a defined team planning process that also facilitates team-building.

The team reaches a common understanding of the work and the approach.

They produce a detailed plan to guide the next development phase or cycle.



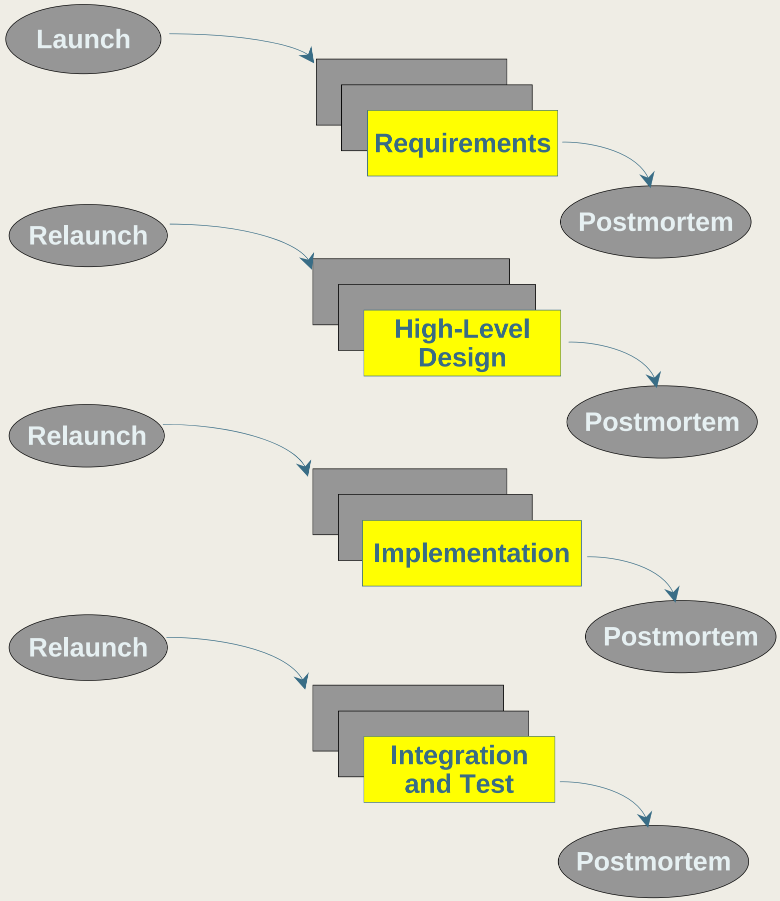
TSP Structure and Flow -2

TSP has four principal development phases.

- Requirements, High-Level Design, Implementation, Test (TSP default)
- *or a project-defined lifecycle*

TSP projects can start or end on any phase.

- from requirements through system test
- requirements only
- high-level design only
- *as needed to do the work*

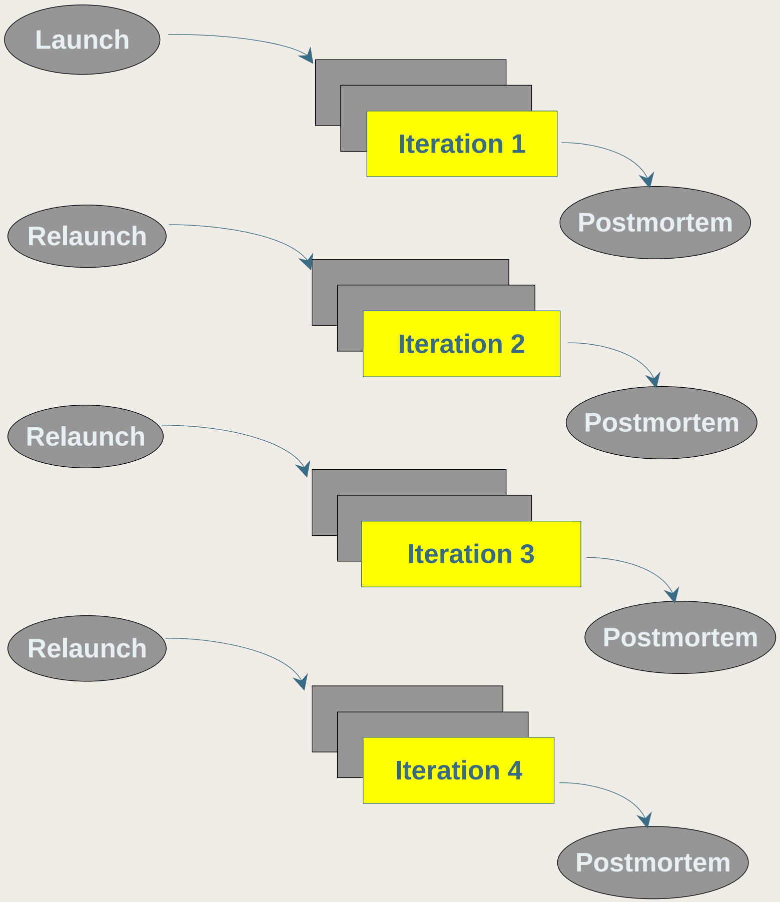


TSP Structure and Flow -3

The TSP phases can and should overlap.

The TSP development strategy encourages
- incremental development
- iterative development
- multiple builds or cycles
- work-ahead

TSP permits whatever process structure makes the most business and technical sense to the team.



TSP Process Elements

Checklists, specifications, standards, and other process assets (22), including

- TSP introduction sequence
- Launch planning guidance
- Executive tools such as checklists for planning assessment and quarterly reviews

Forms (22), including

- Time Recording Log
- Defect Recording Log
- Inspection Report
- Process Inventory
- Quality Summary

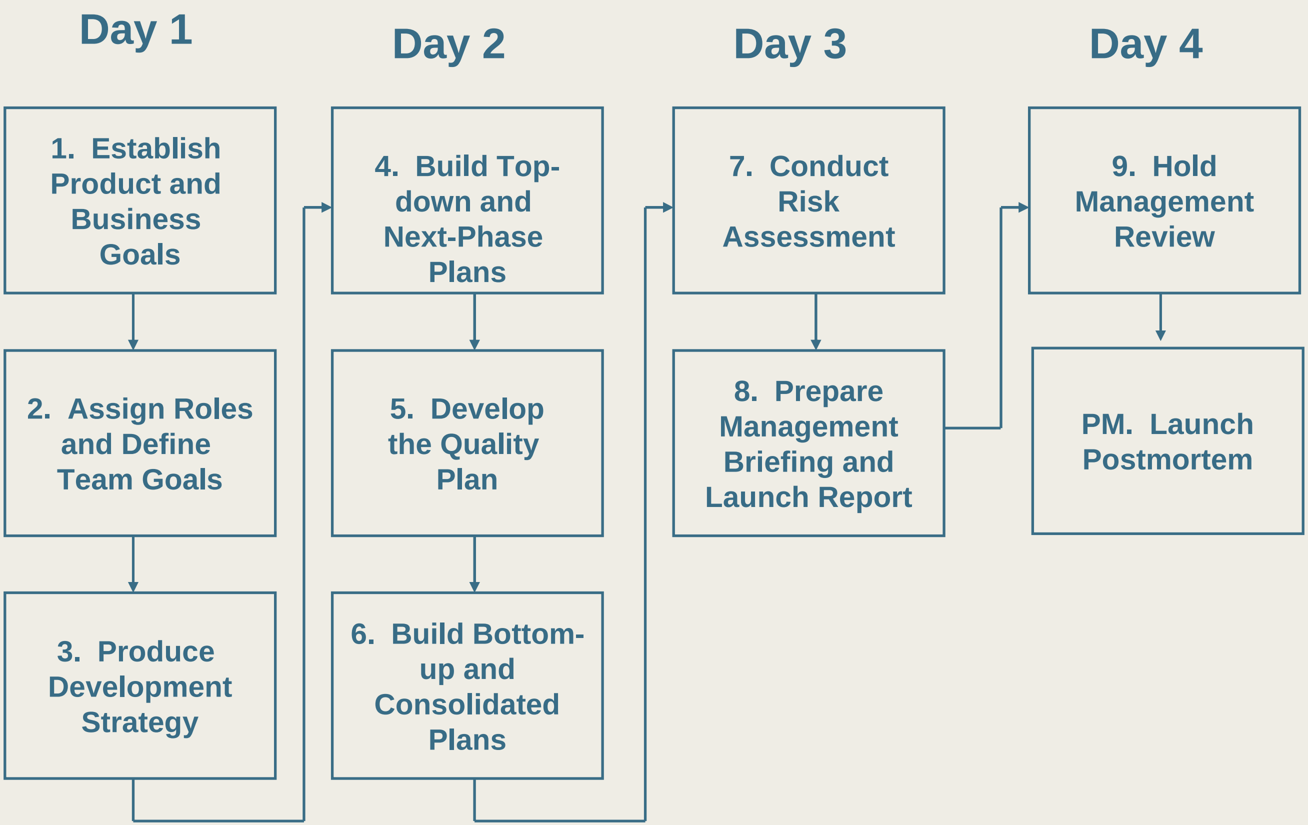
TSP role specifications (12), including

- Meeting roles and responsibilities
- Inspection roles and responsibilities
- Customer interface manager role and responsibilities
- Process manager role and responsibilities

Process Scripts (30), including

- Overall development and enhancement process
- Overall maintenance and enhancement process
- Launch process
- Test defect handling

The Launch Process Meetings



The TSP Launch Artifacts

Business needs
Management goals
Product requirements

What?	How?	When?	Who?	How well?	What if?
Team goals	Team strategy	Task hour plan	Team roles	Quality plan	Risk evaluation
Conceptual design	Team process	Schedule plan	Task plans		Alternative plans
Planned products		Earned-value plan	Detailed plans		
Size estimates					

TSP Project Tracking -1

Project tracking in the TSP is based on the principles and measures used in the PSP.

The detailed team and individual plans facilitate precise project tracking.

Each team member is responsible for

- gathering data on their work
- tracking status against their personal plan
- keeping the team informed
- the quality of the work they produce

TSP Weekly Meeting

Manager's report (team leader)
- new issues and developments

Role reports (8, more or less)
- customer/requirements, design, implementation, test, planning, process, quality, support

Risk report
- status and changes in assigned risks
- impending flag dates and required actions

Project status
- individual and team (planning manager)

Next week's plans
- individual tasks
- dependencies (e.g. reviews needed)
- task, hour, EV goals

Agenda

Why PSP and TSP for Systems Engineering?
Things That Change, Things That Don't
Time Logging Exercise
The TSP Launch
➡ The TSP Management Framework
TSP Quality Management

TSP Project Tracking -2

Project tracking in TSP is based on

- the team's plan
- task hour and task completion data
- plan and earned value

Individual plans facilitate precise project tracking.

Team members are each responsible for

- gathering data on their work
- tracking status against their personal plans
- the quality of the work that they produce
- keeping the team informed of their progress

Individual team member data are consolidated each week so that the team can assess progress against goals.

The WEEK Summary

The weekly team meeting is the forum that the team uses to

- track progress against the plan
- track the status on the project's issues and risks
- communicate with each other

TSP Week Summary - Form WEEK

Name	Consolidated Team Plan	**Date**	2/7/2000
Team	Security System Upgrade		
Status for Week	5	**Cycle**	
Week Date	1/31/2000		

Weekly Data	Plan	Actual	Plan/ Actual
Project hours for this week	80.0	69.0	1.16
Project hours this cycle to date	400.0	344.8	1.16
Earned value for this week	10.3	3.1	3.37
Earned value this cycle to date	40.2	30.0	1.34
To-date hours for tasks completed	293.0	303.8	0.96

Assembly	Phase	Tasks Completed	Resource	Plan Hours	Actual Hours	Earned Value	Planned Week	Plan Hrs./ Actual Hrs.
SYSTEM	REQ	Write SRS general sections	tmc	14.0	12.0	1.4	4	1.17
SYSTEM	REQ	Weekly requirements analysis meeting 5	tma	4.0	4.0	0.4	5	1.00
SYSTEM	REQ	Weekly requirements analysis meeting 5	tmb	4.0	4.0	0.4	5	1.00
SYSTEM	REQ	Weekly requirements analysis meeting 5	tmc	4.0	4.0	0.4	5	1.00
SYSTEM	REQ	Weekly requirements analysis meeting 5	tmd	4.0	4.0	0.4	5	1.00
		TASKS DUE THROUGH WEEK 7						
SYSTEM	REQ	Review SRS general sections	tmc	5.0		0.0	4	
SYSTEM	STP	Complete Validation Test Plan	tmd	8.0	8.5	0.0	4	0.94

Maintaining the Team's Schedule

The team manages its commitments by using the data it collects.

The team determines how it is doing against its plan.

If the team is falling behind, it determines

- what is the likely cause
- what the team can do to maintain its commitment

The team informs management if the commitment cannot be maintained or if management help is needed.

Determining Status Against Plan -1

Two things are important here.

- the team's current project status
- the team's projected completion date

Current status is determined using data on the WEEK form.

TSP Week Summary - Form WEEK

Name	Consolidated Team Plan
Team	Security System Upgrade
Status for Week	5
Week Date	1/31/2000
Date	2/7/2000
Cycle	

Weekly Data	Plan	Actual	Plan/ Actual
Project hours for this week	80.0	69.0	1.16
Project hours this cycle to date	400.0	344.8	1.16
Earned value for this week	10.3	3.1	3.37
Earned value this cycle to date	40.2	30.0	1.34
To-date hours for tasks completed	293.0	303.8	0.96

$$weeks\,behind = \frac{(plan\ EV\ todate - actual\ EV\ todate)}{(actual\ EV\ todate / current\ week)}$$

Determining Status Against Plan -2

Projected completion date can be determined using data on the WEEK form and the original planned weeks.

TSP Week Summary - Form WEEK

Name: Consolidated Team Plan
Team: Millenium Upgrade
Status for Week: 5
Week Date: 1/31/2000
Date: 2/7/2000
Cycle:

Weekly Data	Plan	Actual	Plan/ Actual
Project hours for this week	80.0	69.0	1.16
Project hours this cycle to date	400.0	344.8	1.16
Earned value for this week	10.3	3.1	3.37
Earned value this cycle to date	40.2	30.0	1.34
To-date hours for tasks completed	293.0	303.8	0.96

$$weeks\,to\,go = \frac{(100 - actual\,EV\,todate)}{(actual\,EV\,todate)/(current\,week)}$$

$$\left\{\begin{matrix} weeks\,behind \\ at\,completion \end{matrix}\right\} = (weeks\,to\,go + current\,week) - \left\{\begin{matrix} original \\ planned\,weeks \end{matrix}\right\}$$

Identifying Estimating Problems

The cost performance index (CPI) shows how the team is performing with respect to the effort estimates in the plan.

$$CPI = \frac{\textit{plan hours for completed tasks}}{\textit{actual hours for completed tasks}}$$

The CPI is available on the WEEK form.

TSP Week Summary - Form WEEK

Name Consolidated Team Plan
Team Security System Upgrade
Status for Week 5
Week Date 1/31/2000
Date 2/7/2000
Cycle

Weekly Data	**Plan**	**Actual**	**Plan/ Actual**
Project hours for this week	80.0	69.0	1.16
Project hours this cycle to date	400.0	344.8	1.16
Earned value for this week	10.3	3.1	3.37
Earned value this cycle to date	40.2	30.0	1.34
To-date hours for tasks completed	293.0	303.8	0.96

Interpreting the CPI

A CPI of 1 means

sum of the effort estimates for the completed tasks	=	sum of the actual effort for the completed tasks

What does this imply about the accuracy of the individual estimates?

Assuming the team is achieving the planned task hours, what does this imply about schedule performance?

What does a CPI of 0.5 imply about

- effort estimates?
- schedule performance (assuming the team is achieving the planned task hours)?

Interpreting the CPI (continued)

What does a CPI of 2 imply about

- effort estimates?
- schedule performance (assuming that the team is achieving the planned task hours)?

What general characterization can be made about schedule performance based on the CPI?

Schedule growth (due to effort estimates) = 1/CPI

Projected schedule = Original plan weeks/CPI

Interpreting Task Hour Data

The task hour data is in the form WEEK and can be interpreted similar to the effort for completed tasks data.

•TSP Week Summary - Form WEEK

•Name: Consolidated Team Plan
•Team: Security System Upgrade
•Status for Week: 5
•Week Date: 1/31/2000
•Date: 2/7/2000
•Cycle:

•Weekly Data	•Plan	•Actual	•Plan/ •Actual
•Project hours for this week	138.0	69.0	2.00
•Project hours this cycle to date	689.6	344.8	2.00
•Earned value for this week	10.3	3.1	3.37
•Earned value this cycle to date	80.4	30.0	2.68
•To-date hours for tasks completed	293.0	303.8	0.96

If (Plan hours to date)/(Actual hours to date) = 2

- What does it mean?
- What is the effect on schedule performance?

Interpreting Task Hour Data (continued)

If (Plan hours to date)/(Actual hours to date) = 0.5

- What does it mean?
- What is the effect on schedule performance?

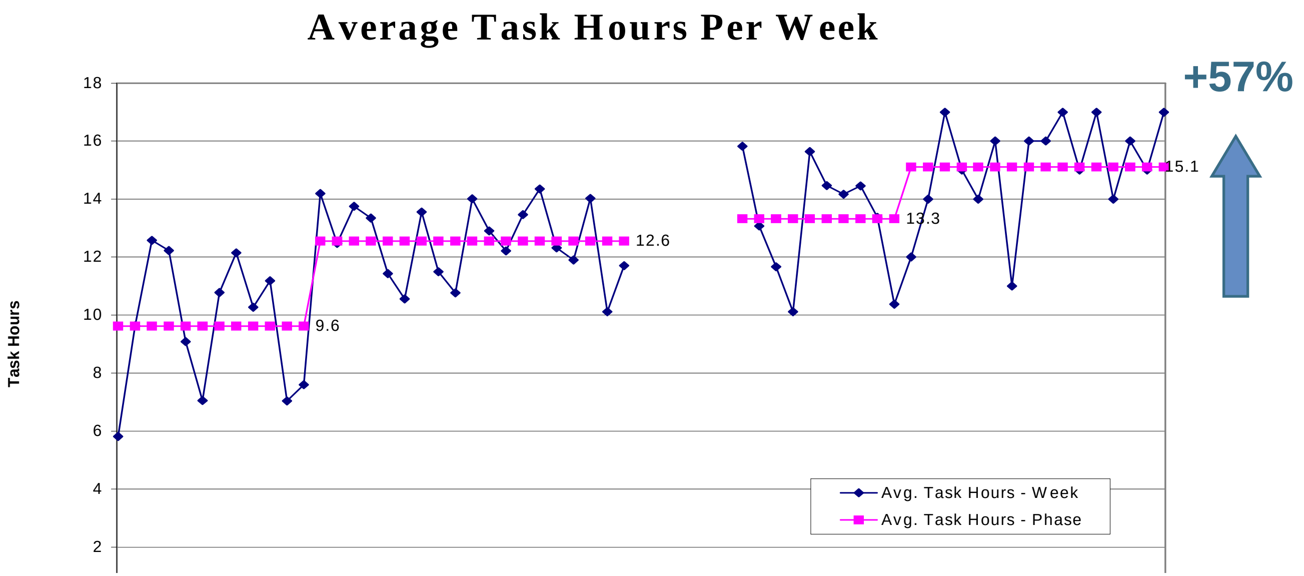
What general characterization can be made about schedule performance based on the plan/actual task hours?

Schedule growth (due to task hours) = plan/actual

Projected schedule = Original plan weeks * (plan/actual)

Improving Task Hours

Average task hours per developer per week were improved from 9.6 hours to 15.1 hours through quiet time, process documentation, more efficient meetings, etc.



Source: Allied Signal

Agenda

Why PSP and TSP for Systems Engineering?

Things That Change, Things That Don't

Time Logging Exercise

The TSP Launch

The TSP Management Framework

➡ TSP Quality Management

What is Quality?

Basic definition: Meeting the user's needs

There are three categories of product quality.

- functionality
- properties (e.g., safety, security, privacy, usability)
- defects

A software-intensive product can't be safe or secure until it is nearly defect-free.

Most current software-intensive processes are preoccupied with removing defects.

Little or no time is left for the other aspects of quality.

The System Quality Problem

Software quality problems are largely caused by defects.

- Defects are injected by the product's developers.
- Even experienced and capable developers inject many defects.
- Each defect is a potential system failure.
- A significant fraction of software defects can be avoided or mitigated by effective systems engineering.

Current practices often rely on testing to remove these defects.

Testing is necessary but, for finding and fixing defects, it is

- time-consuming
- expensive
- ineffective

The Defect Problem

Programs are complex products.

- Small programs have thousands of instructions.
- Large programs have millions of instructions.
- These instructions are individually produced.
- Each instruction must be precisely correct, beginning with the problem statement.

Software effort has a multiplying effort on systems engineering defects.

On average, even experienced programmers inject a defect about every 10-to-12 instructions.

Testing

A single test

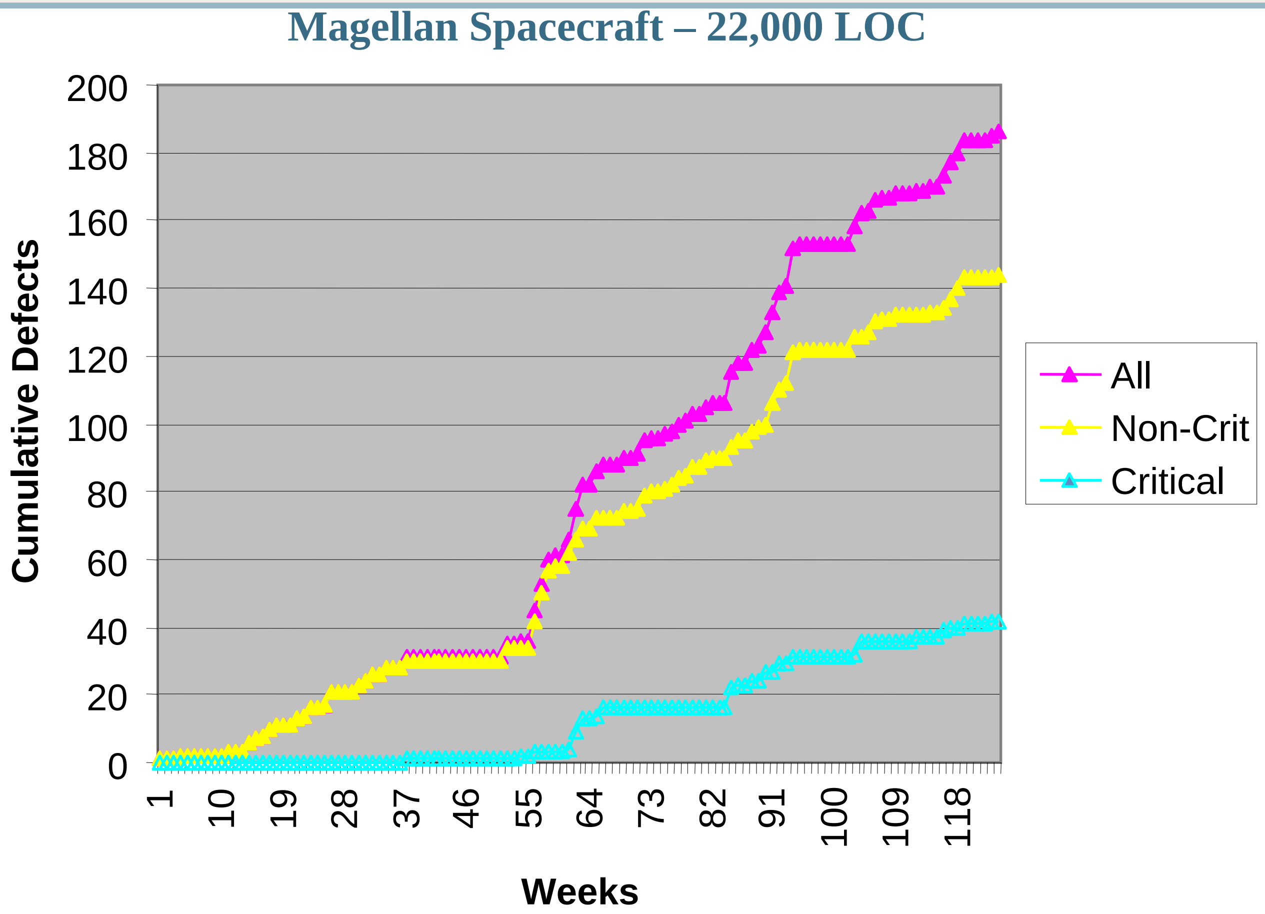
- exercises the product under one set of conditions
- produces correct or incorrect results

If there is a problem, developers must find the defect, fix it, and then test the fix.

For products with many possible operating conditions, many tests are required. *How many of these tests are defective?*

Projects that rely on testing for quality spend a lot of time and money on testing.

Testing Takes a Long Time



Testing Effectiveness

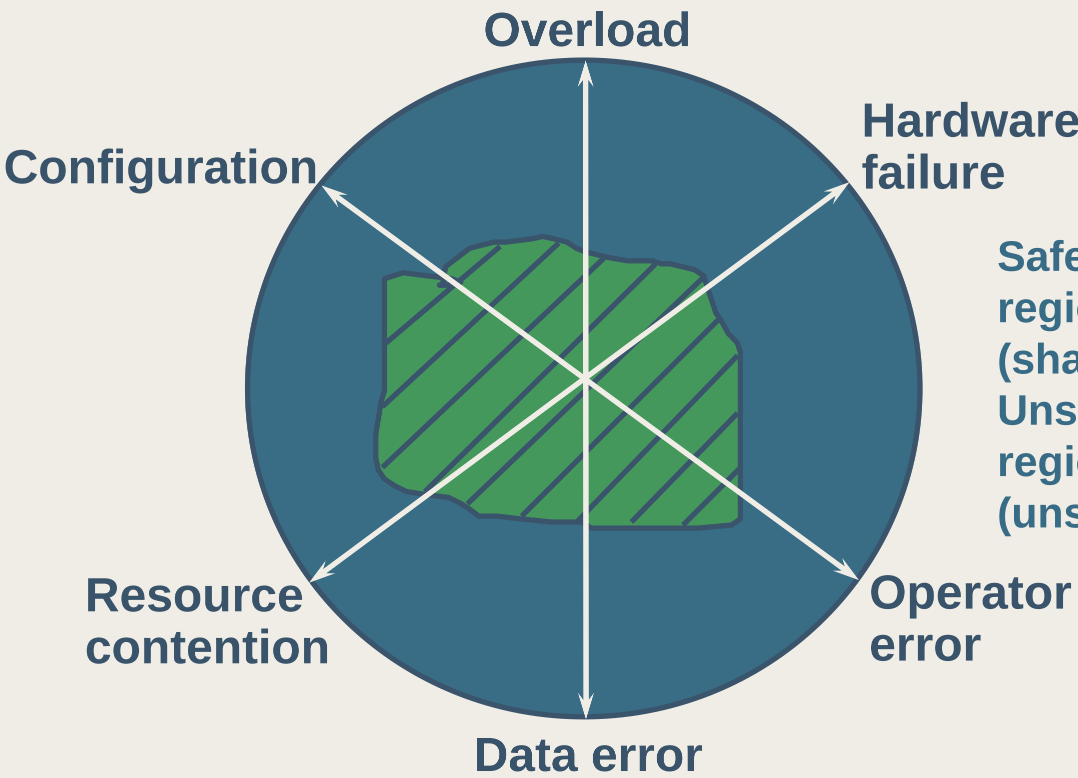
Large complex systems *cannot* be exhaustively tested.

- It is impossible to test every operating condition.
- Testing must focus on only the most frequent conditions.
- Extensive user testing finds even more defects.

Testing finds a percentage of the defects in a product, usually less than 50%.

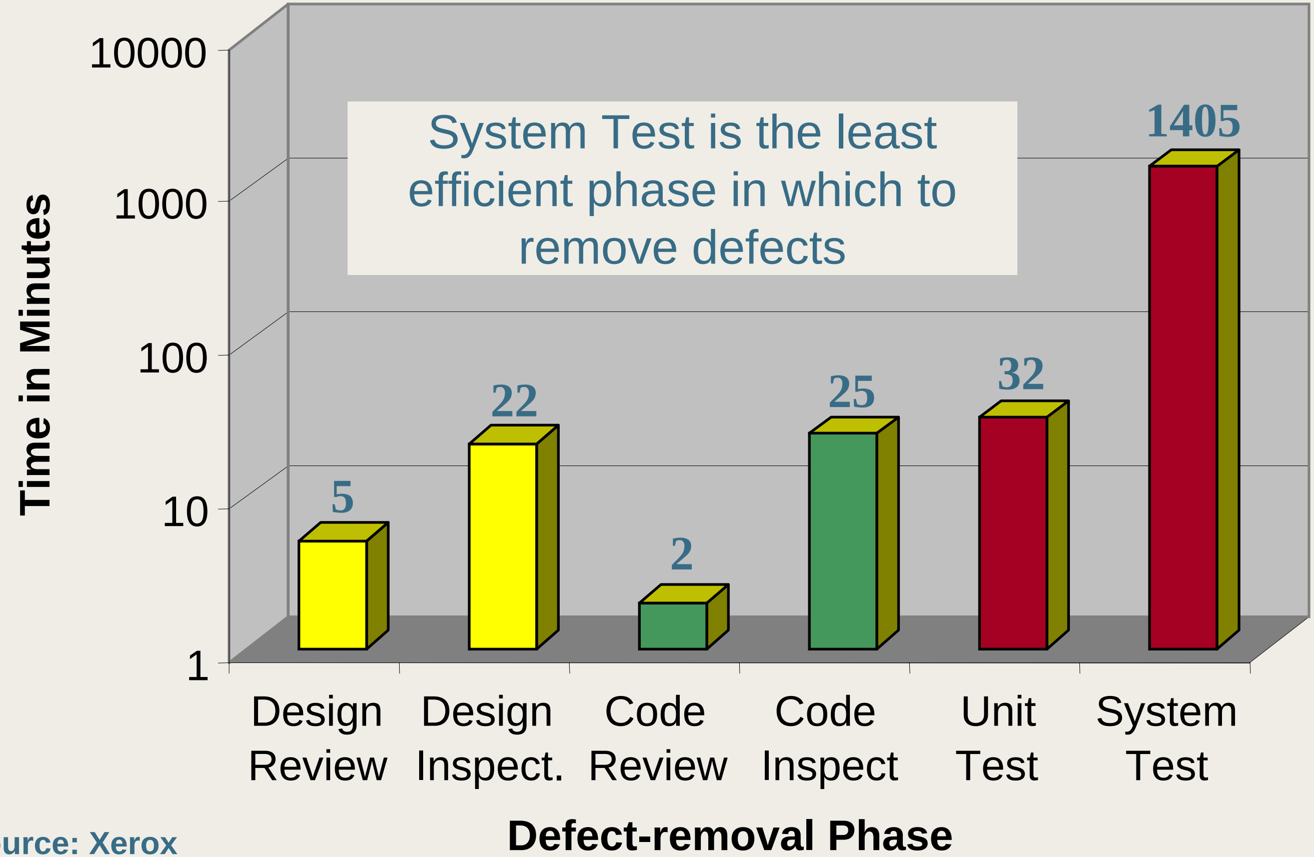
To get a quality product out of test, you must put a quality product into test.

Testing is Ineffective



Safe and secure region = tested (shaded)
Unsafe and insecure region = untested (unshaded)

Reviews and Inspections Save Time



Source: Xerox

Why TSP is Faster and Better

With TSP

- most defects are removed by reviews and inspections
- few defects are left for testing
- testing takes relatively little time

By using TSP, organizations can

- cut testing times by 80% or more
- shorten schedules
- reduce costs
- produce better products

Testing should verify that the development process worked well, rather than fix its exported problems.

Measuring Quality

To produce quality systems, the quality of all its parts must be measured and managed.

These measures must be made at every step in the process.

With TSP and the underlying PSP principles, developers use quality measures to manage the quality of their work. The developers

- inject fewer defects
- remove most defects soon after injecting them

TSP Quality Measures

There are many potential quality measures.

With the TSP, every product element and every process step can be measured.

Quality Measure		Description
Product Quality	Total defect density	The number of defects found in development, per unit of size
	Compile defect density	The number of defects found in compile, per unit of size
	Test defect density	The number of defects found in test, per unit of size
	Percent defect free	The percent of system modules or components that had no defects in a defect removal phase
Process Quality	Phase yield	The percent of defects in a product that are found during the phase
	Review rate	The volume of code or design that is reviewed per hour
	Defect removal rate - defects/hour	The hourly rate at which defects are removed in reviews or inspections
	Quality profile	Composite picture of a module's process quality
	Process quality index (PQI)	A composite value representing the five quality profile dimensions

Quality Implications

With proper training, guidance, and motivation, most developers can produce near-defect-free programs.

Does the same hold true for systems engineers?

With essentially defect-free products

- testing times are sharply reduced
- delivered products work
- maintenance costs are reduced

The key is the engineer's ability to produce defect-free products.

- measure quality
- manage quality
- personal quality commitment

Quality Goals and Plans

With data, TSP teams can

- set measurable quality goals
- make quality plans to meet these goals
- estimate the defects injected and removed in each phase
- track the work to see if they are meeting their quality plans

The TSP Defect Model

At each step of development, defects are injected, removed, or possibly both.

For each step:

Defects Out = Defects In + Defects Injected – Defects Removed

Defects In = Defects Out from the previous step

Defects Injected = function of time in production activities

Defects Removed = percentage (usually much less than 100%) of Defects In + Defects Injected

Example: Planning for Quality -1

A TSP team plans to develop 20 KLOC.

The goal is a design review yield of at least 70%.

- The plan shows 442 hours in detailed design.
- Data show that developers inject 1.3 defects per hour in detailed design.
- Data show that they remove 3 defects per hour in detailed design reviews.

What is the minimum design review time required to remove these defects?

Example: Planning for Quality -2

Defects injected

- 442 hours of design
- 1.3 defects injected per hour
- 1.3*442 = 574.6 defects injected

Defect removal

- 574.6 defects total
- 3 defects removed per hour
- 574.6/3 = 191.5 hours of design review time

The team should plan on 191.5 hours of review time.

To achieve a 70% yield, they must spend at least 0.7*191.5 = 134.1 hours in design reviews.

Example: Planning for Quality -3

Assume that

- no design reviews are done
- ½ of the design defects (.5 * 574 = 287) can be found by integration testing at 5 hours/defect
- ½ of the remaining defects (i.e. ½ of ½ or .5 * 287 = 144) can be found in system testing at 10 hours/defect

How much time will integration and system testing take?

How much time will be saved by doing design reviews?

How many design defects will likely remain for your customers to find?

Maintain Process Discipline

To produce quality systems, every part must be of high quality.

This is possible only if every developer consistently follows a quality process.

To consistently follow a quality process, each member of the development team must

- be properly trained (with the PSP or equivalent)
- work on a disciplined team (with the TSP or equivalent)
- have coaching support and management guidance

Management Support

People do not naturally do disciplined work.

To ensure disciplined work, management must
- train and support the developers
- ensure that the developers' work is guided and monitored
- provide coaching assistance

Management must also
- build and maintain effective teams
- ensure that all team members are trained and willing to follow the process
- recognize and reward quality work

TSP Quality Messages

High-quality processes produce high-quality products.

Quality work is not done by accident; it requires discipline, commitment, management, and measurement.

Quality work saves time and money.

The cornerstone of a high-quality software process is early defect removal.

TSP shows teams how to efficiently remove defects at the earliest possible point in the process.

Your Organization is Unique…

…but most organizations share common problems.

An organization can change under duress, or it can change in response to leadership.

Duress can lead to undesirable consequences since, by definition, it is trying to get away from whatever is causing the duress.

Only leadership can take an organization reliably in a desired direction.

Where will you lead your organization?

Thank you!

Contact information: jdm@sei.cmu.edu

Contact a PSP or TSP transition partner:
http://www.sei.cmu.edu/collaborating/partners/trans.part.psp.html

Contact SEI customer relations:

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213-3890
Phone, voice mail, and on-demand FAX: 412/268-5800
E-mail: customer-relations@sei.cmu.edu

Trademarks and Service Marks

The following are service marks of Carnegie Mellon University.

- Team Software Process^{SM}
- TSP^{SM}
- Personal Software Process^{SM}
- PSP^{SM}

The following are registered in the U.S. Patent & Trademark Office by Carnegie Mellon University.

- Capability Maturity Model®
- CMM®
- CMMI®