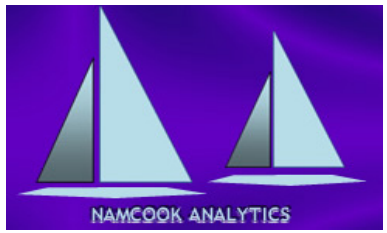
Namcook Analytics LLC

KEYS TO SUCCESS:

SOFTWARE MEASUREMENT
SOFTWARE ESTIMATING
SOFTWARE QUALITY

Capers Jones, VP and CTO



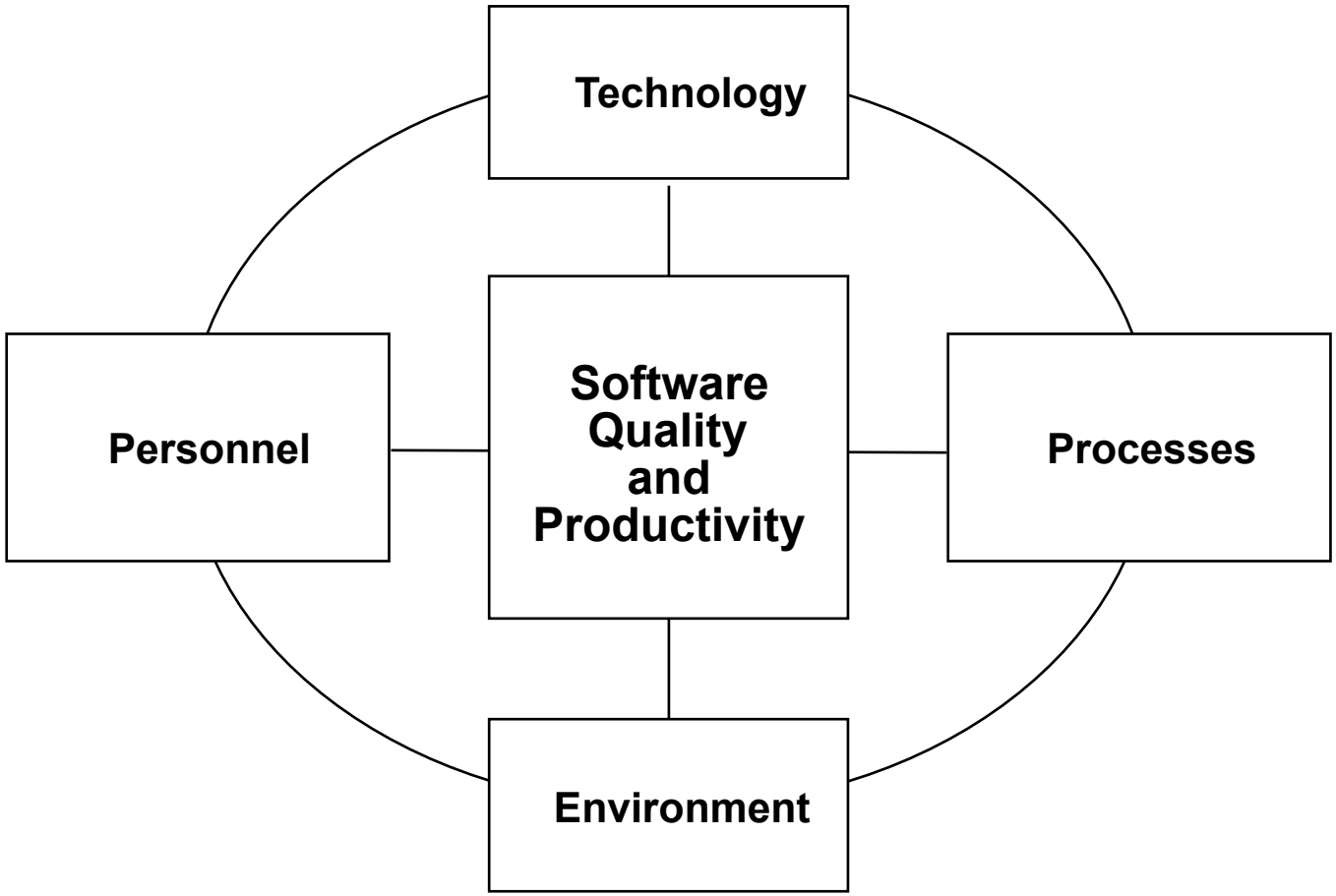
Blog: http://Namcookanalytics.com
Web: http://www.Namcook.com
Email: Capers.Jones3@gmail.com

August 4, 2015

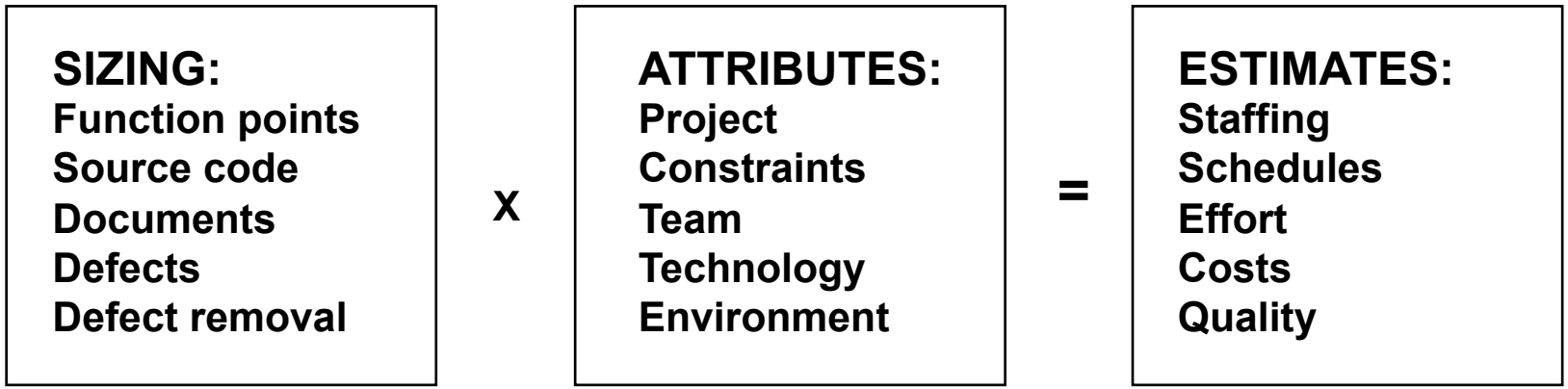
MEASUREMENT, ESTIMATION AND QUALITY

- **Software measurement, estimation, and quality form a "tripod."**
- **Attempting to improve one leg without improving the other two leads to imbalance and the structure will topple.**
- **Good measurements lead to good estimation.**
- **Good measurements and good estimation lead to good quality.**
- **Good quality leads to lower costs, shorter schedules, and more accurate estimates.**

Four Key Areas That Influence Software Cost Estimates



ESTIMATING PRINCIPLES



SOFTWARE COST ESTIMATION TOOLS

TOOL	YEAR AVAILABLE	ORGANIZATION
PRICE-S	1973	RCA
SEER	1974	Hughes
SLIM	1979	Air Force
COCOMO	1981	TRW
SPQR/20	1985	SPR *
CHECKPOINT - CHECKMARK	1989	SPR *
CostXpert	1992	Roetzheim
ExcelerPlan	1993	Roetzheim
KNOWLEDGEPLAN	1995	SPR *
COCOMO II	1995	USC
ISBSG	2005	ISBSG
SOFTWARE RISK MASTER (SRM)	2011	Namcook *

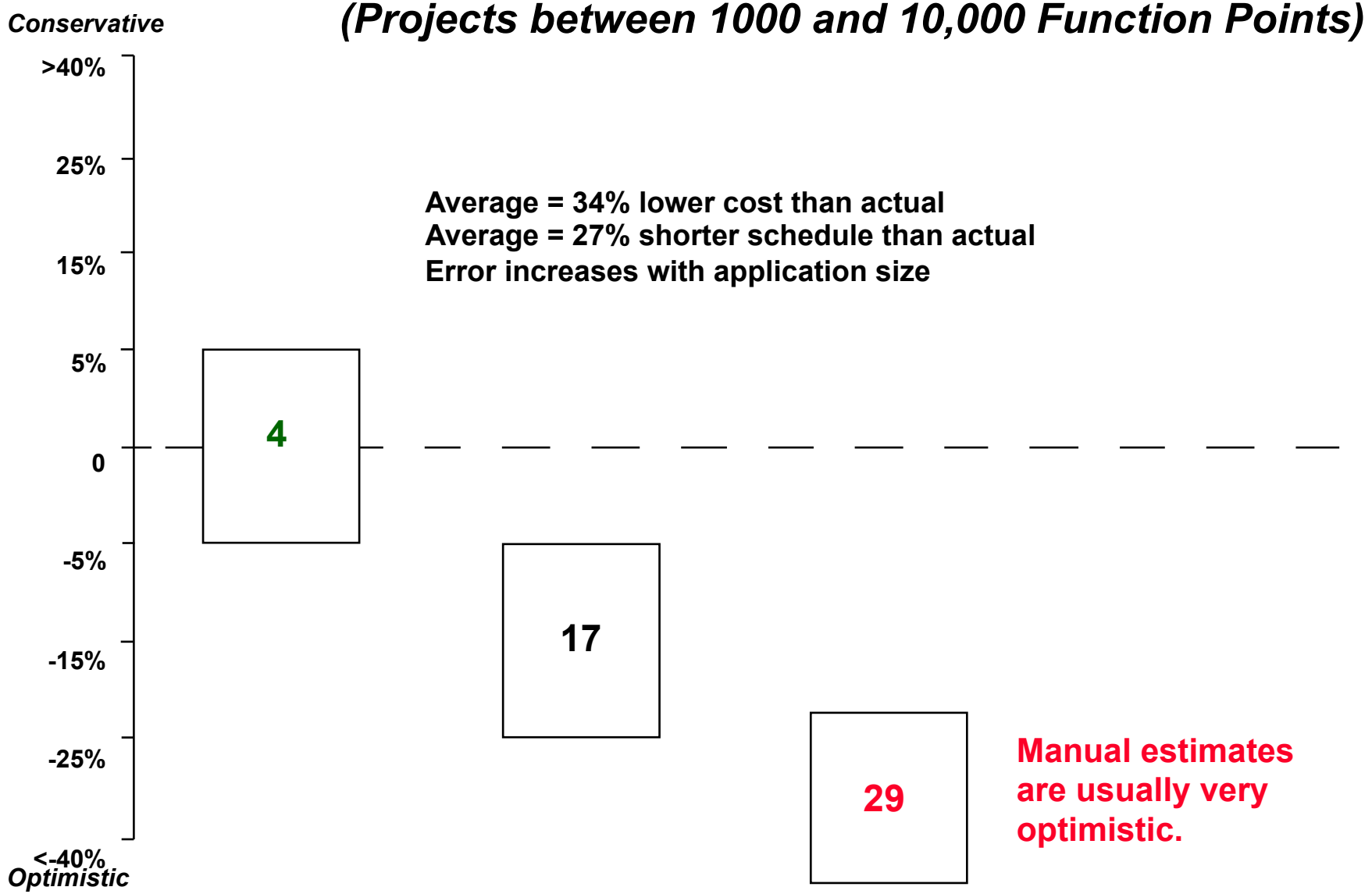
* Estimating tools designed by Capers Jones

50 MANUAL VS. 50 AUTOMATED COST ESTIMATES

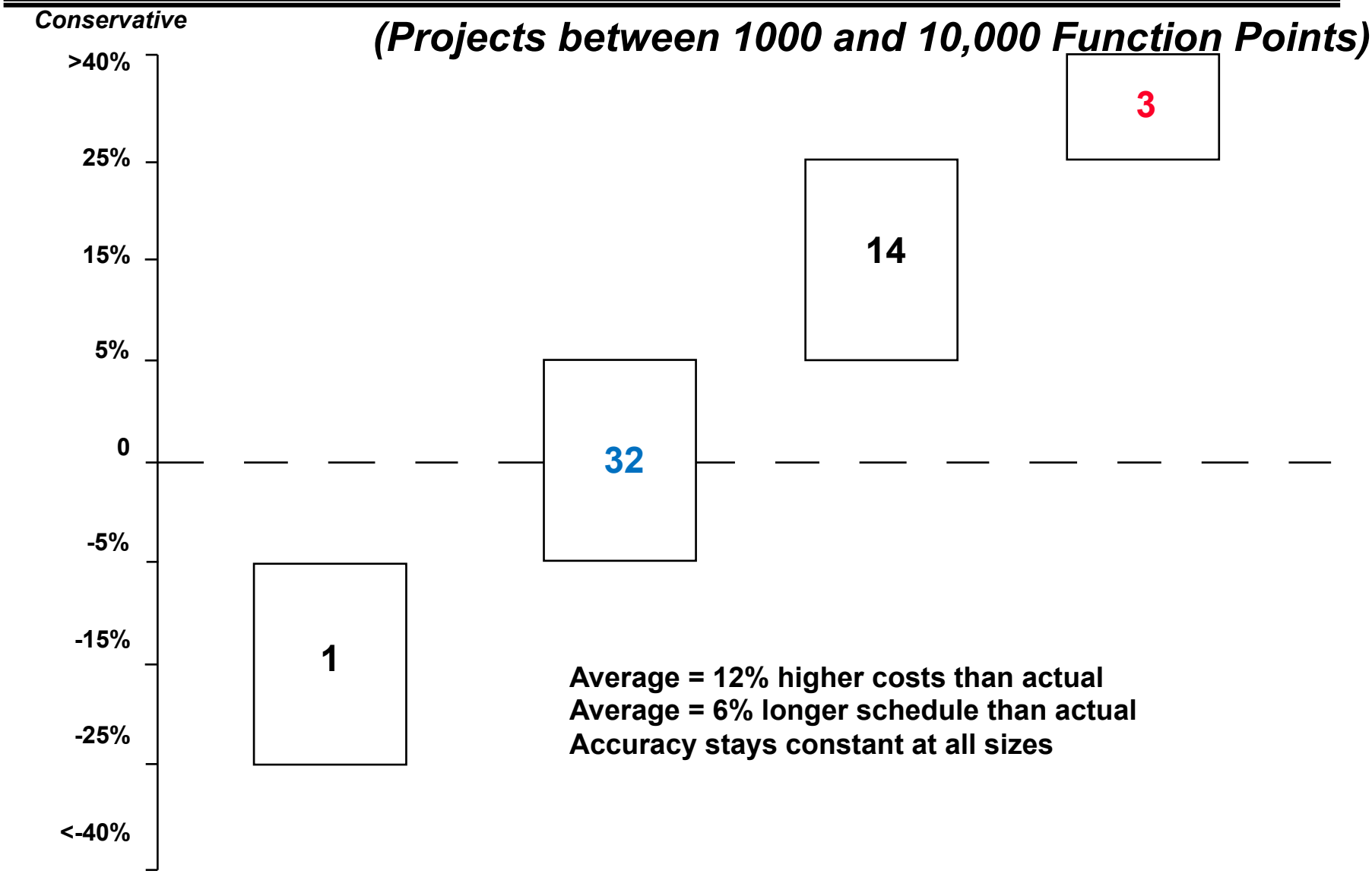
- **Manual estimates work well < 250 function points.**
- **Automated estimates are more accurate > 250 function points**
- **> 5,000 function points manual estimates are dangerous.**
- **Automated estimates are usually accurate or conservative.**
- **Manual estimates grow more optimistic as size increases.**

ACCURACY RANGES FOR 50 MANUAL ESTIMATES

(Projects between 1000 and 10,000 Function Points)



ACCURACY RANGES FOR 50 AUTOMATED ESTIMATES



PROBLEMS WITH MANUAL ESTIMATION

- **Manual sizing with function points is slow and later than needed:**
  Function point analysis < 500 function points per staff day.

- **Manual estimates are slow, expensive, and hard to change.**

- **Usually omit or under estimate requirements creep.**

- **Usually too optimistic with quality and test schedule prediction**

- **Usually omit or under estimate specialist tasks:**
  Technical writing, quality assurance, architecture, etc.

- **Non-coding work and quality become larger percentages of total effort > 1000 function points as do requirements creep.**

ESTIMATING ACCURACY BY GRANULARITY

+/- 30% for project-level estimates with no granularity or inner structure. These estimates cannot be validated since no one knows what the work consisted of.

+/- 15% for phase-level estimates for requirements, design, development, and testing. Too many activities span phases such as integration, documentation, and management.

+/- 5% for activity-based estimates that include at least these: 1) requirements, 2) design, 3) coding, 4) testing, 5) quality assurance; 6) documentation; 7) management. This level is useful for both measurement and estimation. Up to 25 activities are common.

+/- 3% for task-level estimates that include 25 to 2,000 specific tasks. Hard to measure but can be precise.

ESTIMATING ACCURACY BY GRANULARITY

At the project level "testing" is included in the total project in an unknown amount.

At the phase level "testing" is broken out into its own cost bucket.

At the activity level "testing" can include unit test, function test, regression test, component test, performance test, security test, usability test, platform test, system test, beta test, acceptance test, supply chain test, and others.

At the task level each form of testing can include test case design, test case construction, test case execution, defect logging, defect routing, defect repairs, repair integration, and others.

ESTIMATING ACCURACY BY PROJECT SIZE

Function Pts.	Accuracy	Reasons
10	15%	Human differences; sickness
100	12%	Human differences; sickness
1,000	5%	Easiest, most accurate to predict
10,000	10%	Requirements creep; business changes; poor quality control
100,000	15%	Requirements creep; business changes; poor quality control

ESTIMATING ACCURACY BY TECHNOLOGY

3% for excellent projects with good teams, good methods, good change control, and good quality control, high CMMI levels.

10% for average projects with average teams, adequate methods, adequate change control and adequate testing but marginal pre-test quality control. Testing usually runs late.

20% for marginal projects with inexperienced teams, marginal methods, poor change control and poor quality control without any pre-test inspections or static analysis. These projects run late due to excessive bugs. Low CMMI or none at all.

40% for poor projects with novice teams, novice clients, poor methods, poor change control, and poor quality control without any pre-test inspections or static analysis. These projects get canceled due to negative ROI. Low CMMI or none at all.

PRIMARY REASONS FOR SCHEDULE DELAYS

- **Omitting pre-test inspections and static analysis stretches out test schedules by more than 200%. This adds 3 to 6 months to the schedules of applications > 1000 function points.**

- **Unplanned requirements creep of > 2% per calendar month or > 20% of initial size adds 3 to 6 months to the schedules of applications > 1000 function points.**

- **These two problems are minor < 100 function points but grow in importance > 1000 function points and are critical >10,000 function points. Both problems are predictable and avoidable.**

EXAMPLES OF REQUIREMENTS CREEP

SIZE AT START (Function Pts.)	CREEP %	SIZE WITH CREEP (Function Pts.)
10	0%	10
100	3%	103
1,000	8%	1,080
10,000	12%	11,200
100,000	21%	121,000

Agile Sprints	Number of Sprints	Sprint Size
10	1	10
100	1	100
1,000	5	200
10,000	44	227 *
100,000	431	232 *

*** Agile is seldom used > 2,500 function points**

ESTIMATING ACCURACY BY METHODOLOGY

	Manual	Automated
Team Software Process (TSP)	5%	5%
Rational Unified Process (RUP)	5%	5%
IntegraNova	5%	5%
T-VEC	5%	5%
Prince2	10%	5%
Merise	10%	5%
Structured analysis and design (SADT)	12%	7%
Iterative	13%	7%
Extreme Programming (XP)	15%	10%
Spiral	15%	10%
Agile/Scrum	20%	15%
Waterfall	30%	15%
Cowboy	50%	30%

ESTIMATING ACCURACY BY METRICS USED

	Manual	Automated
IFPUG function points with SNAP	5%	5%
IFPUG function points without SNAP	10%	7%
COSMIC function points	10%	7%
NESMA function points	10%	7%
FISMA function points	10%	7%
Mark II function points	12%	10%
Use-case points	17%	15%
Unadjusted function points	17%	15%
Agile velocity	25%	15%
Story points	30%	25%
Backfired function points	45%	35%
Uncertified function point counts	45%	35%
Logical code statements	50%	35%
Physical lines of code	65%	40%

APPROXIMATE BENCHMARKS CIRCA 2015

	Measured Projects	
IFPUG function points without SNAP	60,000	(1978 through 2014)
Backfired function points (IFPUG)	45,000	(1973 through 2014)
Uncertified function points	15,000	
COSMIC function points	5,000	
Agile velocity	4,500	
Story points	4,000	
Mark II function points (UK primarily)	2,000	
Physical lines of code (military primarily)	2,000	
Logical code statements	1,500	
NESMA function points	1,500	
FISMA function points	500	
Use-case points	500	
Unadjusted function points	200	
IFPUG function points with SNAP	300	(new metric in 2012)

BENCHMARK OBSERVATIONS CIRCA 2015

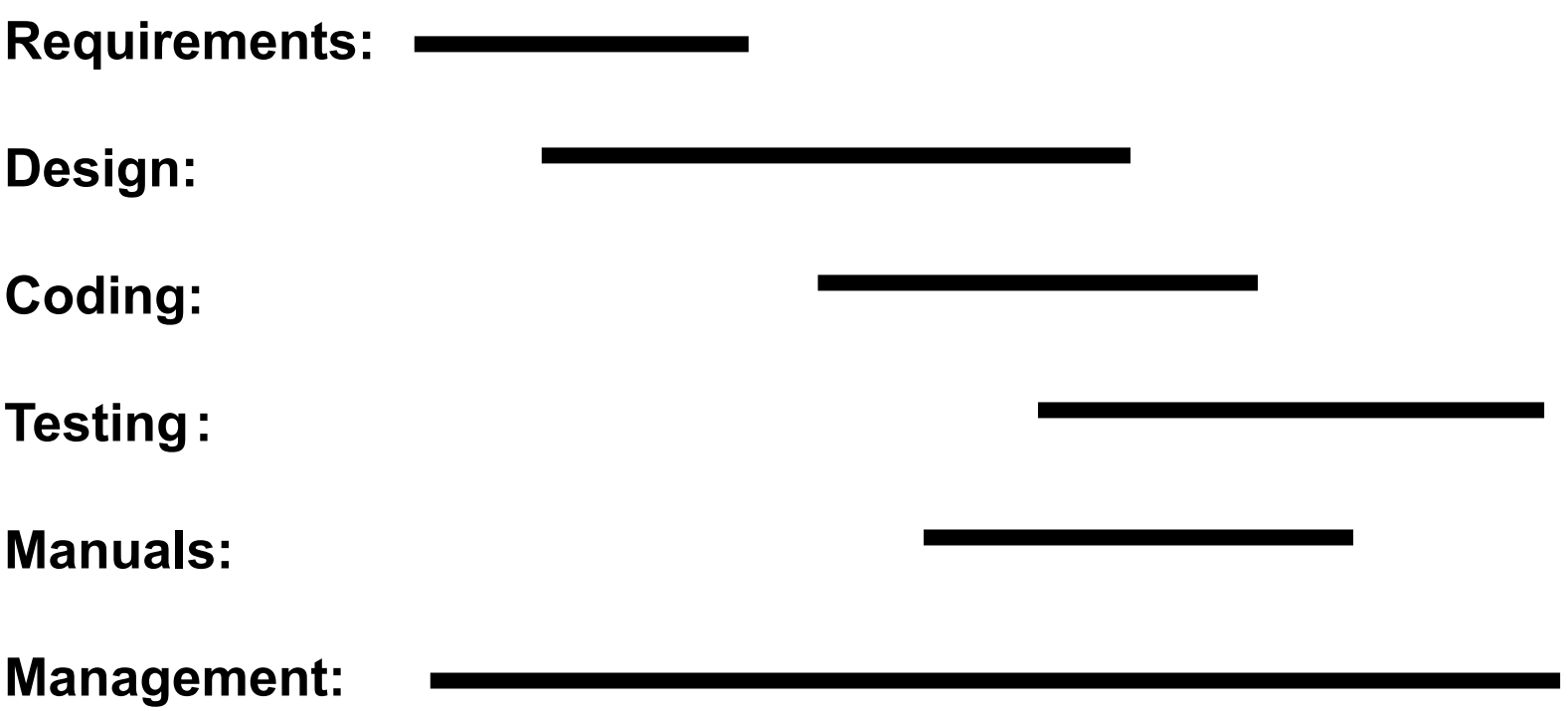
- **The Software Industry has too many metrics.**
- **The software industry uses inaccurate metrics such as "lines of code" and "cost per defect."**
- **The Software Industry has sparse data on schedules above 10,000 function points due to high cost of function points.**
- **The Software Industry has sparse data on productivity above 10,000 function points due to high cost of function points.**
- **The Software Industry has sparse data on quality at every size, due to leakage and poor measurement practices.**
- **Leakage from productivity and quality data are endemic.**

BEST CASE ESTIMATING ACCURACY

(Assumes experienced personnel, good methodologies, good quality control, good languages, normal requirements creep.)

Function Pts.	Manual	Automated	Team Sizes
10	3%	3%	1
100	7%	3%	2
1,000	15%	5%	7
10,000	25%	7%	45
100,000	45%	10%	350

SCHEDULING OVERLAP EXAMPLE



Schedule without overlap (waterfall model) = 48 months

Schedule with overlap = 24 months

ESTIMATING GOALS

1 The economic value of quality is not well understood.

2 “Cost per defect” and “lines of code” hide the value of quality.

3 As quality goes up schedules, costs, and risks come down.

4 Methodologies such as Agile help but are not sufficient.

5 Testing alone helps but is not sufficient.

6 Good methods combined with pre-test inspections and static analysis plus formal testing by trained test personnel are all necessary to achieve best-case results.

ESTIMATING FUNDAMENTAL CONCEPTS

1 Finding and fixing bugs is the # 1 cost driver for all software.

2 Producing paper documents is the # 2 cost driver for software.

3 Coding is the # 3 cost driver for software.

4 Meetings, travel is the # 4 cost driver for software.

5 Project management, oversight is the # 5 cost driver for software.

6 Requirements creep is the # 6 cost driver for software.

- **Estimates need to be accurate for all 6 cost drivers!**

FEATURES THAT AID ESTIMATING ACCURACY

1 Measure historical projects to better than 1% precision.

2 Predict all six major software cost drivers plus reuse.

3 Include requirements creep in sizing logic.

4 Include deferred functions left out to meet schedule goals.

5 Include new SNAP size rules for non-functional requirements.

6 Adjust estimates based on team skills, methodologies, programming languages, CMMI levels, project type, project class, and project size, international differences.

FACTORS THAT REDUCE ESTIMATE ACCURACY

1 Major business interruptions such as layoffs of development teams, strikes, or natural disasters such as floods, earthquakes.

2 Unavoidable requirements changes such as new Federal or State laws that impact financial applications on short notice.

3 Patent or IP litigation where court orders freeze software.

4 Bankruptcy of software development company (as shown by Studio 38 in Rhode Island in 2012.)

5 Mandated changes in tools, methods, or languages after estimates are created and approved.

FACTORS THAT NEGATE GOOD ESTIMATES

1 At month 9 on a 12 month order-entry system the company acquired a competitor. This added 3 months to the schedule.

2 Due to replacement of a popular CIO in a manufacturing company 65% of the IT staff resigned. This delayed 25 projects for 2 months.

3 The air-conditioning system in a software office building burst and destroyed the data center and offices. This caused a 2 week delay.

4 When Alcatel acquired the telecom business of ITT more than 30 projects were canceled and 400 software personnel laid off.

5 Many accurate estimates are rejected by higher management and replaced by arbitrary and impossible schedule demands.

ESTIMATES VERSUS MEASURED ACCURACY

Automated estimates are often more accurate than "historical data."

Historical data "leaks" and omits cost and resource elements:

- **Unpaid overtime**
- **Management effort**
- **Business analysts**
- **Architects**
- **Creeping requirements**
- **Technical writers**
- **Integration**
- **Data base analysts**
- **Others**

RANGES OF HISTORICAL DATA ACCURACY

1% On-site interviews with teams using a formal chart of accounts and accurate calibration tools such as SRM.

1% Remote interviews with teams using a formal chart of accounts and accurate calibration tools such as SRM.

5% Self-reported data by military and defense agencies with formal measures, CMMI 5, and earned-value tools.

5% Outsource contract measures using billable hours and function points with good measurement tools.

7% Software organizations that operate as profit centers and charge for services.

RANGES OF HISTORICAL DATA ACCURACY

10% Software for embedded devices such as smart phones.

12% Commercial software projects that do not track unpaid overtime but measure paid time.

12% Small projects that have < 3 people and track costs.

15% Software built under fixed-price contracts.

35% Companies without formal cost collection and resource tracking tools that submit data to benchmark collections.

50% Companies that only measure "DCUT" or "design, code, and unit test" and do not measure other work.

55% Projects built under time and material contracts.

RANGES OF HISTORICAL DATA ACCURACY

65% Companies that use lines of code or LOC measures and omit requirements, design, and other paper activities.

75% Companies that only measure "code development" and do not measure other work.

80% Companies where software operates as a cost center without charges for software development or maintenance.

100% Companies without formal measurement programs.

U.S. average for historical data across 13,000 projects is about 37%; i.e. 63% of project effort is not recorded or measured based on team reconstruction during interviews.

COMPARISONS OF ACCURACY

3% **Best results for formal measurements using full charts of accounts that capture all activities and unpaid overtime.**

5% **Formal automated cost estimates of best-case projects by expert teams using state of the art methods.**

12% **Automated cost estimates of average projects with mediocre quality control and change control.**

15% **Manual estimates of small software projects < 1,000 function points in size.**

35% **Manual estimates of large software projects > 1,000 function points in size. Estimates are optimistic.**

COMPARISONS OF GLOBAL WORK HOURS

Country	Hours per month	Unpaid overtime	Total
India	190	8	200
China	186	12	198
Colombia	176	6	180
Russia	164	4	168
United States	149	10	159
France	123	2	125
Netherlands	115	0	115

COMPARISONS OF GLOBAL QUALITY LEVELS

Country	Defects per FP	Removal %	Delivered
Japan	4.30	93.50%	0.29
South Korea	4.90	92.00%	0.39
Israel	5.10	92.00%	0.41
United States	4.82	90.15%	0.47
Germany	4.95	88.00%	0.59
Russia	5.15	86.50%	0.70
North Korea	5.20	84.00%	0.86

COMPARISONS OF INDUSTRY QUALITY LEVELS

Industry	Defects per FP	Removal %	Delivered
Medical devices	5.20	99.50%	0.03
Telecom	5.50	96.50%	0.19
Software	3.50	94.00%	0.21
Banks/finance	4.50	95.00%	0.23
Defense	6.00	96.00%	0.26
Autos	4.90	94.50%	0.29
Government	6.00	84.70%	0.70

COMMON MEASUREMENT GAPS AND OMISSIONS

Activities	Completeness
01 Requirements	Missing or incomplete
02 Prototyping	Missing or incomplete
03 Architecture	Missing or incomplete
04 Project Planning	Missing or incomplete
05 Initial analysis/design	Missing or incomplete
06 Detailed design	Complete
07 Design reviews	Missing or incomplete
08 Coding	Complete
09 Reusable code acquisition	Missing or incomplete
10 COTS acquisitions	Missing or incomplete
11 Code inspections (if performed)	Complete
12 IV&V (if performed)	Complete

COMMON MEASUREMENT GAPS AND OMISSIONS

Activities	Completeness
13 Configuration control	Missing or incomplete
14 Integration	Missing or incomplete
15 User documentation	Missing or incomplete
16 Unit testing	Missing or incomplete
17 Function testing	Missing or incomplete
18 Integration testing	Missing or Incomplete
19 System testing	Complete
20 Field testing	Missing or incomplete
21 Acceptance testing	Missing or incomplete
22 Independent testing (if performed)	Complete
23 Quality assurance	Missing or incomplete
24 Installation and training	Missing or incomplete

COMMON MEASUREMENT GAPS AND OMISSIONS

Activities	Completeness
25 Project management	Missing or incomplete
26 Project office (if used)	Missing or incomplete
27 Function points (if used)	Complete
28 Unpaid overtime	Missing or incomplete
29 Requirements creep	Missing or incomplete
30 Administrative support	Missing or Incomplete
31 Travel and meetings	Missing or incomplete
32 Training/education	Missing or incomplete
33 Data base analysis	Missing or incomplete

Overall:

Historical data is only about 37% complete; 63% of effort is not recorded.

LEAKAGE DAMAGES MEASUREMENT ACCURACY

SIZE IN FP	LEAK	EFFORT MONTHS (Measured)		PRODUCTIVITY (FP per month)
1000	0%	200	(No leakage)	5.00
1000	25%	150	(Omits Mgt & unpaid overtime)	6.67
1000	50%	100	(Measures DCUT only)	10.00
1000	75%	50	(Measures coding only)	20.00

Leakage from historical data makes productivity rates look higher than they really are.
This is a chronic problem for the software industry.

QUALITY MEASUREMENTS ALSO LEAK

	Percent	Accuracy
Full quality measures from requirements	5.00%	100.00 %
Quality measures from testing	5.00%	55.00%
Quality measures from testing (minus unit)	15.00%	45.00%
Quality measures from release : all defects	20.00%	25.00%
Quality measures from release : severe defects	25.00%	10.00%
No quality measures at all	30.00%	0.00%
Total/Average	100.00%	39.17%

Incomplete quality measures are a chronic problem for the software industry.

QUALITY MEASUREMENT LEAKS BY ORIGINS

DEFECT ORIGINS	MEASURE COMPLETENESS
Requirement defects	Missing or incomplete
Design defects	Missing or incomplete
Code defects	Usually omits unit test
Structural defects	Missing or incomplete
Document defects	Missing or incomplete
Bad fixes or secondary defects	Missing or incomplete
Test case defects	Missing or incomplete
Performance defects	Missing or incomplete
Usability defects	Missing or incomplete
Serviceability defects	Missing or incomplete
Severity 3 and 4 defects	Missing or incomplete
Security vulnerabilities	Missing until exploited

MEASURES AND ESTIMATES BY INDUSTRY

SECTOR	MEASURES	ESTIMATES	METHOD
Defense CMMI 5	5%	5%	Automated
Commercial (large)	5%	5%	Automated
Telecommunications	5%	5%	Automated
Outsource (large)	5%	5%	Automated
Outsource (medium)	10%	20%	Auto/manual
Defense CMMI 3	10%	20%	Auto/manual
Banks (large)	15%	25%	Auto/manual
Defense CMMI 1	25%	35%	Auto/Manual
Manufacture (medium)	25%	35%	Auto/manual
Civilian governments	30%	35%	Manual
Banks (medium)	35%	35%	Manual
Small companies	50%	50%	Manual
Venture startups	50%	50%	Manual

SUMMARY OF 150 MAJOR CORPORATIONS

Companies that top 95% in defect removal efficiency; 5% in measurement accuracy, and 5% in estimation accuracy = 10

(6.67%) Technology, defense, and software companies.

Companies between 85% and 95% in defect removal efficiency; 5% to 15% in measurement accuracy; and 5% to 15% in estimation accuracy = 25 (16.67%)

Companies below 85% in defect removal efficiency; 15% to 30% in measurement accuracy, and 15% to 30% in estimation accuracy = 65 (43.33%)

Companies with unknown defect removal efficiency; no measurements, and worse than 40% errors in estimation = 50

(33.33%) Manufacturing, retail, energy

SUMMARY OF 50 SMALL/MEDIUM CORPORATIONS

Companies that top 95% in defect removal efficiency; 5% in measurement accuracy, and 5% in estimation accuracy = 5

(10.00%) Software security, quality, and tools.

Departments between 85% and 95% in defect removal efficiency; 5% to 15% in measurement accuracy; and 5% to 15% in estimation accuracy = 8 (16.00%)

Companies below 85% in defect removal efficiency; 15% to 30% in measurement accuracy, and 15% to 30% in estimation accuracy = 13 (26.00%)

Companies with unknown defect removal efficiency; no measurements, and worse than 40% errors in estimation = 24

(48.00%) Games, service groups, web and smart phone aps.

SUMMARY OF 30 GOVERNMENT DEPARMENTS

Departments that top 95% in defect removal efficiency; 5% in measurement accuracy, and 5% in estimation accuracy = 5 (16.67%) All in security, space, and military weapons sectors.

Departments between 85% and 95% in defect removal efficiency; 5% to 15% in measurement accuracy; and 5% to 15% in estimation accuracy = 4 (13.33%) Financial departments.

Departments below 85% in defect removal efficiency; 15% to 30% in measurement accuracy, and 15% to 30% in estimation accuracy = 6 (20.00%) Both state and federal departments in this group; no municipal departments.

Departments with unknown defect removal efficiency; no measurements, and worse than 40% errors in estimation = 15 (50.00%) Includes federal, state, municipal departments.

MEASURES BY INDUSTRY SECTORS

Systems/embedded software	**Best quality measures**
	Best software quality
Information systems	**Best productivity measures**
	Best use of function points
Outsource vendors	**Best benchmark measures**
	Best cost tracking
Commercial software	**Best user satisfaction measures**
	Best maintenance measures
Military software	**Best reliability measures**
	Best use of CMMI assessments
	Best use of earned-value measures

INDUSTRY SECTORS >95% IN DEFECT REMOVAL

Industry Sectors	Year 95% Exceeded
Telecommunications	1975
Computer manufacturing	1977
Aero-space manufacturing	1979
Military/Defense contractors	1980
Medical devices	1983
Commercial software	1992

SOFTWARE NEEDING BETTER QUALITY CONTROL

1) ERP packages are buggy, hard to install, and hard to customize. Data migration is also difficult.

2) Stock market programmed trading applications are buggy and have caused major disruptions of stock markets with billions In losses.

3) Property assessment software is buggy and proprietary. Algorithms are concealed from tax officials. Many lawsuits.

4) Voting machine and vote tabulation software is buggy and easy to hack. Some election results have been wrong. Common Cause keeps a master list of major voting software errors. Software algorithms are proprietary and secret.

SOFTWARE NEEDING BETTER QUALITY CONTROL

6) Embedded applications controlling automobile engines, brakes, airbags, etc. are often buggy. There have been hundreds of recalls and dozens of accidents.

7) Software controlling electric power generation and transmission is fragile and prone to outages, sometimes impacting wide areas.

8) Software controlling nuclear generators is not always validated in countries with major nuclear plants.

9) Air-traffic control software is aging, unreliable, and so complex that repairs or replacement are expensive and hazardous.

WHY SOFTWARE MEASURES HAVE VALUE

Companies that measure:		Companies that don't:	
On-time projects =	75%	On-time projects =	45%
Late projects =	20%	Late projects =	40%
Canceled projects =	5%	Canceled projects =	15%
Defect potentials =	< 3.00	Defect potentials =	> 5.00
Defect removal =	> 95%	Defect removal =	Unknown
Cost estimates =	+/- 5%	Cost estimates =	+/- 35%
CEO satisfaction =	High	CEO satisfaction =	Low
User satisfaction =	High	User satisfaction =	Low
Competitiveness =	High	Competitiveness =	Low

NEED FOR HIGH-SPEED FUNCTION POINTS

Software Package	Size in Funct. Pts.	Count Days	Costs
Oracle	229,434	458	$1,374,000
Windows 7	202,150	404	$1,212,000
Microsoft Office 2003	33,736	67	$201,000
F15 Avionics	23,019	46	$138,000
VA medical records	19,810	40	$120,000
Apple I-Phone	19,366	39	$117,000
Google search engine	18,640	37	$111,000
Linux	17,505	35	$105,000

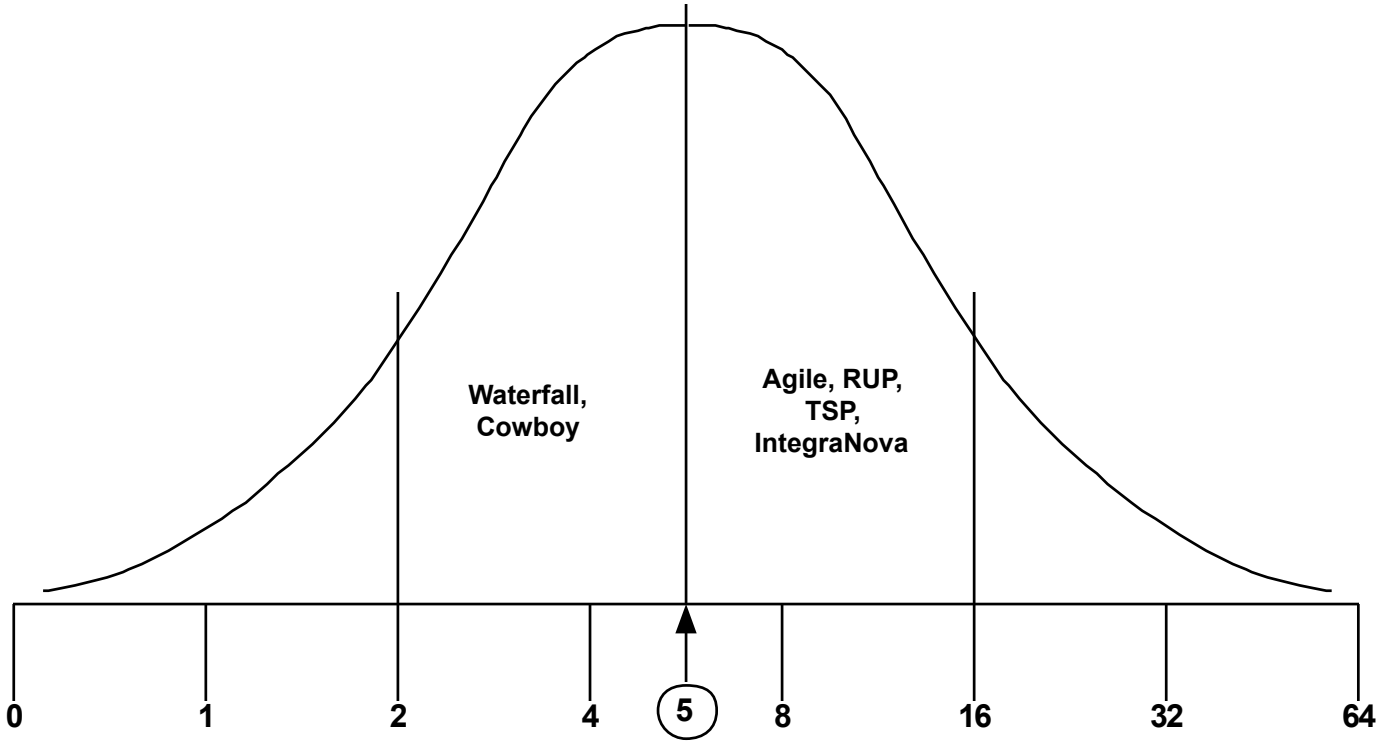
Note: Manual function point analysis is too slow and costly to be used above 10,000 function points.

SIZING BY MEANS OF "PATTERN MATCHING"

Software Package	Size in Funct. Pts.	Count Days	Costs
Oracle	229,434	.07	$208
Windows 7	202,150	.07	$208
Microsoft Office 2003	33,736	.07	$208
F15 Avionics	23,019	.07	$208
VA medical records	19,810	.07	$208
Apple I-Phone	19,366	.07	$208
Google search engine	18,640	.07	$208
Linux	17,505	.07	$208

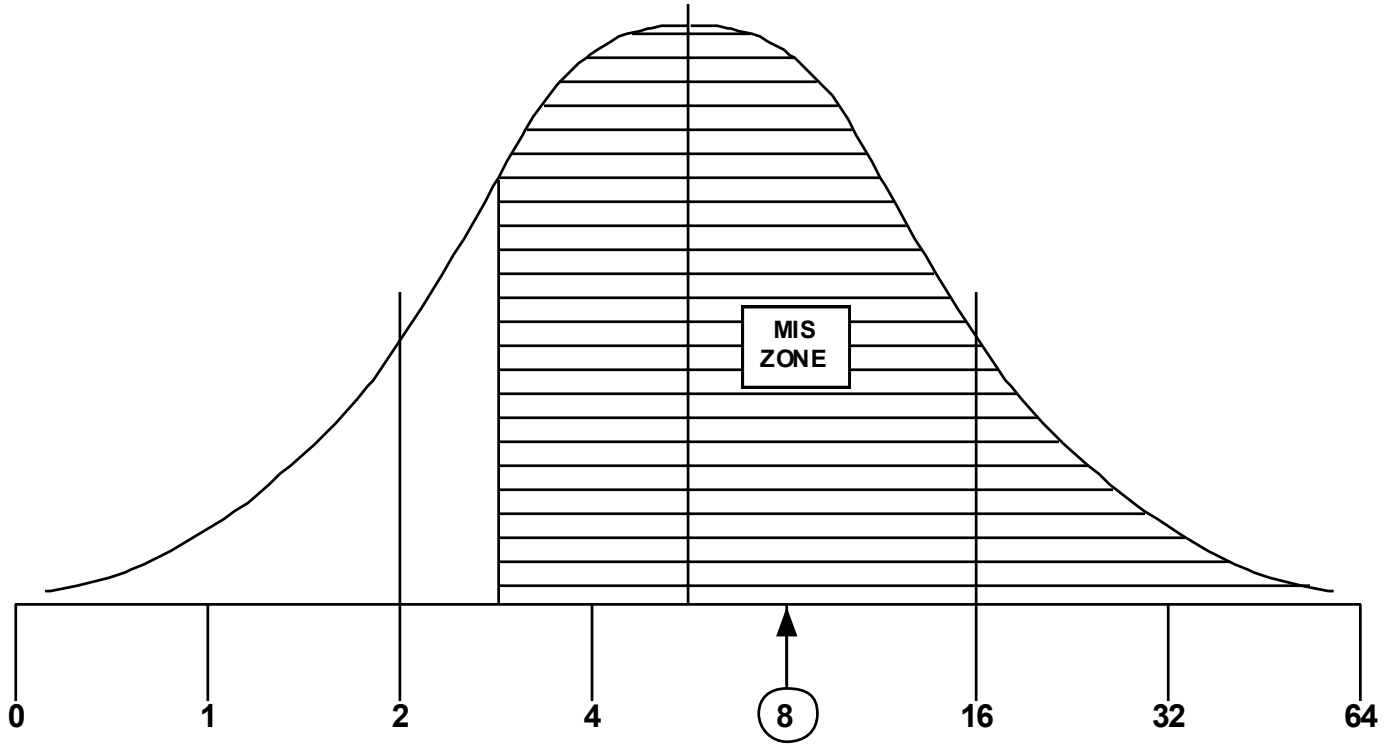
Note: Sizing with pattern matching takes between 4 and 7 minutes regardless of the number of function points in the application.

OVERALL U.S. SOFTWARE PRODUCTIVITY

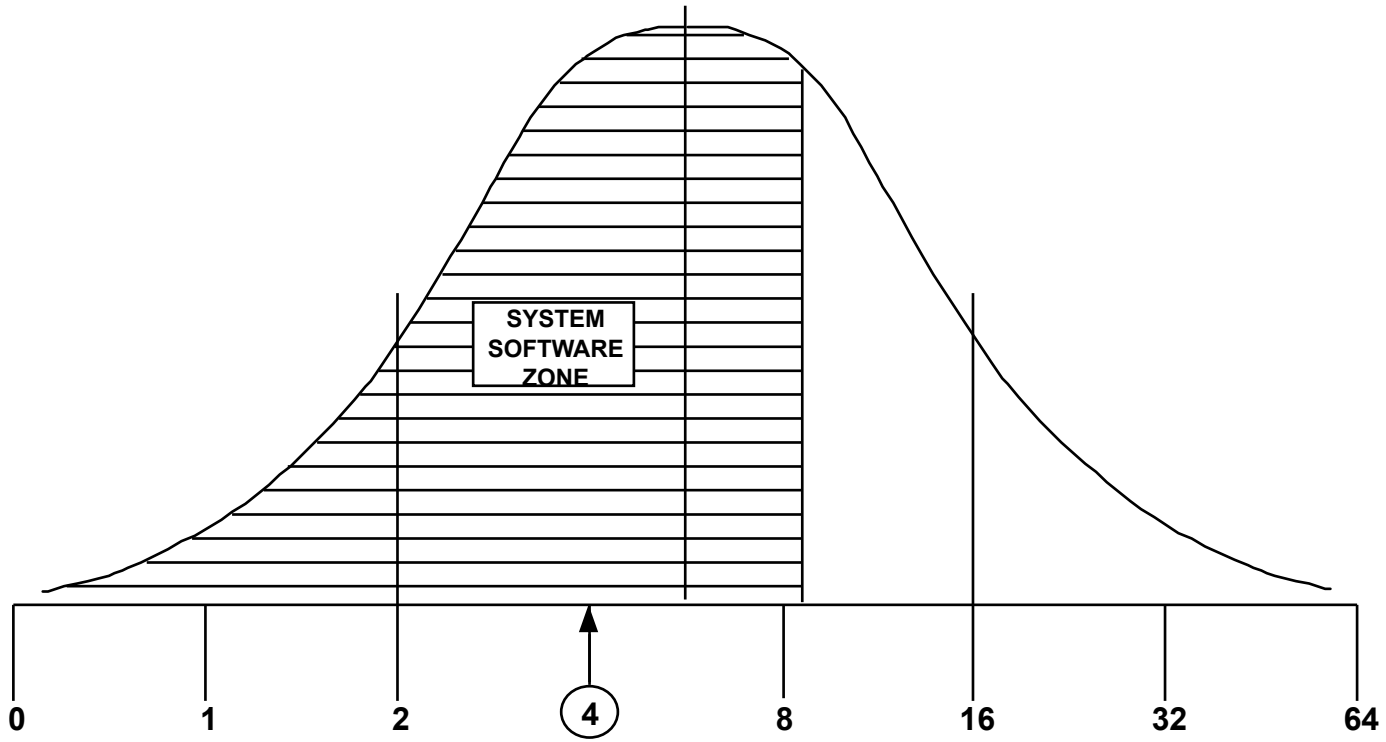


Function Points per Staff Month

DISTRIBUTION OF U.S. IT SOFTWARE PRODUCTIVITY

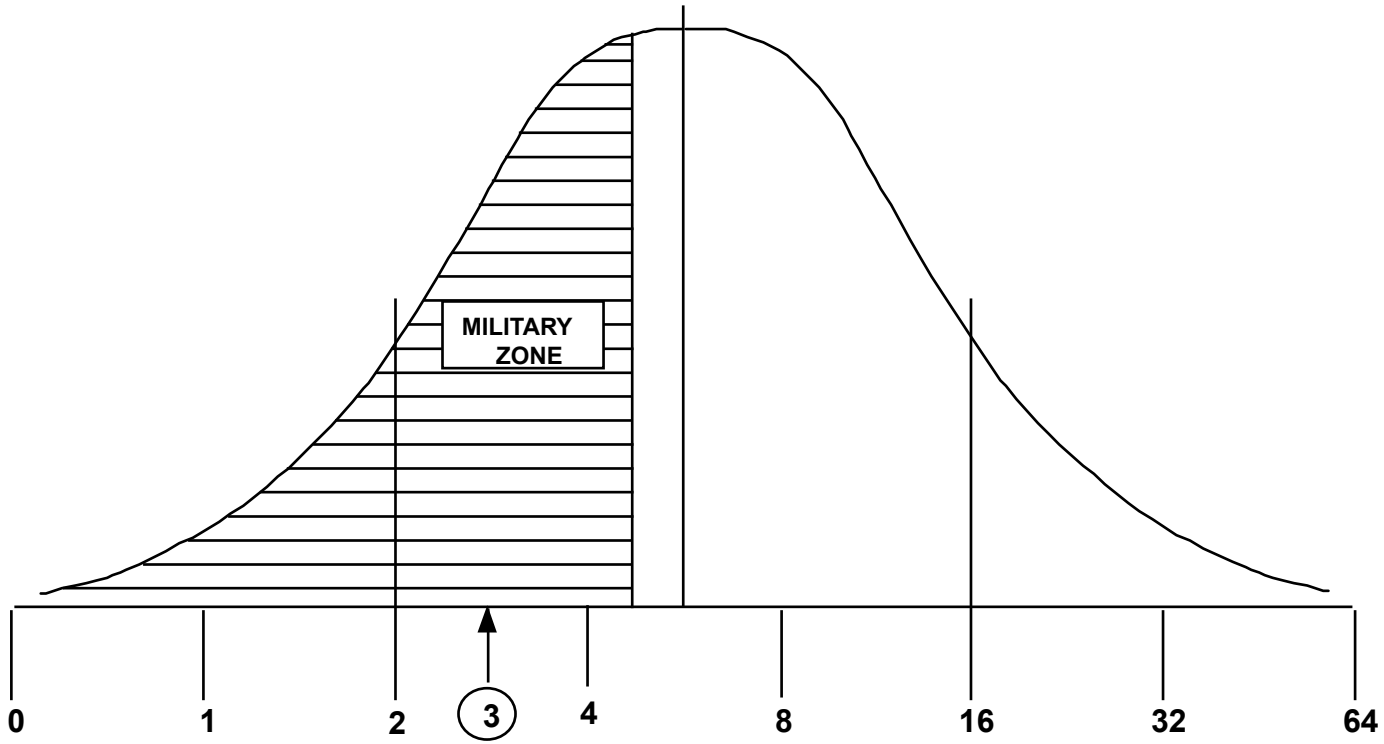


SYSTEMS & EMBEDDED SOFTWARE PRODUCTIVITY



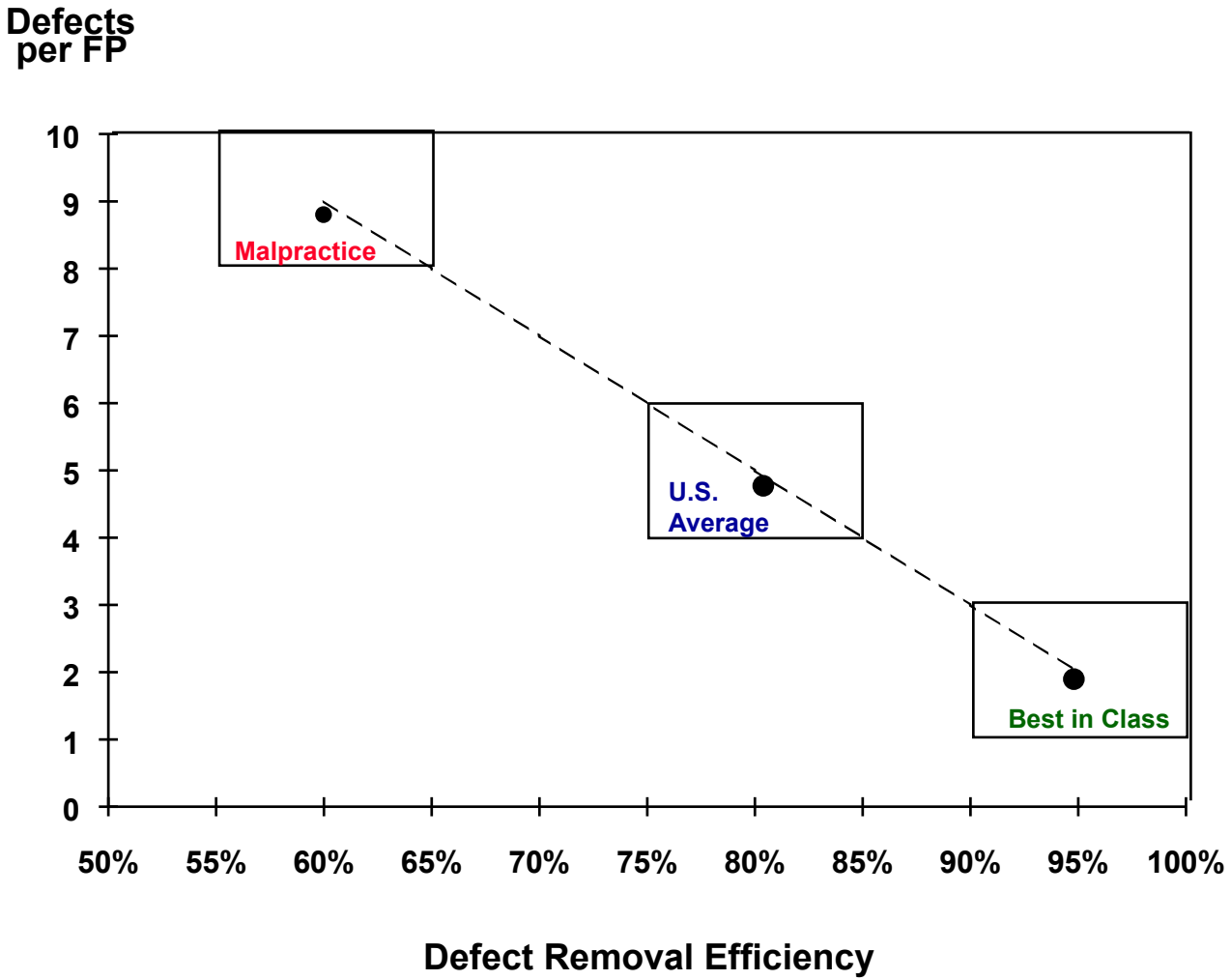
Function Points per Staff Month

U.S. MILITARY SOFTWARE PRODUCTIVITY

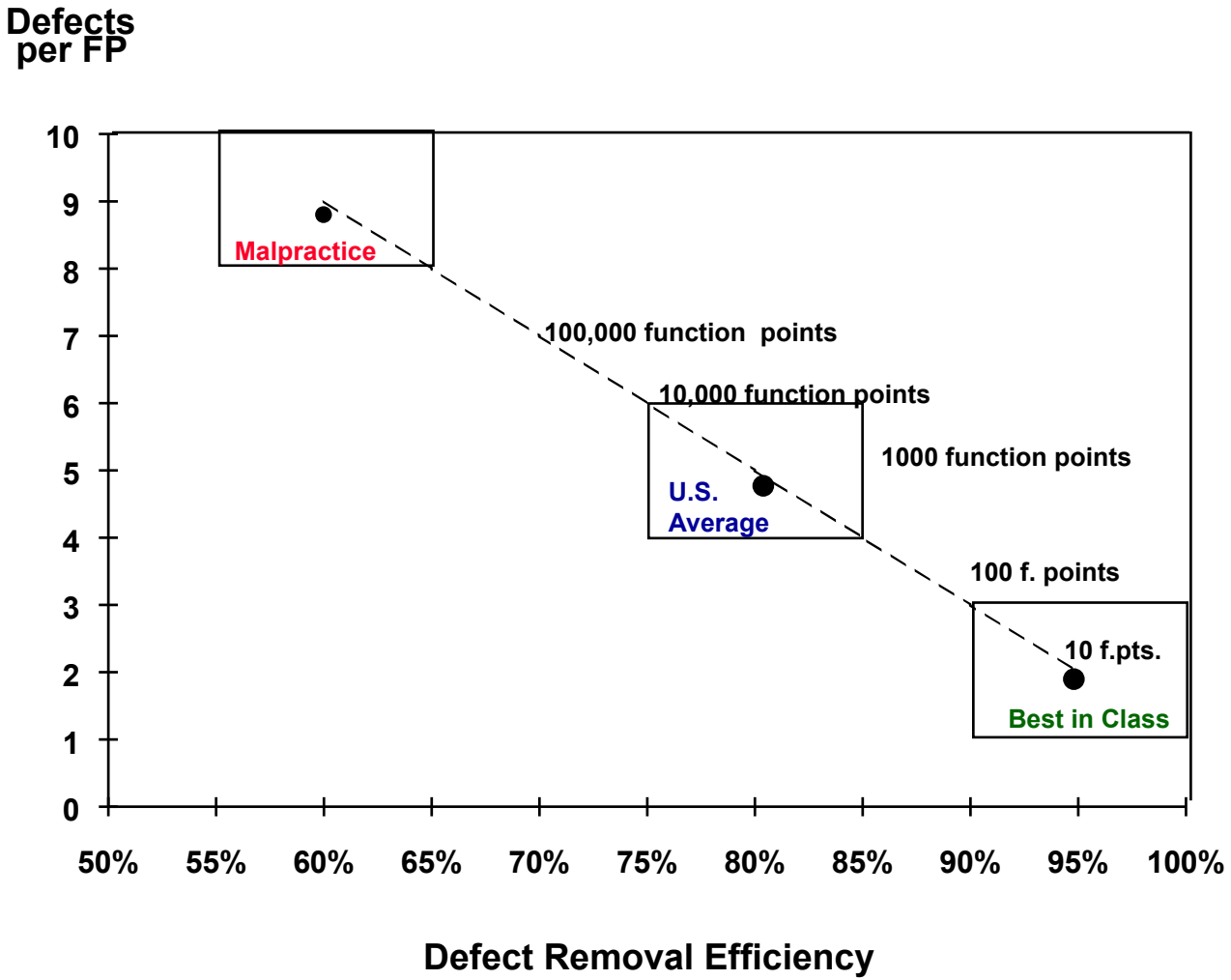


Function Points per Staff Month

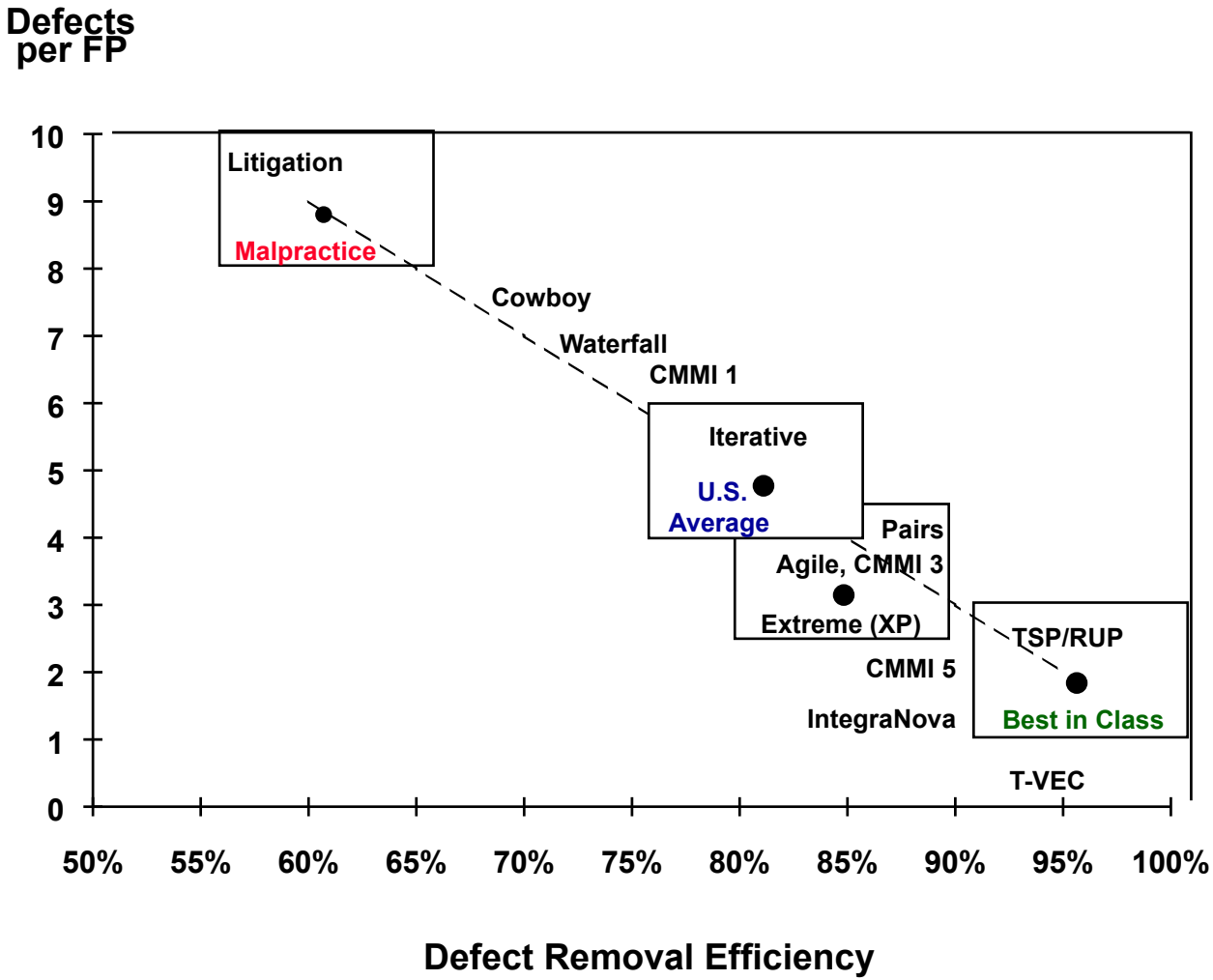
SOFTWARE QUALITY IMPROVEMENT



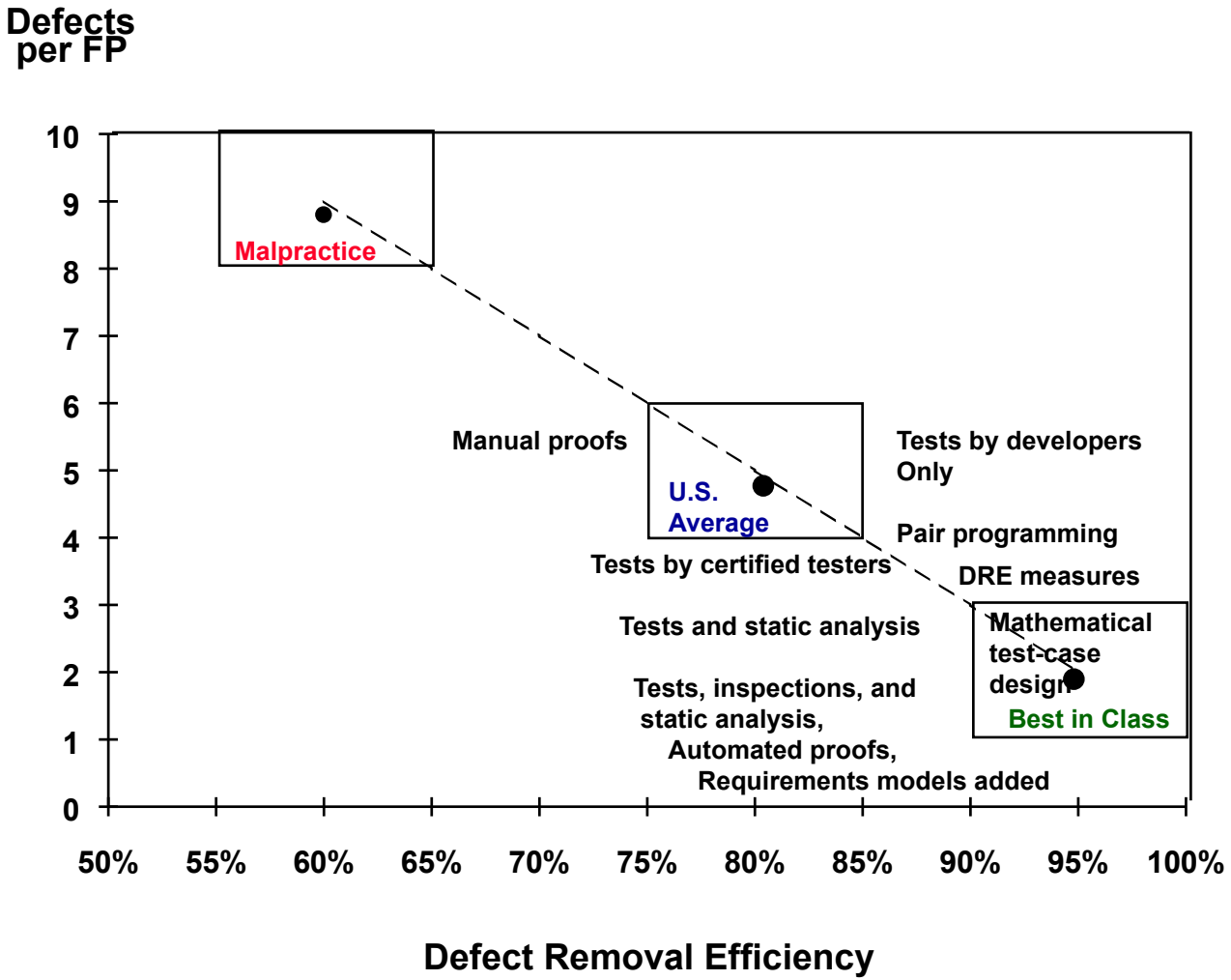
SOFTWARE QUALITY IMPROVEMENT



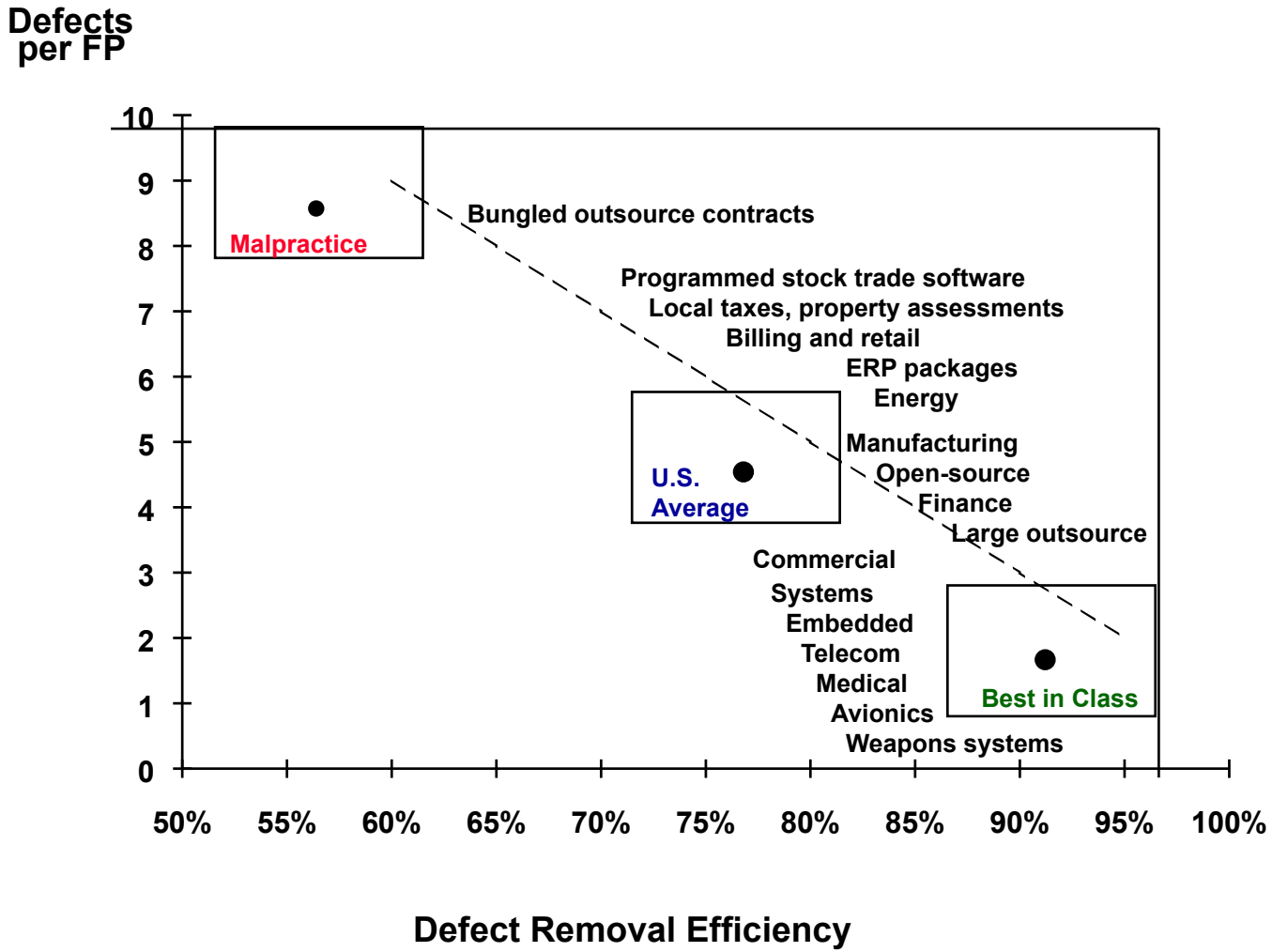
SOFTWARE QUALITY IMPROVEMENT (cont.)



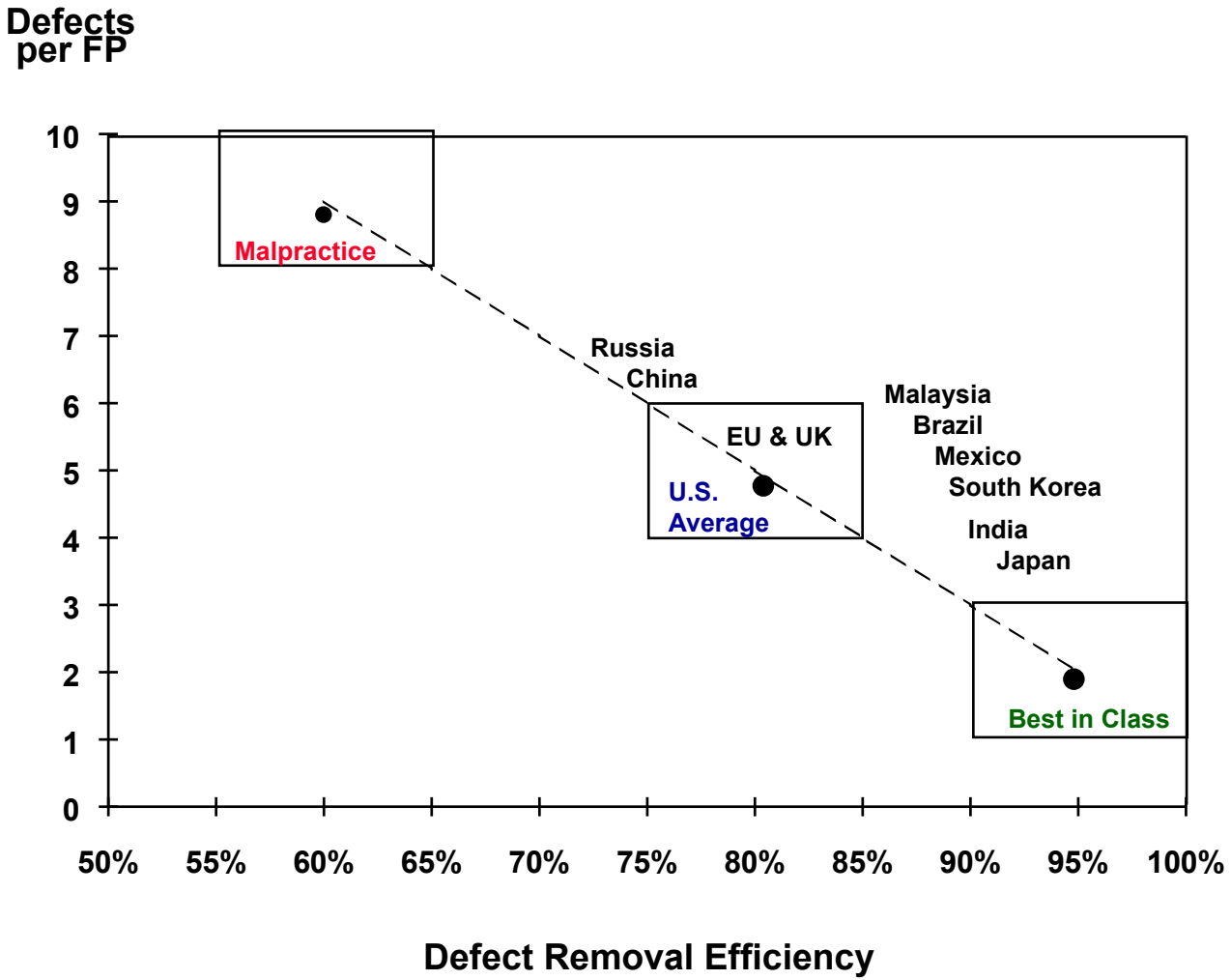
SOFTWARE QUALITY IMPROVEMENT



SOFTWARE QUALITY IMPROVEMENT



SOFTWARE QUALITY IMPROVEMENT



SUMMARY AND CONCLUSIONS

- **Software estimating is important and complex**
- **Automated estimates are more accurate than manual**
- **Function points make estimating easier**
- **High-speed function points allow early estimates**
- **Lines of code are inadequate and hazardous**
- **Cost per defect is inadequate and hazardous**
- **Good estimates lead to successful projects**
- **Bad estimates lead to disaster**

SUMMARY AND CONCLUSIONS

The accuracy of a software estimate cannot be known unless the accuracy of the historical data used for comparison is also known.

SUMMARY AND CONCLUSIONS

- **Competence is predictable.**
- **Incompetence has random results.**

SUMMARY AND CONCLUSIONS

- **Good measurements lead to accurate estimation.**
- **Accurate estimation leads to high quality.**
- **High quality leads to controlled projects.**
- **Controlled projects can be estimated within 3%.**
- **Uncontrolled projects are hazardous and experience:**
  - **Schedule delays**
  - **Cost overruns**
  - **Inaccurate, optimistic estimates**
  - **Negative ROI**
  - **Cancellations**
  - **Poor quality**
  - **Unhappy clients**

REFERENCES TO SOFTWARE COST ESTIMATING

Boehm, Dr. Barry; Software Engineering Economics; Prentice Hall, 1981.

Gack, Gary; Managing the Black Hole; Business Expert Publisher, 20`0

Galorath, Software Sizing, Estimating, and Risk Management, Auerbach Publishing, 2006

Garmus, David & Herron David, Function Point Analysis, Addison Wesley, 2001.

Hill, Peter, Practical Software Estimation; McGraw Hill, 2011

Jones, Capers; and Bonsighour, Oliveir; The Economics of Software Quality; Addison Wesley; 2011 (July)

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

Jones, Capers; Applied Software Measurement; 3rd edition; McGraw Hill, 2008.

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

Jones, Capers; Estimating Software Costs; 2nd edition; McGraw Hill,2007.

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

McConnell, Steve; Software Estimating-Demystifying the Black Art; Microsoft Press, 2006.

Pressman, Roger; Software Engineering – A Practitioners Approach; McGraw Hill, 2005.

Putnam, Larry; Measures for Excellence; Yourdon Press, Prentice Hall, 1992.

Royce, Walker; Software Project Management: A Unified Framework; Addison Wesley, 1998

Strassmann, Paul; Information Productivity; Information Economics Press, 1999.

Stutzke, Richard D.; Estimating Software Intensive Systems; Pearson, 2005