SOFTWARE QUALITY IN 2011:

A SURVEY OF THE STATE OF THE ART

Capers Jones, President

http://www.spr.com
Capers.Jones3@GMAILcom

June 11, 2011

ADVISORY BOARDS FOR CAPERS JONES

- **Chief Scientist Emeritus**
  Software Productivity Research LLC

- **Advisory Board**
  Software Improvement Group (SIG), Amsterdam

- **Advisor**
  Consortium for IT Software Quality (CISQ)

- **Advisor**
  Software Engineering Methods and Theory (SEMAT)

SOURCES OF QUALITY DATA

Data collected from 1984 through 2011

- About 675 companies (150 clients in Fortune 500 set)
- About 35 government/military groups
- About 13,500 total projects
- New data = about 50-75 projects per month
- Data collected from 24 countries
- Observations during more than 15 lawsuits

BASIC DEFINITIONS OF SOFTWARE QUALITY

- **Functional Software Quality**
  Software that combines low defect rates and high levels Of user satisfaction. The software should also meet all user requirements and adhere to international standards.

- **Structural Software Quality**
  Software that exhibits a robust architecture and can operate In a multi-tier environment without failures or degraded performance. Software has low cyclomatic complexity levels.

- **Aesthetic Software Quality**
  Software with elegant and easy to use commands and Interfaces, attractive screens, and well formatted outputs.

ECONOMIC DEFINITIONS OF SOFTWARE QUALITY

- **"Technical debt"**
  The assertion (by Ward Cunningham in 1992) that quick and careless development with poor quality leads To many years of expensive maintenance and enhancements.

- **Cost of Quality (COQ)**
  The overall costs of prevention, appraisal, internal failures, and external failures. For software these mean defect prevention, pre-test defect removal, testing, and post-release defect repairs. (Consequential damages are usually not counted.)

- **Total Cost of Ownership (TCO)**
  The sum of development + enhancement + maintenance + support from day 1 until application is retired. (Recalculation at 5 year intervals is recommended.)

SOFTWARE QUALITY HAZARDS IN TEN INDUSTRIES

INDUSTRY	HAZARD
Airlines	**Safety hazards**
Air traffic control problems	
Flight schedule confusion	
Navigation equipment failures	
Maintenance schedules thrown off	
Delay in opening Denver airport	
Passengers booked into non-existent seats	
Passengers misidentified as terror suspects	

SOFTWARE QUALITY HAZARDS IN TEN INDUSTRIES

INDUSTRY	HAZARD
Defense	**Security hazards**
Base security compromised	
Computer security compromised	
Strategic weapons malfunction	
Command, communication network problems	
Aircraft maintenance records thrown off	
Logistics and supply systems thrown off	
Satellites malfunction	

SOFTWARE QUALITY HAZARDS IN TEN INDUSTRIES

INDUSTRY	HAZARD
Finance	**Financial transaction hazards**
Interest calculations in error	
Account balances thrown off	
Credit card charges in error	
Funds transfer thrown off	
Mortgage/loan interest payments in error	
Hacking and identity theft due to software security flaws	
Denial of service attacks due to software security flaws	

SOFTWARE QUALITY HAZARDS IN TEN INDUSTRIES

INDUSTRY	HAZARD
Health Care	**Safety hazards**
Patient monitoring devices malfunction	
Operating room schedules thrown off	
Medical instruments malfunction	
Prescription refill problems	
Hazardous drug interactions	
Billing problems	
Medical records stolen or released by accident	

SOFTWARE QUALITY HAZARDS IN TEN INDUSTRIES

INDUSTRY	HAZARD
Insurance	**Liability, benefit hazards**
Policy due dates in error	
Policies cancelled in error	
Benefits and interest calculation errors	
Annuities miscalculated	
Errors in actuarial studies	
Payment records in error	

SOFTWARE QUALITY HAZARDS IN TEN INDUSTRIES

INDUSTRY	HAZARD
State, Local Governments	**Local economic hazards**
School taxes miscalculated	
Jury records thrown off	
Real-estate transactions misfiled	
Divorce, marriage records misfiled	
Alimony, child support payment records lost	
Death records filed for wrong people	
Traffic light synchronization thrown off	
Errors in property tax assessments	

SOFTWARE QUALITY HAZARDS IN TEN INDUSTRIES

INDUSTRY	HAZARD
Manufacturing	**Operational hazards**
Subcontract parts fail to arrive	
Purchases of more or less than economic order quantities	
Just-in-time arrivals thrown off	
Assembly lines shut down	
Aging errors for accounts receivable and cash flow	
Aging errors for accounts payable and cash flow	
Pension payments miscalculated	

SOFTWARE QUALITY HAZARDS IN TEN INDUSTRIES

INDUSTRY	HAZARD
National Government	Citizen record hazards
	Tax records in error
	Annuities and entitlements miscalculated
	Social Security payments miscalculated or cancelled
	Disbursements miscalculated
	Retirement benefits miscalculated
	Personal data stolen or released by accident

SOFTWARE QUALITY HAZARDS IN TEN INDUSTRIES

INDUSTRY	HAZARD
Public Utilities	**Safety hazards**
Electric meters malfunction	
Gas meters malfunction	
Distribution of electric power thrown off	
Billing records in error	
Nuclear power plants malfunction	

SOFTWARE QUALITY HAZARDS IN TEN INDUSTRIES

INDUSTRY	HAZARD
Telecommunications	**Service disruption hazards**
Intercontinental switching disrupted	
Domestic call switching disrupted	
Billing records in error	

SOFTWARE QUALITY HAZARDS ALL INDUSTRIES

1. **Software is blamed for more major business problems than any other man-made product.**
2. **Poor software quality has become one of the most expensive topics in human history: > $150 billion per year in U.S.; > $500 billion per year world wide.**
3. **Projects cancelled due to poor quality >15% more costly than successful projects of the same size and type.**
4. **Software executives, managers, and technical personnel are regarded by many CEO's as a painful necessity rather than top professionals.**
5. **Improving software quality is a key topic for all industries.**

FUNDAMENTAL SOFTWARE QUALITY METRICS

- **Defect Potentials**
  - **Sum of requirements errors, design errors, code errors, document errors, bad fix errors, test plan errors, and test case errors**
- **Defect Discovery Efficiency (DDE)**
  - **Percent of defects discovered before release**
- **Defect Removal Efficiency (DRE)**
  - **Percent of defects removed before release**
- **Defect Severity Levels (Valid unique defects)**

  Severity 1 = Total stoppage
  Severity 2 = Major error
  Severity 3 = Minor error
  Severity 4 = Cosmetic error

FUNDAMENTAL SOFTWARE QUALITY METRICS (cont.)

- **Standard Cost of Quality**
  - **Prevention**
  - **Appraisal**
  - **Internal failures**
  - **External failures**
- **Revised Software Cost of Quality**
  - **Defect Prevention**
  - **Pre-Test Defect Removal (inspections, static analysis)**
  - **Testing Defect Removal**
  - **Post-Release Defect Removal**
- **Error-Prone Module Effort**
  - **Identification**
  - **Removal or redevelopment**
  - **repairs and rework**

QUALITY MEASUREMENT PROBLEMS

- **Cost per defect penalizes quality!**
- **(Buggiest software has lowest cost per defect!)**
- **Lines of code penalize high-level languages!**
- **Lines of code ignore non-coding defects!**
- **Most companies don't measure all defects!**
- **Most common omissions are requirement bugs, design bugs, and bugs found by desk checks and unit testing. Real bugs can outnumber measured bugs by more than 5 to 1!**

COST PER DEFECT PENALIZES QUALITY

	Case A High quality	Case B Low quality
Defects found	50	500
Test case creation	$10,000	$10,000
Test case execution	$10,000	$10,000
Defect repairs	$10,000	$70,000
TOTAL	$30,000	$90,000
Cost per Defect	$600	$180
$ Cost savings	$60,000	$0.00

A BASIC LAW OF MANUFACTURING ECONOMICS

"If a manufacturing cycle has a high proportion of fixed costs and there is a decline in the number of units produced the cost per unit will go up."

1. **As quality improves the number of defects goes down.**
2. **Test preparation and test execution act like fixed costs.**
3. **Therefore the "cost per defect" must go up.**
4. **Late defects must cost more than early defects.**
5. **Defects in high quality software cost more than in bad quality software.**

LINES OF CODE HARM HIGH-LEVEL LANGUGES

	Case A JAVA	Case B C
KLOC	50	125
Function points	1,000	1,000
Code defects found	500	1,250
Defects per KLOC	10.00	10.00
Defects per FP	0.5	1.25
Defect repairs	$70,000	$175,000
$ per KLOC	$1,400	$1,400
$ per Defect	$140	$140
$ per Function Point	$70	$175
$ cost savings	$105,000	$0.00

A BASIC LAW OF MANUFACTURING ECONOMICS

"If a manufacturing cycle has a high proportion of fixed costs and there is a decline in the number of units produced the cost per unit will go up."

1) As language levels go up the number of lines of code produced comes down.

2) The costs of requirements, architecture, design, and documentation act as fixed costs.

3) Therefore the "cost per line of code" must go up.

4) Cost per line of code penalizes languages in direct proportion to their level.

U.S. AVERAGES FOR SOFTWARE QUALITY

(Data expressed in terms of defects per function point)

Defect Origins	Defect Potential	Removal Efficiency	Delivered Defects
Requirements	1.00	77%	0.23
Design	1.25	85%	0.19
Coding	1.75	95%	0.09
Documents	0.60	80%	0.12
Bad Fixes	0.40	70%	0.12
TOTAL	5.00	85%	0.75

(Function points show all defect sources - not just coding defects)
(Code defects = 35% of total defects)

BEST IN CLASS SOFTWARE QUALITY

(Data expressed in terms of defects per function point)

Defect Origins	Defect Potential	Removal Efficiency	Delivered Defects
Requirements	0.40	85%	0.08
Design	0.60	97%	0.02
Coding	1.00	99%	0.01
Documents	0.40	98%	0.01
Bad Fixes	0.10	95%	0.01
TOTAL	2.50	96%	0.13

OBSERVATIONS

(Most often found in systems software > SEI CMM Level 3 or in TSP projects)

POOR SOFTWARE QUALITY - MALPRACTICE

(Data expressed in terms of defects per function point)

Defect Origins	Defect Potential	Removal Efficiency	Delivered Defects
Requirements	1.50	50%	0.75
Design	2.20	50%	1.10
Coding	2.50	80%	0.50
Documents	1.00	70%	0.30
Bad Fixes	0.80	50%	0.40
TOTAL	8.00	62%	3.05

OBSERVATIONS

(Most often found in large water fall projects > 10,000 Function Points).

GOOD QUALITY RESULTS > 90% SUCCESS RATE

- **Formal Inspections (Requirements, Design, and Code)**
- **Static analysis (for about 25 languages out of 2,500 in all)**
- **Joint Application Design (JAD)**
- **Functional quality metrics using function points**
- **Structural quality metrics such as cyclomatic complexity**
- **Defect Detection Efficiency (DDE) measurements**
- **Defect Removal Efficiency (DRE) measurements**
- **Automated Defect tracking tools**
- **Active Quality Assurance (> 3% SQA staff)**
- **Utilization of effective methods (Agile, XP, RUP, TSP, etc.)**
- **Mathematical test case design based on design of experiments**
- **Quality estimation tools**
- **Testing specialists (certified)**
- **Root-Cause Analysis**
- **Quality Function Deployment (QFD)**

MIXED QUALITY RESULTS: < 50% SUCCESS RATE

- **CMMI level 3 or higher (some overlap among CMMI levels: Best CMMI 1 groups better than worst CMMI 3 groups)**
- **ISO and IEEE quality standards (Prevent low quality; Little benefit for high-quality teams)**
- **Six-Sigma methods (unless tailored for software projects)**
- **Independent Verification & Validation (IV & V)**
- **Quality circles in the United States (more success in Japan)**
- **Clean-room methods for rapidly changing requirements**
- **Kaizan (moving from Japan to U.S. and elsewhere)**
- **Cost of quality without software modifications**

POOR QUALITY RESULTS: < 25% SUCCESS RATE

- **Testing as only form of defect removal**
- **Informal Testing and uncertified test personnel**
- **Testing only by developers; no test specialists**
- **Passive Quality Assurance (< 3% QA staff)**
- **Token Quality Assurance (< 1% QA staff)**
- **LOC Metrics for quality (omits non-code defects)**
- **Cost per defect metric (penalizes quality)**
- **Failure to estimate quality or risks early**
- **Quality measurement "leakage" such as unit test bugs**

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 75% efficient (1 of 4 bugs remain)**
- **Design Reviews and Code Inspections -- often more than 65% efficient; have topped 90%**
- **Static analysis –often more than 65% efficient; has topped 95%**
- **Inspections, static analysis, and testing combined lower costs and schedules by > 20%; lower total cost of ownership (TCO) by > 45%.**

SOFTWARE DEFECT ORIGINS

1.	Requirements	Hardest to prevent and repair
2.	Requirements creep	Very troublesome source of bugs
3.	Architecture	Key to structural quality
4.	Design	Most severe and pervasive
5.	Source code	Most numerous; easiest to fix
6.	Security flaws	Hard to find and hard to fix
7.	Documentation	Can be serious if ignored
8.	Bad fixes	Very difficult to find
9.	Bad test cases	Numerous but seldom measured
10.	Data errors	Very common but not measured
11.	Web content	Very common but not measured
12.	Structure	Hard to find by testing; inspections and static analysis can identify multi-tier platform defects

TOTAL SOFTWARE DEFECTS IN RANK ORDER

Defect Origins	Defects per Function Point
1. Data defects	2.50 *
2. Code defects	1.75
3. Test case defects	1.65 *
4. Web site defects	1.40 *
5. Design defects	1.25 **
6. Requirement Defects	1.00 **
7. Structural defects	0.70 **
8. Document defects	0.60 **
9. Bad-fix defects	0.40 **
10. Requirement creep defects	0.30 **
11. Security defects	0.25 **
12. Architecture Defects	0.20 *
TOTAL DEFECTS	12.00

* NOTE 1: Usually not measured due to lack of size metrics

** NOTE 2: Often omitted from defect measurements

ORIGINS OF HIGH-SEVERITY SOFTWARE DEFECTS

Defect Origins	Percent of Severity 1 and 2 Defects
1. Design defects	17.00%
2. Code defects	15.00%
3. Structural defects	13.00%
4. Data defects	11.00%
5. Requirements creep defects	10.00%
6. Requirements defects	9.00%
7. Web site defects	8.00%
8. Security defects	7.00%
9. Bad fix defects	4.00%
10. Test case defects	2.00%
11. Document defects	2.00%
12. Architecture Defects	2.00%
TOTAL DEFECTS	100.00%

Severity 1 = total stoppage; Severity 2 = major defects

ORIGINS OF LOW-SEVERITY SOFTWARE DEFECTS

Defect Origins	Percent of Severity 3 and 4 Defects
1. Code defects	35.00%
2. Data defects	20.00%
3. Web site defects	10.00%
4. Design defects	7.00%
5. Structural defects	6.00%
6. Requirements defects	4.00%
7. Requirements creep defects	4.00%
8. Security defects	4.00%
9. Bad fix defects	4.00%
10. Test case defects	3.00%
11. Document defects	2.00%
12. Architecture Defects	1.00%
TOTAL DEFECTS	100.00%

Severity 3 = minor defects; Severity 4 = cosmetic defects

ORIGINS OF DUPLICATE DEFECTS

Defect Origins	Percent of Duplicate Defects (Many reports of the same bugs)
1. Code defects	30.00%
2. Structural defects	20.00%
3. Data defects	20.00%
4. Web site defects	10.00%
5. Security defects	4.00%
6. Requirements defects	3.00%
7. Design defects	3.00%
8. Bad fix defects	3.00%
9. Requirements creep defects	2.00%
10. Test case defects	2.00%
11. Document defects	2.00%
12. Architecture Defects	1.00%
TOTAL DEFECTS	100.00%

Duplicate = Multiple reports for the same bug (> 10,000 can occur)

ORIGINS OF INVALID DEFECTS

Defect Origins	Percent of Invalid Defects (Defects not caused by software itself)
1. Data defects	25.00%
2. Structural defects	20.00%
3. Web site defects	13.00%
4. User errors	12.00%
5. Document defects	10.00%
6. External software	10.00%
7. Requirements creep defects	3.00%
8. Requirements defects	1.00%
9. Code defects	1.00%
10. Test case defects	1.00%
11. Security defects	1.00%
12. Design defects	1.00%
13. Bad fix defects	1.00%
14. Architecture Defects	1.00%
TOTAL DEFECTS	100.00%

Invalid = Defects caused by platforms or external software applications

WORK HOURS AND COSTS FOR DEFECT REPAIRS

	Defect Origins	Work Hours		Costs ($75 per hour)	
1.	Security defects		10.00		$750.00
2.	Design defects		8.50		$637.50
3.	Requirements creep defects		8.00		$600.00
4.	Requirements defects		7.50		$562.50
5.	Structural defects		7.25		$543.75
6.	Architecture defects	7.00		$525.00	
7.	Data defects		6.50		$487.50
8.	Bad fix defects		6.00		$450.00
9.	Web site defects		5.50		$412.50
10.	Invalid defects		4.75		$356.25
11.	Test case defects		4.00		$300.00
12.	Code defects		3.00		$225.00
13.	Document defects		1.75		$131.50
14.	Duplicate defects		1.00		$75.00
	AVERAGES	5.77		$432.69	

Maximum can be > 10 times greater

DEFECT DAMAGES AND RECOVERY COSTS

Defect Origins

1.	**Security defects**	**$200,000,000**
2.	**Design defects**	**$175,000,000**
3.	**Requirements defects**	**$150,000,000**
4.	**Data defects**	**$125,000,000**
5.	**Code defects**	**$100,000,000**
6.	**Structural defects**	**$95,000,000**
7.	**Requirements creep defects**	**$90,000,000**
8.	**Web site defects**	**$80,000,000**
9.	**Architecture defects**	**$80,000,000**
10.	**Bad fix defects**	**$60,000,000**
11.	**Test case defects**	**$50,000,000**
12.	**Document Defects**	**$25,000,000**
AVERAGES		**$102,500,000**

Defect recovery costs for major applications in large companies and government agencies

WORK HOURS AND COSTS BY SEVERITY

Defect Severity	Work Hours	Costs ($75 per hour)
Severity 1 (total stoppage)	6.00	$450.00
Severity 2 (major errors)	9.00	$675.00
Severity 3 (minor errors)	3.00	$225.00
Severity 4 (cosmetic errors)	1.00	$75.00
Abeyant defects (special case)	40.00	$3,000.00
Invalid defects	4.75	$355.25
Duplicate defects	1.00	$75.00

Maximum can be > 10 times greater

DEFECT REPAIRS BY APPLICATION SIZE

Function. Points	Sev 1 Hours	Sev 2 Hours	Sev 3 Hours	Sev 4 Hours	AVERAGE HOURS
10	2.00	3.00	1.50	0.50	1.75
100	4.00	6.00	2.00	0.50	3.13
1000	6.00	9.00	3.00	1.00	4.75
10000	8.00	12.00	4.00	1.50	6.38
100000	18.00	24.00	6.00	2.00	12.50

Function Points	Sev 1 $	Sev 2 $	Sev 3 $	Sev 4 $	AVERAGE COSTS
10	$150	$220	$112	$38	$132
100	$300	$450	$150	$38	$234
1000	$450	$675	$225	$75	$356
10000	$600	$900	$300	$113	$478
100000	$1350	$1800	$450	$150	$938

DEFECT REPORTS IN FIRST YEAR OF USAGE

Function Points	10	100	1000	10,000	100,000
Users					
1	55%	27%	12%	3%	1%
10	65%	35%	17%	7%%	3%
100	75%	42%	20%	10%	7%
1000	85%	50%	***27%***	12%	10%
10,000	95%	75%	35%	20%	12%
100,000	99%	87%	45%	35%	20%
1,000,000	100%	96%	77%	45%	32%
10,000,000	100%	100%	90%	65%	45%

ELAPSED TIME IN DAYS FOR DEFECT RESOLUTION

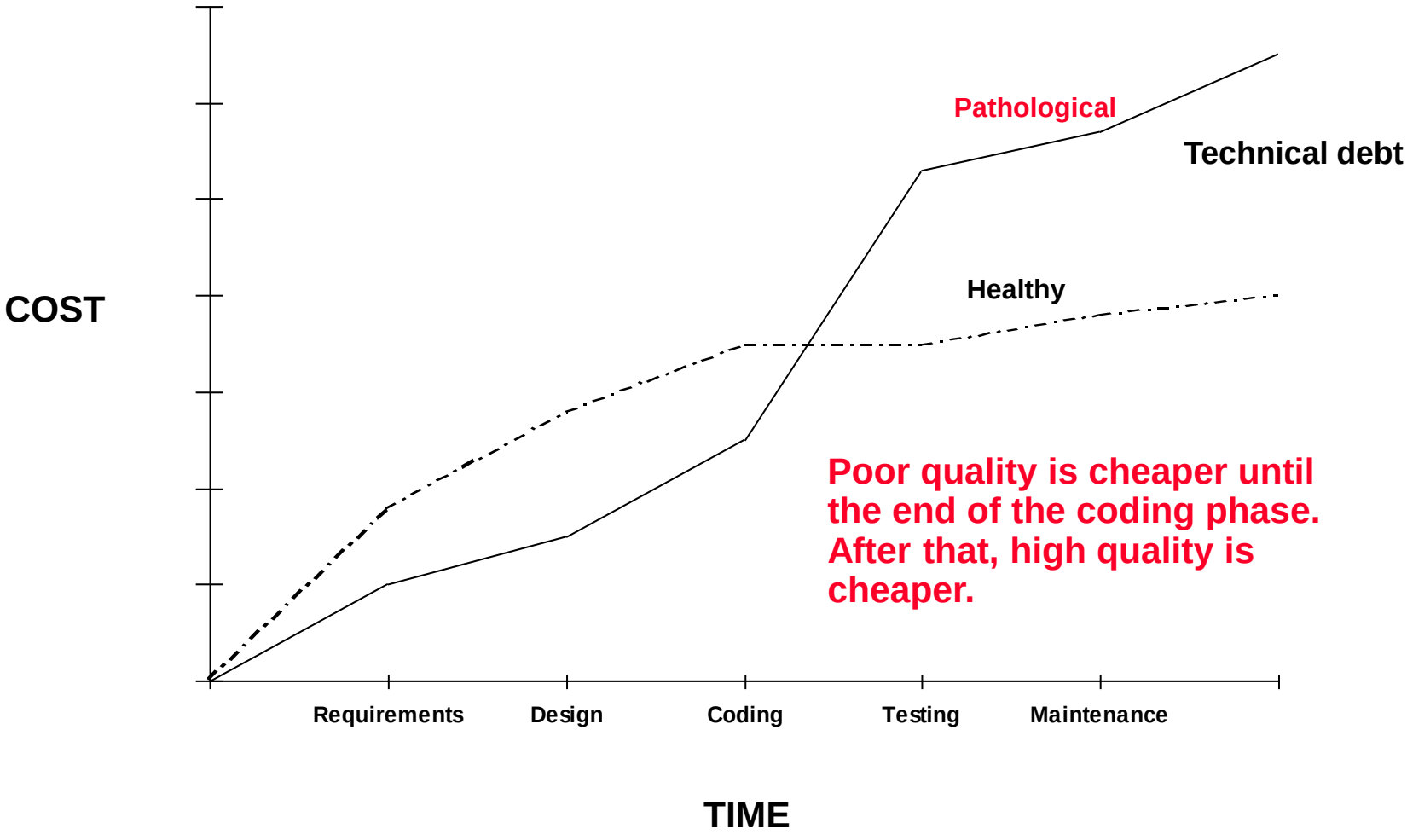
Removal method	Stat. Analy.	Unit Test	Inspect.	Funct. Test	Sys. Test	Maint.
Preparation	1	2	5	6	8	7
Execution	1	1	2	4	6	3
Repair	1	1	1	1	2	2
Validate	1	1	1	2	4	5
Integrate	1	1	1	2	4	6
Distribute	1	1	1	2	3	7
TOTAL DAYS	6	7	11	17	27	30

Defect repairs take < 12% of elapsed time

SOFTWARE DEFECT SEVERITY CATEGORIES

Severity 1:	**TOTAL FAILURE S**	**1% at release**
Severity 2:	**MAJOR PROBLEMS**	**20% at release**
Severity 3:	**MINOR PROBLEMS**	**35% at release**
Severity 4:	**COSMETIC ERRORS**	**44% at release**
STRUCTURAL	**MULTI-TIER DEFECTS**	**15% of reports**
INVALID	**USER OR SYSTEM ERRORS**	**15% of reports**
DUPLICATE	**MULTIPLE REPORTS**	**30% of reports**
ABEYANT	**CAN'T RECREATE ERROR**	**5% of reports**

HOW QUALITY AFFECTS SOFTWARE COSTS



U. S. SOFTWARE QUALITY AVERAGES CIRCA 2011

(Defects per Function Point)

	System Software	Commercial Software	Information Software	Military Software	Outsource Software
Defect Potentials	6.0	5.0	4.5	7.0	5.2
Defect Removal Efficiency	94%	90%	73%	96%	92%
Delivered Defects	0.36	0.50	1.22	0.28	0.42
First Year Discovery Rate	65%	70%	30%	75%	60%
First Year Reported Defects	0.23	0.35	0.36	0.21	0.25

U. S. SOFTWARE QUALITY AVERAGES CIRCA 2011

(Defects per Function Point)

	Web Software	Embedded Software	SEI-CMM 3 Software	SEI-CMM 1 Software	Overall Average
Defect Potentials	4.0	5.5	5.0	5.75	5.1
Defect Removal Efficiency	72%	95%	95%	83%	86.7%
Delivered Defects	1.12	0.3	0.25	0.90	0.68
First Year Discovery Rate	95%	90%	60%	35%	64.4%
First Year Reported Defects	1.06	0.25	0.15	0.34	0.42

SOFTWARE SIZE VS DEFECT REMOVAL EFFICIENCY

(Data Expressed in terms of Defects per Function Point)

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

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)

SEI CMM Levels	Defect Potentials	Removal Efficiency	Delivered Defects
SEI CMMI 1	5.25	80%	1.05
SEI CMMI 2	5.00	85%	0.75
SEI CMMI 3	4.75	90%	0.48
SEI CMMI 4	4.50	93%	0.32
SEI CMMI 5	4.25	96%	0.17

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

(Data Expressed in Terms of Defects per Function Point
For projects 10,000 function points in size)

SEI CMM Levels	Defect Potentials	Removal Efficiency	Delivered Defects
SEI CMMI 1	6.50	75%	1.63
SEI CMMI 2	6.25	82%	1.13
SEI CMMI 3	5.50	87%	0.71
SEI CMMI 4	5.25	90%	0.53
SEI CMMI 5	4.75	94%	0.29

DEFECTS AND SOFTWARE METHODOLGOIES

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

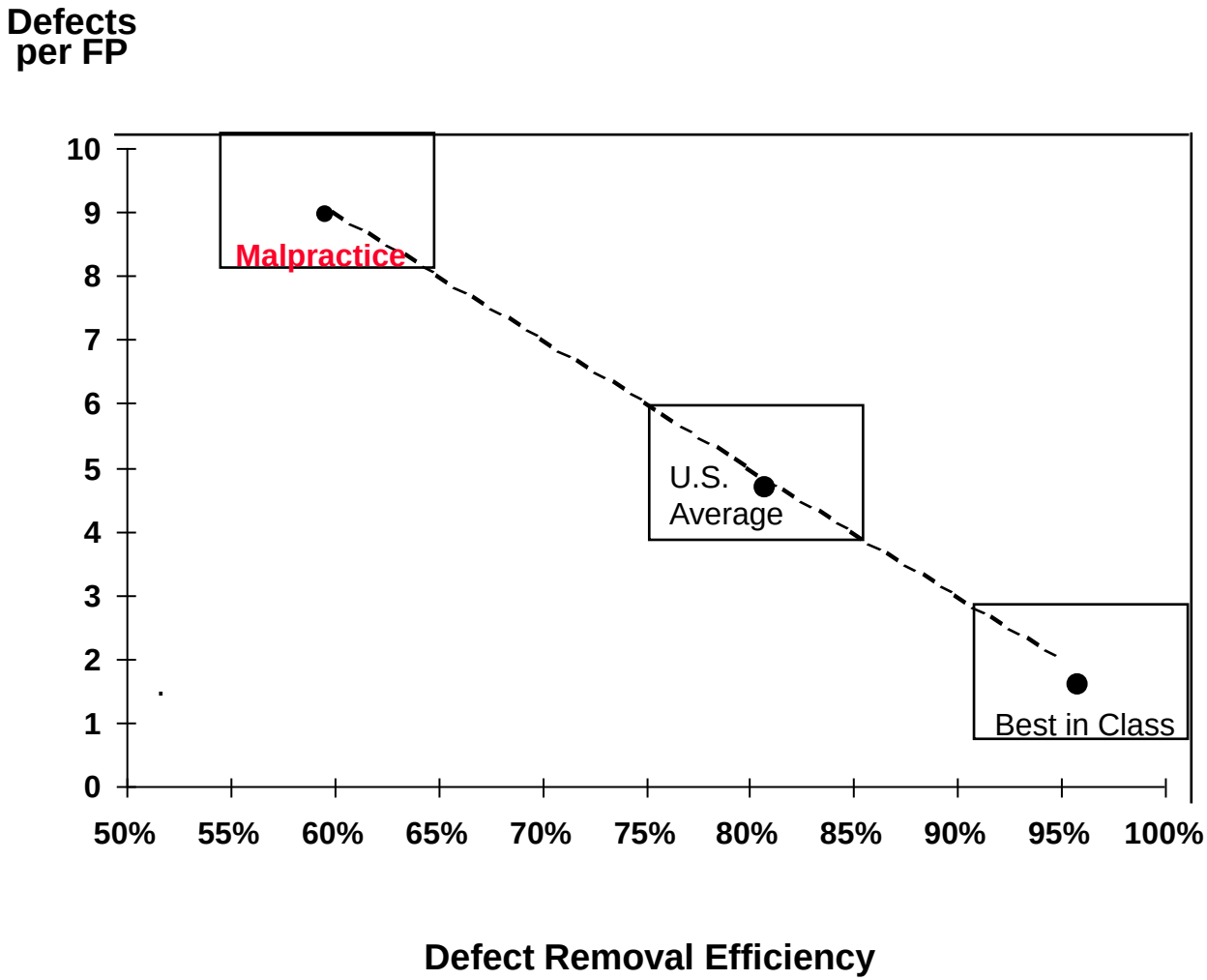
Software methods	Defect Potential	Removal Efficiency	Delivered Defects
Waterfall	5.50	80%	1.10
Iterative	4.75	87%	0.62
Object-Oriented	4.50	88%	0.54
Rational Unified Process (RUP)	4.25	92%	0.34
Agile	4.00	90%	0.40
PSP and TSP	3.50	96%	0.14
85% Certified reuse	1.75	99%	0.02

DEFECTS AND SOFTWARE METHODOLGOIES

(Data Expressed in Terms of Defects per Function Point
For projects nominally 10,000 function points in size)

Software methods	Defect Potential	Removal Efficiency	Delivered Defects
Waterfall	7.00	75%	1.75
Iterative	6.25	82%	1.13
Object-Oriented	5.75	85%	0.86
Rational Unified Process (RUP)	5.50	90%	0.55
Agile	5.50	87%	0.72
PSP and TSP	5.00	94%	0.30
85% Certified reuse	2.25	96%	0.09

MAJOR SOFTWARE QUALITY ZONES



INDUSTRY-WIDE DEFECT CAUSES

Ranked in order of effort required to fix the defects:

1. **Requirements problems (omissions; changes, errors)**
2. **Design problems (omissions; changes; errors)**
3. **Security flaws and vulnerabilities**
4. **Interface problems between modules**
5. **Logic, branching, and structural problems**
6. **Memory allocation problems**
7. **Testing omissions and poor coverage**
8. **Test case errors**
9. **Stress/performance problems**
10. **Bad fixes/Regressions**

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 and team satisfaction**
4) **Low levels of delivered defects**
5) **Low levels of maintenance costs**
6) **Low risk of litigation**

INDUSTRY DATA ON DEFECT ORIGINS

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

IBM Corporation (MVS)

45%	Design errors
25%	Coding errors
20%	Bad fixes
5%	Documentation errors
5%	Administrative errors
100%	

SPR Corporation (client studies)

20%	Requirements errors
30%	Design errors
35%	Coding errors
10%	Bad fixes
5%	Documentation errors
100%	

TRW Corporation

60%	Design errors
40%	Coding errors
100%	

Mitre Corporation

64%	Design errors
36%	Coding errors
100%	

Nippon Electric Corp.

60%	Design errors
40%	Coding errors
100%	

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 <u>prevention</u> technologies**
     - **Structured design and JAD**
     - **Structured code**
     - **Use of inspections, static analysis**
     - **Reuse of certified components**

  2. **Defect <u>removal</u> technologies**
     - **Design inspections**
     - **Code inspections, static analysis**
     - **Formal Testing using mathematical test case design**

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 for software**
- **Usage of TSP/PSP methods**
- **Climbing > Level 3 on the SEI CMMI**
- **Static analysis, inspections**

DEFECT PREVENTION - Continued

DEFECT PREVENTION

- **Life-cycle quality measurements**
- **Kaizen, Poka Yoke, Kanban, Quality Circles (from Japan)**
- **Prototypes of final application (disposable are best)**
- **Defect tracking tools**
- **Formal design inspections**
- **Formal code inspections**
- **Embedding users with development team (Agile methods)**
- **SCRUM (issue-oriented team meetings)**

DEFECT REMOVAL METHODS

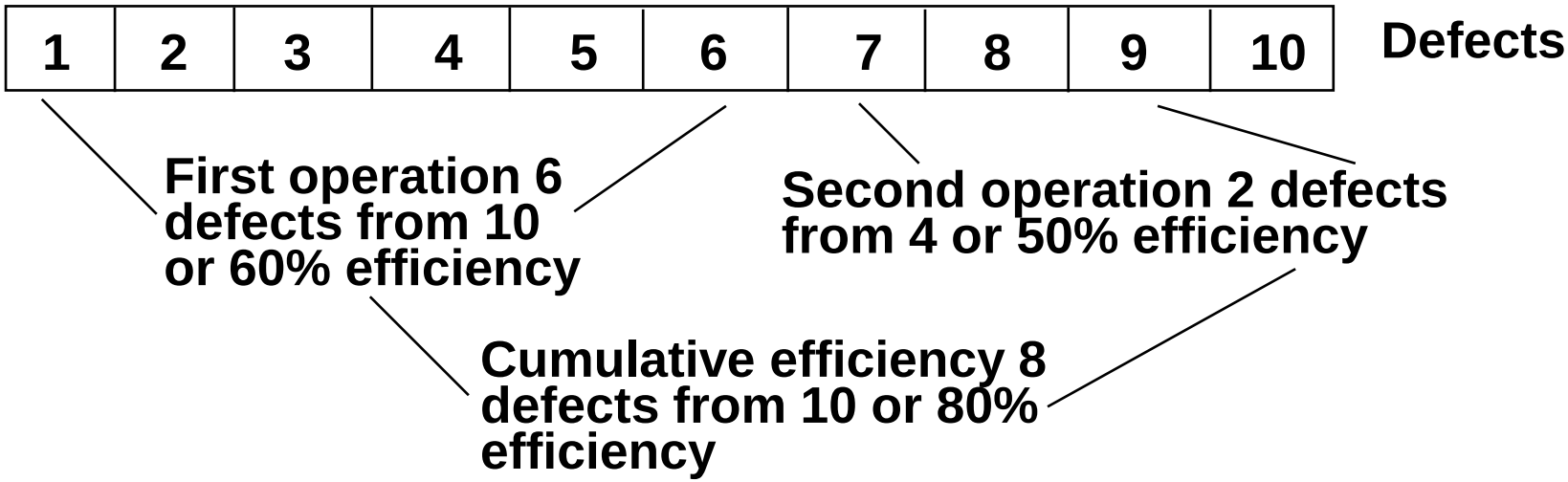
DEFECT REMOVAL

- **Requirements inspections**
- **Design inspections**
- **Test plan inspections**
- **Test case inspections**
- **Static analysis (C, Java, COBOL, SQL etc.)**
- **Code inspections**
- **Automated testing (unit, performance)**
- **All forms of manual testing (more than 40 kinds of test)**

DEFECT REMOVAL EFFICIENCY

- **Defect removal efficiency is a key quality measure**
- **Removal efficiency =** $\dfrac{\text{Defects found}}{\text{Defects present}}$
- **"Defects present" is the critical parameter**

DEFECT REMOVAL EFFICIENCY - continued



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

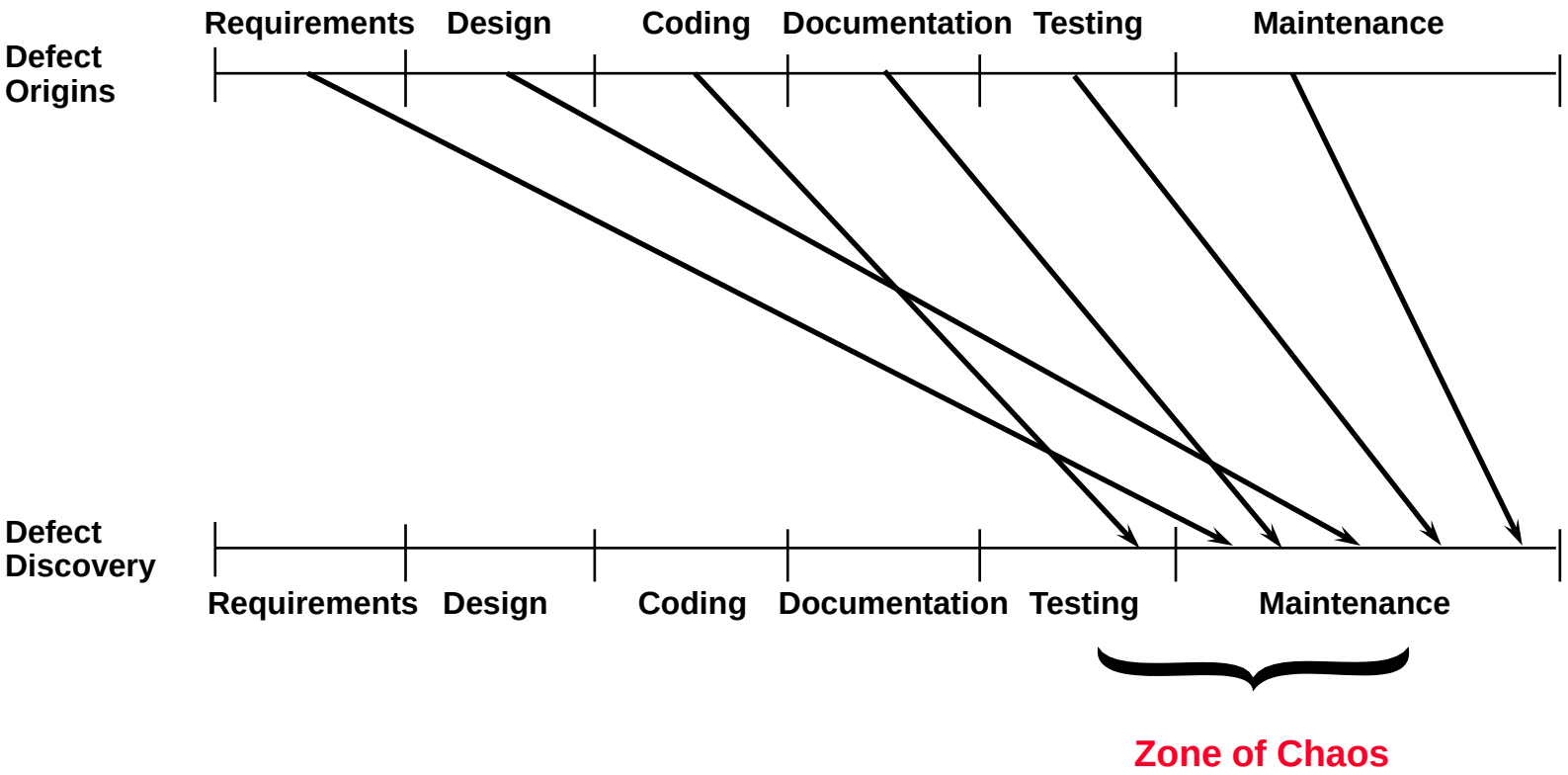
DEFECT REMOVAL EFFICIENCY EXAMPLE

DEVELOPMENT DEFECTS REMOVED		
Inspections		**350**
Static analysis		**300**
Testing		**250**
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%**

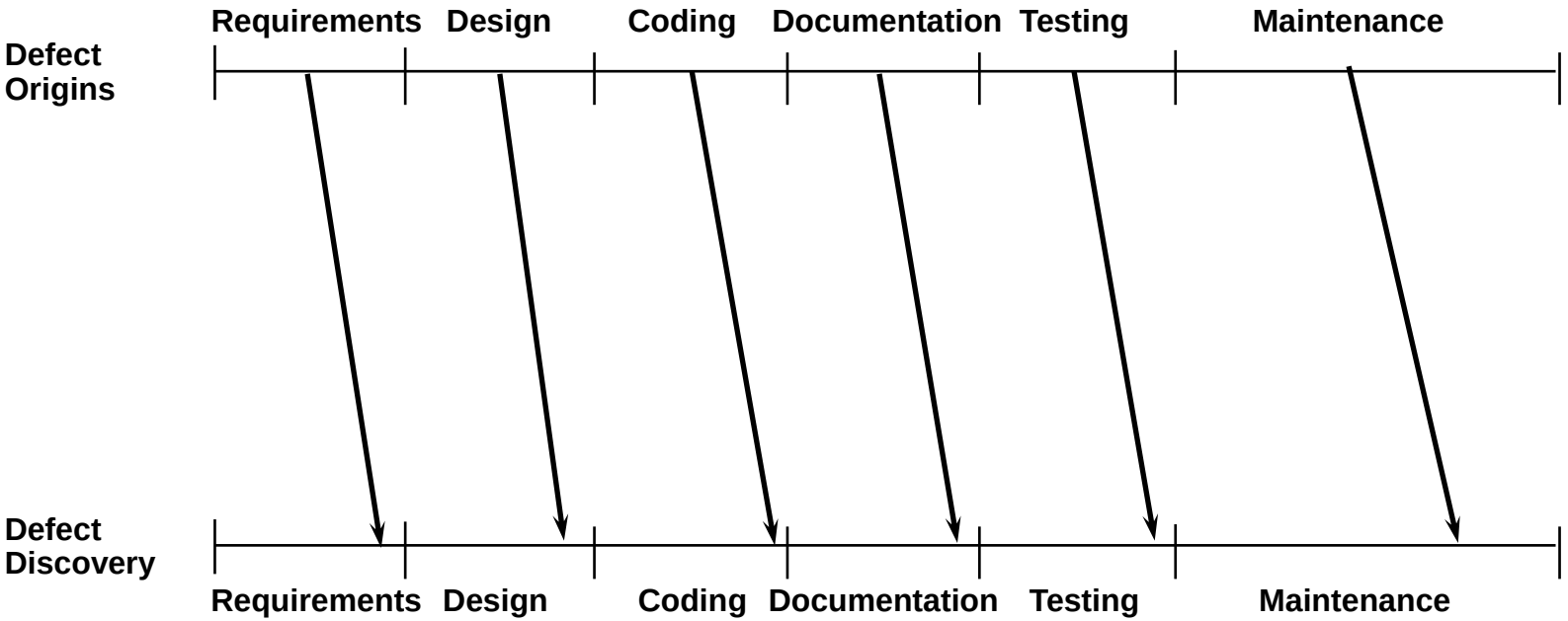
RANGES OF DEFECT REMOVAL EFFICIENCY

	Lowest	Median	Highest
1 Requirements review (informal)	20%	30%	50%
2 Top-level design reviews (informal)	30%	40%	60%
3 Detailed functional design inspection	30%	65%	85%
4 Detailed logic design inspection	35%	65%	75%
5 Code inspection or static analysis	35%	60%	90%
6 Unit tests	10%	25%	50%
7 New Function tests	20%	35%	65%
8 Integration tests	25%	45%	60%
9 System test	25%	50%	65%
10 External Beta tests	15%	40%	75%
CUMULATIVE EFFICIENCY	75%	98%	99.99%

NORMAL DEFECT ORIGIN/DISCOVERY GAPS



DEFECT ORIGINS/DISCOVERY WITH INSPECTIONS



SOFTWARE DEFECT REMOVAL RANGES

WORST CASE RANGE

TECHNOLOGY COMBINATIONS	DEFECT REMOVAL EFFICIENCY		
	Lowest	Median	Highest
1. No Design Inspections No Code Inspections or static analysis No Quality Assurance No Formal Testing	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 or static analysis FORMAL QUALITY ASSURANCE No formal testing	32%	45%	55%
3. No design inspections No code inspections or static analysis No quality assurance FORMAL TESTING	37%	53%	60%
4. No design inspections CODE INSPECTIONS/STATIC ANALYSIS No quality assurance No formal testing	43%	57%	65%
5. FORMAL DESIGN INSPECTIONS No code inspections or static analysis 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 No code inspections or static analysis FORMAL QUALITY ASSURANCE FORMAL TESTING	50%	65%	75%
7. No design inspections FORMAL CODE INSPECTIONS/STAT. AN. FORMAL QUALITY ASSURANCE No formal testing	53%	68%	78%
8. No design inspections FORMAL CODE INSPECTIONS/STAT.AN. No quality assurance FORMAL TESTING	55%	70%	80%

SOFTWARE DEFECT REMOVAL RANGES (cont.)

TWO TECHNOLOGY CHANGES - continued

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

SOFTWARE DEFECT REMOVAL RANGES (cont.)

THREE TECHNOLOGY CHANGES

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

SOFTWARE DEFECT REMOVAL RANGES (cont.)

BEST CASE RANGE

TECHNOLOGY COMBINATIONS	DEFECT REMOVAL EFFICIENCY		
	Lowest	Median	Highest
16. FORMAL DESIGN INSPECTIONS STATIC ANALYSIS FORMAL CODE INSPECTIONS FORMAL QUALITY ASSURANCE FORMAL TESTING	95%	99%	99.99%

DISTRIBUTION OF 1500 SOFTWARE PROJECTS BY DEFECT REMOVAL EFFICIENCY LEVEL

Defect Removal Efficiency Level (Percent)	Number of Projects	Percent of Projects
> 99	6	0.40%
95 - 99	104	6.93%
90 - 95	263	17.53%
85 - 90	559	37.26%
80 - 85	408	27.20%
< 80	161	10.73%
Total	1,500	100.00%

SOFTWARE QUALITY UNKNOWNS IN 2011

SOFTWARE QUALITY TOPICS NEEDING RESEARCH:

Errors in software test plans and test cases

Errors in web content such as graphics and sound

Correlations between security flaws and quality flaws

Errors in data and creation of a "data point" metric

Error content of data bases, repositories, warehouses

Causes of bad-fix injection rates

Impact of complexity on quality and defect removal

Impact of creeping requirements on quality

CONCLUSIONS ON SOFTWARE QUALITY

- **No single quality method is adequate by itself.**
- **Formal inspections, static analysis are most efficient**
- **Inspections + static analysis + testing > 97% efficient.**
- **Defect prevention + removal best overall**
- **Quality function deployment & six-sigma prevent defects**
- **Higher CMMI levels, TSP, RUP, Agile, XP are effective**
- **Quality excellence has ROI > $15 for each $1 spent**
- **High quality benefits schedules, productivity, users**

REFERENCES ON SOFTWARE QUALITY

Black, Rex; Managing the Testing Process; Microsoft Press, 1999.

Crosby, Phiip B.; Quality is Free; New American Library, Mentor Books, 1979.

Gack Gary, Managing the Black Hole; Business Expert Publishing, 2009

Gilb, Tom & Graham, Dorothy; Software Inspections; Addison Wesley, 1983.

Jones, Capers & Bonsignour, Olivier; The Economics of Software Quality; Addison Wesley, 2011 (summer)

Jones, Capers; Software Engineering Best Practices; McGraw Hill, 2010

Jones, Capers; Applied Software Measurement; McGraw Hill, 2008.

Jones, Capers; Estimating Software Costs, McGraw Hill, 2007.

Jones, Capers; Assessments, Benchmarks, and Best Practices, Addison Wesley, 2000.

REFERENCES ON SOFTWARE QUALITY

Kan, Steve; Metrics and Models in Software Quality Engineering, Addison Wesley, 2003.

McConnell; Steve; Code Complete 2; Microsoft Press, 2004

Pirsig, Robert; Zen and the Art of Motorcycle Maintenance; Bantam; 1984

Radice, Ron; High-quality, Low-cost Software Inspections, Paradoxican Publishing, 2002.

Wiegers, Karl; Peer Reviews in Software, Addison Wesley, 2002.

REFERENCES ON SOFTWARE QUALITY

www.ASQ.org	**(American Society for Quality)**
www.IFPUG.org	**(Int. Func. Pt. Users Group)**
www.ISBSG.org	**(Int. Software Bench. Standards Group)**
www.ISO.org	**(International Organization for Standards)**
www.ITMPI.org	**(Infor. Tech. Metrics and Productivity Institute)**
www.PMI.org	**(Project Management Institute)**
www.processfusion.net	**(Process Fusion)**
www.SEI.org	**(Software Engineering Institute)**
www.SPR.com	**(Software Productivity Research LLC)**
www.SSQ.org	**(Society for Software Quality)**

REFERENCES ON SOFTWARE QUALITY

www.SEMAT.org (Software Engineering Methods and Theory)

www.CISQ.org (Consortium for IT software quality)

www.SANS.org Sans Institute listing of software defects

www.eoqsg.org European Institute for Software Qualiy

www,galorath.com Galorath Associates

www.associationforsoftwaretesting.org
Association for Software Testing

www.qualityassuranceinstitute.com
Quality Assurance Institute (QAI)