SOFTWARE QUALITY IN 2002:
A SURVEY OF
THE STATE OF THE ART

Capers Jones, Chief Scientist Emeritus

SOURCES OF SPR’S QUALITY DATA

SPR clients from 1984 through 2002
- About 600 companies (150 clients in Fortune 500 set)
- About 30 government/military groups
- About 12,000 total projects
- New data = about 75 projects per month
- Data collected from 24 countries
- Observations during more than a dozen lawsuits

BASIC DEFINITIONS

SOFTWARE QUALITY
Software that combines the characteristics of low defect rates and high user satisfaction

USER SATISFACTION
Clients who are pleased with a vendor’s products, quality levels, ease of use, and support

DEFECT PREVENTION
Technologies that minimize the risk of making errors in software deliverables

DEFECT REMOVAL
Activities that find and correct defects in software deliverables

BAD FIXES
Secondary defects injected as a byproduct of defect repairs

FUNDAMENTAL SOFTWARE QUALITY METRICS

- Defect Potentials
  - requirements errors, design errors, code errors, document errors, bad fix errors, test plan errors, and test case errors

- Defects Removed
  - Characteristics
    - By origin
      - before testing
      - during testing
      - during deployment
  - Defect Removal Efficiency
    - ratio of defects removed to defect potentials

- Defect Severity Levels
  - fatal, serious, minor, cosmetic
FUNDAMENTAL SOFTWARE QUALITY METRICS (cont.)

- Duplicate Defects
- Invalid Defects
- Defect Removal Effort and Costs
  - preparation
  - execution
  - repairs and rework
  - effort on duplicates and invalids
- Supplemental Quality Metrics
  - complexity
  - test case volumes
  - test case coverage
  - IBM’s orthogonal defect classification (Ram Chillarege)

FUNDAMENTAL SOFTWARE QUALITY METRICS (cont.)

- Standard Cost of Quality
  - Prevention
  - Appraisal
  - Failures
- Revised Software Cost of Quality
  - Defect Prevention
  - Non-Test Defect Removal
  - Testing Defect Removal
  - Post-Release Defect Removal
- Error-Prone Module Effort
  - Identification
  - Removal or redevelopment
  - Repairs and rework

HAZARDOUS QUALITY DEFINITIONS

Should quality mean “conformance to requirements?”

Requirements contain > 15% of software errors.
Requirements sometimes grow at > 2% per month.
Do you conform to requirements errors?
Do you conform to totally new requirements?

HAZARDOUS QUALITY METRICS

Cost per Defect

- Approaches infinity as defects near zero
- Conceals real economic value of quality
**COST PER DEFECT PENALIZES QUALITY**

<table>
<thead>
<tr>
<th>Function Points</th>
<th>Poor Quality</th>
<th>Good Quality</th>
<th>Excellent Quality</th>
<th>Zero Defects</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bugs Discovered</td>
<td>500</td>
<td>50</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>Initial work</td>
<td>$5,000</td>
<td>$5,000</td>
<td>$5,000</td>
<td>$5,000</td>
</tr>
<tr>
<td>Defect detection</td>
<td>$5,000</td>
<td>$2,500</td>
<td>$1,000</td>
<td>$0</td>
</tr>
<tr>
<td>Defect repair</td>
<td>$25,000</td>
<td>$5,000</td>
<td>$1,000</td>
<td>$0</td>
</tr>
<tr>
<td>Total</td>
<td>$35,000</td>
<td>$12,500</td>
<td>$7,000</td>
<td>$5,000</td>
</tr>
<tr>
<td>Cost per Defect Removed</td>
<td>$70</td>
<td>$250</td>
<td>$1,400</td>
<td>∞</td>
</tr>
<tr>
<td>Cost per Function Point</td>
<td>$350</td>
<td>$125</td>
<td>$70</td>
<td>$50</td>
</tr>
</tbody>
</table>

**HAZARDS OF “DEFECTS PER KLOC” METRICS**

**Defects per KLOC**

Software defects are found in:
- Requirements
- Design
- Source code
- User documents
- Bad fixes (secondary defects)

Requirements and design defects often outnumber code defects.

The metric “Defects per KLOC” ignores the complexity and importance of all deliverables other than code.

**FOUR LANGUAGE COMPARISON OF SOFTWARE DEFECT POTENTIALS**

<table>
<thead>
<tr>
<th>Defect Origin</th>
<th>Assembly</th>
<th>Ada</th>
<th>C++</th>
<th>C++ and Reuse</th>
</tr>
</thead>
<tbody>
<tr>
<td>Function points</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>KLOC</td>
<td>30</td>
<td>7.5</td>
<td>5.5</td>
<td>2.5</td>
</tr>
<tr>
<td>Requirements</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>Design</td>
<td>50</td>
<td>50</td>
<td>35</td>
<td>15</td>
</tr>
<tr>
<td>Code</td>
<td>150</td>
<td>45</td>
<td>35</td>
<td>15</td>
</tr>
<tr>
<td>Documents</td>
<td>25</td>
<td>25</td>
<td>25</td>
<td>25</td>
</tr>
<tr>
<td>Bad Fixes</td>
<td>20</td>
<td>10</td>
<td>7</td>
<td>4</td>
</tr>
<tr>
<td>TOTAL DEFECTS</td>
<td>265</td>
<td>150</td>
<td>122</td>
<td>79</td>
</tr>
<tr>
<td>Defects per KLOC</td>
<td>10.6</td>
<td>20.0</td>
<td>22.2</td>
<td>31.6</td>
</tr>
<tr>
<td>Defects/Function Point</td>
<td>3.0</td>
<td>2.0</td>
<td>1.22</td>
<td>0.79</td>
</tr>
</tbody>
</table>

Use of the metric “Defect per KLOC” may be considered professional malpractice.

**U.S. AVERAGES FOR SOFTWARE QUALITY**

(Data expressed in terms of defects per function point)

<table>
<thead>
<tr>
<th>Defect Origins</th>
<th>Defect Potential</th>
<th>Removal Efficiency</th>
<th>Delivered Defects</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirements</td>
<td>1.00</td>
<td>77%</td>
<td>0.23</td>
</tr>
<tr>
<td>Design</td>
<td>1.25</td>
<td>85%</td>
<td>0.19</td>
</tr>
<tr>
<td>Coding</td>
<td>1.75</td>
<td>95%</td>
<td>0.09</td>
</tr>
<tr>
<td>Documents</td>
<td>0.60</td>
<td>80%</td>
<td>0.12</td>
</tr>
<tr>
<td>Bad Fixes</td>
<td>0.40</td>
<td>70%</td>
<td>0.12</td>
</tr>
<tr>
<td>TOTAL</td>
<td>5.00</td>
<td>85%</td>
<td>0.75</td>
</tr>
</tbody>
</table>

(Function points show all defect sources - not just coding defects)
BEST IN CLASS SOFTWARE QUALITY

(Data expressed in terms of defects per function point)

<table>
<thead>
<tr>
<th>Defect Origins</th>
<th>Defects</th>
<th>Potential Removal Efficiency Delivered Defects</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirements</td>
<td>0.40</td>
<td>85%</td>
</tr>
<tr>
<td>Design</td>
<td>0.60</td>
<td>97%</td>
</tr>
<tr>
<td>Coding</td>
<td>1.00</td>
<td>99%</td>
</tr>
<tr>
<td>Documents</td>
<td>0.40</td>
<td>98%</td>
</tr>
<tr>
<td>Bad Fixes</td>
<td>0.10</td>
<td>95%</td>
</tr>
<tr>
<td>TOTAL</td>
<td>2.50</td>
<td>96%</td>
</tr>
</tbody>
</table>

OBSERVATION
Most often found in systems software > SEI CMM Level 3

POOR SOFTWARE QUALITY - MALPRACTICE

(Data expressed in terms of defects per function point)

<table>
<thead>
<tr>
<th>Defect Origins</th>
<th>Defects</th>
<th>Potential Removal Efficiency Delivered Defects</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirements</td>
<td>1.50</td>
<td>50%</td>
</tr>
<tr>
<td>Design</td>
<td>2.20</td>
<td>50%</td>
</tr>
<tr>
<td>Coding</td>
<td>2.50</td>
<td>80%</td>
</tr>
<tr>
<td>Documents</td>
<td>1.00</td>
<td>70%</td>
</tr>
<tr>
<td>Bad Fixes</td>
<td>0.80</td>
<td>50%</td>
</tr>
<tr>
<td>TOTAL</td>
<td>8.00</td>
<td>62%</td>
</tr>
</tbody>
</table>

OBSERVATIONS
Most often found in large client-server projects (> 5000 FP).

GOOD QUALITY RESULTS > 90% SUCCESS RATE

- Formal Inspections (Requirements, Design, and Code)
- Joint Application Design (JAD)
- Quality Function Deployment (QFD)
- Quality Metrics using function points
- Quality Metrics using IBM’s Orthogonal defect classification
- Defect Removal Efficiency Measurements
- Automated Defect tracking tools
- Active Quality Assurance (> 5% SQA staff)
- Formal change controls
- User Satisfaction Surveys
- Formal Test Plans for Major Projects
- Quality Estimation Tools
- Automated Test Support Tools
- Testing Specialists
- Root-Cause Analysis

MIXED QUALITY RESULTS: < 50% SUCCESS RATE

- Total Quality Management (TQM)
- Independent Verification & Validation (IV & V)
- Independent quality audits
- Six-Sigma quality programs
- Baldrige Awards
- IEEE Quality Standards
- Testing only by developers
- DOD 2167A and DOD 498
- Reliability Models
- Quality circles
- Clean-room methods
- Cost of quality without software modifications
POOR QUALITY RESULTS: < 25% SUCCESS RATE

- ISO 9000 - 9004 Quality Standards
- Informal Testing
- Manual Testing
- Passive Quality Assurance (< 3% QA staff)
- Token Quality Assurance (< 1% QA staff)
- LOC Metrics for quality
- Cost per defect metric
- Rapid Application Development (RAD)

A PRACTICAL DEFINITION OF SOFTWARE QUALITY (PREDICTABLE AND MEASURABLE)

- Low Defect Potentials (< 2.5 per Function Point)
- High Defect Removal Efficiency (> 95%)
- Unambiguous, Stable Requirements (< 2.5% change)
- Explicit Requirements Achieved (> 97.5% achieved)
- High User Satisfaction Ratings (> 90% “excellent”)
  - Installation
  - Ease of learning
  - Ease of use
  - Functionality
  - Compatibility
  - Error handling
  - User information (screens, manuals, tutorials)
  - Customer support
  - Defect repairs

SOFTWARE QUALITY OBSERVATIONS

Quality Measurements Have Found:

- Individual programmers -- Less than 50% efficient in finding bugs in their own software
- Normal test steps -- often less than 70% efficient (1 of 3 bugs remain)
- Design Reviews and Code Inspections -- often more than 85% efficient; have topped 85%
- Reviews or inspections plus formal testing -- are often more than 96% efficient; have hit 99%
- Reviews and Inspections -- lower costs and schedules by as much as 30%

SOFTWARE DEFECT ORIGINS

1) Requirements: Hardest to prevent and repair
2) Design: Most severe and pervasive
3) Code: Most numerous; easiest to fix
4) Documentation: Can be serious if ignored
5) Bad Fixes: Very difficult to find
6) Bad Test Cases: Common and troublesome
7) Data quality: Common but hard to measure
8) Web content: Unmeasured circa 2002
SOFTWARE DEFECT SEVERITY CATEGORIES

Severity 1: TOTAL FAILURES 1% at release
Severity 2: MAJOR PROBLEMS 20% at release
Severity 3: MINOR PROBLEMS 35% at release
Severity 4: COSMETIC ERRORS 44% at release

INVALID USER OR SYSTEM ERRORS 15% of reports
DUPLICATE MULTIPLE REPORTS 30% of reports
ABEYANT CAN’T RECREATE ERROR 5% of reports

PERCENTAGE OF SOFTWARE EFFORT BY TASK

<table>
<thead>
<tr>
<th>Size in Function Points</th>
<th>Management/ Support</th>
<th>Defect Removal</th>
<th>Paperwork</th>
<th>Coding</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>10,240</td>
<td>18%</td>
<td>36%</td>
<td>34%</td>
<td>12%</td>
<td>100%</td>
</tr>
<tr>
<td>5,120</td>
<td>17%</td>
<td>33%</td>
<td>32%</td>
<td>18%</td>
<td>100%</td>
</tr>
<tr>
<td>2,580</td>
<td>16%</td>
<td>31%</td>
<td>29%</td>
<td>24%</td>
<td>100%</td>
</tr>
<tr>
<td>1,280</td>
<td>15%</td>
<td>29%</td>
<td>26%</td>
<td>30%</td>
<td>100%</td>
</tr>
<tr>
<td>640</td>
<td>14%</td>
<td>27%</td>
<td>23%</td>
<td>36%</td>
<td>100%</td>
</tr>
<tr>
<td>320</td>
<td>13%</td>
<td>25%</td>
<td>20%</td>
<td>42%</td>
<td>100%</td>
</tr>
<tr>
<td>160</td>
<td>12%</td>
<td>23%</td>
<td>17%</td>
<td>48%</td>
<td>100%</td>
</tr>
<tr>
<td>80</td>
<td>11%</td>
<td>21%</td>
<td>14%</td>
<td>54%</td>
<td>100%</td>
</tr>
<tr>
<td>40</td>
<td>10%</td>
<td>19%</td>
<td>11%</td>
<td>60%</td>
<td>100%</td>
</tr>
<tr>
<td>20</td>
<td>9%</td>
<td>17%</td>
<td>8%</td>
<td>66%</td>
<td>100%</td>
</tr>
<tr>
<td>10</td>
<td>8%</td>
<td>15%</td>
<td>5%</td>
<td>72%</td>
<td>100%</td>
</tr>
</tbody>
</table>

HOW QUALITY INFLUENCES SOFTWARE COSTS

Pathological
Healthy

Poor quality is cheaper until the end of the coding phase. After that, high quality is cheaper.

U. S. SOFTWARE QUALITY AVERAGES CIRCA 2002

<table>
<thead>
<tr>
<th>Defect Potentials</th>
<th>System Software</th>
<th>Commercial Software</th>
<th>Information Software</th>
<th>Military Software</th>
<th>Outsource Software</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.0</td>
<td>5.0</td>
<td>4.5</td>
<td>7.0</td>
<td>5.2</td>
<td></td>
</tr>
<tr>
<td>Defect Removal Efficiency</td>
<td>94%</td>
<td>90%</td>
<td>73%</td>
<td>96%</td>
<td>92%</td>
</tr>
<tr>
<td>Delivered Defects</td>
<td>0.4</td>
<td>0.5</td>
<td>1.2</td>
<td>0.3</td>
<td>0.4</td>
</tr>
<tr>
<td>First Year Discovery Rate</td>
<td>65%</td>
<td>70%</td>
<td>30%</td>
<td>75%</td>
<td>60%</td>
</tr>
<tr>
<td>First Year Reported Defects</td>
<td>0.26</td>
<td>0.35</td>
<td>0.36</td>
<td>0.23</td>
<td>0.30</td>
</tr>
</tbody>
</table>
### U. S. SOFTWARE QUALITY AVERAGES CIRCA 2002

(Defects per Function Point)

<table>
<thead>
<tr>
<th></th>
<th>Web Software</th>
<th>Embedded Software</th>
<th>SEI-CMM 3 Software</th>
<th>SEI-CMM 1 Software</th>
<th>Overall Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Defect Potentials</td>
<td>4.0</td>
<td>5.5</td>
<td>3.0</td>
<td>5.5</td>
<td>5.1</td>
</tr>
<tr>
<td>Defect Removal Efficiency</td>
<td>72%</td>
<td>95%</td>
<td>95%</td>
<td>73%</td>
<td>86.7%</td>
</tr>
<tr>
<td>Delivered Defects</td>
<td>1.1</td>
<td>0.3</td>
<td>0.15</td>
<td>1.5</td>
<td>0.68</td>
</tr>
<tr>
<td>First Year Discovery Rate</td>
<td>95%</td>
<td>90%</td>
<td>60%</td>
<td>35%</td>
<td>64.4%</td>
</tr>
<tr>
<td>First Year Reported Defects</td>
<td>1.0</td>
<td>0.27</td>
<td>0.09</td>
<td>0.52</td>
<td>0.43</td>
</tr>
</tbody>
</table>

### SOFTWARE SIZE VS DEFECT REMOVAL EFFICIENCY

(Data Expressed in terms of Defects per Function Point)

<table>
<thead>
<tr>
<th>Size</th>
<th>Defect Potential</th>
<th>Defect Removal Efficiency</th>
<th>Delivered Defects</th>
<th>1st Year Discovery Rate</th>
<th>1st Year Reported Defects</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.85</td>
<td>95.00%</td>
<td>0.09</td>
<td>90.00%</td>
<td>0.08</td>
</tr>
<tr>
<td>10</td>
<td>2.45</td>
<td>92.00%</td>
<td>0.20</td>
<td>80.00%</td>
<td>0.16</td>
</tr>
<tr>
<td>100</td>
<td>3.68</td>
<td>90.00%</td>
<td>0.37</td>
<td>70.00%</td>
<td>0.26</td>
</tr>
<tr>
<td>1000</td>
<td>5.00</td>
<td>85.00%</td>
<td>0.75</td>
<td>50.00%</td>
<td>0.38</td>
</tr>
<tr>
<td>10000</td>
<td>7.60</td>
<td>78.00%</td>
<td>1.67</td>
<td>40.00%</td>
<td>0.67</td>
</tr>
<tr>
<td>100000</td>
<td>9.55</td>
<td>75.00%</td>
<td>2.39</td>
<td>30.00%</td>
<td>0.72</td>
</tr>
<tr>
<td>AVERAGE</td>
<td>5.02</td>
<td>85.83%</td>
<td>0.91</td>
<td>60.00%</td>
<td>0.38</td>
</tr>
</tbody>
</table>

### SOFTWARE DEFECT POTENTIALS AND DEFECT REMOVAL EFFICIENCY FOR EACH LEVEL OF SEI CMM

(Data Expressed in Terms of Defects per Function Point
For projects nominally 1000 function points in size)

<table>
<thead>
<tr>
<th>SEI CMM Levels</th>
<th>Defect Potentials</th>
<th>Removal Efficiency</th>
<th>Delivered Defects</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEI CMM 1</td>
<td>5.00</td>
<td>80%</td>
<td>1.00</td>
</tr>
<tr>
<td>SEI CMM 2</td>
<td>4.00</td>
<td>90%</td>
<td>0.40</td>
</tr>
<tr>
<td>SEI CMM 3</td>
<td>3.00</td>
<td>95%</td>
<td>0.15</td>
</tr>
<tr>
<td>SEI CMM 4</td>
<td>2.00</td>
<td>97%</td>
<td>0.08</td>
</tr>
<tr>
<td>SEI CMM 5</td>
<td>1.00</td>
<td>99%</td>
<td>0.01</td>
</tr>
</tbody>
</table>

### SOFTWARE DEFECT POTENTIALS AND DEFECT REMOVAL EFFICIENCY FOR EACH LEVEL OF SEI CMM

(Data Expressed in Terms of Defects per Function Point
For projects > 5000 function points in size)

<table>
<thead>
<tr>
<th>SEI CMM Levels</th>
<th>Defect Potentials</th>
<th>Removal Efficiency</th>
<th>Delivered Defects</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEI CMM 1</td>
<td>5.50</td>
<td>73%</td>
<td>1.48</td>
</tr>
<tr>
<td>SEI CMM 2</td>
<td>4.00</td>
<td>90%</td>
<td>0.40</td>
</tr>
<tr>
<td>SEI CMM 3</td>
<td>3.00</td>
<td>95%</td>
<td>0.15</td>
</tr>
<tr>
<td>SEI CMM 4</td>
<td>2.50</td>
<td>97%</td>
<td>0.08</td>
</tr>
<tr>
<td>SEI CMM 5</td>
<td>2.25</td>
<td>98%</td>
<td>0.05</td>
</tr>
</tbody>
</table>
MAJOR SOFTWARE QUALITY ZONES

The SEI CMM levels overlap.

SOFTWARE QUALITY IMPROVEMENT (cont.)

Client-server projects are worse than U.S. averages.

Telecommunications projects are better than U.S. averages.
SOFTWARE QUALITY IMPROVEMENT (cont.)

Defect Removal Efficiency

OO projects can be hazardous due to shallow learning curve

DEFECTS PER FP
0 1 2 3 4 5

50% 55% 60% 65% 70% 75% 80% 85% 90% 95% 100%

SOFTWARE QUALITY IMPROVEMENT (cont.)

Defect Removal Efficiency

ISO 9000-9004 have uncertain results

DEFECTS PER FP
0 1 2 3 4 5

50% 55% 60% 65% 70% 75% 80% 85% 90% 95% 100%

SOFTWARE QUALITY IMPROVEMENT (cont.)

Defect Removal Efficiency

Military projects are better than U.S. averages

DEFECTS PER FP
0 1 2 3 4 5

50% 55% 60% 65% 70% 75% 80% 85% 90% 95% 100%

SOFTWARE QUALITY IMPROVEMENT (cont.)

Defect Removal Efficiency

Best in Class

INDUSTRY-WIDE DEFECT CAUSES

Ranked in order of effort required to fix the defects:
1. Requirements problems (omissions; changes)
2. Design problems
3. Interface problems between modules
4. Logic, branching, and structural problems
5. Memory allocation problems
6. Testing omissions and poor coverage
7. Test case errors
8. Stress/performance problems
9. Bad fixes/Regressions
10. Documentation errors
SOFTWARE QUALITY UNKNOWNS

SOFTWARE QUALITY TOPICS NEEDING RESEARCH:

• ERRORS IN SOFTWARE TEST PLANS AND TEST CASES
• ERRORS IN WEB “CONTENT” (I.E. GRAPHICS, SOUNDS)
• MASS-UPDATE TESTING
• SUPPLY-CHAIN TESTING (MULTI-NATIONAL)
• ERRORS IN DATABASES AND DATA WAREHOUSES
• CAUSES OF BAD FIX INJECTION RATES
• IMPACT OF COMPLEXITY ON QUALITY
• IMPACT OF CREEPING REQUIREMENTS

DEFECT REMOVAL AND TESTING STAGES NOTED DURING LITIGATION FOR POOR QUALITY

<table>
<thead>
<tr>
<th>Reliable Software</th>
<th>Software Involved in Litigation for Poor Quality</th>
</tr>
</thead>
<tbody>
<tr>
<td>Formal design inspections</td>
<td>Used</td>
</tr>
<tr>
<td>Formal code inspections</td>
<td>Used</td>
</tr>
<tr>
<td>Subroutine testing</td>
<td>Used</td>
</tr>
<tr>
<td>Unit testing</td>
<td>Used</td>
</tr>
<tr>
<td>New function testing</td>
<td>Used</td>
</tr>
<tr>
<td>Regression testing</td>
<td>Used</td>
</tr>
<tr>
<td>Integration testing</td>
<td>Used</td>
</tr>
<tr>
<td>System testing</td>
<td>Used</td>
</tr>
<tr>
<td>Performance testing</td>
<td>Used</td>
</tr>
<tr>
<td>Capacity testing</td>
<td>Used</td>
</tr>
</tbody>
</table>

SOFTWARE QUALITY AND LITIGATION CLAIMS

PLAINTIFF CLAIMS:  DEFENDANT CLAIMS:

Schedule overrun Requirements changes
Cost overrun New demands by clients
Poor quality Rushed by clients
False claims Refusal to cooperate

PROBLEMS ON BOTH SIDES

Ambiguous clauses in contract
Informal software cost estimates
No formal quality estimates at all
No use of formal inspections
Inadequate milestone tracking
Friction and severe personal disputes
Independent audits too late to solve issues

INDEPENDENT ASSESSMENTS AND AUDITS

• Often used for military projects
• Can be an effective defense for litigation
• Effective quality assessments are formal
• Effective quality assessments cover defect prevention
• Effective quality assessments cover defect removal
• Effective quality assessments cover defect measures
• Effective assessments should cover 100% of projects
• Samples or partial assessments not safe for litigation
**OPTIMIZING QUALITY AND PRODUCTIVITY**

Projects that achieve 95% cumulative Defect Removal Efficiency will find:

1. Minimum schedules
2. Maximum productivity
3. High levels of user satisfaction
4. Low levels of delivered defects
5. Low levels of maintenance costs
6. Low risk of litigation

**ORIGINS OF SOFTWARE DEFECTS**

Because defect removal is such a major cost element, studying defect origins is a valuable undertaking.

<table>
<thead>
<tr>
<th>IBM Corporation (MVS)</th>
<th>SPR Corporation (client studies)</th>
</tr>
</thead>
<tbody>
<tr>
<td>45% Design errors</td>
<td>20% Requirements errors</td>
</tr>
<tr>
<td>25% Coding errors</td>
<td>30% Design errors</td>
</tr>
<tr>
<td>20% Bad fixes</td>
<td>35% Coding errors</td>
</tr>
<tr>
<td>5% Documentation errors</td>
<td>10% Bad fixes</td>
</tr>
<tr>
<td>5% Administrative errors</td>
<td>5% Documentation errors</td>
</tr>
<tr>
<td>100%</td>
<td>100%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TRW Corporation</th>
<th>Mitre Corporation</th>
<th>Nippon Electric Corp.</th>
</tr>
</thead>
<tbody>
<tr>
<td>60% Design errors</td>
<td>64% Design errors</td>
<td>60% Design errors</td>
</tr>
<tr>
<td>40% Coding errors</td>
<td>36% Coding errors</td>
<td>40% Coding errors</td>
</tr>
<tr>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
</tbody>
</table>

**FUNCTION POINTS AND DEFECT POTENTIALS**

Function points raised to the 1.15 power can predict the probable number of software defects. The range is from 1.1 to 1.25 power.

(Defects in requirements, design, code, documents, and bad fix categories.)

<table>
<thead>
<tr>
<th>FUNCTION POINTS</th>
<th>POTENTIAL DEFECTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>14</td>
</tr>
<tr>
<td>100</td>
<td>200</td>
</tr>
<tr>
<td>1,000</td>
<td>2,818</td>
</tr>
<tr>
<td>10,000</td>
<td>39,811</td>
</tr>
<tr>
<td>100,000</td>
<td>316,228</td>
</tr>
</tbody>
</table>

**SOFTWARE QUALITY AND PRODUCTIVITY**

- The most effective way of improving software productivity and shortening project schedules is to reduce defect levels.
- Defect reduction can occur through:
  1. Defect prevention technologies
  - Structured design and JAD
  - Structured code
  - Reuse of certified components
  2. Defect removal technologies
  - Design inspections
  - Code inspections
  - Formal Testing
DEFECT PREVENTION METHODS

DEFECT PREVENTION
- Joint Application Design (JAD)
- Quality function deployment (QFD)
- Software reuse (high-quality components)
- Root cause analysis
- Six-Sigma quality programs
- ISO 9000-9004 audits
- SEI CMM level greater than 2
- IBM “clean room” methods

DEFECT PREVENTION - Continued

DEFECT PREVENTION
- SEI CMM assessments
- SPR assessments
- TickIT assessments
- SPICE assessments
- Kaizen methodology
- Quality circles
- Independent Verification & Validation (IV&V)

DEFECT PREVENTION - Continued

DEFECT PREVENTION
- Total quality management (TQM)
- Quality measurements
- Orthogonal defect classification
- Defect tracking tools
- Formal design inspections
- Formal code inspections

DEFECT REMOVAL METHODS

DEFECT REMOVAL
- Requirements inspections
- Design inspections
- Test plan inspections
- Test case inspections
- Code inspections
- User manual inspections
- Data quality inspections
DEFECT REMOVAL - Continued

DEFECT REMOVAL
• Independent audits
• Testing: normal forms
• Testing: special forms
• Testing: user-based forms
• Testing: independent
• Testing: clean-room

DEFECT PREVENTION MATRIX

<table>
<thead>
<tr>
<th>Requirements</th>
<th>Design</th>
<th>Code</th>
<th>Document</th>
<th>Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>JAD’s</td>
<td>Excellent</td>
<td>Good</td>
<td>Not Applicable</td>
<td>Fair</td>
</tr>
<tr>
<td>Prototypes</td>
<td>Excellent</td>
<td>Excellent</td>
<td>Fair</td>
<td>Not Applicable</td>
</tr>
<tr>
<td>Structured Methods</td>
<td>Fair</td>
<td>Good</td>
<td>Excellent</td>
<td>Fair</td>
</tr>
<tr>
<td>ISO 9000-9004</td>
<td>Fair</td>
<td>Good</td>
<td>Fair</td>
<td>Fair</td>
</tr>
<tr>
<td>Blueprints &amp; Reusable Code</td>
<td>Excellent</td>
<td>Excellent</td>
<td>Excellent</td>
<td>Excellent</td>
</tr>
<tr>
<td>QFD</td>
<td>Good</td>
<td>Excellent</td>
<td>Fair</td>
<td>Poor</td>
</tr>
</tbody>
</table>

QUALITY MEASUREMENT EXCELLENCE

<table>
<thead>
<tr>
<th>Defect Estimation</th>
<th>Defect Tracking</th>
<th>Usability Measures</th>
<th>Complexity Measures</th>
<th>Test Coverage Measures</th>
<th>Removal Measures</th>
<th>Maintenance Measures</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Excellent</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>2. Good</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>3. Average</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>4. Marginal</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>5. Poor</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>
DEFECT REMOVAL EFFICIENCY

- Defect removal efficiency is a key quality measure
- Removal efficiency = Defects found / Defects present
- "Defects present" is the critical parameter

DEFECT REMOVAL EFFICIENCY - continued

First operation 6 defects from 10 or 60% efficiency
Second operation 2 defects from 4 or 50% efficiency
Cumulative efficiency 8 defects from 10 or 80% efficiency

Defect removal efficiency = Percentage of defects removed by a single level of review, inspection or test
Cumulative defect removal efficiency = Percentage of defects removed by a series of reviews, inspections or tests

DEFECT REMOVAL EFFICIENCY EXAMPLE

DEVELOPMENT DEFECTS
- Inspections 500
- Testing 400
- Subtotal 900

USER-REPORTED DEFECTS IN FIRST 90 DAYS
- Valid unique defects 100

TOTAL DEFECT VOLUME
- Defect totals 1000

REMOVAL EFFICIENCY
- Dev. (900) / Total (1000) = 90%

RANGES OF DEFECT REMOVAL EFFICIENCY

<table>
<thead>
<tr>
<th>Review Type</th>
<th>Lowest</th>
<th>Median</th>
<th>Highest</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Requirements review</td>
<td>20%</td>
<td>30%</td>
<td>50%</td>
</tr>
<tr>
<td>2 Top-level design reviews</td>
<td>30%</td>
<td>40%</td>
<td>60%</td>
</tr>
<tr>
<td>3 Detailed functional design reviews</td>
<td>30%</td>
<td>45%</td>
<td>65%</td>
</tr>
<tr>
<td>4 Detailed logic design reviews</td>
<td>35%</td>
<td>55%</td>
<td>75%</td>
</tr>
<tr>
<td>5 Code inspections</td>
<td>35%</td>
<td>60%</td>
<td>85%</td>
</tr>
<tr>
<td>6 Unit tests</td>
<td>10%</td>
<td>25%</td>
<td>50%</td>
</tr>
<tr>
<td>7 New Function tests</td>
<td>20%</td>
<td>35%</td>
<td>55%</td>
</tr>
<tr>
<td>8 Integration tests</td>
<td>25%</td>
<td>45%</td>
<td>60%</td>
</tr>
<tr>
<td>9 System test</td>
<td>25%</td>
<td>50%</td>
<td>65%</td>
</tr>
<tr>
<td>10 External Beta tests</td>
<td>15%</td>
<td>40%</td>
<td>75%</td>
</tr>
</tbody>
</table>

CUMULATIVE EFFICIENCY
- 75% 97% 99.99%
NORMAL DEFECT ORIGIN/DISCOVERY GAPS

Defect Origins
Requirements Design Coding Documentation Testing Maintenance

Defect Discovery
Requirements Design Coding Documentation Testing Maintenance

Zone of Chaos

DEFECT ORIGINS/DISCOVERY WITH INSPECTIONS

TECHNOLOGY COMBINATIONS   DEFECT REMOVAL EFFICIENCY

Lowest Median Highest

1. No Design Inspections
   No Code Inspections
   No Quality Assurance
   No Formal Testing

Lowest 30% 40% 50%

SOFTWARE DEFECT REMOVAL RANGES (cont.)

SINGLE TECHNOLOGY CHANGES

TECHNOLOGY COMBINATIONS   DEFECT REMOVAL EFFICIENCY

Lowest Median Highest

2. No design inspections
   No code inspections
   NO QUALITY ASSURANCE
   No formal testing

37% 53% 60%

3. No design inspections
   No code inspections
   No quality assurance
   FORMAL TESTING

43% 57% 65%

4. No design inspections
   FORMAL CODE INSPECTIONS
   No quality assurance
   No formal testing

5. FORMAL DESIGN INSPECTIONS
   No code inspections
   No quality assurance
   No formal testing

45% 60% 68%
### SOFTWARE DEFECT REMOVAL RANGES (cont.)

#### TWO TECHNOLOGY CHANGES

| TECHNOLOGY COMBINATIONS | DEFECT REMOVAL EFFICIENCY |  |
|-------------------------|---------------------------|
| Lowest                  | Median                    | Highest |
| 6. No design inspections| 50%                       | 65%     | 75%     |
| No code inspections     |                           |         |
| FORMAL QUALITY ASSURANCE|                           |         |
| FORMAL TESTING          |                           |         |
| 7. No design inspections| 53%                       | 68%     | 78%     |
| FORMAL CODE INSPECTIONS |                           |         |
| FORMAL QUALITY ASSURANCE|                           |         |
| No formal testing       |                           |         |
| 8. No design inspections| 55%                       | 70%     | 80%     |
| FORMAL CODE INSPECTIONS |                           |         |
| No quality assurance    |                           |         |
| FORMAL TESTING          |                           |         |

#### TWO TECHNOLOGY CHANGES - continued

| TECHNOLOGY COMBINATIONS | DEFECT REMOVAL EFFICIENCY |  |
|-------------------------|---------------------------|
| Lowest                  | Median                    | Highest |
| 9. FORMAL DESIGN INSPECTIONS | 60%    | 75%     | 85%     |
| No code inspections     |                           |         |
| FORMAL QUALITY ASSURANCE|                           |         |
| No formal testing       |                           |         |
| 10. FORMAL DESIGN INSPECTIONS | 65%    | 80%     | 87%     |
| No code inspections     |                           |         |
| No quality assurance    |                           |         |
| FORMAL TESTING          |                           |         |
| 11. FORMAL DESIGN INSPECTIONS | 70%    | 85%     | 90%     |
| FORMAL CODE INSPECTIONS |                           |         |
| No quality assurance    |                           |         |
| No formal testing       |                           |         |

#### THREE TECHNOLOGY CHANGES

| TECHNOLOGY COMBINATIONS | DEFECT REMOVAL EFFICIENCY |  |
|-------------------------|---------------------------|
| Lowest                  | Median                    | Highest |
| 12. No design inspections| 75%                       | 87%     | 93%     |
| FORMAL DESIGN INSPECTIONS|                           |         |
| FORMAL CODE INSPECTIONS |                           |         |
| FORMAL QUALITY ASSURANCE|                           |         |
| FORMAL TESTING          |                           |         |
| 13. FORMAL DESIGN INSPECTIONS | 77%   | 90%     | 95%     |
| No code inspections     |                           |         |
| FORMAL QUALITY ASSURANCE|                           |         |
| FORMAL TESTING          |                           |         |
| 14. FORMAL DESIGN INSPECTIONS | 83%  | 95%     | 97%     |
| FORMAL CODE INSPECTIONS |                           |         |
| FORMAL QUALITY ASSURANCE|                           |         |
| No formal testing       |                           |         |
| 15. FORMAL DESIGN INSPECTIONS | 85%  | 97%     | 99%     |
| FORMAL CODE INSPECTIONS |                           |         |
| No quality assurance    |                           |         |
| FORMAL TESTING          |                           |         |

#### BEST CASE RANGE

<table>
<thead>
<tr>
<th>TECHNOLOGY COMBINATIONS</th>
<th>DEFECT REMOVAL EFFICIENCY</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lowest</td>
<td>Median</td>
</tr>
<tr>
<td>1. FORMAL DESIGN INSPECTIONS</td>
<td>95%</td>
</tr>
</tbody>
</table>
PATTERNS OF SOFTWARE QUALITY

SOFTWARE QUALITY METHODS VARY BY CLASS:

1) Systems software
2) Embedded software
3) Military software
4) Commercial software
5) Outsourced software
6) Information Technology (IT) software
7) End-User developed personal software
8) Web-based software

PATTERNS OF SOFTWARE QUALITY

SYSTEMS SOFTWARE QUALITY METHODS

- Usually > 96% defect removal efficiency
- Overall, best software quality results
- Best quality results > 10,000 function points
- Formal design and code inspections
- Formal software quality assurance groups
- Formal software quality measurements
- Formal change control
- Formal test plans
- Unit test by developers
- 6 to 10 test stages by test specialists
- Use of six-sigma or SEI methods

PATTERNS OF SOFTWARE QUALITY

EMBEDDED SOFTWARE QUALITY METHODS

- Usually > 94% defect removal efficiency
- Most projects < 500 function points in size
- Wide range of software quality results
- Should use formal inspections, but may not
- Should use formal SQA teams, but may not
- Informal software quality measurements
- Should use formal change control
- Should use formal test plans
- Unit test by developers
- 3 to 6 test stages
- Should use test specialists, but may not
PATTERNS OF SOFTWARE QUALITY

MILITARY SOFTWARE QUALITY METHODS

• USUALLY > 95% DEFECT REMOVAL EFFICIENCY
• OVERALL GOOD SOFTWARE QUALITY RESULTS
• BEST QUALITY RESULTS > 100,000 FUNCTION POINTS
• FORMAL DESIGN AND CODE INSPECTIONS
• FORMAL SOFTWARE QUALITY ASSURANCE GROUPS
• FORMAL SOFTWARE QUALITY MEASUREMENTS
• FORMAL CHANGE CONTROL
• FORMAL TEST PLANS
• USE OF SEI ASSESSMENTS AND CMM APPROACHES
• 6 TO 15 TEST STAGES BY TEST SPECIALISTS
• ONLY CLASS TO USE INDEPENDENT VERIF. AND VALID.
• ONLY CLASS TO USE INDEPENDENT TESTING

PATTERNS OF SOFTWARE QUALITY

COMMERCIAL SOFTWARE QUALITY METHODS

• USUALLY > 90% DEFECT REMOVAL EFFICIENCY
• MOST PROJECTS > 5000 FUNCTION POINTS IN SIZE
• WIDE RANGE OF SOFTWARE QUALITY RESULTS
• SHOULD USE FORMAL INSPECTIONS, BUT MAY NOT
• SHOULD USE FORMAL SQA TEAMS, BUT MAY NOT
• INFORMAL SOFTWARE QUALITY MEASUREMENTS
• FORMAL CHANGE CONTROL
• FORMAL TEST PLANS
• UNIT TEST BY DEVELOPERS
• 3 TO 8 TEST STAGES
• SHOULD USE TEST SPECIALISTS, BUT MAY NOT
• OFTEN EXTENSIVE BETA TESTING BY USERS

PATTERNS OF SOFTWARE QUALITY

OUTSOURCE SOFTWARE QUALITY METHODS

• USUALLY > 94% DEFECT REMOVAL EFFICIENCY
• OVERALL BETTER SOFTWARE QUALITY THAN CLIENTS
• GOOD QUALITY > 1000 FUNCTION POINTS
• SHOULD USE FORMAL INSPECTIONS, BUT MAY NOT
• SHOULD USE FORMAL SQA GROUPS, BUT MAY NOT
• SHOULD USE FORMAL QUALITY MEASUREMENTS
• SHOULD USE FORMAL CHANGE CONTROL
• SHOULD USE FORMAL TEST PLANS
• UNIT TEST BY DEVELOPERS
• 4 TO 8 TEST STAGES BY TEST SPECIALISTS
• ACCEPTANCE TESTING BY CLIENTS
• MANY LATE CHANGES DEMANDED BY CLIENTS

PATTERNS OF SOFTWARE QUALITY

IT SOFTWARE QUALITY METHODS

• USUALLY < 90% DEFECT REMOVAL EFFICIENCY
• OFTEN MEDIOCRE SOFTWARE QUALITY
• POOR QUALITY > 1000 FUNCTION POINTS
• SELDOM USES FORMAL DESIGN AND CODE INSPECTIONS
• SELDOM USES FORMAL SQA GROUPS
• SELDOM USES SOFTWARE QUALITY MEASUREMENTS
• FORMAL CHANGE CONTROL
• INFORMAL TEST PLANS
• UNIT TEST BY DEVELOPERS
• 2 TO 6 TEST STAGES BY DEVELOPERS
• ACCEPTANCE TESTING BY CLIENTS
**PATTERNS OF SOFTWARE QUALITY**

**END-USER SOFTWARE QUALITY METHODS**

- Usually < 50% defect removal efficiency
- Often dangerously poor software quality
- All projects < 100 function points
- No use of formal design and code inspections
- No use of SQA groups
- No use of software quality measurements
- Informal change control
- Seldom any test plans
- Unit test by developer may be only test stage

**WEB SOFTWARE QUALITY METHODS**

- Usually < 90% defect removal efficiency
- Most projects < 1000 function points in size
- Wide range of software quality results
- Should use formal inspections, but may not need
- Web “content” is a special topic
- Informal software quality measurements
- Should use formal change control
- Should use formal test plans
- Unit test by developers
- 2 to 4 test stages
- Should use test specialists, but may not need

**CONCLUSIONS ON SOFTWARE QUALITY**

- No single method is adequate.
- Testing alone is insufficient.
- Formal inspections and tests combined give high efficiency, low costs and short schedules.
- Defect prevention plus inspections and tests give highest cumulative efficiency and best economics.
- Bad fix injection needs special solutions.
- Database errors need special solutions.
- Web “content” needs special solutions.

**END**