TSP Teams Transform Test

2009 QUEST Chicago

James D. McHale
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

jdm@sei.cmu.edu

A Nightmare Scenario

The development team has been promising their “final” code drop for weeks, ever since their latest due date (months after the original date).

Your test team has been filling in with clean-up, make-work tasks for most of that time.

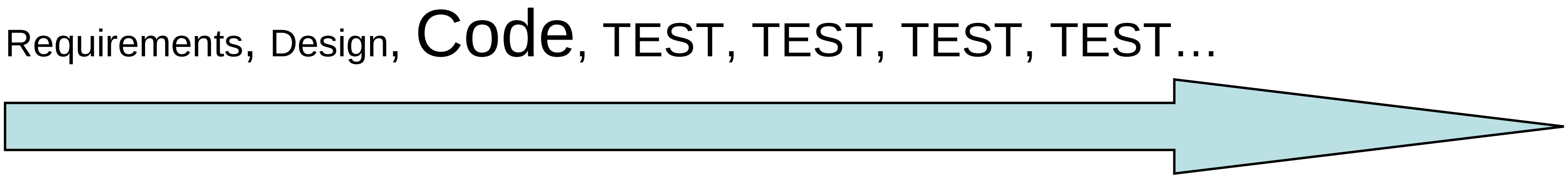
Every day represents one less day for your test team to complete their work, since the final ship date is not moving.

Finally an email arrives handing off the code. All the proper links to the configuration management system are there…

But they point to software that doesn’t build. A few more days pass.

Another email. The software builds this time. The first day of testing cannot complete due to irrecoverable defects… and management wants *you* to commit to the original ship date.

Current Test Reality



Requirements, Design, Code, TEST, TEST, TEST, TEST…

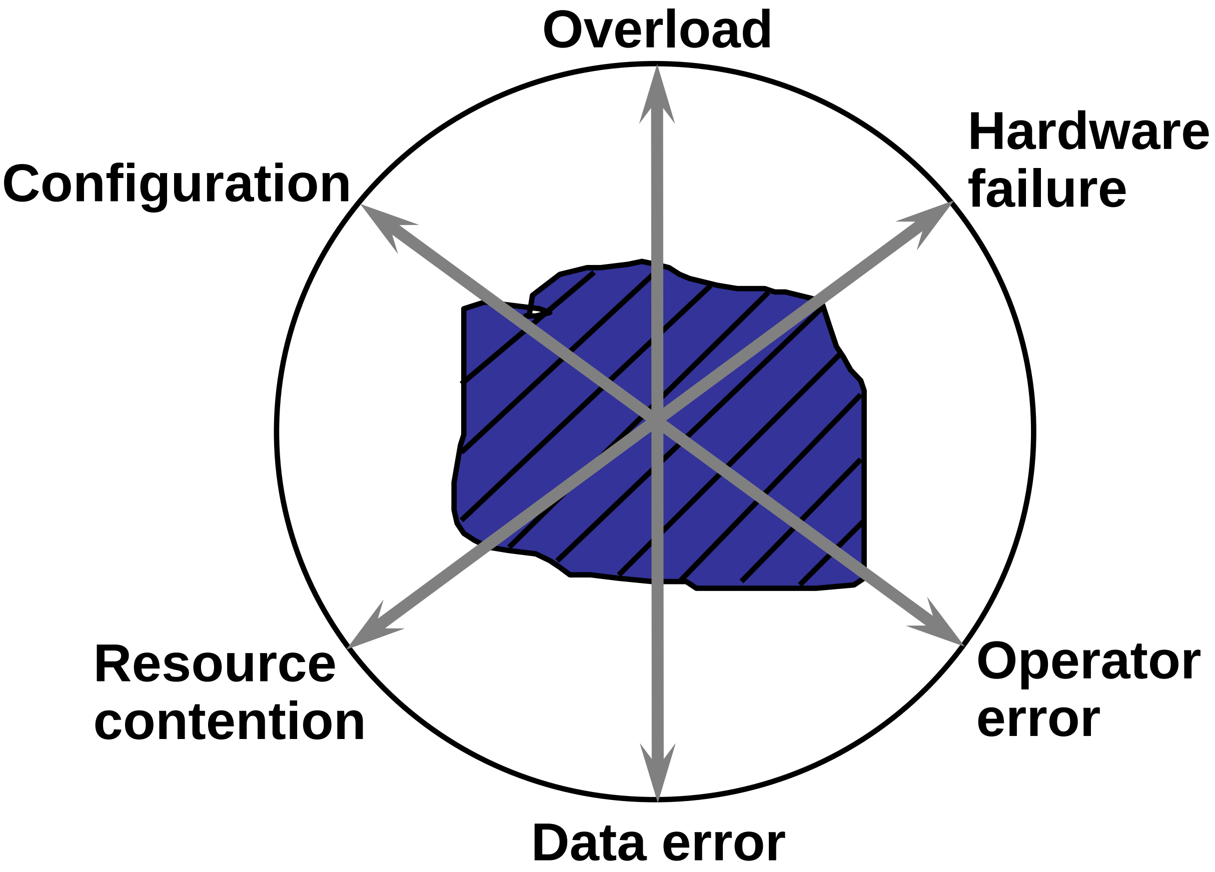
The focus is on writing code and getting it into test.

Most working software developers learned this model implicitly, either in school or on the job, and have no other mental model available.

Counting ALL of the testing costs for most modern-day applications *still* shows that 50% or more of development effort is spent in debugging.

Is the purpose of testing to find defects? Or to verify function?

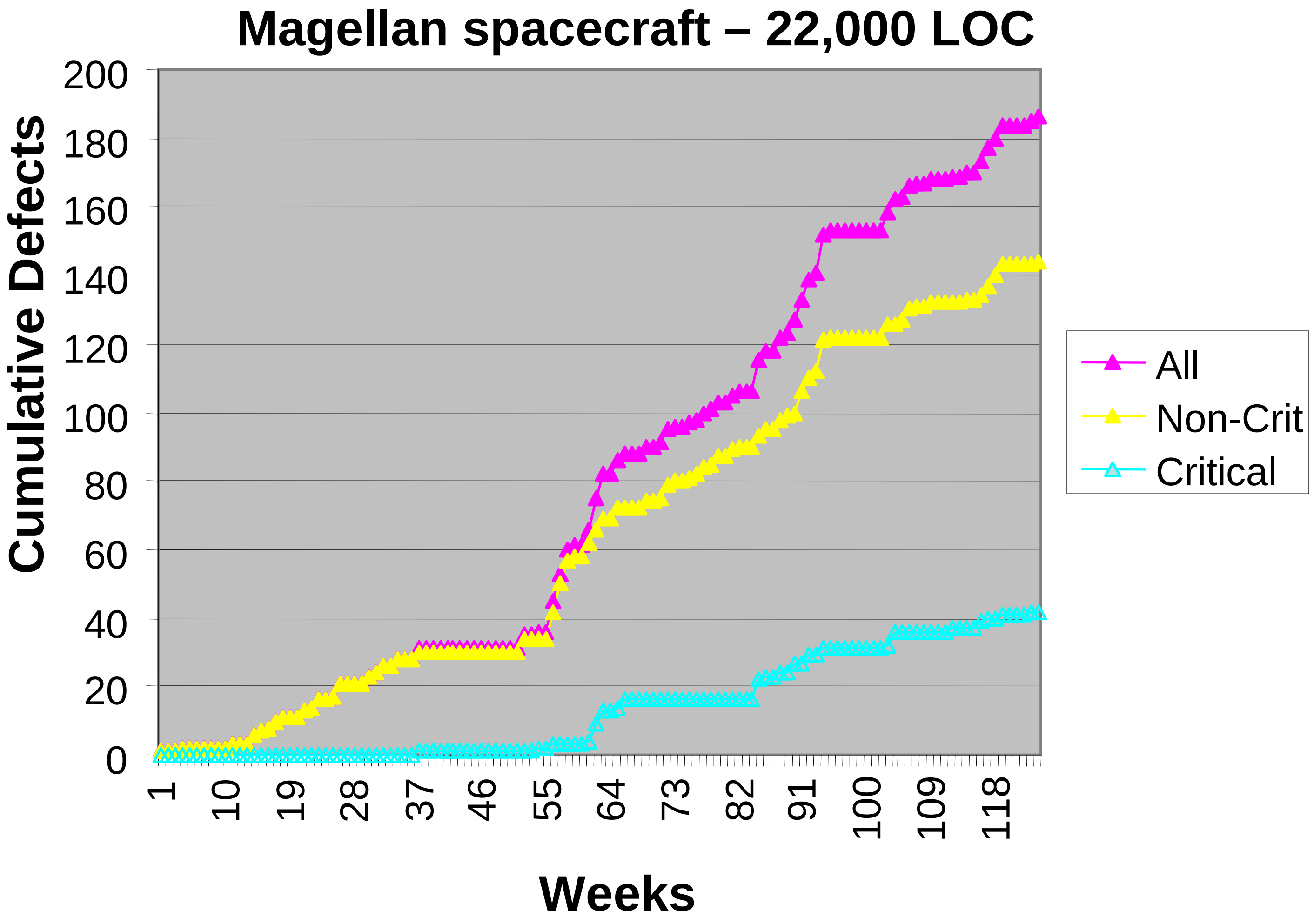
The Problem With Testing



Good region = tested (shaded)

Unknown region = untested (unshaded)

Testing Takes a Long Time



Answers For Testing?

Testing is necessary (and not a necessary evil).

However there are few good answers for test teams in an organization whose management allows poor quality software into test.

- Typical development practices rely on test to find all but the simplest defects.
- Incentives to “get into test” are unwittingly counter-productive.
- With very buggy code, it is sometimes difficult to separate development defects from testing defects.

Testing better and faster only encourages the root problem.

Where Do Defects Come From?

While a relatively small number of expensive defects are inherent in requirements (often outside the control of development), most defects are injected by the developers.

For a small program (100 – 200 LOC), it may be only 5 or 10 defects, and many of those are caught by modern IDEs. (This is the mode in which almost everyone learns to program, regardless of academic discipline.)

For a larger program (1000 – 2000 LOC), 50 to 100 defects is still tractable given those modern IDEs.

For a small system (10,000 – 100,000 LOC), 500 to 5,000 defects cause long nights, lost weekends, and death march projects.

For large systems (>1 million LOC), 50,000 defects per MLOC (even with 80% caught "automatically") translates to worst-nightmare scenarios.

What Can Be Done About Defects?

Test teams cannot prevent the “import” of defects that development teams “export”, they can only highlight the problem.

Development teams cannot today buy tools that reduce the number of defects going to test by a factor of 10.

But… a different attitude on the part of developers concerning quality CAN reduce defects going to test by that factor of 10, or better.

How can this be achieved?

Personal Software Process (PSP)

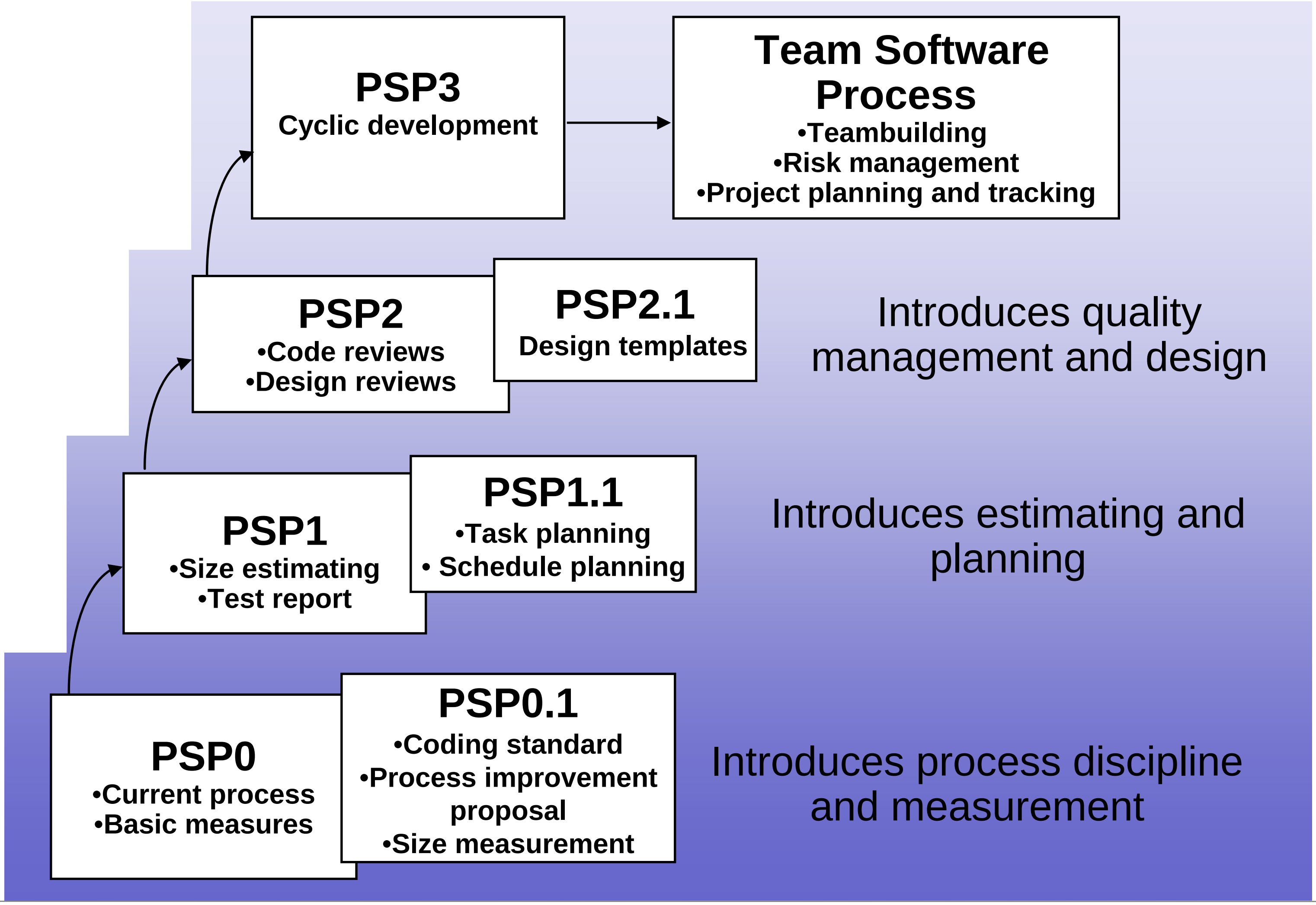
The Personal Software Process (PSP)

- developed by Watts Humphrey at the SEI in the early 1990s
- an interlocking set of measurement, quality, and planning frameworks
- applicable to the smallest project (one person)

PSP is taught incrementally through a series of 7 or 8 programming assignments, typically two one-week blocks about a month apart.

One version of the training aims at starting to use a subset of these skills *immediately* following the first week.

Basic PSP Skills



PSP3
Cyclic development

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

PSP2
- Code reviews
- Design reviews

PSP2.1
Design templates

Introduces quality management and design

PSP1
- Size estimating
- Test report

PSP1.1
- Task planning
- Schedule planning

Introduces estimating and planning

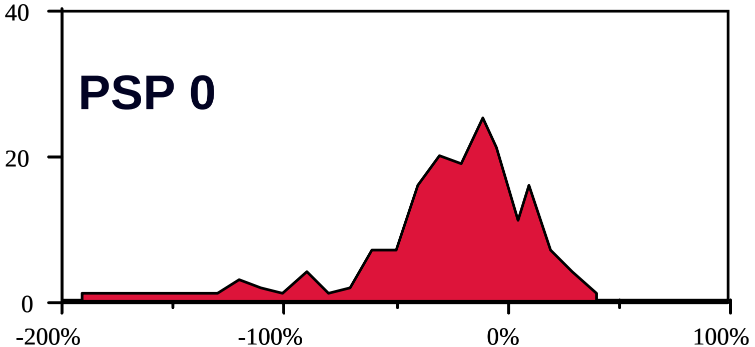
PSP0
- Current process
- Basic measures

PSP0.1
- Coding standard
- Process improvement proposal
- Size measurement

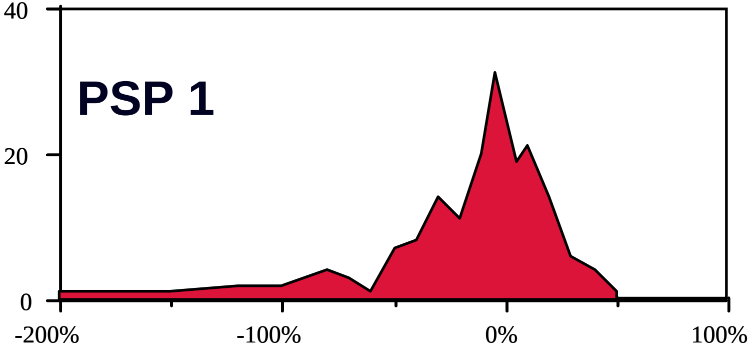
Introduces process discipline and measurement

PSP Estimating Accuracy

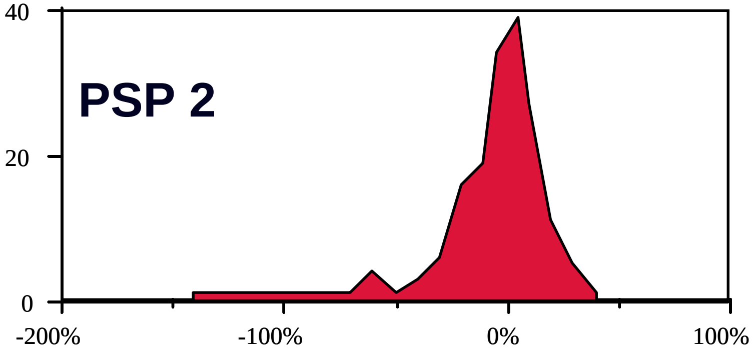
Majority are underestimating



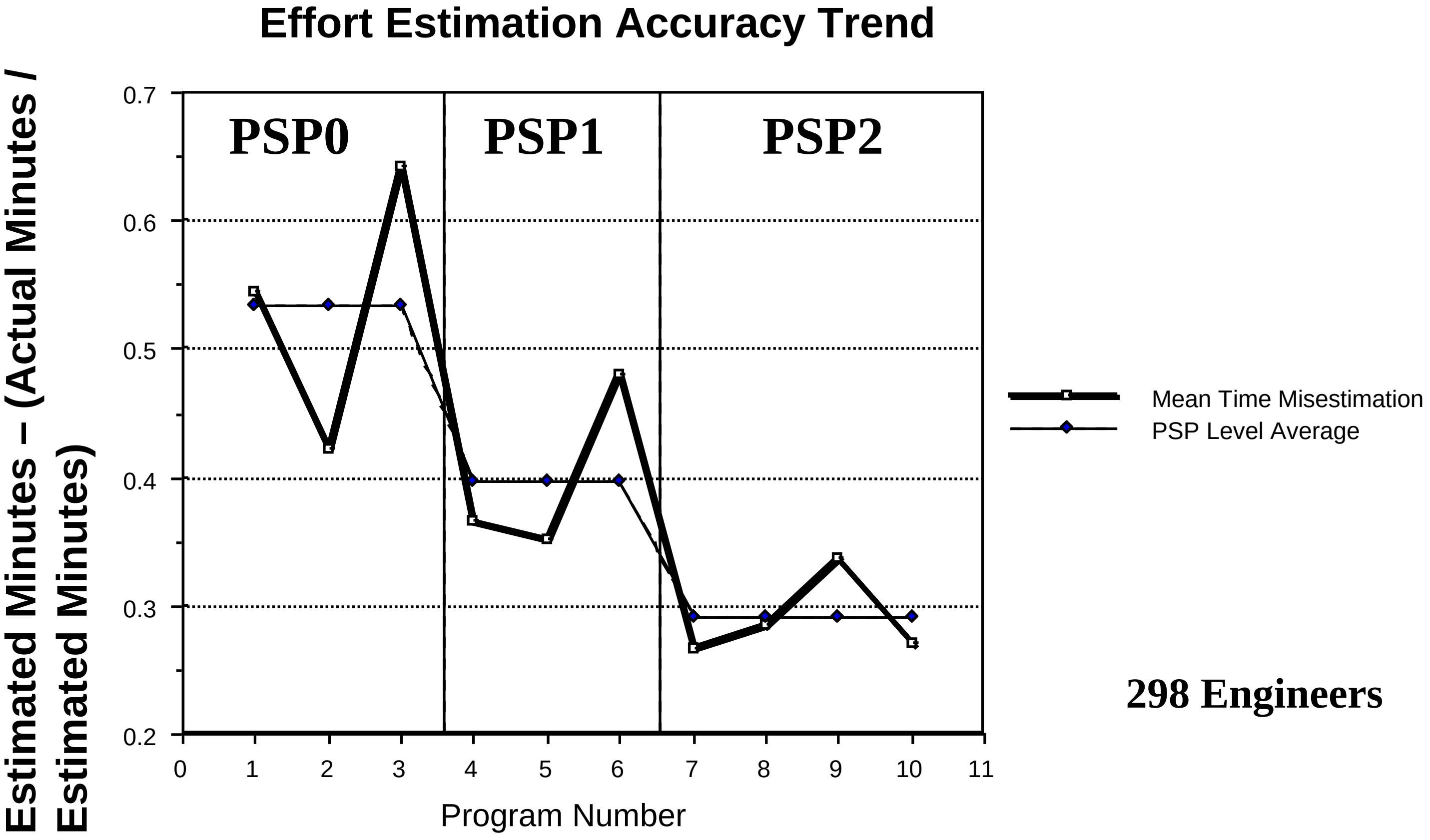
Balance of over- and under-estimates



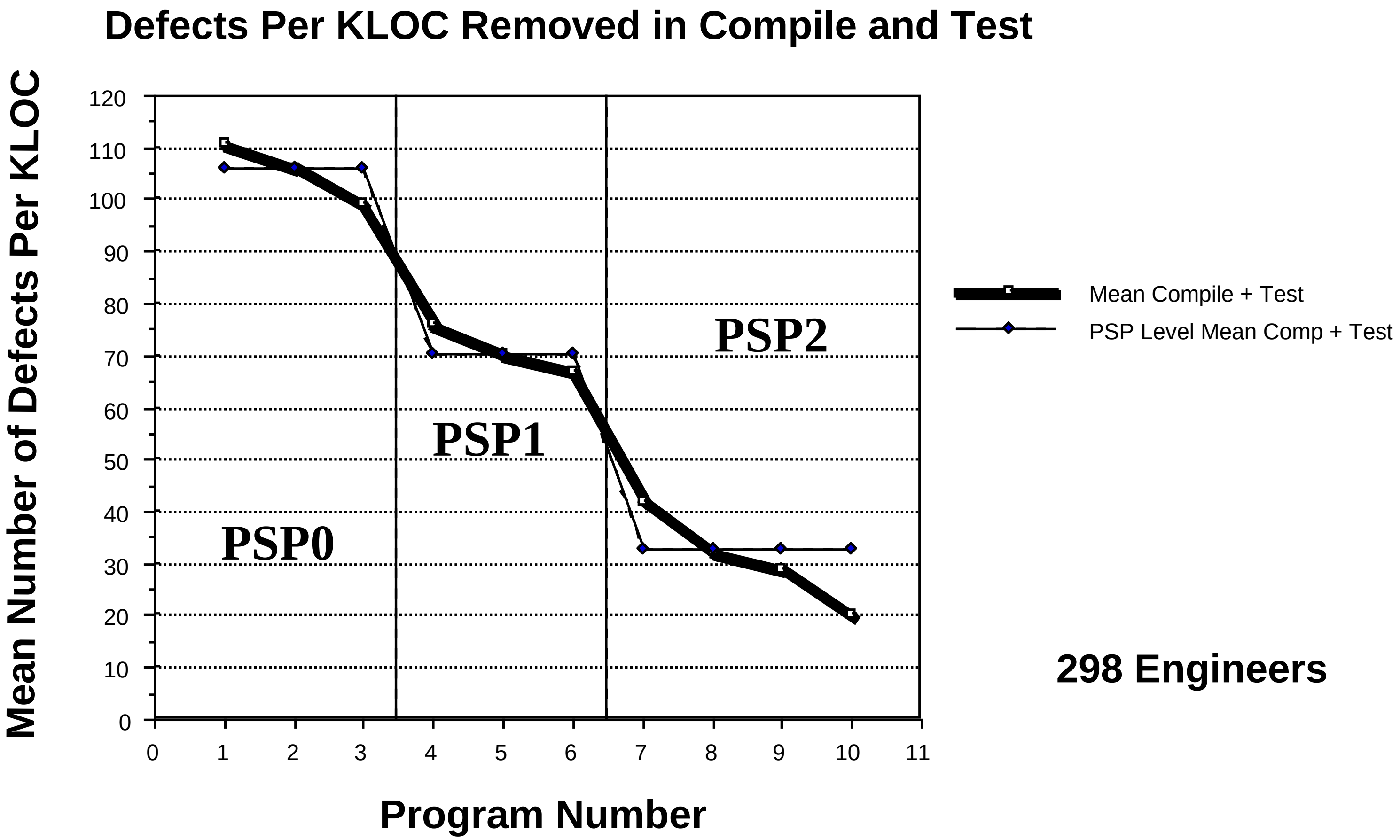
Much tighter balance around zero error



Improving Estimating Accuracy

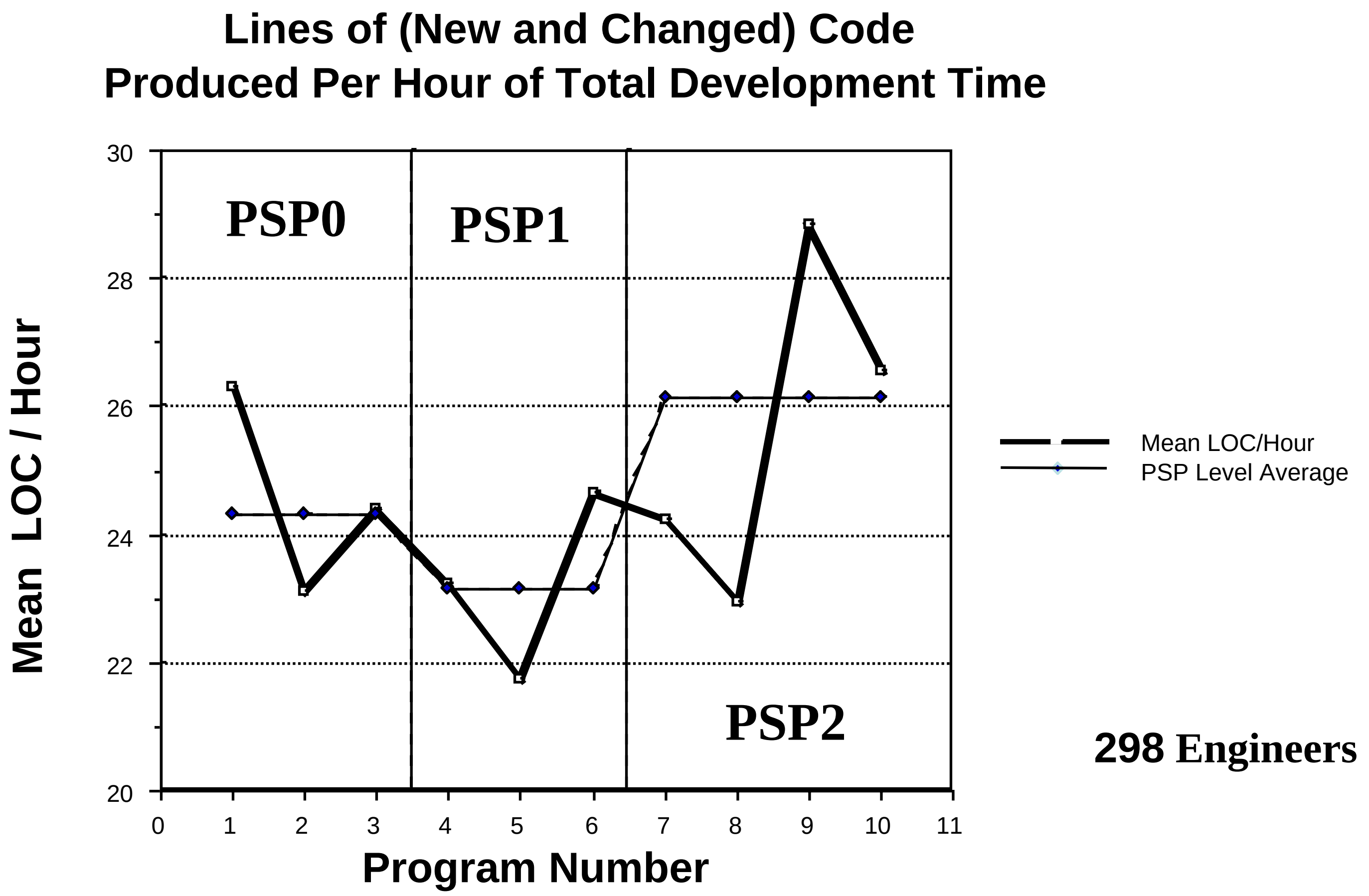


PSP Quality Results



298 Engineers

PSP Productivity Results



Problems With PSP

In the classroom, PSP works brilliantly.

In practice, PSP works only in a supportive environment.

That environment must acknowledge

- that the individual’s contribution to quality is critical
- that the people doing the work must have the proper quality training
- that the project team is the minimum support structure for such work
- “Quality without numbers is just talk.” – *Watts Humphrey*

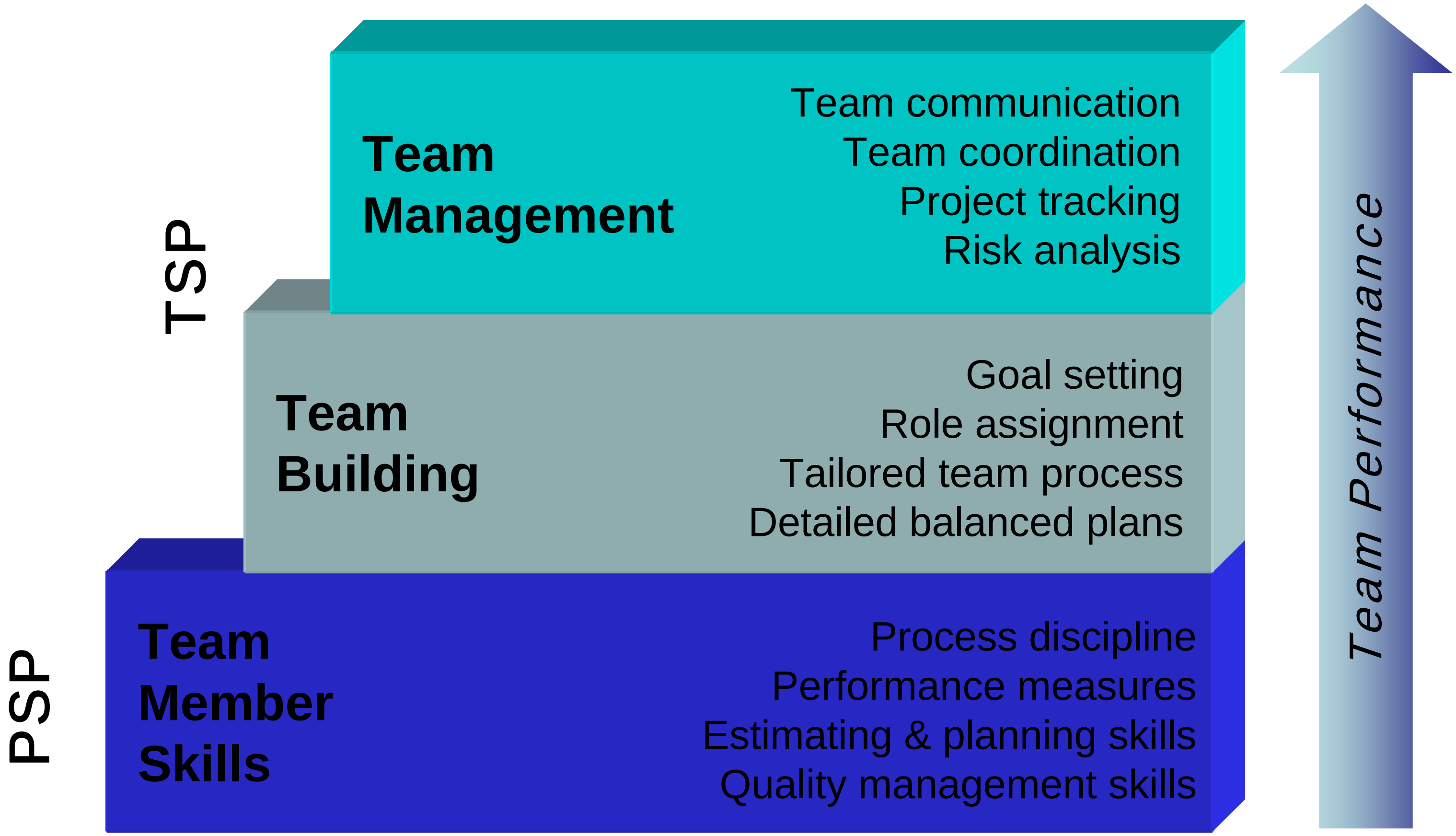
Team Software Process (TSP)

The Team Software Process (TSP) was developed in the late 1990s to address the problems seen in early field use of the PSP.

TSP applies the PSP measurement, quality, and planning framework at the level of a project team, adding

- a team-owned project management method that generates extremely accurate and reliable tracking and exception information
- a team commitment discipline that involves each individual in defining and accepting the group's responsibilities, not just for timely delivery but also for the quality of the final product
- a team-working structure based on pre-defined roles (similar to the positions on a sports team or the sections of an orchestra)

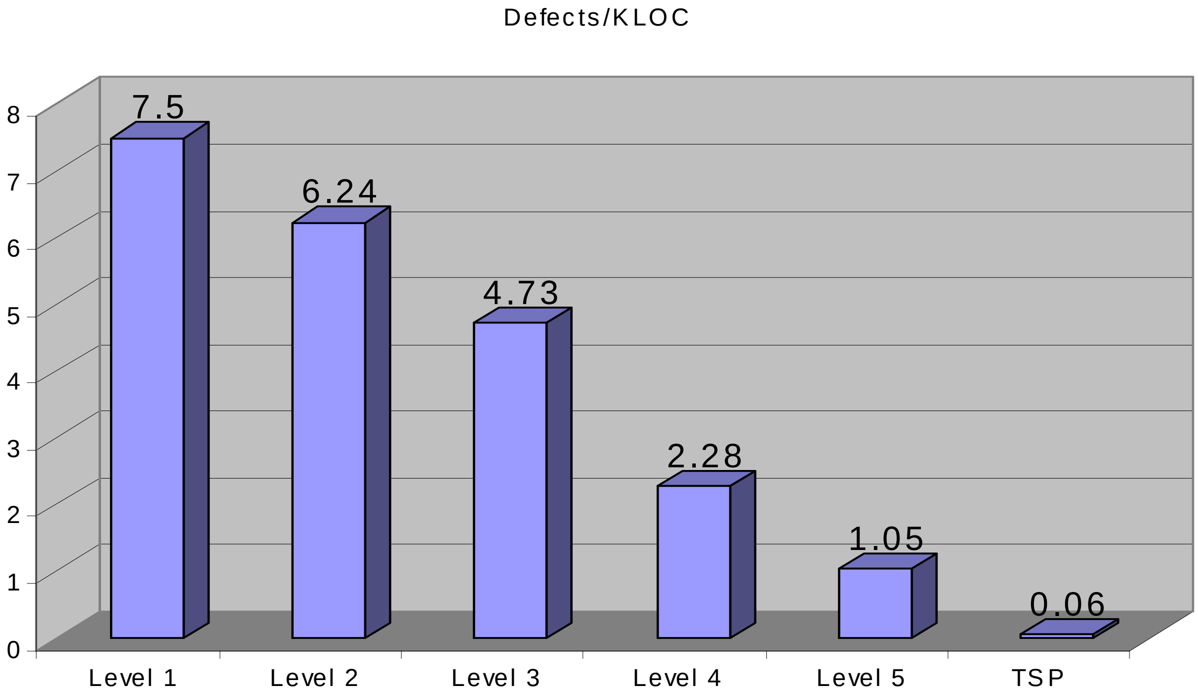
Building High-Performance Teams



Improved Quality (Delivered Defect Density)

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

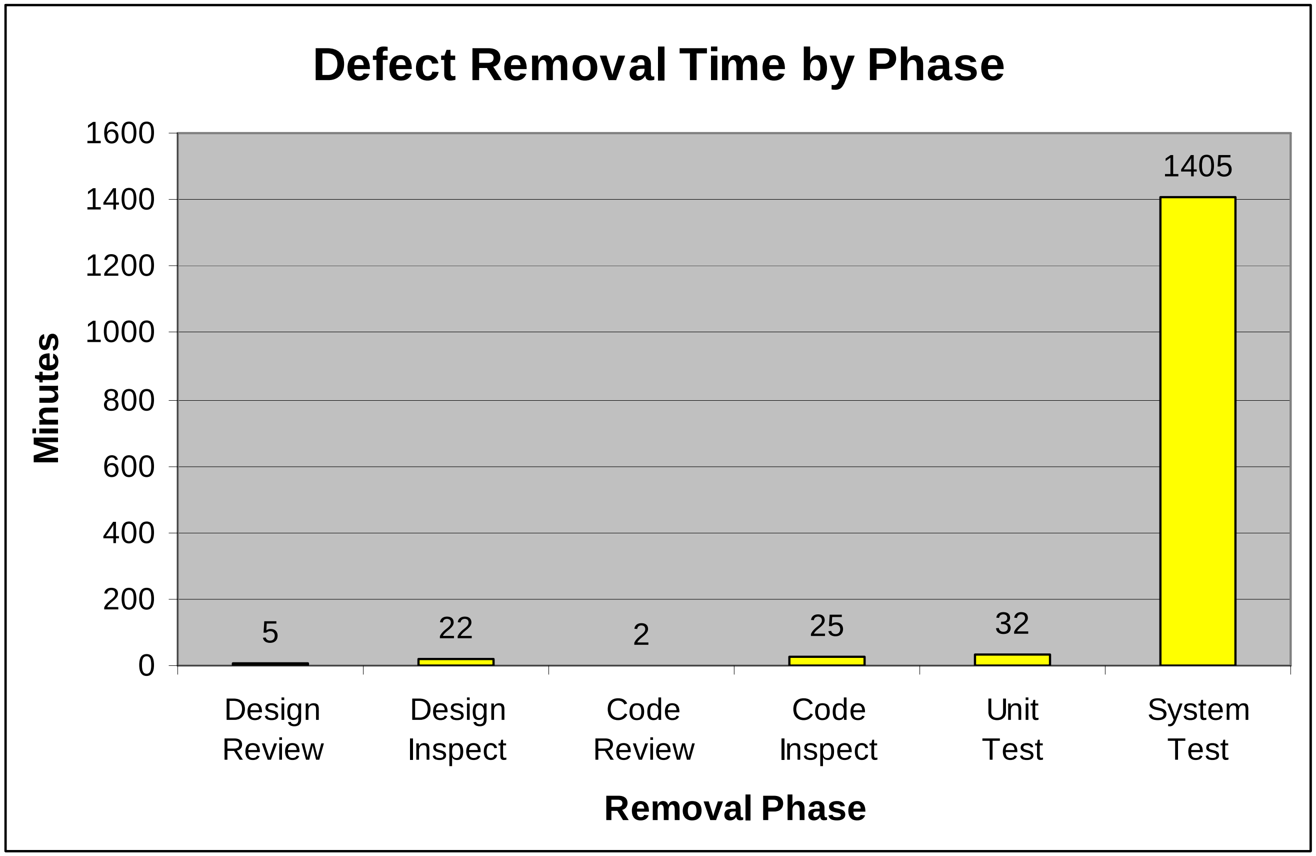
Approximately 1/3 of these projects were defect-free.



Source: CMU/SEI-2003-TR-014

Reviews and Inspections Save Time

Xerox found that TSP quality management practices reduced the cost of poor quality by finding and removing defects earlier when costs are lower.

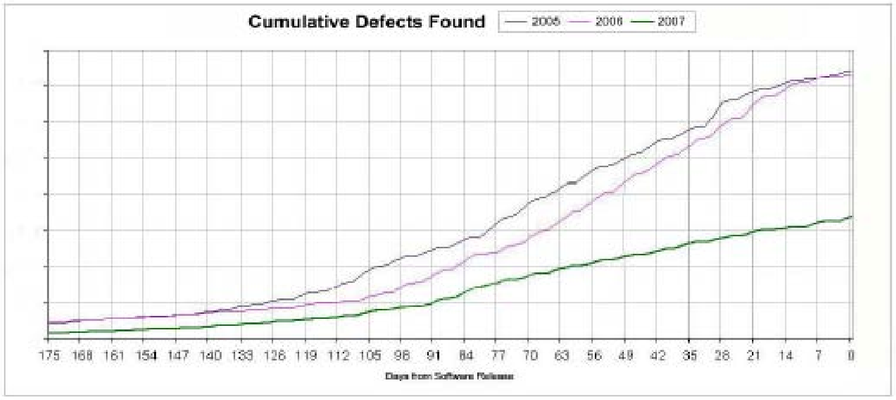


Intuit Quality Improvement

TSP reduced defects found in system test by 60% over the previous two releases of QuickBooks 2007.

Intuit has also recently reported a savings of $20M from a reduction in customer support calls on QuickBooks 2007.

Results at Intuit: Improved Quality

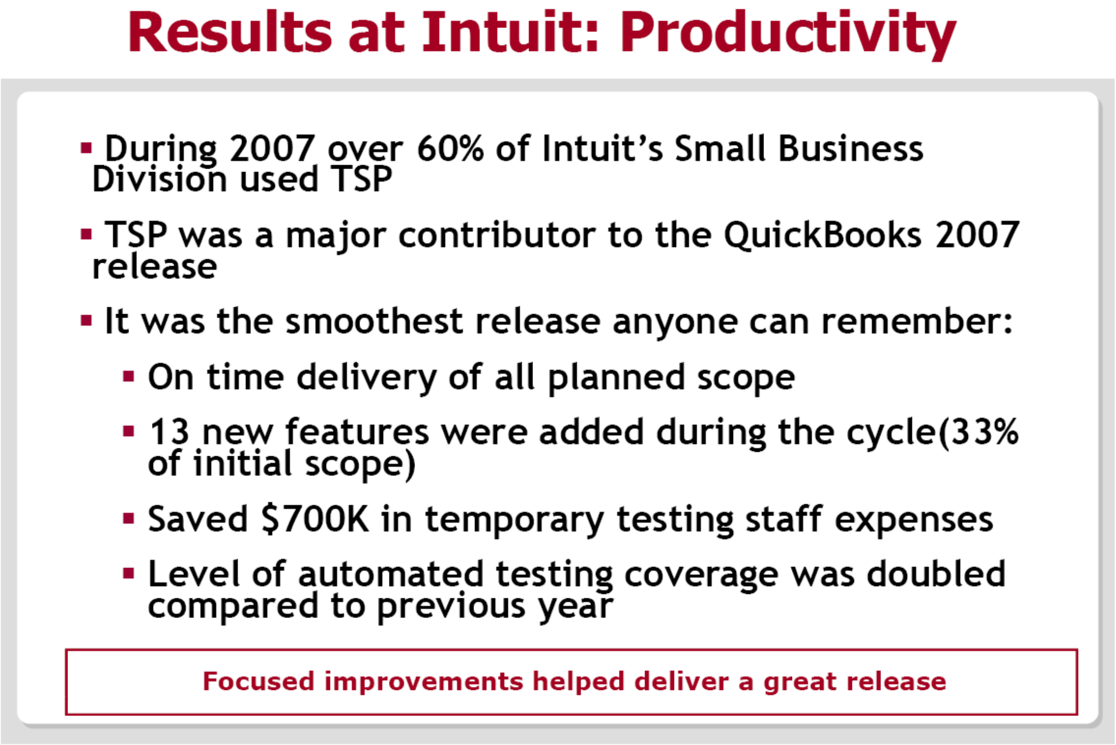


In 2007 ~60% fewer defects were found
in System Test than the previous two releases

Source: Intuit

Intuit Productivity Improvement

By putting a quality product into system test Intuit improved productivity and reduced cost while delivering 33% more functionality than planned.



Results at Intuit: Productivity

- During 2007 over 60% of Intuit's Small Business Division used TSP
- TSP was a major contributor to the QuickBooks 2007 release
- It was the smoothest release anyone can remember:
  - On time delivery of all planned scope
  - 13 new features were added during the cycle(33% of initial scope)
  - Saved $700K in temporary testing staff expenses
  - Level of automated testing coverage was doubled compared to previous year

Focused improvements helped deliver a great release

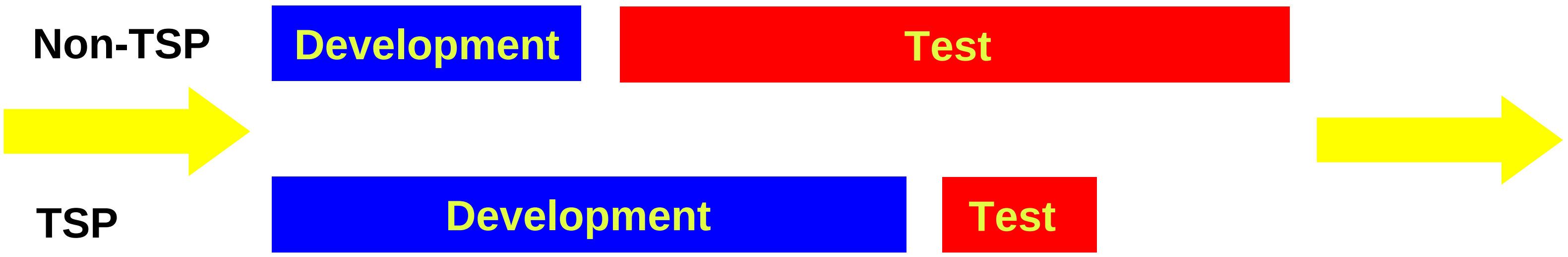
Source: Intuit

Intuit Test Schedule Reduction

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

- post code-complete effort is 8% instead of 33% of the project
- for TSP projects, standard test times are cut from 4 months to 1 week

Testing time is reduced from four months to one month.

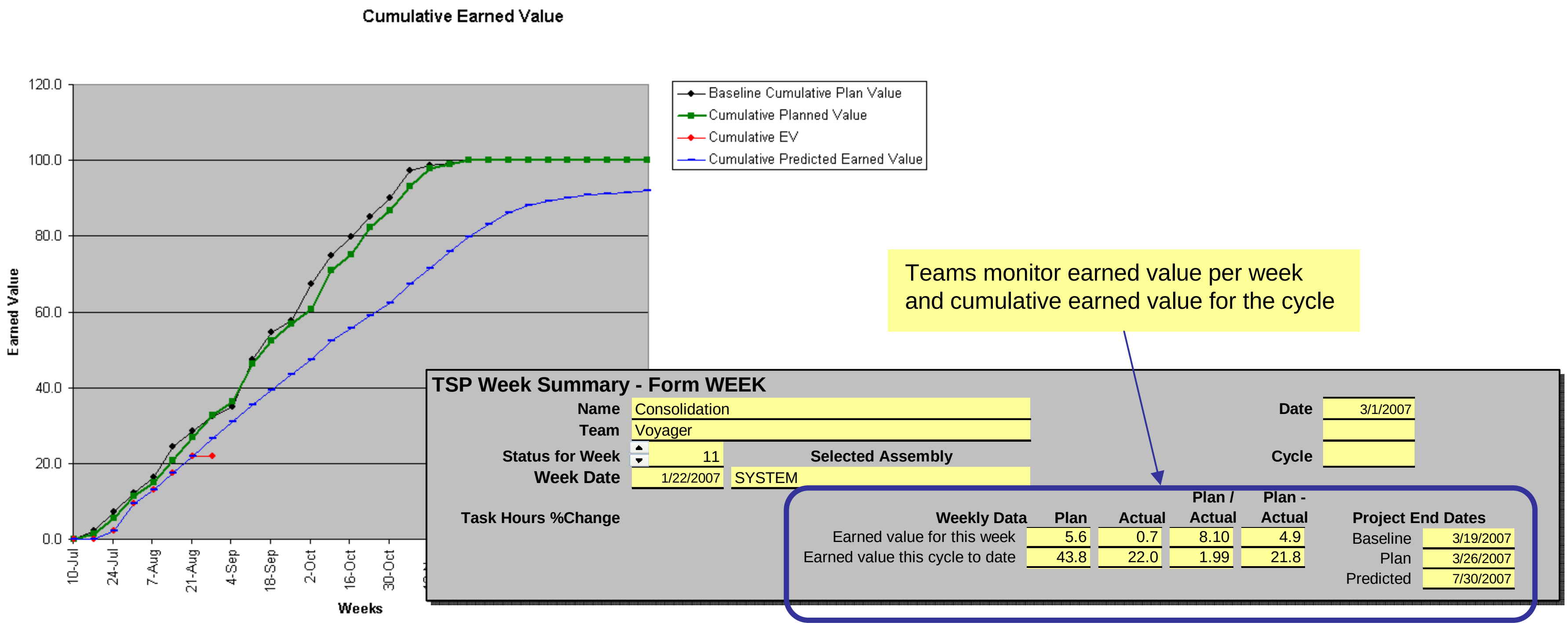


Source: Intuit

Schedule Management -1

TSP teams routinely meet their schedule commitments.

They use earned value management, task hour management, and quality management at the team and personal level to help manage schedule.



Teams monitor earned value per week and cumulative earned value for the cycle

TSP Week Summary - Form WEEK

Name	Consolidation	**Date**	3/1/2007
Team	Voyager		
Status for Week	11	**Selected Assembly**	**Cycle**
Week Date	1/22/2007	SYSTEM	

Task Hours %Change

Weekly Data	**Plan**	**Actual**	**Plan / Actual**	**Plan - Actual**
Earned value for this week	5.6	0.7	8.10	4.9
Earned value this cycle to date	43.8	22.0	1.99	21.8

Project End Dates	
Baseline	3/19/2007
Plan	3/26/2007
Predicted	7/30/2007

Schedule Management -2

Intuit’s 2007 release of QuickBooks met every major milestone and delivered 33% more functionality than planned.

First-time TSP projects at Microsoft had a 10 times better mean schedule error than non-TSP projects at Microsoft as reflected in the following table.

Microsoft Schedule Results	Non-TSP Projects	TSP Projects
Released on Time	42%	66%
Average Days Late	25	6
Mean Schedule Error	10%	1%
Sample Size	80	15

Source: Microsoft

A New Reality -1

What if…?

- development teams delivered software when originally promised?
- the software worked as advertised when delivered?
- the test team could execute the entire test suite and report a relative few defects that could be listed on one or two pages?
- development returned a second version of the code in a day or two with all the defects fixed?
- testing was complete on the SECOND pass through the test suite?

A New Reality -2

Test teams would

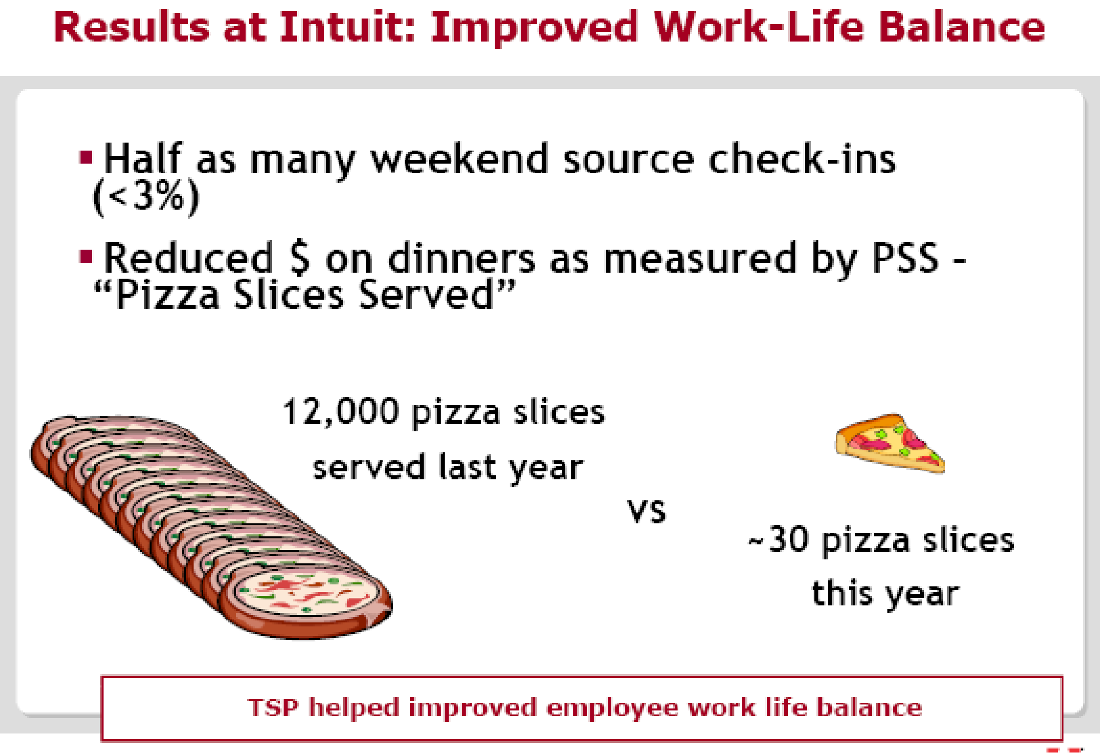
- not be under the gun constantly
- concentrate on verifying full functionality rather than constantly running most-used scenarios
- be able to test more exhaustively (the meaning of “full functionality” most likely would change)

Work-Life Balance

People are your most important resource.

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"

12,000 pizza slices served last year

vs

~30 pizza slices this year

TSP helped improved employee work life balance

Source: Intuit

Questions?

4th Annual Software Engineering Institute (SEI)

Team Software Process (TSP) Symposium

September 21-24, 2009

Royal Sonesta Hotel, New Orleans, LA

Theme: Establishing a Competitive Advantage

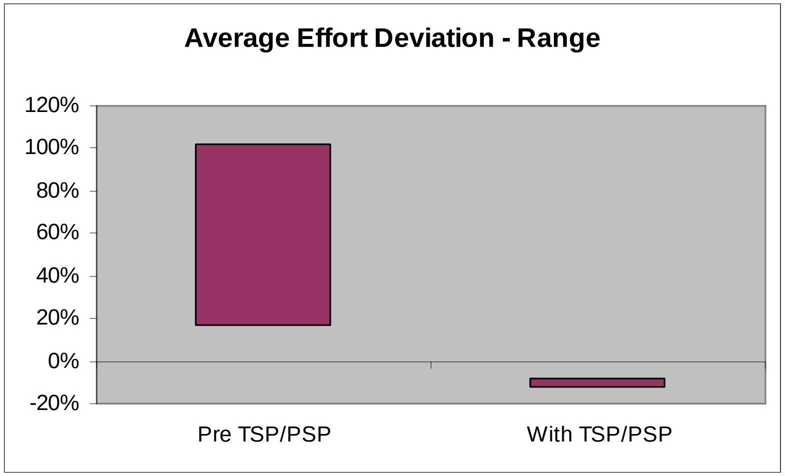
Backup slides

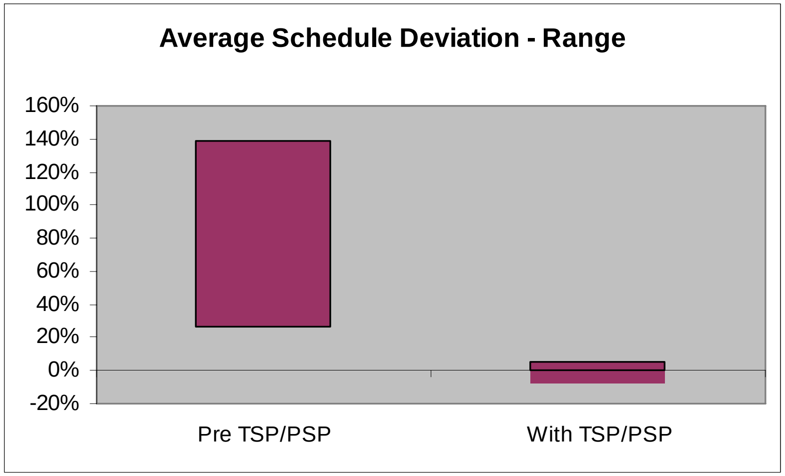
TSP Performance Comparison -1

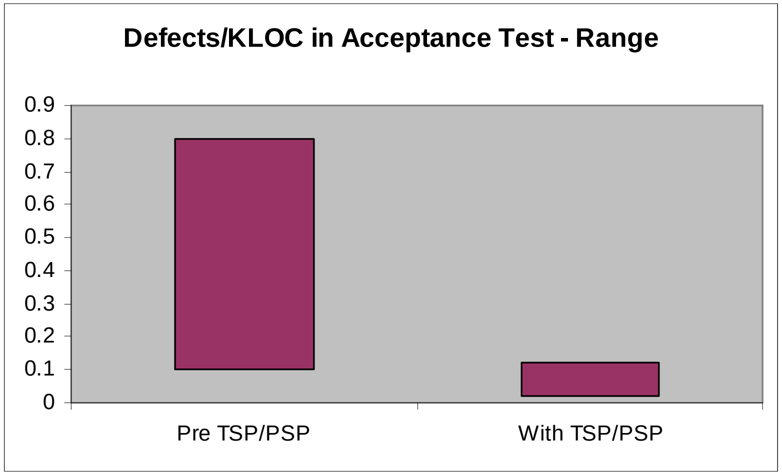
Performance Category	Typical Industry Performance[1]	Study Baseline (2000) [2]	TSP Impact study (2000)[3]	TSP Impact study (2003)[4]
Schedule Error	180% avg.	27% to 112%	5% avg.	6% avg.
Effort Error	130% avg.	17% to 85%	-4% avg.	26% avg.
System Test defects per KLOC	5 to 25 approx.	NA	0.0 to 0.9	0.4 avg. 0.0 to 0.9
Released defects per KLOC	1 to 7	0.2 to 1+	0.0 to 0.35	0.06 avg. 0.0 to 0.2

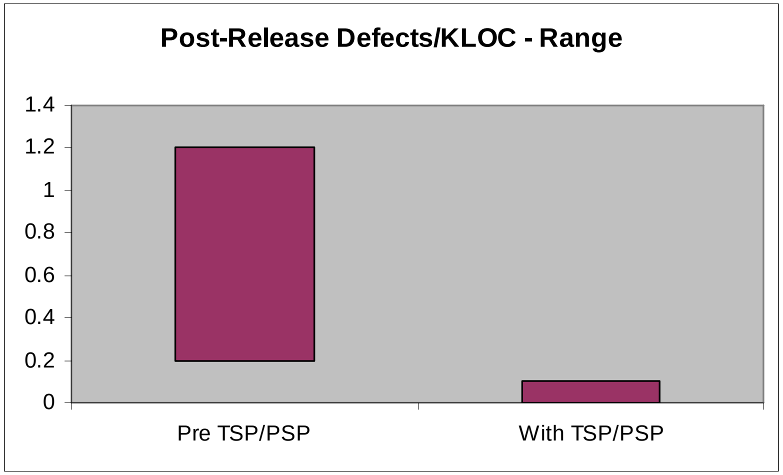
1. Gartner Group
2. Control projects from the CMU/SEI-2000-TR-015 report, a study of 18 TSP projects in four organizations conducted in 2000
3. TSP projects from the CMU/SEI-2000-TR-015 report, a study of 18 TSP projects in four organizations conducted in 2000
4. CMU/SEI-2003-TR-014, a study of 20 projects in 13 organizations conducted in 2003

TSP Performance Comparison -2


Average Effort Deviation - Range


Average Schedule Deviation - Range


Defects/KLOC in Acceptance Test - Range


Post-Release Defects/KLOC - Range

*From a study of 18 TSP projects in four organizations CMU/SEI-2000-TR-015

Quality Benefits

TSP dramatically reduces the effort and schedule for system test.

Most defects are removed during reviews and inspections at a cost of 2 to 25 minutes per defect.

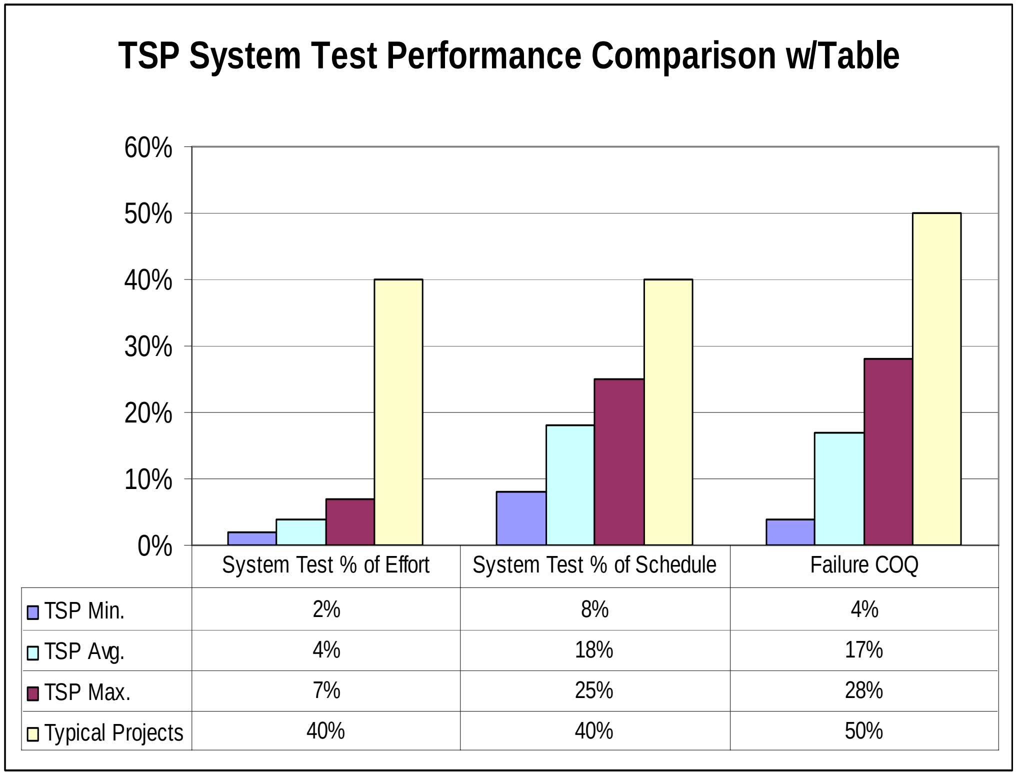
System test removal costs run from

to 2 to 20 hours per defect.

These benefits continue

after delivery.

- lower support costs
- satisfied customer
- better resource utilization


TSP System Test Performance Comparison w/Table

	System Test % of Effort	System Test % of Schedule	Failure COQ
■ TSP Min.	2%	8%	4%
□ TSP Avg.	4%	18%	17%
■ TSP Max.	7%	25%	28%
□ Typical Projects	40%	40%	50%

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.