

# Managing Software Quality with the Team Software Process

James W. Over

April 13, 2010



# Key Message

---

Society depends on software.

As software professionals we have an obligation to produce reliable, secure software.

The methods exist to achieve this goal, but they aren't widely used.

Software quality professionals should help shift the profession from its ad-hoc, “test-in quality” mindset, towards a measured, disciplined, “build-in quality” approach.



# Team Software Process (TSP)

---

TSP is a process that is specifically designed for software teams.

It's purpose is to help teams

- plan their work
- negotiate their commitments with management
- manage and track projects to a successful conclusion
- produce quality products in less time
- achieve their best performance without the “death march” ending



# TSP Quality Improvements at Microsoft

## Background information

- two consecutive releases of the same system
- same six month schedule
- same seven member team
- similar functionality produced
- TSP used on release 2.5

<b>Post code complete defects</b>		
<b>Phase</b>	<b>Version 2.4</b>	<b>Version 2.5</b>
<b>Integration Test</b>	<b>237</b>	<b>4</b>
<b>System Test</b>	<b>473</b>	<b>10</b>
<b>User Acceptance Test</b>	<b>153</b>	<b>3</b>
<b>Total</b>	<b>1072</b>	<b>17</b>

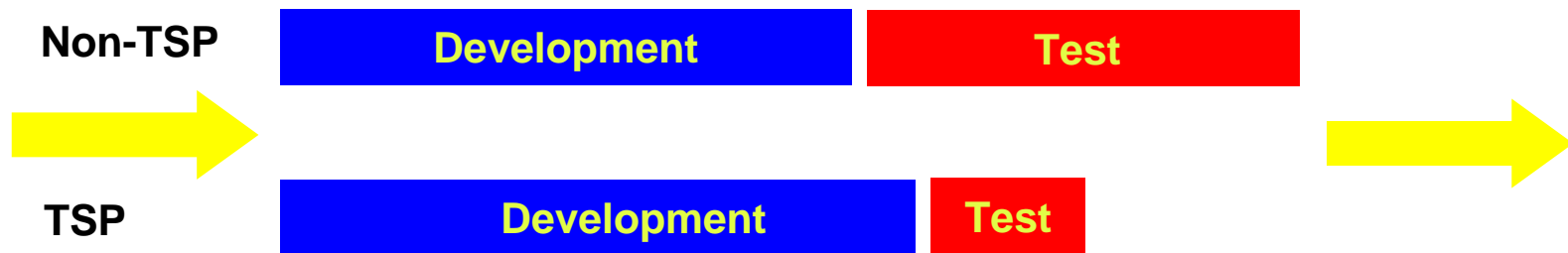


# Quality Improvement at Intuit

---

From data on over 40 TSP teams, Intuit has found that

- sixty percent fewer defects after code-complete
- post code-complete effort is 8% instead of 33% of the project
- standard test times are cut from 4 months to 1 month or less



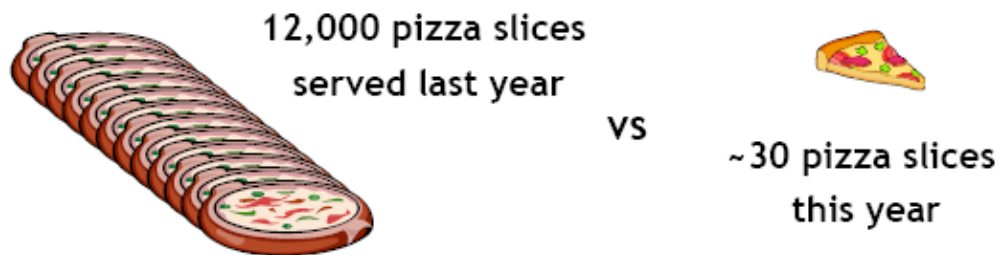
# Quality and Work-Life Balance

Finding and retaining good people is critical to long-term success.

Intuit found that TSP improved work-life balance, a key factor in job satisfaction.

## Results at Intuit: Improved Work-Life Balance

- Half as many weekend source check-ins (<3%)
- Reduced \$ on dinners as measured by PSS - "Pizza Slices Served"



TSP helped improved employee work life balance

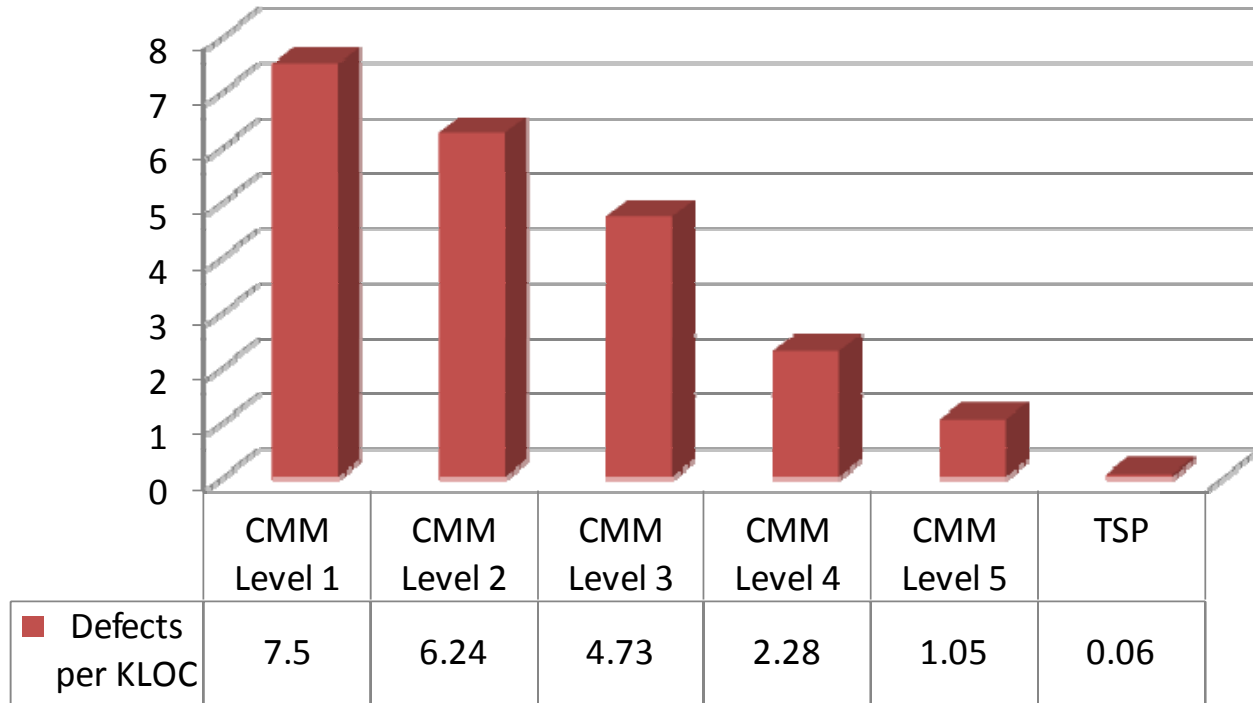
Source: Intuit



# TSP Quality Performance

In a study of 20 projects in 13 organizations TSP teams averaged 0.06 defects per thousand lines of new or modified code.

Average Defect Density of Delivered Software



Source: CMU/SEI-2003-TR-014



# Topics

---

## How is software quality managed today?

What motivates and convinces software teams to manage quality?

What methods should software teams use to manage quality throughout development and testing?

What data should teams collect?

- How is the data used by development teams and QA?
- What leading indicators can be used to identify quality problems early in development?



# Software Industry Quality Performance

The quality of software products is worse than most other hi-tech products.

Many important software products have 1 to 2 defects per thousand lines of code, or higher.

- operating systems
- communications systems
- database systems

Application software is usually worse.



Depicted above: Linux system crash screen on an Airbus entertainment system



# Software Industry Quality Strategy

---

The software industry is the only modern high-tech industry that relies heavily on testing to remove defects.

Many software defects are found in or after test when defect removal costs are the highest and the methods are the least effective.

This strategy results in defective products and unnecessary rework that inflates development costs by 30% to 40% or more.

This strategy is also a principal cause of unexpected delays, system failures, and software security vulnerabilities.



# Software Quality Practice

---

Formal inspections are not widely used.

- Peer review by another developer is the most common review practice
- Often only the “critical” code is reviewed or inspected.
- Inspections aren’t measured or managed to improve effectiveness.

Quantitative quality management is not common practice.

- Quality plans are generally qualitative not quantitative.
- Defects generally aren’t counted before test or code inspection.
- Quality cannot be managed or tracked before testing begins due to a lack of plans and data.



# To Engineer is Human\*

---

Software engineering is the art of turning ambiguous requirements into precise instructions.

On average, most software developers inject one defect in every 7 to 12 lines of code.

Typically 20% to 25% of these defects escape into system testing where they will take 1 to 2 days each to find and fix.

\* To Engineer is Human: The Role of Failure in Successful Design, by Henry Petroski.



# The Risk of Poor Quality

---

Computers are involved in nearly every aspect of our lives.

While computers are very reliable, software is not.

The risk of loss of life or property is increasing due to software in

- medical and healthcare systems
- financial systems
- network and communications systems
- aircraft and air traffic control systems
- power generation and distribution systems

*"If GM had kept up with technology like the computer industry has, we would all be driving twenty-five dollar cars that got 1,000 miles to the gallon." – Bill Gates*

*"If GM had developed technology like Microsoft, we would all be driving cars ...that for no reason whatsoever would crash twice a day" – General Motors*



# The Cost of Poor Quality

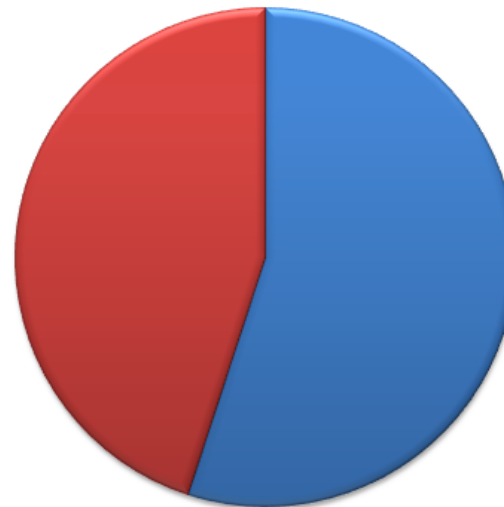
Without reviews or inspections a 50,000 LOC system has

- 20+ defects/KLOC at test entry
- that is 1000+ defects
- at the typical 10+ hours per defect, that is 10,000+ programmer hours to find and fix

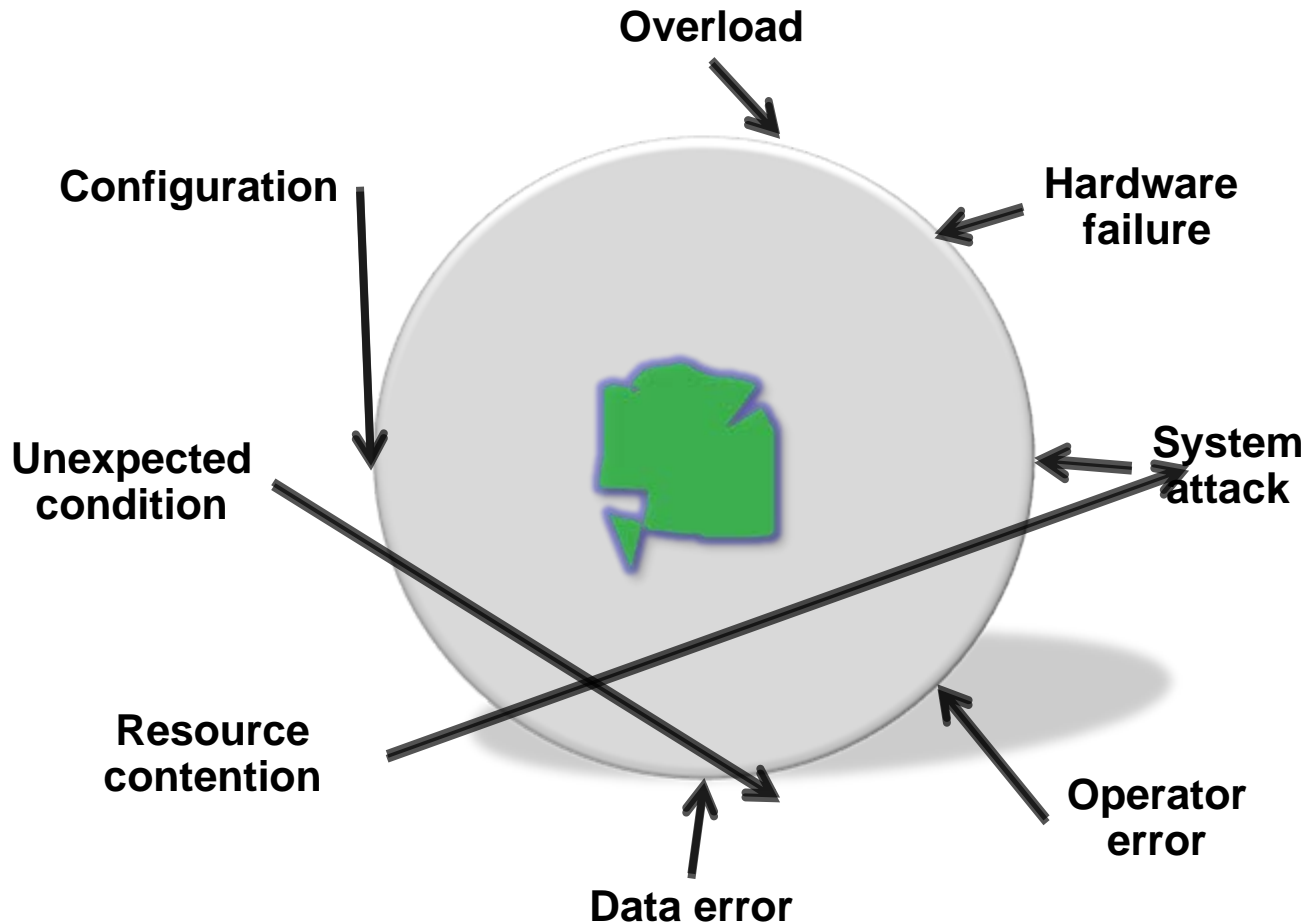
*The cost of removing these defects is about 5 programmer years, or nearly half the cost of developing 50,000 LOC.*

## Software Development Cost

■ Development Costs ■ Testing and Rework



# Why Testing Isn't Enough



# Software Quality Management

---

IBM's Dr. Harlan Mills said, "*How do you know that you've found the last defect in system test?*"

*"You never find the first one."*

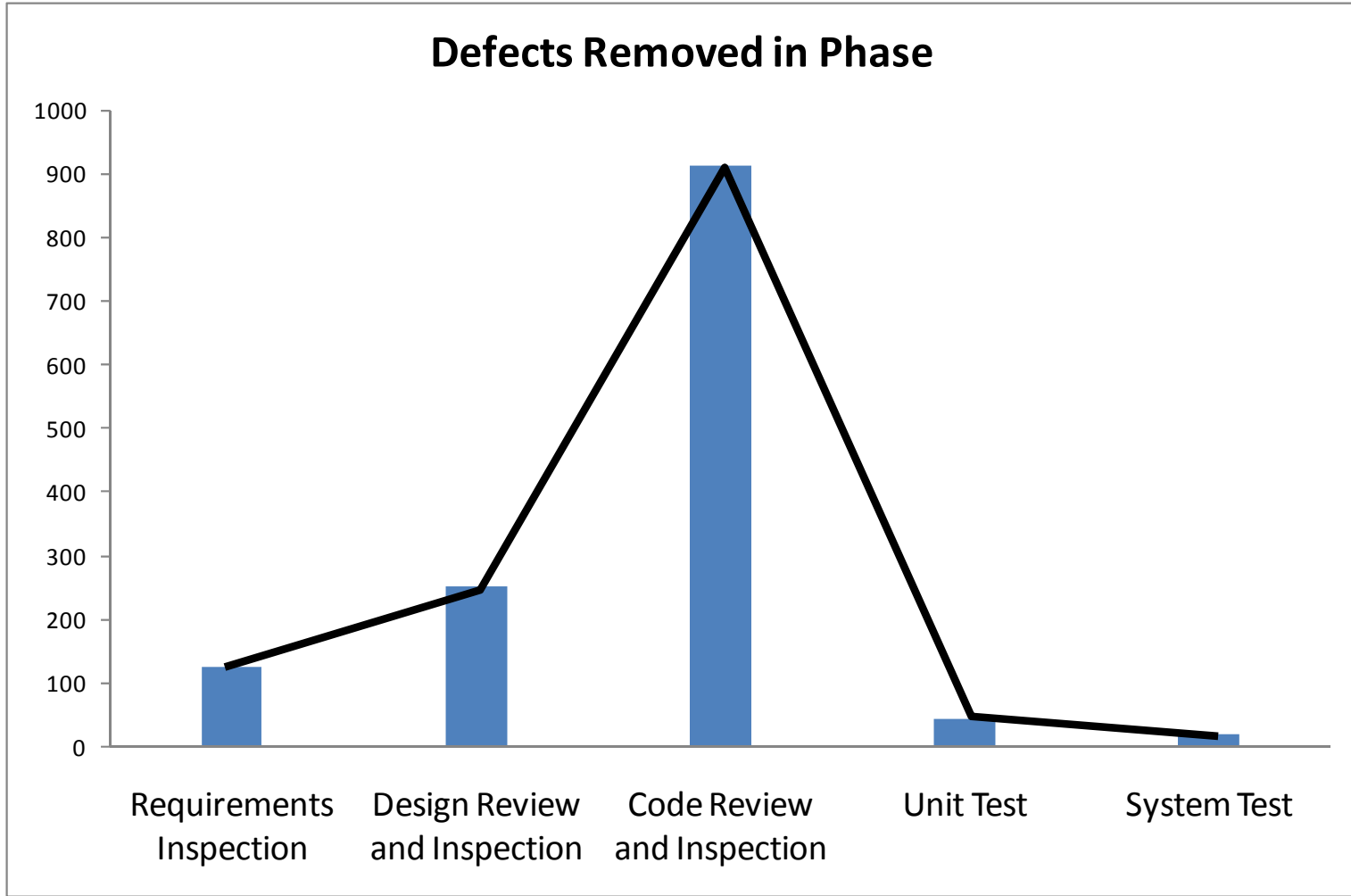
If you want a quality product out of test, you must put a quality product into test.

*How do you put a quality product into test?*

Measure and manage quality at every step, from requirements through system test.



# Early Defect Removal Strategy



# Topics

---

How is software quality managed today?

**What motivates and convinces software teams to manage quality?**

What methods should software teams use to manage quality throughout development and testing?

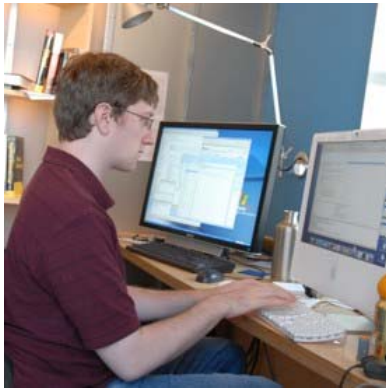
What data should teams collect?

- How is the data used by development teams and QA?
- What leading indicators can be used to identify quality problems early in development?



# Commitment to Quality

“The system test engineers became convinced that TSP was worthwhile when they realized that they were going from tracking down software bugs in the lab to just confirming functionality. Our first project: certified with ten times increase in quality with significant drop in cost to develop. Follow-on project: certified with NO software defects delivered to system test or customer.”



“My first TSP-based team recently finished their system test. They had three system test defects in 7400 lines of new code. No defects were code- or design-related; they were either install or documentation— each of which took about five minutes to fix. System test took less than five percent of the overall project effort.”



# Catch-22

---

To use new methods, software professionals must believe the methods will help them do better work.

To believe that, they must have used the methods.

To break this conundrum, TSP has a course where professionals

- use new methods to write several small programs
- plan, measure, track, and analyze their work

They then learn from their own data that the new methods work.



# Learning to Develop Software

---

In computer science and software engineering education,

- the emphasis is on technical knowledge and individual performance.
- evaluation emphasizes code that runs, not how the student got there.
- the prevailing ethic is to code quickly and fix the problems in test.

Developers then use these same practices on the job resulting in

- missed commitments
- lengthy testing schedules
- buggy software



# Personal Software Process

---

The PSP is a process for structured personal tasks.

Developers learn PSP in a hands-on course where they use a defined and measured process to estimate, plan, track, and manage quality.

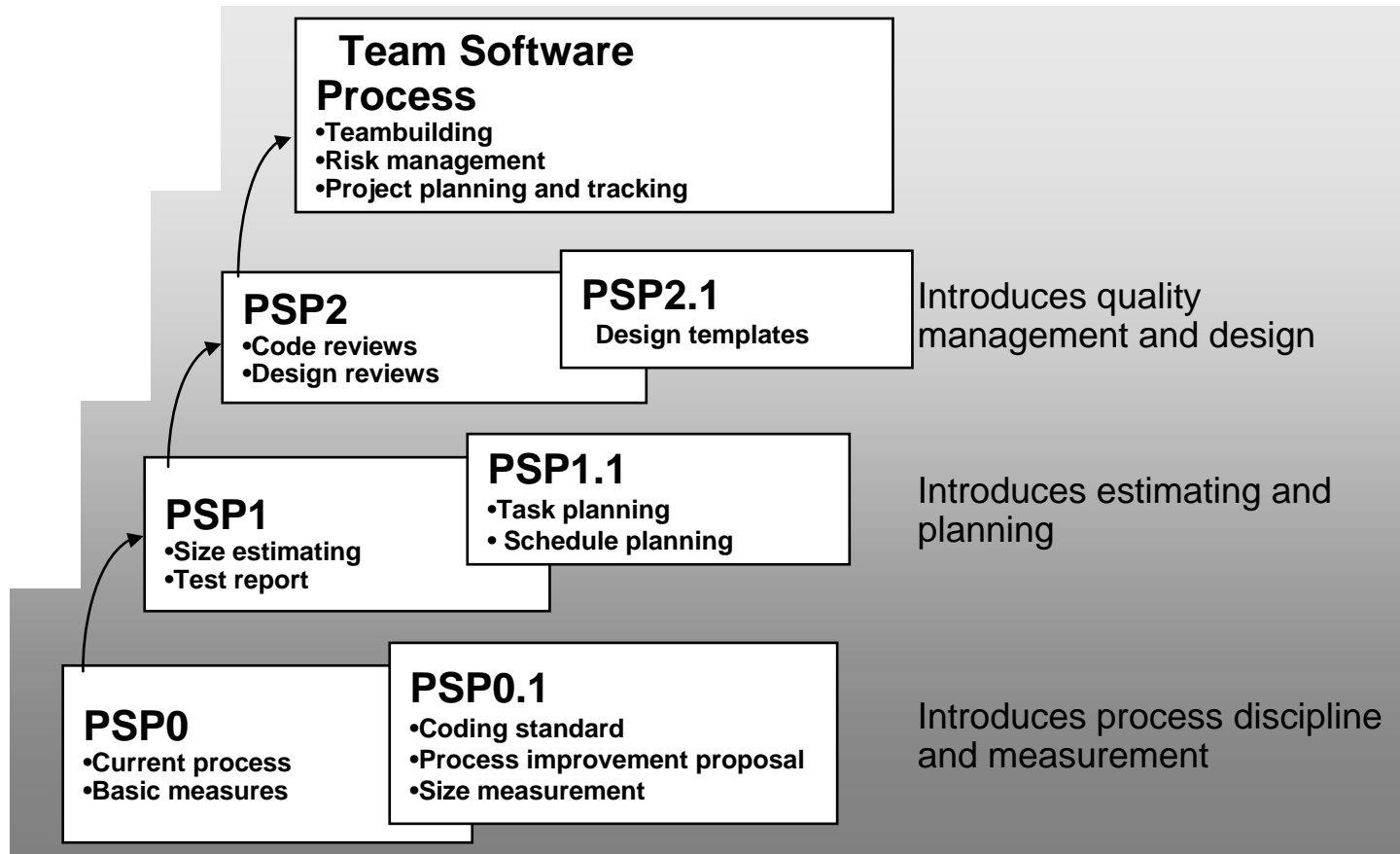
This leads to

- better estimating, planning, and tracking
- protection against over-commitment
- a personal commitment to quality

The training provides the self-convincing evidence of the benefits that developers need to use these methods in practice.



# PSP Learning Stages



Developers write one or more programs at each PSP level

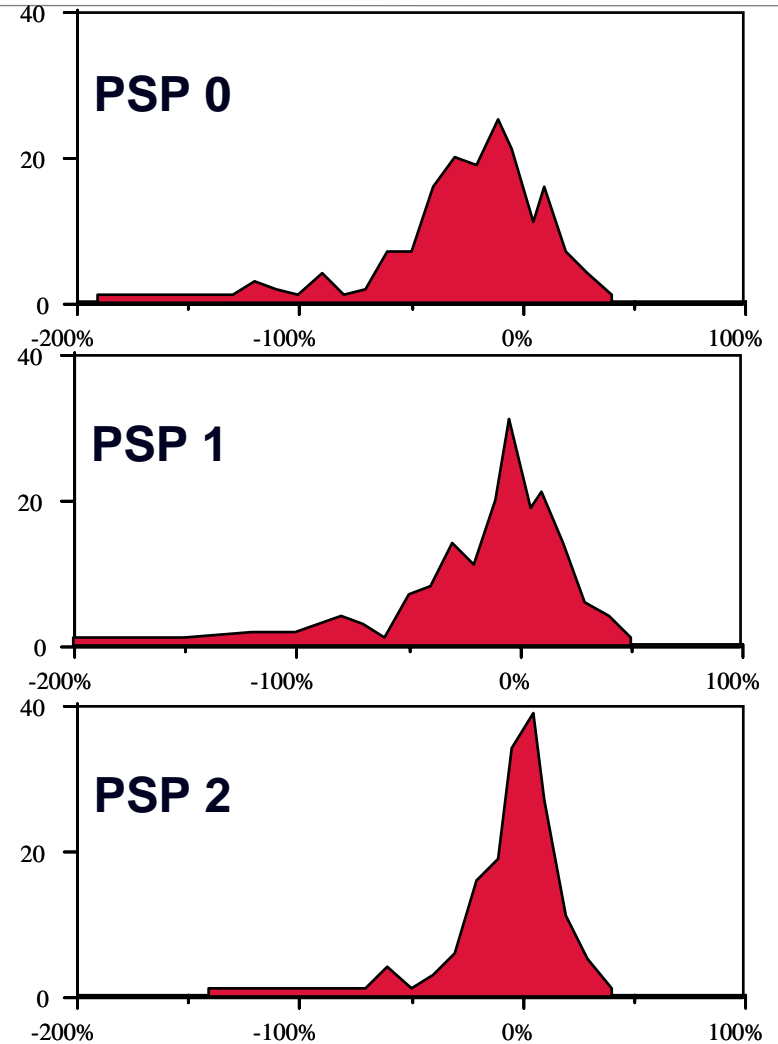


# PSP Effort Estimating Accuracy

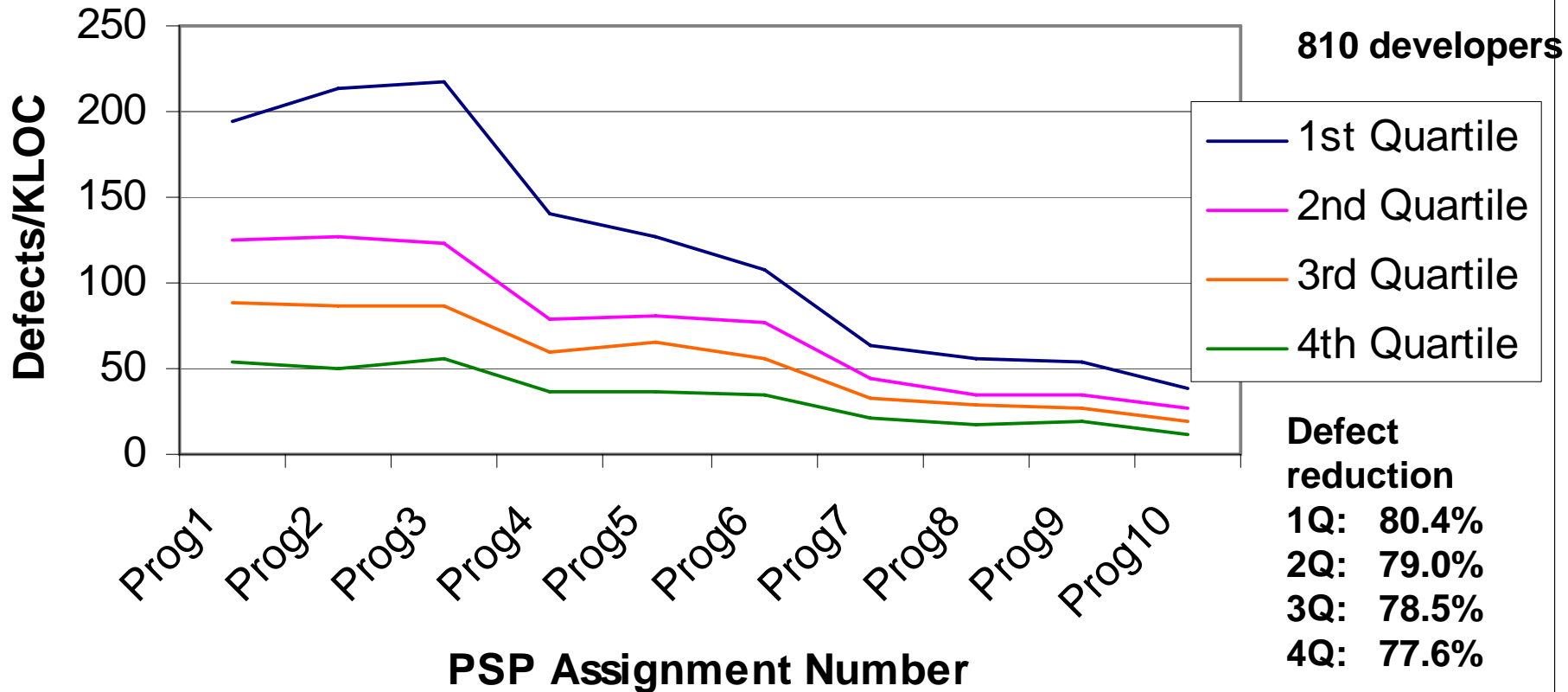
Majority are under-estimating

Balance of over-estimates and under-estimates

Much tighter balance around zero

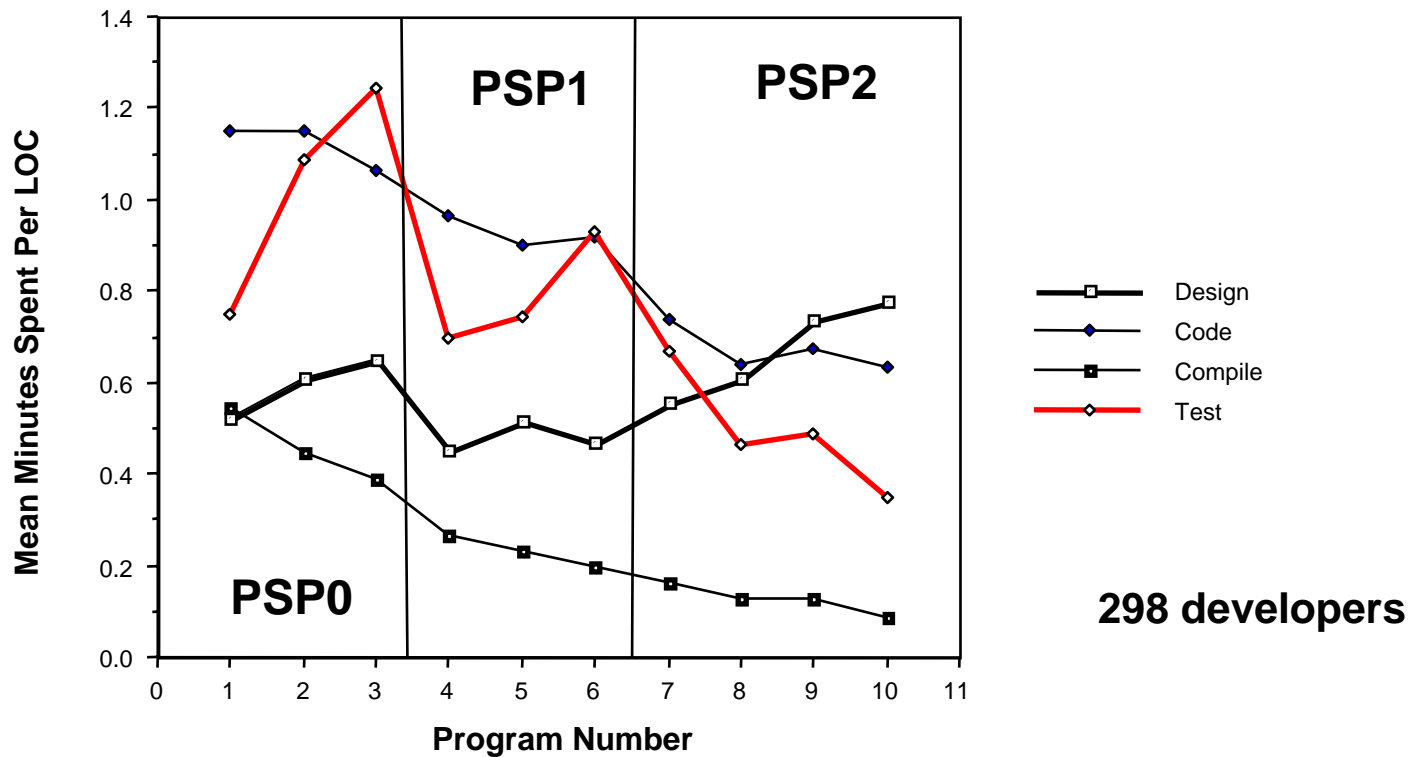


# Compile and Test Defects - from PSP Training



# PSP Design Time Results

Time Invested Per (New and Changed) Line of Code



# Quality and the Team

---

Quality doesn't happen by accident.

Quality software is possible only when every member of a development team makes a personal commitment.

To build a high-quality product they must

- be properly trained and motivated
- understand their personal quality data
- have control of their process and plans
- have the proper data to track quality



# Topics

---

How is software quality managed today?

What motivates and convinces software teams to manage quality?

What methods should software teams use to manage quality throughout development and testing?

What data should teams collect?

- How is the data used by development teams and QA?
- What leading indicators can be used to identify quality problems early in development?



# Building Quality Products

---



# Management Styles

---

The principal management styles have been:



## Body Management

People as oxen that must be driven, directed, and motivated through fear.



Frederick Taylor

## Task Management

People as machines. Management knows the best way to get the work done. The workers follow.



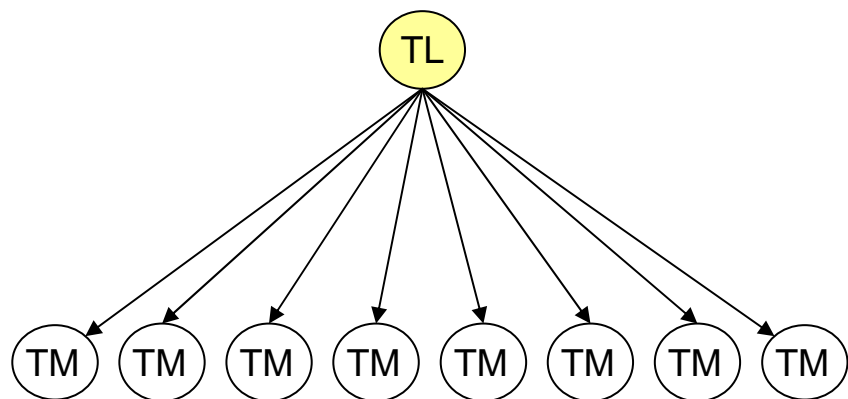
Peter Drucker

## Knowledge management

People as individuals. The knowledge worker knows the best way to get the work done. Management motivates, leads, and coaches.

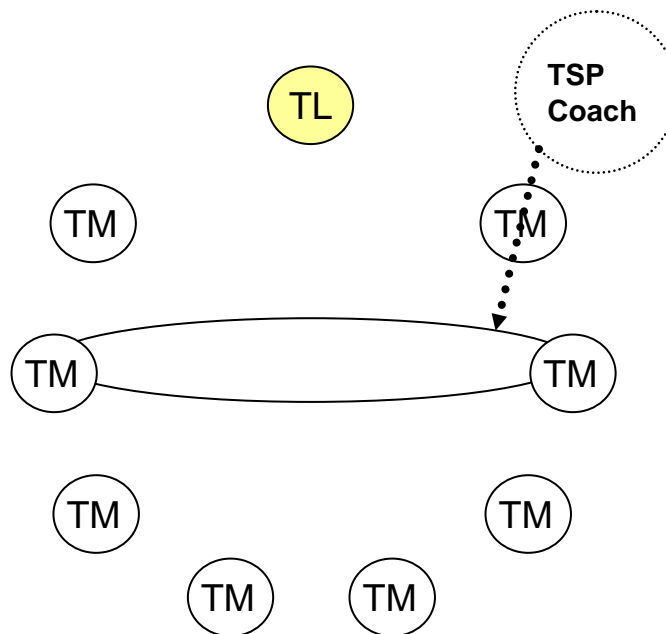


# Software Team Management Styles



## Traditional team

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

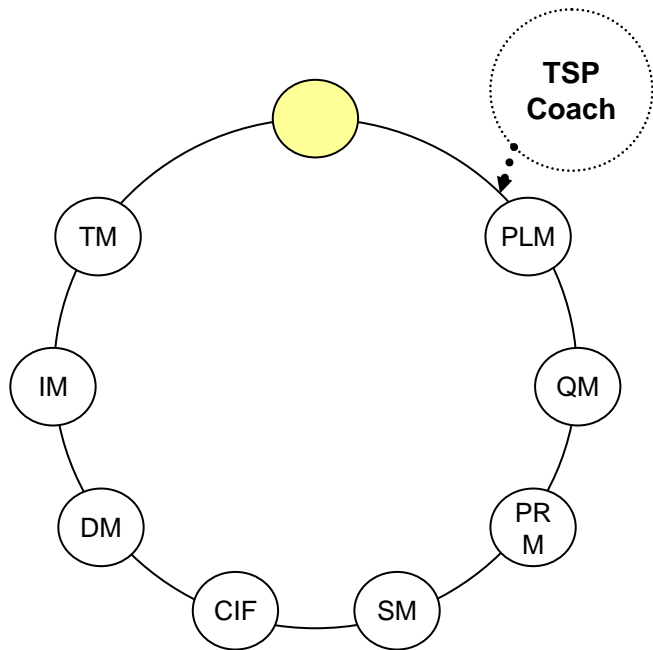


## Self-directed team

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



# Sharing the Team Management Responsibilities



## Self-directed team roles

Eight pre-defined roles distribute traditional project management responsibilities across the team.

All team members have traditional roles, e.g. developer, tester, etc.

## Project Management Roles

**Planning manager** – responsible for tracking the plan.

**Quality manager** – responsible for tracking the quality plan.

**Process manager** – responsible for ensuring process discipline and for process improvement.

**Support manager** – responsible for ensuring that support needs are met and for configuration management.

## Technical Roles

**Customer interface manager** – responsible for the interface to the customer or customer representative.

**Design manager** – responsible for the design practices and quality.

**Implementation manager** – responsible for implementation practices and quality.

**Test manager** – responsible for test practices and quality.

# The Coaching Role

---

## The coach

- trains and facilitates the adoption of team-based practices
- works with the team leader to build the team
- observer that uses data to guide the team



Tiger Woods and his coach Hank Haney.

### Team Leader vs. Coach

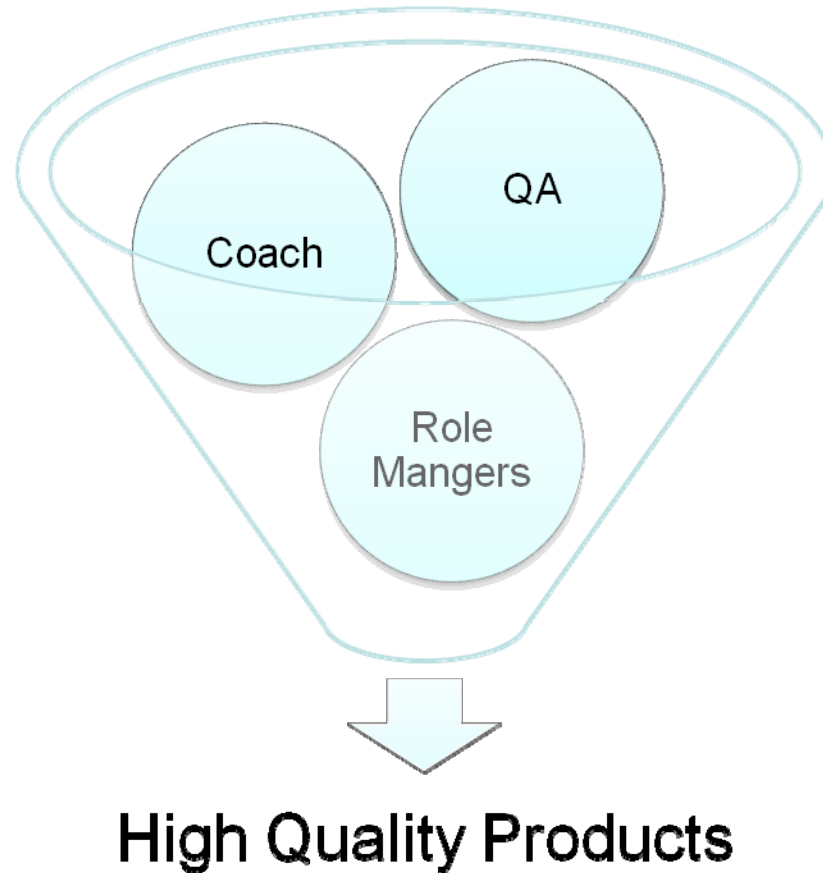
The team leader's job is to use the team to build the product.

The coaches job is to use the project to build the team.



# Working Together to Improve Product Quality

---



# Planning Accuracy and Quality

---

Most software projects are underestimated and are therefore late before they start.

When projects are running behind schedule, managers and developers will abandon the process and look for shortcuts.

- hurry through design
- code quickly and deliver to test
- rush through test and fix only the most critical bugs

Without reasonably accurate plans, quality suffers.



# Guidelines for Improving Plan Accuracy

---

Create a conceptual design as the basis for the estimate.

Estimate size first, then effort, to reduce estimating bias.

Use historical data for size and effort estimates to further reduce bias.

Estimate in detail to further reduce cumulative error.

Use historical data to estimate resource availability.

Make and use a quantitative quality plan to reduce the risk of schedule delays caused by “buggy” software.

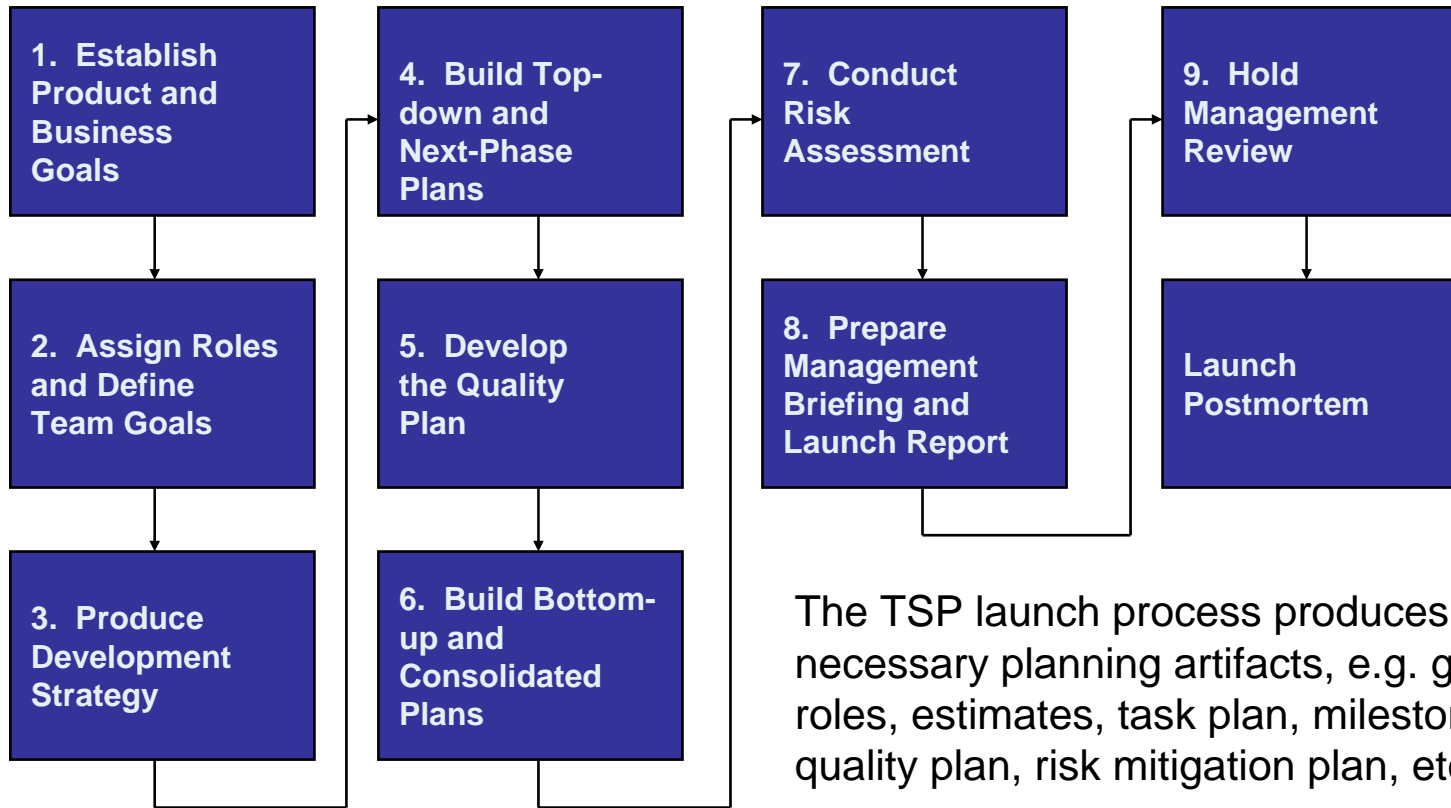
Do workload balancing.

Good plans are dynamic, plan early and often.

The best plans are made by the people assigned to do the work.



# Self-directed Team Planning: The TSP Launch Process



The TSP launch process produces necessary planning artifacts, e.g. goals, roles, estimates, task plan, milestones, quality plan, risk mitigation plan, etc.

*The most important outcome is a committed team.*



# TSP Quality Management Practices

---

TSP incorporates several quality management practices

- planning for quality
- yield management
- capture/recapture
- defect prevention

Quality is measured and tracked throughout the process.



# Quality Planning

Quality guidelines are used during TSP planning to

- estimate defects injected
- make a plan for their removal

Estimates are based on historical defect densities or, injection rates and phase yields.

Quality indicators are then calculated from these data and used to track plan vs. actual quality during execution.

Quality Guideline	Benchmark Value
Review rate	200 LOC/Hr.
Design injection rate	0.75/Hr.
Design review removal rate	1.5/Hr.
Design inspection removal rate	0.5/Hr.
Design review/inspection yield	70%
Code injection rate	2/Hr.
Code review removal rate	4/Hr.
Code inspection removal rate	1/Hr.
Code review/inspection yield	70%
Unit test removal rate	3/Hr.
Unit test yield	50%
Expected defect density	
Unit Test	< 5/KLOC
Integration Test	< 0.5/KLOC
System Test	< 0.2/KLOC
Acceptance Test	< 0.1/KLOC

# Yield Management

---

Every process phase has the potential to inject new defects.

There are many techniques for finding and fixing these defects.

- walkthroughs, reviews, and inspections
- manual and automated testing tools

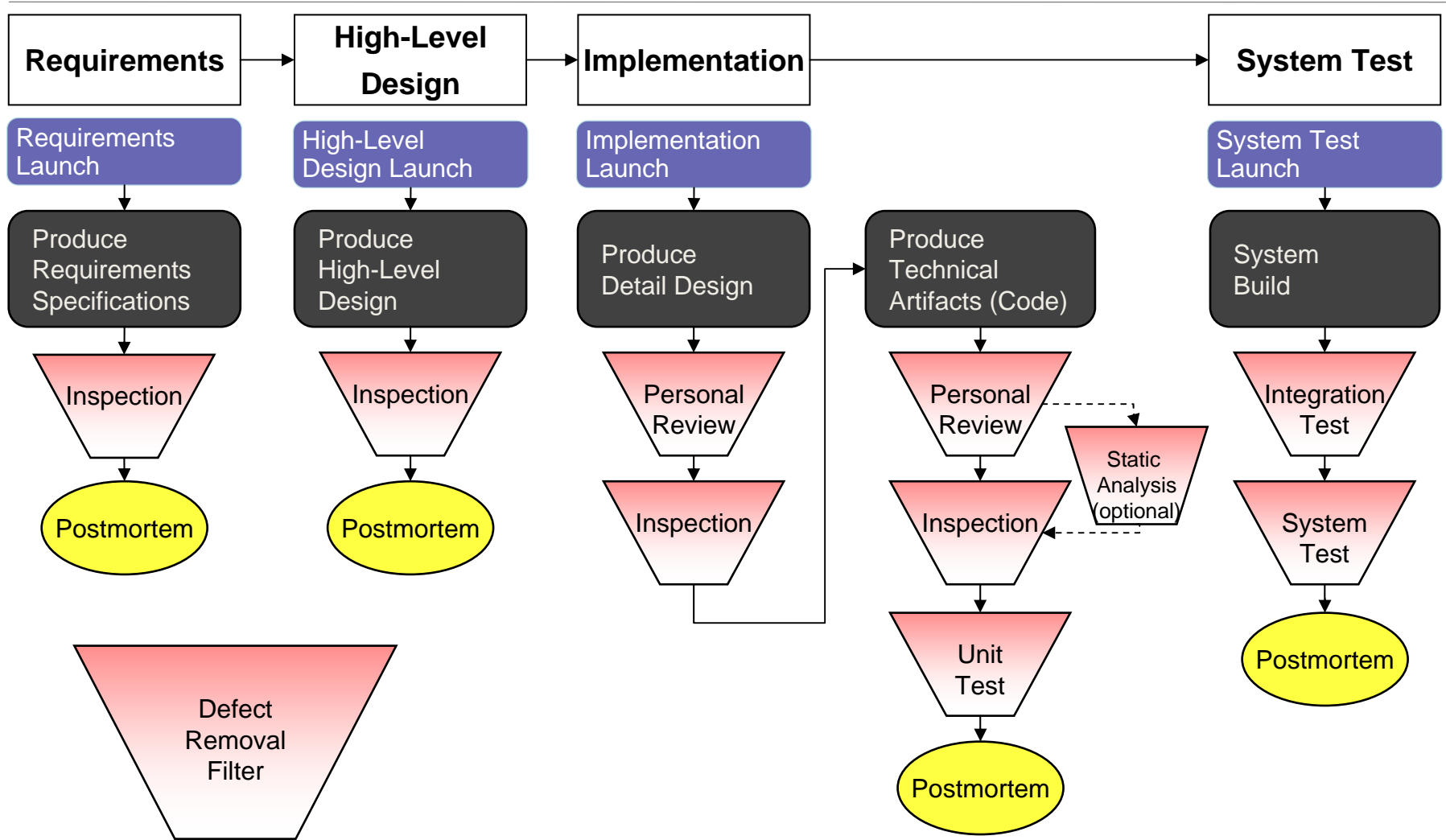
Think of these techniques as defect removal filters

The cleanest software is produced by using multiple filters.

- test-only process yield: less than 99%
- multi-stage defect filter process yield: 99.9% to 100%



# TSP Process with Defect Removal Filters

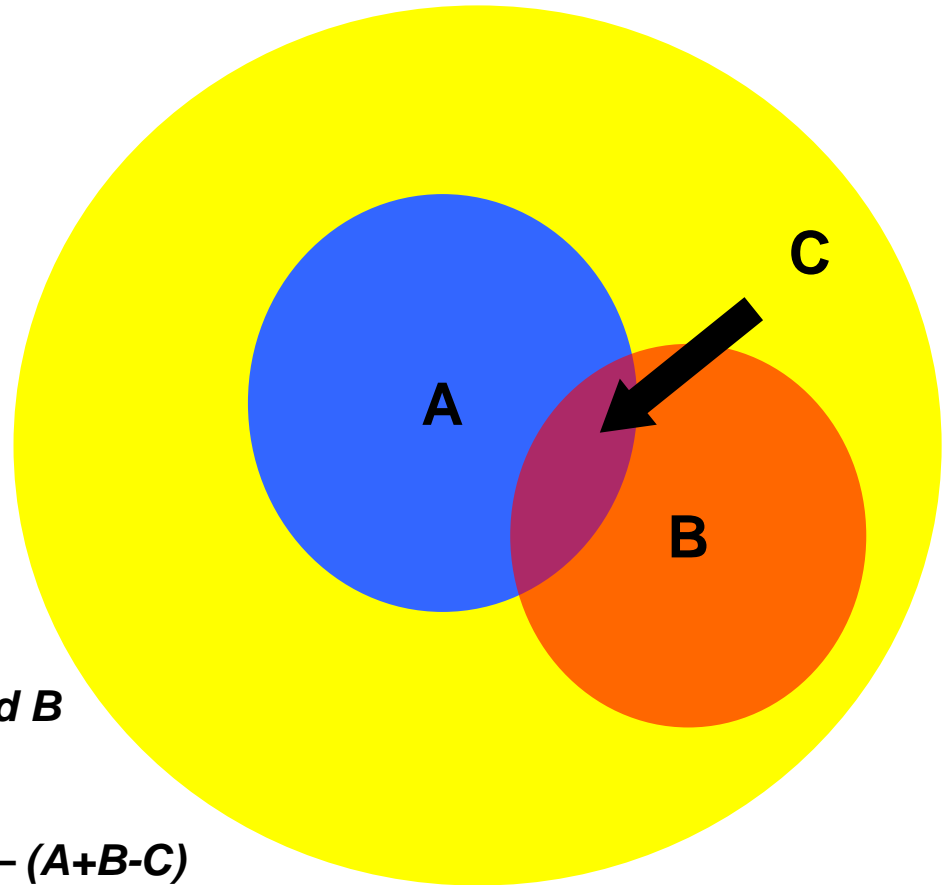


# Capture-Recapture

The capture-recapture method uses sampled data to estimate populations.

It can be used to estimate the defects in a product.

***A = Defects in test A***  
***B = Defects in test B***  
***C = Defects common to A and B***  
***Est. total defects =  $A*B/C$***   
***Total found =  $A+B-C$***   
***Est. total remaining =  $A*B/C - (A+B-C)$***



# System Test and Defect Prevention

---

What is the typical response when a defect is found in system test?

- The defect is reported.
- Someone is assigned to find and fix it.
- When fixed the module is checked back in for testing.

System test yields are low (~50%), so system test defects get special treatment in the TSP.

- Every defective module is re-inspected.
- A defect prevention process is invoked if defects are found during or after system test.
- Each defect is analyzed to prevent future escapes.



# Topics

---

How is software quality managed today?

What motivates and convinces software teams to manage quality?

What methods should software teams use to manage quality throughout development and testing?

What data should teams collect?

- How is the data used by development teams and QA?
- What leading indicators can be used to identify quality problems early in development?



# The TSP Measurement Framework



**Schedule**



**Effort**



**Size**



**Quality**

*Four direct measures apply to all processes and products*

*Estimates made during planning*

*Directly measured by team members while working*

*The data are used to track project status and to analyze and improve performance.*

*Direct measures, integrated into a measurement framework, provide flexibility.*



# Schedule

Schedule is the most commonly used project measure.

Schedule accuracy depends on granularity.



TSP schedule granularity is in hours, not days, weeks, or months.

TSP Task Planning Template - Form TASK			Total Plan Hours		Total Actual Hours								
Name Prasad Perini			318.9										
Team PSP Ghost			Reminder: Estimated Hours can be entered manually - OR - calculated based on Estimated Size and Rate. If Size and Rate are present, this field will be recalculated when you Update Task and Schedule.										
Date 2/3/2004													
Cycle													
<input type="button" value="Generate Task List"/> <input type="button" value="Update Task and Schedule"/>			Resources	Estimated Size	Size Measure	Rate (per Hr.)	Estimated Hours	Engrs	Plan Hours	Plan Date	Plan Week	Actual Hours	Actual Date
Assembly	Phase	Task											
Main Form	DLDINSP	Main Form DLD Inspection	SA, PP	300	LOC	200.0	1.5	1.0	1.5	3/10/2003	15	5.0	3/7/2003
Main Form	CODEINSP	Main Form Code Inspection	SA, PP	300	LOC	200.0	1.5	1.0	1.5	3/10/2003	15	4.8	3/10/2003
Filter Object	CODEINSP	Filter Object Code Inspection	SA, PP	300	LOC	200.0	1.5	1.0	1.5	3/10/2003	15	3.2	1/22/2003
Task Panel Control	DLDINSP	Task Panel Control DLD Inspection	NK, PP	250	LOC	200.0	1.3	1.0	1.3	3/10/2003	15	0.0	3/7/2003
Task Panel Control	CODEINSP	Task Panel Control Code Inspection	NK, PP	250	LOC	200.0	1.3	1.0	1.3	3/10/2003	15	0.0	3/10/2003
ProfileUserList.aspx	DLDINSP	ProfileUserList.aspx DLD Inspection	PP, VY	1010	LOC	200.0	5.1	1.0	5.1	3/17/2003	16	2.0	2/4/2003
ProfileUserList.aspx	CODEINSP	ProfileUserList.aspx Code Inspection	PP, VY	1010	LOC	200.0	5.1	1.0	5.1	3/17/2003	16	4.4	2/27/2003



# Time

Time is a measure of time on task.

The TSP time measure is task hours, i.e. the time spent on a project task, minus interruption time.

TSP team members record their time as they work, not at the end of the day, week, or month.



TSP Time Recording Log - Form LOGT							
Name Prasad Perini				Date 2/3/2004			
Team PSP Ghost							
				Cycle			
				Hours 321.2			
Assembly	Phase	Task	Date	Start	Int.	Stop	Delta
OEM-ChangeR	PLAN	OEM-ChangeRequest-7 PLAN	03/13/03	15:45:10		16:22:43	37.6
OEM-ChangeR	HLD	OEM-ChangeRequest-7 HLD	03/13/03	16:53:08		17:30:40	37.5
OEM-ChangeR	DLD	OEM-ChangeRequest-7 DLD	03/13/03	17:30:49		18:02:59	32.2
OEM-ChangeR	DLD	OEM-ChangeRequest-7 DLD	03/13/03	18:55:20		19:54:35	59.3
OEM-ChangeR	DLDR	OEM-ChangeRequest-7 DLDR	03/14/03	10:00:43		10:31:59	31.3
OEM-ChangeR	DLDINSP	OEM-ChangeRequest-7 DLDINSP	03/17/03	14:37:36		15:13:56	36.3
OEM-ChangeR	DLD	OEM-ChangeRequest-7 DLD	03/17/03	15:46:18		16:00:51	14.6
OEM-ChangeR	DLD	OEM-ChangeRequest-7 DLD	03/17/03	16:11:56		16:33:34	21.6
OEM-ChangeR	DLDR	OEM-ChangeRequest-7 DLDR	03/17/03	16:46:49		17:04:20	17.5
OEM-ChangeR	CODE	OEM-ChangeRequest-7 CODE	03/17/03	17:45:47		18:47:23	61.6
OEM-ChangeR	CODE	OEM-ChangeRequest-7 CODE	03/17/03	18:50:51		19:01:18	10.5
OEM-ChangeR	CODE	OEM-ChangeRequest-7 CODE	03/18/03	09:38:54		10:10:35	31.7
OEM-ChangeR	CR	OEM-ChangeRequest-7 CR	03/18/03	11:50:46		12:04:33	13.8
OEM-ChangeR	CR	OEM-ChangeRequest-7 CR	03/18/03	12:53:56		13:29:14	35.3

# Size

Size is a measure of the magnitude of the deliverable, e.g. lines of code or function points, pages.



TSP size measures are selected based on their correlation with time.

TSP also uses size data to

- normalize other measures
- track progress

TSP Size Summary - Form SUMS												
		Name Prasad Perini										
		Team PSP Ghost										
		Date 2/3/2004										
		Cycle				Actual Size						
ID	Assembly, Sub-Assembly, or Part Name	(A)ssembly or (P)art	Parent Assembly Name	Owner	Size Measure	Base	Deleted	Modified	Added	Reused	New and Changed	Total
25	DeliveryOEMPartValidate-Files	A	OEM MOO Integration RSM	PP	LOC	0	0	0	489	0	489	489
26	DeliveryOEMPartList(SQL)	A	OEM MOO Integration RSM	PP	LOC	0	0	0	613	0	613	613
27	AppDataExchangeCreate(SQL)	A	OEM MOO Integration RSM	PP	LOC	0	0	0	178	0	178	178
28	AppDataExchangeGet(SQL)	A	OEM MOO Integration RSM	PP	LOC	0	0	0	153	0	153	153
29	OEM MOO Integration RSM	A	SYSTEM	NK	Text Pages	0	0	0	4	0	4	4
30	Build Doc for OEM MOO Team	A	OEM MOO Integration RSM	NK	Text Pages	0	0	0	0	0	0	0
31	Build Script for OEM MOO Team	A	OEM MOO Integration RSM	NK	LOC	0	0	0	0	0	0	0



# Defects

Defects are the measure of quality in the TSP.

Any change to an interim or final work product, made to ensure proper design, implementation, test, use, or maintenance, is a defect in the TSP.



Defects are logged as they are found and fixed.

Defect tracking takes place throughout the process.

TSP Defect Recording Log - Form LOGD								
Name		Prasad Perini			Date		2/3/2004	
Team		PSP Ghost			Cycle			
Date	Num	Type	Assembly	Injected	Removed	Fix Time	Fix Ref.	Description
1/16/2003	66	20	OEM User Groups	CODE	CR	5.0		Missing ' ' between parameters
1/16/2003	67	70	OEM User Groups	CODE	CR	5.0		Permissions don't match for objects and its attrib
1/23/2003	68	70	OEM User Groups	DLD	CODEINSP	5.0		SRFile, SRProperty objects need create permission
1/23/2003	69	70	OEM User Groups	DLD	CODEINSP	10.0		Permissions don't match for objects and its attrib
1/23/2003	70	70	OEM User Groups	CODE	CODEINSP	2.0		211-212 Wrong Sproc (iGrpApp should be iCode)
1/24/2003	71	70	OEM User Groups	CODE	UT	25.0		Wrong Database Name for UserAccount Object
1/24/2003	72	70	OEM User Groups	DLD	UT	3.0		Extra Attribute name in User.Account ObjectAttribu
1/24/2003	73	90	AppDataExchangeG	DLD	DLDR	1.0		Granted permissions to OEMUsers instead of Phoc
1/24/2003	74	40	AppDataExchangeG	DLD	DLDR	5.0		Step names in Logic don't match with error table
1/24/2003	75	40	AppDataExchangeG	DLD	DLDR	1.0		Change record to IsActive in step 2
1/24/2003	76	70	AppDataExchangeG	DLD	DLDR	1.0		Column names were not specified in step 4
1/24/2003	77	90	AppDataExchangeG	DLD	DLDR	1.0		Every condition was not specified after update



# What the Measurement Framework Provides...

---

## Sample of Derived Measures

Estimation accuracy (size/time)  
Prediction intervals (size/time)  
Time in phase distribution  
Defect injection phase distribution  
Defect removal phase distribution  
Productivity  
%Reuse  
%New Reusable  
Cost performance index  
Planned value  
Earned value  
Predicted earned value  
Defect density

## Sample of Derived Measures (continued)

Defect density by phase  
Defect removal rate by phase  
Defect removal leverage  
Review rates  
Process yield  
Phase yield  
Failure cost of quality  
Appraisal cost of quality  
Appraisal/Failure COQ ratio  
Percent defect free  
Defect removal profiles  
Quality profile  
Quality profile index

# Quality Summary -1

TSP form SUMQ displays key plan and actual quality data for the entire project or any module.

- Percent Defect Free
- Defect Density/Page
- Defect Density/KLOC
- Defect Ratios

TSP Quality Plan - Form SUMQ		
Name	Carol	
Team	PSP Ghost	
Assembly	SYSTEM	
<b>SYSTEM Percent Defect Free</b>		
In Compile		<b>Actual</b> 62.3%
In Unit Test		53.6%
In Build and Integration Test		73.9%
In System Test		98.6%
In Acceptance Test		100.0%
In Product Life		100.0%
<b>Defects/Page</b>		
	<b>Plan</b>	<b>Actual</b>
REQ Inspection	0.00	0.00
HLD Inspection	0.00	0.00
<b>Defects/KLOC</b>		
	<b>Plan</b>	<b>Actual</b>
DLD Review	8.33	9.36
DLD Inspection	4.77	6.54
Code Review	19.19	10.33
Compile	4.92	3.43
Code Inspection	4.21	5.79
Unit Test	1.20	4.25
Build and Integration Test	0.48	0.89
System Test	0.36	0.04
Total Development	43.75	41.10
Acceptance Test	0.00	0.00
Product Life	0.36	0.00
Total	44.11	41.10
<b>Defect Ratios</b>		
	<b>Plan</b>	<b>Actual</b>
DLD Review/Unit Test	6.94	2.20
Code Review/Compile	3.90	3.01



# Quality Summary -2

TSP form SUMQ displays key plan and actual quality data for the entire project or any module.

- Development Time Ratios
- Inspection and Review Rates
- A/FR (Cost of Quality Ratio)
- Phase Yields

TSP Quality Plan - Form SUMQ		
Name	Carol	
Team	PSP Ghost	
Assembly	SYSTEM	
<b>Development Time Ratios</b>	<b>Plan</b>	<b>Actual</b>
REQ Inspection/Requirements	0.00	0.00
HLD Inspection/High-Level Design	0.02	0.03
Detailed Design/Code	1.09	1.57
DLD Review/Detailed Design	0.45	0.32
Code Review/Code	0.46	0.41
<b>Inspection/Review Rates</b>	<b>Plan</b>	<b>Actual</b>
REQ Inspection	0.00	0.00
HLD Inspection	0.00	0.00
DLD Review	148.45	225.53
DLD Inspection	111.08	143.76
Code Review	156.86	272.99
Code Inspection	114.69	132.31
<b>A/FR</b>	<b>2.82</b>	<b>3.49</b>
<b>Phase Yields</b>	<b>Plan</b>	<b>Actual</b>
Planning	0%	0%
Requirements	0%	0%
System Test Plan	0%	0%
REQ Inspection	50%	0%
High-Level Design	0%	0%
Integration Test Plan	0%	0%
HLD Inspection	60%	0%
Detailed Design	0%	0%
DLD Review	55%	47%
Test Development	0%	0%
DLD Inspection	70%	62%

# Quality Summary -3

TSP form SUMQ displays key plan and actual quality data for the entire project or any module.

- Process Yields
- Defect Injection Rates
- Defect Removal Rates

TSP Quality Plan - Form SUMQ		
Name	Carol	
Team	PSP Ghost	
Assembly	SYSTEM	
<b>Process Yields</b>	<b>Plan</b>	<b>Actual</b>
% Before Compile	76%	65%
% Before Unit Test	96%	88%
% Before Build and Integration Test	97%	98%
% Before System Test	98%	100%
% Before Acceptance Test	99%	100%
<b>Defect Injection Rates (Defects Injected Per Hour)</b>	<b>Plan</b>	<b>Actual</b>
Planning	0	0.05
Requirements	0.25	0.00
System Test Plan	0	0.00
REQ Inspection	0	0.00
High-Level Design	0.25	0.38
Integration Test Plan	0	0.00
HLD Inspection	0	0.93
Detailed Design	1	1.35
DLD Review	0	0.03
Test Development	0	0.00
DLD Inspection	0	0.01
Code	2	2.36
Code Review	0	0.01
Compile	0.3	0.00
Code Inspection	0	0.00
Unit Test	0.067	0.00
Build and Integration Test	0	0.06
System Test	0	0.00
<b>Defect Removal Rates</b>	<b>Plan</b>	<b>Actual</b>
Planning	0.00	0.00
Requirements	0.00	0.00



# QA Quality Review in the TSP

---

QA is a stakeholder in these quality reviews:

- Launch process management review meeting
- Weekly meetings
- Inspections
- Cycle, phase, and project postmortems
- Management/customer status meetings



# Leading Indicators

---

TSP has many leading indicators for managing software quality.

- planning for poor quality indicators
- process and measurement quality indicators
- product quality indicators

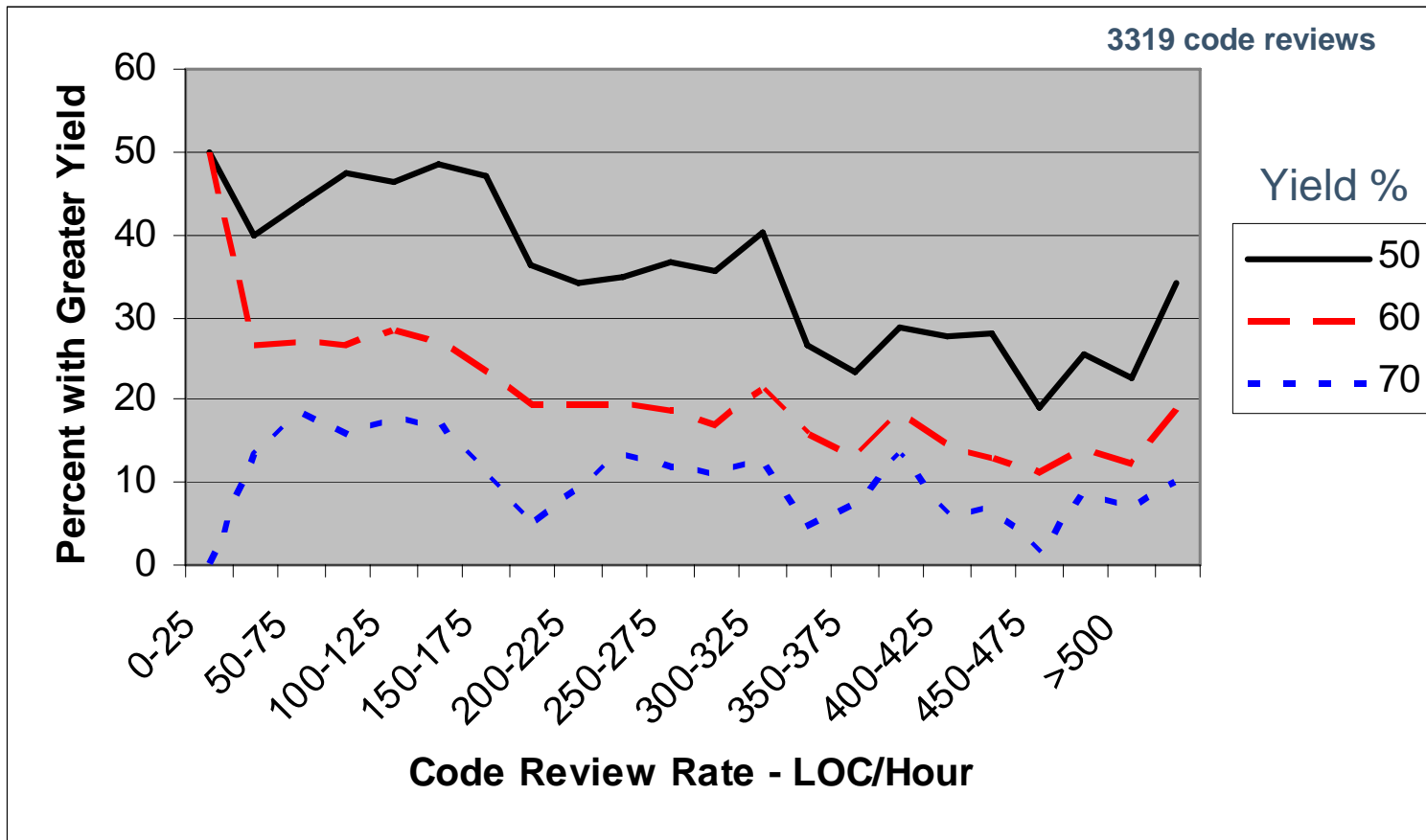
The following material gives examples of plan and actual

- review and inspection rates.
- development time ratios.
- Quality Profiles and the Process Quality Indices.



# Review and Inspection Rates

Review and inspection rates are generally correlated to yield.



# Development Time Ratios

Development time ratios compare time spent in related activities and correlate these ratios with low test defect densities.

## Examples

- designing and coding
- designing and design reviews
- coding and code reviews

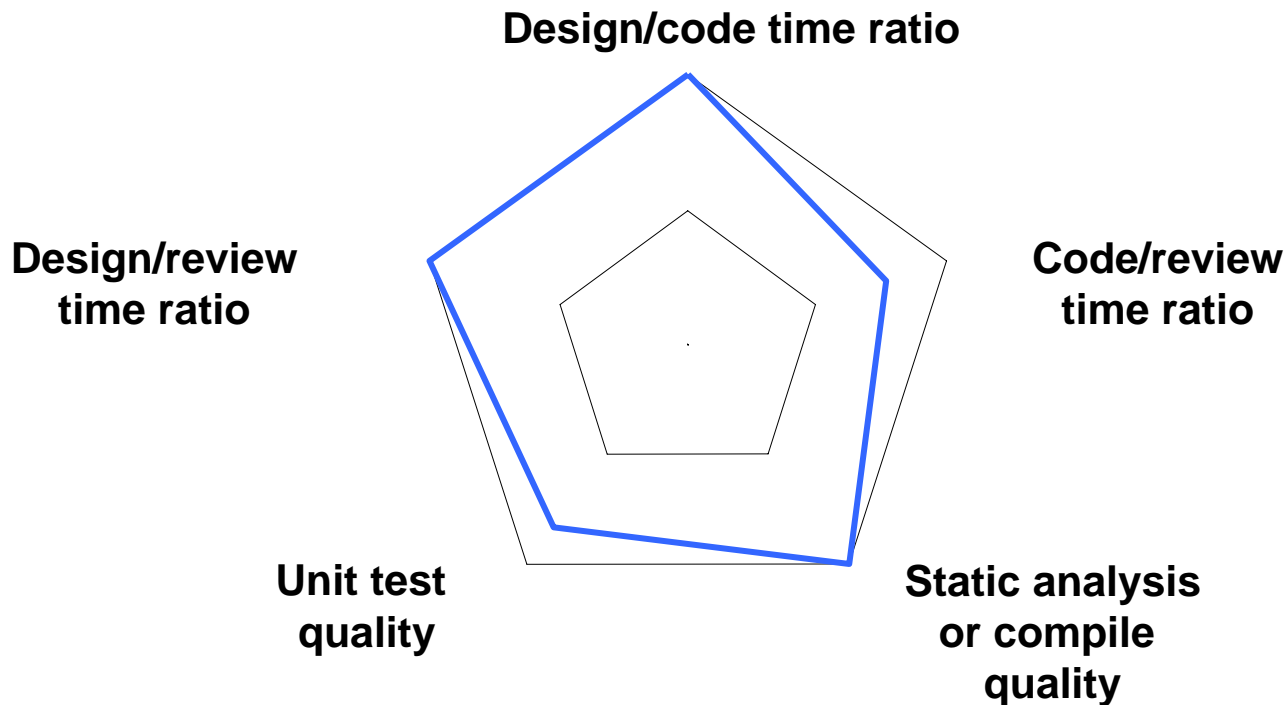
Indicator	Value
Design defect injection rate	0.75/Hr.
Design review defect removal rate	1.5/Hr.
Design/Design Review time ratio	2
Coding defect injection rate	2/Hr.
Code review defect removal rate	4/Hr.
Code/Code Review time ratio	2
Design/Code time ratio (derived from TSP data)	1



# Component Quality Profile

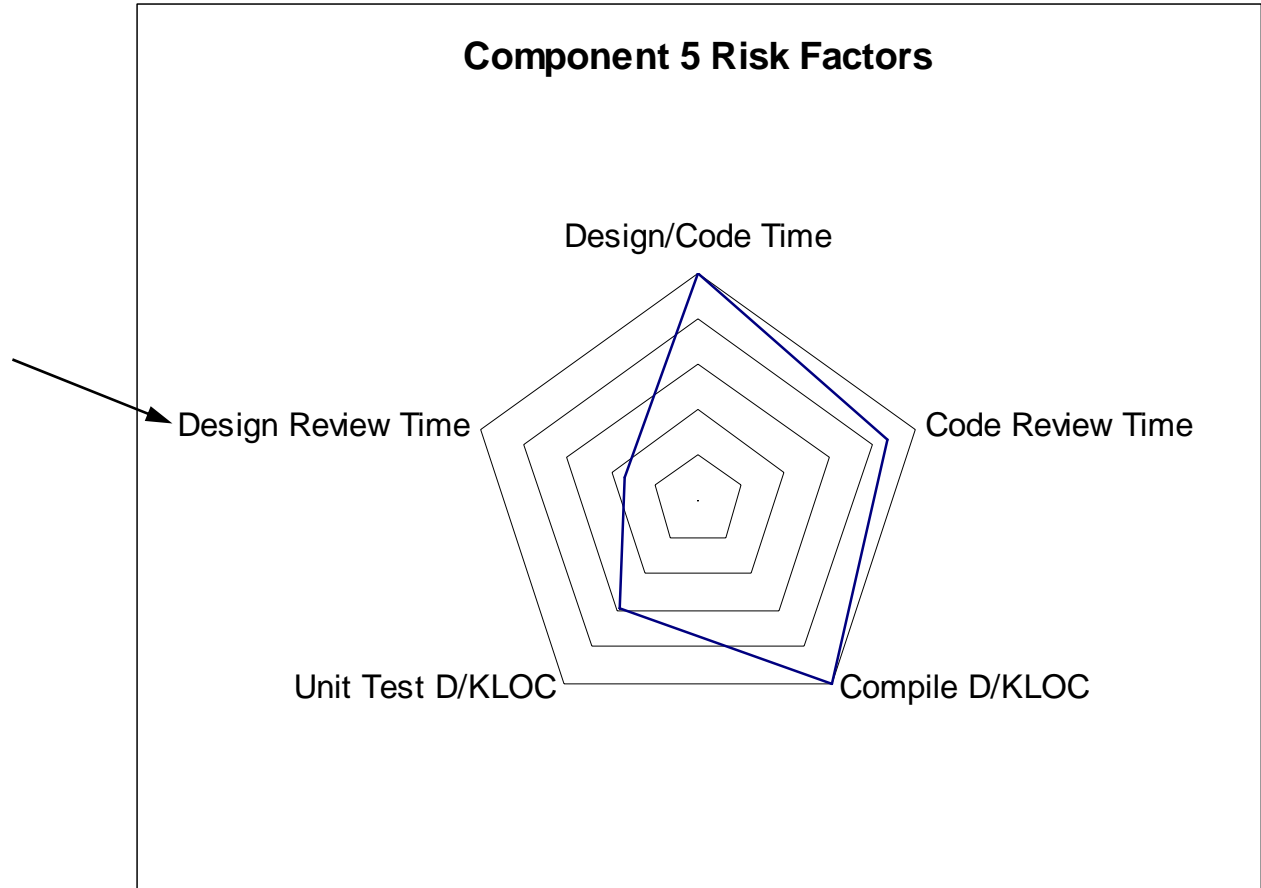
---

The component quality profile is an early warning indicator consisting of five risk factors that indicate the potential for post unit test defects.



# Interpreting the Component Quality Profile

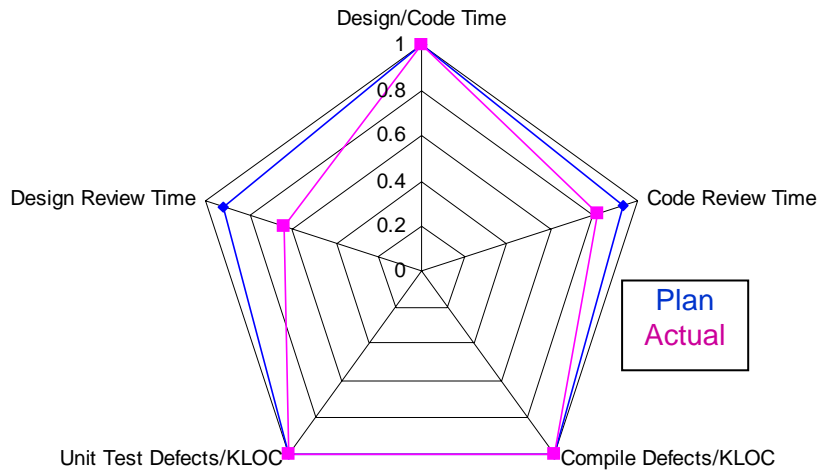
Inadequate design review time results in design defects escaping to test and production.



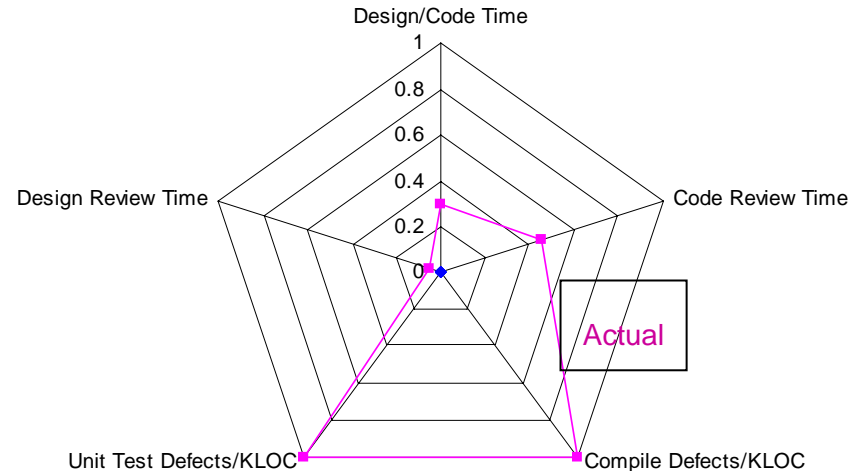
# System Quality Profile Example

The Quality Profile may be applied to the entire system or any part.

In this example, only 14 defects were found during system and user acceptance testing out of 1336 defects found in 28 KSLOC.



System Quality Profile



Previous Release System Quality Profile



# The Process Quality Index

---

The process quality index (PQI) provides a quality figure of merit for every system element.

To calculate PQI, multiply the profile dimensions to produce a composite value that considers

- compile and unit test defect levels
- design and code review times
- time spent in design

Before test entry, PQI indicates the likelihood that a system element will have subsequent defects.

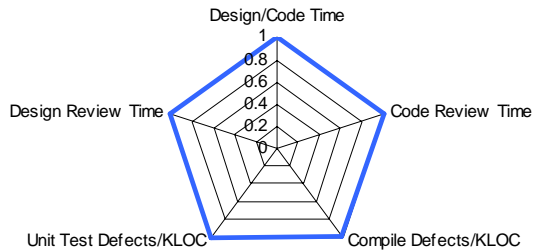
Values above 0.4 are considered to be good.





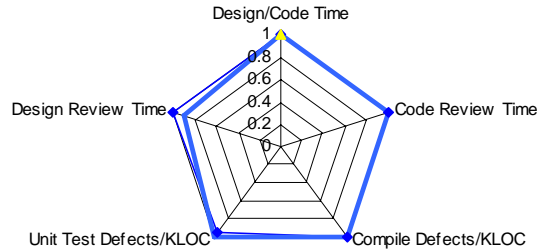
# QA and the TSP Quality Indicators

Quality Profile for Assembly 1



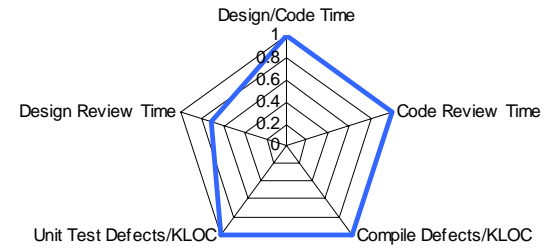
PQI = 0.97

Quality Profile for Assembly 2



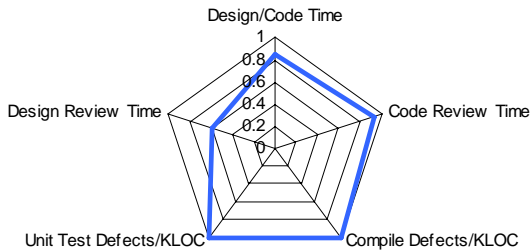
PQI = 0.88

Quality Profile for Assembly 3



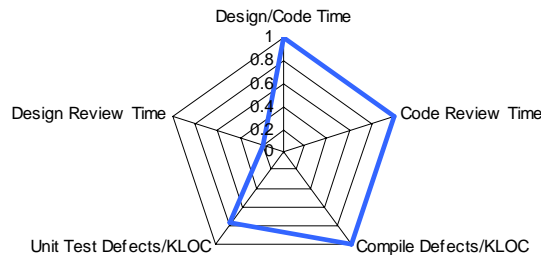
PQI = 0.71

Quality Profile for Assembly 4



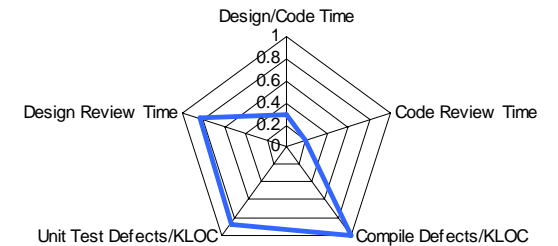
PQI = 0.59

Quality Profile for Assembly 5



PQI = 0.15

Quality Profile for Assembly 6



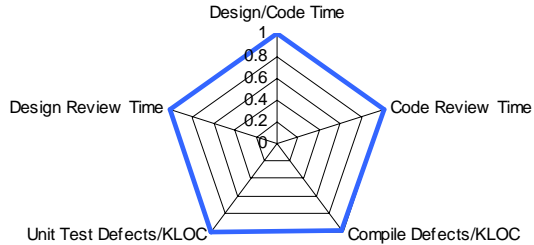
PQI = 0.04



# Defects Found in System Test by QA

Quality Profile for Assembly 1

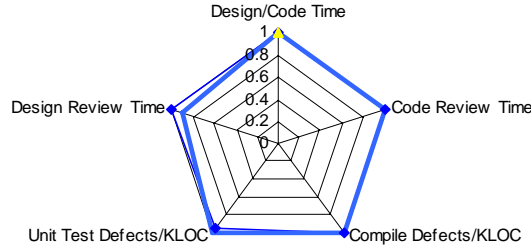
Test defects = 0



PQI = 0.97

Quality Profile for Assembly 2

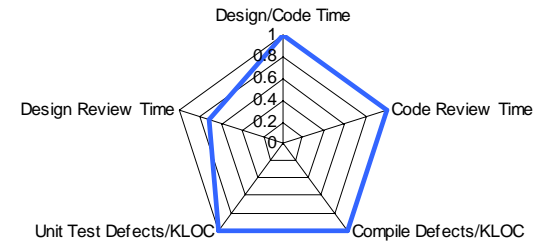
Test defects = 0



PQI = 0.88

Quality Profile for Assembly 3

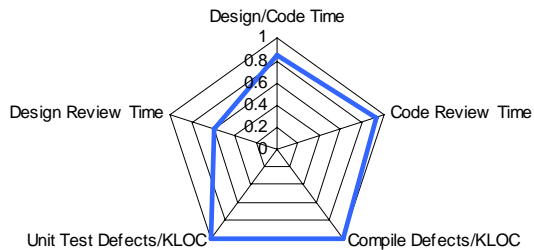
Test defects = 0



PQI = 0.71

Quality Profile for Assembly 4

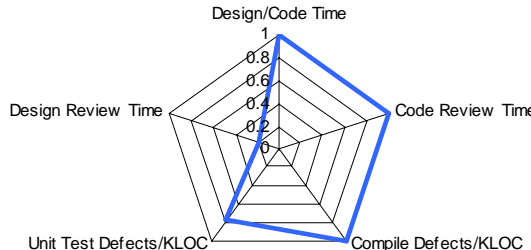
Test defects = 0



PQI = 0.59

Quality Profile for Assembly 5

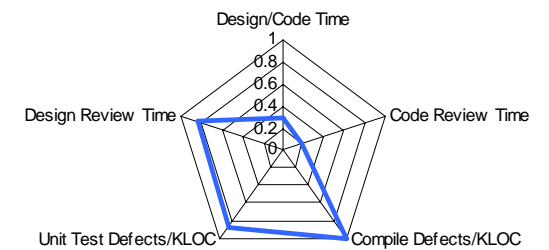
Test defects = 1



PQI = 0.15

Quality Profile for Assembly 6

Test defects = 3



PQI = 0.04



# Summary

---

We can no longer rely on testing as the principal means of improving the quality of software systems.

To get a quality product out of test, we must

- establish a quality ethic at the individual level
- plan for and measure quality at each step
- use disciplined processes that emphasize early defect removal

The role of QA must change from a “test-in quality” focus to a “build-in quality” focus where quality plans and data are used by QA and the development teams to manage quality throughout the process.



---

# Questions?

For more information contact:

Jim Over

+ 1 412-268-7624

[jwo@sei.cmu.edu](mailto:jwo@sei.cmu.edu)

