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

<table>
<thead>
<tr>
<th>Post code complete defects</th>
<th>Version 2.4</th>
<th>Version 2.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phase</td>
<td>Version 2.4</td>
<td>Version 2.5</td>
</tr>
<tr>
<td>Integration Test</td>
<td>237</td>
<td>4</td>
</tr>
<tr>
<td>System Test</td>
<td>473</td>
<td>10</td>
</tr>
<tr>
<td>User Acceptance Test</td>
<td>153</td>
<td>3</td>
</tr>
<tr>
<td>Total</td>
<td>1072</td>
<td>17</td>
</tr>
</tbody>
</table>
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

Source: Intuit
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”

12,000 pizza slices served last year

VS

~30 pizza slices this year

Source: Intuit

TSP helped improved employee work life balance
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.
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.*
Why Testing Isn’t Enough

- Overload
- Hardware failure
- System attack
- Operator error
- Data error
- Resource contention
- Unexpected condition
- Configuration

Tested – paths in the safe region (shaded green)

Untested – paths in the unsafe region (shaded red)
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

Defects Removed in Phase

Requirements Inspection  Design Review and Inspection  Code Review and Inspection  Unit Test  System Test
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.
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

- 1st Quartile
- 2nd Quartile
- 3rd Quartile
- 4th Quartile

Defect reduction:
1Q: 80.4%
2Q: 79.0%
3Q: 78.5%
4Q: 77.6%

810 developers

Defects/KLOC
PSP Design Time Results

Time Invested Per (New and Changed) Line of Code

Mean Minutes Spent Per LOC

Program Number

0 1 2 3 4 5 6 7 8 9 10 11

0.0 0.2 0.4 0.6 0.8 1.0 1.2 1.4

Design Code Compile Test

PSP0 PSP1 PSP2

298 developers
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

PSP

Quality practices

Self-directed teams

Measurement

Coaching

Accurate pans
Management Styles

The principal management styles have been:

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

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

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

Frederick Taylor

Peter Drucker
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

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.

Tiger Woods and his coach Hank Haney.
Working Together to Improve Product Quality

High Quality Products
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.

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

Requirements → High-Level Design → Implementation → System Test

- Requirements Launch
  - Produce Requirements Specifications
    - Inspection
      - Postmortem

- High-Level Design Launch
  - Produce High-Level Design
    - Inspection
      - Postmortem

- Implementation Launch
  - Produce Detail Design
    - Personal Review
      - Inspection

- Produce Technical Artifacts (Code)
  - Personal Review
    - Static Analysis (optional)
  - Inspection
    - Unit Test
      - System Test
        - System Test Launch
          - System Build
            - Integration Test
              - System Test
                - Postmortem

Defect Removal Filter
Capture-Recapture

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

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

\[ A = \text{Defects in test A} \]
\[ B = \text{Defects in test B} \]
\[ C = \text{Defects common to A and B} \]

\[ \text{Est. total defects} = \frac{A \times B}{C} \]
\[ \text{Total found} = A + B - C \]
\[ \text{Est. total remaining} = \frac{A \times B}{C} - (A + B - C) \]
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

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  | Effort
---|---
Size  | Quality

Source: CMU/SEI-92-TR-019
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.
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.
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
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.
## What the Measurement Framework Provides…

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

---

Efficiency, Quality, and Agility © 2010 Carnegie Mellon University
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

<table>
<thead>
<tr>
<th>SYSTEM Percent Defect Free</th>
<th>Actual</th>
</tr>
</thead>
<tbody>
<tr>
<td>In Compile</td>
<td>62.3%</td>
</tr>
<tr>
<td>In Unit Test</td>
<td>53.8%</td>
</tr>
<tr>
<td>In Build and Integration Test</td>
<td>73.9%</td>
</tr>
<tr>
<td>In System Test</td>
<td>98.5%</td>
</tr>
<tr>
<td>In Acceptance Test</td>
<td>100.0%</td>
</tr>
<tr>
<td>In Product Life</td>
<td>100.0%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Defects/Page</th>
<th>Plan</th>
<th>Actual</th>
</tr>
</thead>
<tbody>
<tr>
<td>REQ Inspection</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>HLD Inspection</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Defects/KLOC</th>
<th>Plan</th>
<th>Actual</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLD Review</td>
<td>8.33</td>
<td>9.36</td>
</tr>
<tr>
<td>DLD Inspection</td>
<td>4.77</td>
<td>6.54</td>
</tr>
<tr>
<td>Code Review</td>
<td>19.19</td>
<td>10.33</td>
</tr>
<tr>
<td>Compile</td>
<td>4.92</td>
<td>3.43</td>
</tr>
<tr>
<td>Code Inspection</td>
<td>4.21</td>
<td>5.79</td>
</tr>
<tr>
<td>Unit Test</td>
<td>1.20</td>
<td>4.25</td>
</tr>
<tr>
<td>Build and Integration Test</td>
<td>0.48</td>
<td>0.89</td>
</tr>
<tr>
<td>System Test</td>
<td>0.36</td>
<td>0.04</td>
</tr>
<tr>
<td>Total Development</td>
<td>43.75</td>
<td>41.10</td>
</tr>
<tr>
<td>Acceptance Test</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Product Life</td>
<td>0.36</td>
<td>0.00</td>
</tr>
<tr>
<td>Total</td>
<td>44.11</td>
<td>41.10</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Defect Ratios</th>
<th>Plan</th>
<th>Actual</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLD Review/Unit Test</td>
<td>6.94</td>
<td>2.20</td>
</tr>
<tr>
<td>Code Review/Compile</td>
<td>3.90</td>
<td>3.01</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Development Time Ratios</th>
<th>Plan</th>
<th>Actual</th>
</tr>
</thead>
<tbody>
<tr>
<td>REQ Inspection/Requirements</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>HLD Inspection/High-Level Design</td>
<td>0.02</td>
<td>0.03</td>
</tr>
<tr>
<td>Detailed Design/Code</td>
<td>1.09</td>
<td>1.57</td>
</tr>
<tr>
<td>DLD Review/Detailed Design</td>
<td>0.45</td>
<td>0.32</td>
</tr>
<tr>
<td>Code Review/Code</td>
<td>0.46</td>
<td>0.41</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Inspection/Review Rates</th>
<th>Plan</th>
<th>Actual</th>
</tr>
</thead>
<tbody>
<tr>
<td>REQ Inspection</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>HLD Inspection</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>DLD Review</td>
<td>148.45</td>
<td>225.53</td>
</tr>
<tr>
<td>DLD Inspection</td>
<td>111.08</td>
<td>143.76</td>
</tr>
<tr>
<td>Code Review</td>
<td>156.86</td>
<td>272.99</td>
</tr>
<tr>
<td>Code Inspection</td>
<td>114.09</td>
<td>132.31</td>
</tr>
<tr>
<td><strong>A/FR</strong></td>
<td><strong>2.82</strong></td>
<td><strong>3.49</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Phase Yields</th>
<th>Plan</th>
<th>Actual</th>
</tr>
</thead>
<tbody>
<tr>
<td>Planning</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>Requirements</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>System Test Plan</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>REQ Inspection</td>
<td>50%</td>
<td>0%</td>
</tr>
<tr>
<td>High-Level Design</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>Integration Test Plan</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>HLD Inspection</td>
<td>60%</td>
<td>0%</td>
</tr>
<tr>
<td>Detailed Design</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>DLD Review</td>
<td>55%</td>
<td>47%</td>
</tr>
<tr>
<td>Test Development</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>DLD Inspection</td>
<td>70%</td>
<td>62%</td>
</tr>
</tbody>
</table>
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
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.

![Graph showing the relationship between code review rate (LOC/Hour) and percent with greater yield.](image)
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

<table>
<thead>
<tr>
<th>Indicator</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Design defect injection rate</td>
<td>0.75/Hr.</td>
</tr>
<tr>
<td>Design review defect removal rate</td>
<td>1.5/Hr.</td>
</tr>
<tr>
<td>Design/Design Review time ratio</td>
<td>2</td>
</tr>
<tr>
<td>Coding defect injection rate</td>
<td>2/Hr.</td>
</tr>
<tr>
<td>Code review defect removal rate</td>
<td>4/Hr.</td>
</tr>
<tr>
<td>Code/Code Review time ratio</td>
<td>2</td>
</tr>
<tr>
<td>Design/Code time ratio (derived from TSP data)</td>
<td>1</td>
</tr>
</tbody>
</table>
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.
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.
PQI vs. Post-Development Defects
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