Software Solutions Symposium 2017

March 20–23, 2017

Best Practices for Software Process and Product Measurement

Dr. Bill Curtis

Executive Director, CISQ

SVP & Chief Scientist, CAST Research Labs

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

CISQ
Consortium for IT Software Quality

Instructor — Dr. Bill Curtis

Dr. Bill Curtis is Senior Vice President Chief Scientist of CAST Software, where he heads CAST Research Labs. CAST is a leading provider of technology for measuring the structural quality of business applications. He leads the production of CAST's annual CRASH Report on the global state of structural quality in business applications. He is also the Director of the Consortium for IT Software Quality, a Special Interest Group of the Object Management Group (OMG) where he leads an international effort to establish standards for automating measures of software quality characteristics. Dr. Curtis is a Fellow of the Institute of Electrical and Electronics Engineers (IEEE) for his career contributions to software process improvement and measurement.

Dr. Curtis is a co-author of the Capability Maturity Model for Software (CMM), the People CMM, and the Business Process Maturity Model. He was also a member of the Product Suite Team for CMMI. He is a globally recognized expert in software measurement, process improvement, and workforce development. He has been leader in the application of statistics to software engineering. He was the Chief Process Officer at Borland Software Corporation. Until it's acquisition by Borland he was a Co-founder and Chief Scientist of TeraQuest in Austin, Texas, the premier provider of CMM-based services. He is a former Director of the Software Process Program in the Software Engineering Institute at Carnegie Mellon University. Prior to joining the SEI, Dr. Curtis directed research on advanced user interface technologies and the software design process at MCC, the US's fifth generation computer research center in Austin, Texas. Prior to MCC he developed a global software productivity and quality measurement system at ITT's Programming Technology Center. Prior to ITT he evaluated software development methods in GE Space Division, worked in organizational effectiveness at Weyerhaeuser, and taught statistics at the University of Washington. He has co-authored four books, is on the editorial boards of several journals, and has published over 150 papers on software development and management. Dr. Curtis earned a Ph.D. from Texas Christian University and a M.A. from The University of Texas. He earned his B.A. from Eckerd College in mathematics, psychology, and theater.

CISQ
Consortium for IT Software Quality

Agenda

Dr. Bill Curtis
P.O. Box 126079
Fort Worth, Texas
76126-0079 USA
curtis@acm.org

Section 1
Measuring Software Size and Productivity

1. Practical Software Measurement
2. Size and functional measures
3. Measuring software productivity
4. Analyzing software productivity

CISQ
Consortium for IT Software Quality

Why Does Measurement Languish?

Badly defined measures

Too many measures

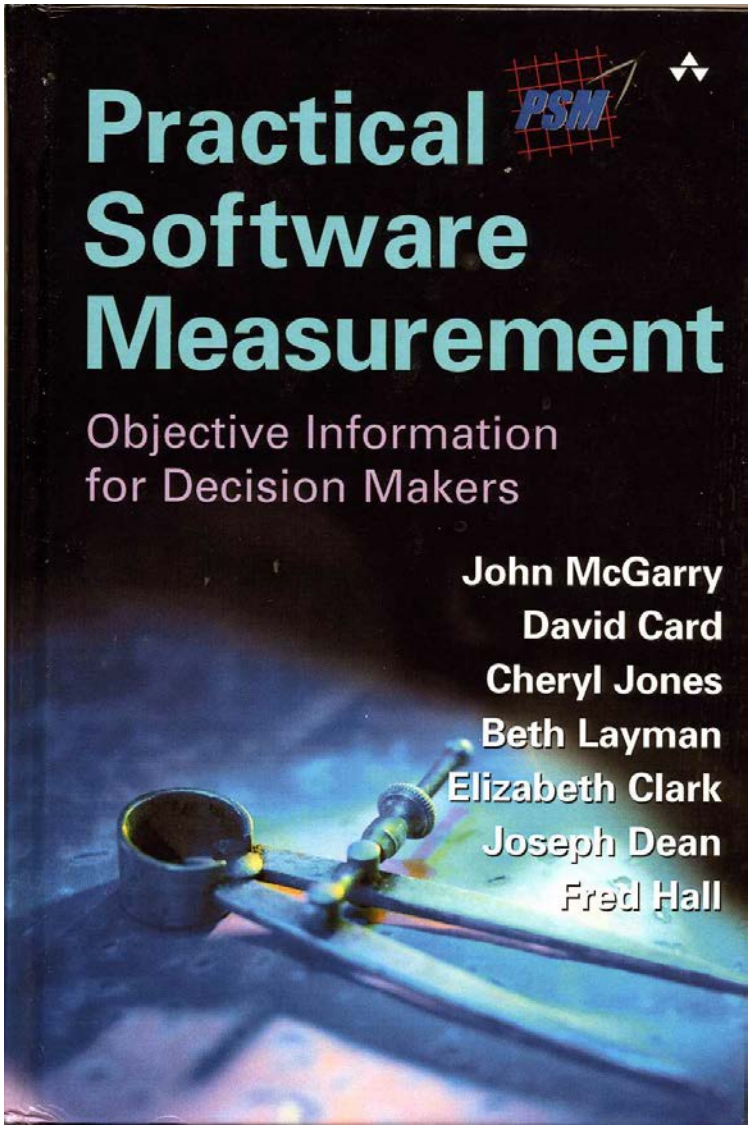
Use in personal appraisals

Unreliable data

Poorly matched to process maturity

Measurement Process Guidance

- Based on decades of software measurement best practice
- Best guidance for starting a measurement program
- Compatible with *ISO 15939 – Software Measurement Process*
- Free guides & licensed training and consulting available
- http://psmsc.com/



McGarry, et al., (2002). Practical Software Measurement.

CISQ
Consortium for IT Software Quality

Measurement Categories

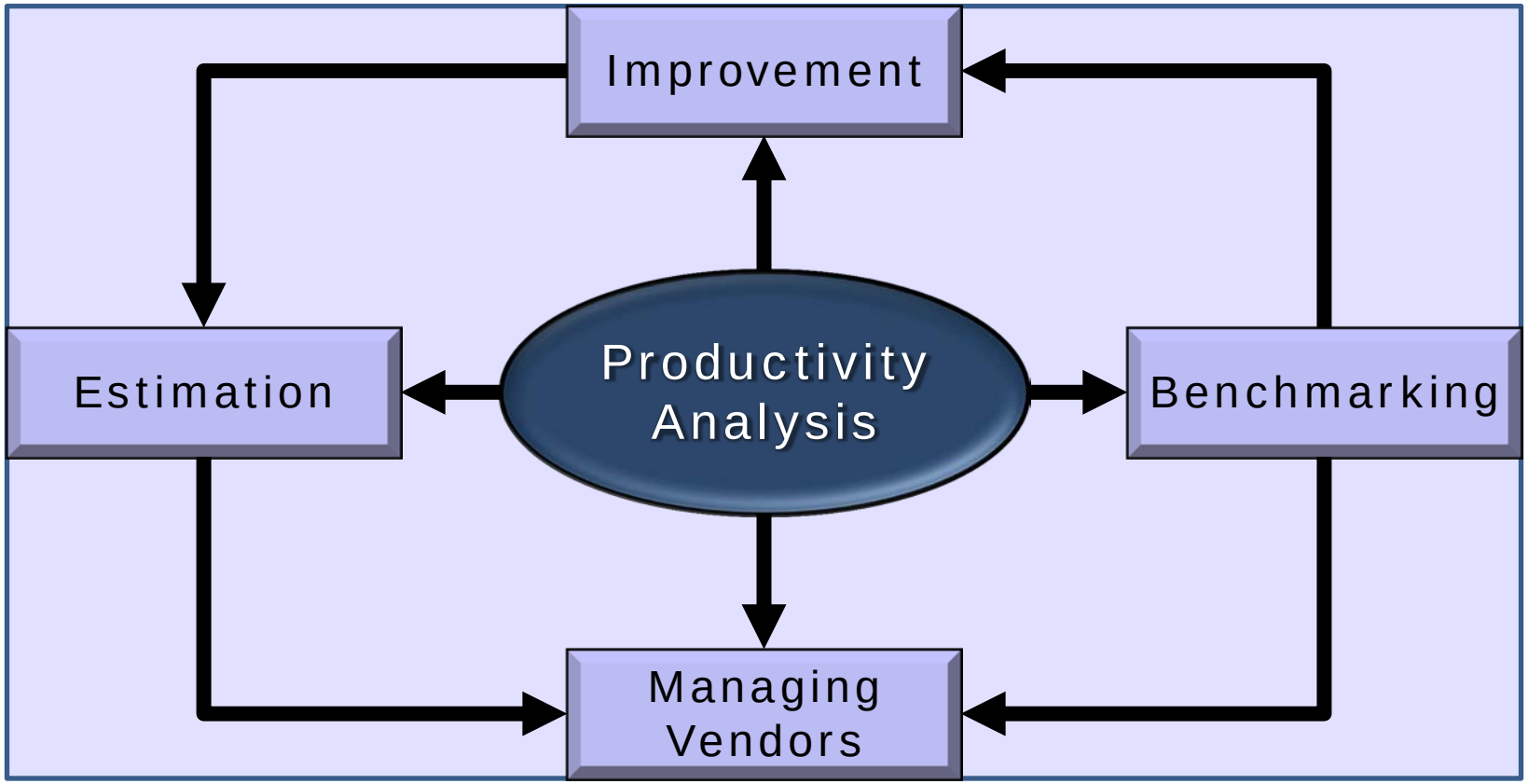
Category	Issue	Example measure
Schedule & Progress	Milestone completion Earned value	% Actual vs. planned dates Actual vs. planned completions
Resources & Cost	Personnel effort Facilities used	Person-hours Test equipment & hours
Product Size	Size of coded product Size of documentation	Function Points Pages of manuals
Product Quality	Functional quality Structural quality	Defects per KLOC Violations of reliability rules
Process Performance	Efficiency Compliance	Defect detection efficiency Audit results
Product Value	Return on investment Risk avoidance	Δ Increase in target revenue Cost of system outage
Customer Satisfaction	Account growth Self–sufficiency	Repeat business within 1 year Number of helpdesk calls

≥ Level 3 | Level 2

McGarry, et al., (2002). *Practical Software Measurement*, p.37

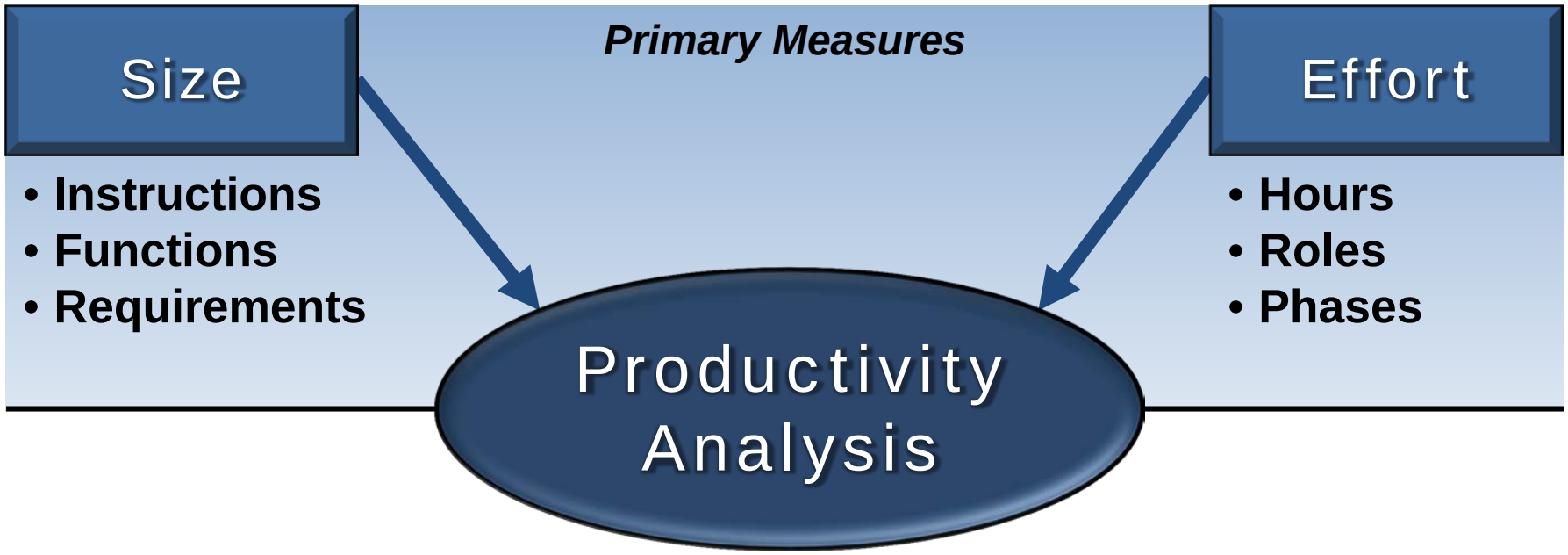
CISQ
Consortium for IT Software Quality

Productivity Analysis Objectives



CISQ
Consortium for IT Software Quality

Productivity Analysis Measures



CISQ
Consortium for IT Software Quality

Software Productivity

$$\text{Software Productivity} = \frac{\text{Size of software produced}}{\text{Total effort expended to produce it}}$$

$$\text{Release Productivity} = \frac{\text{Size of software} \left\{ \begin{array}{l} \text{developed,} \\ \text{deleted, or} \\ \text{modified} \end{array} \right.}{\text{Total effort expended on the release}}$$

CISQ
Consortium for IT Software Quality

Software Size Measures

Instructions	Lines of Code
Most frequently used. Different definitions of a line can cause counts to vary by 10x. Smaller programs often accomplish the same functionality with higher quality coding.	

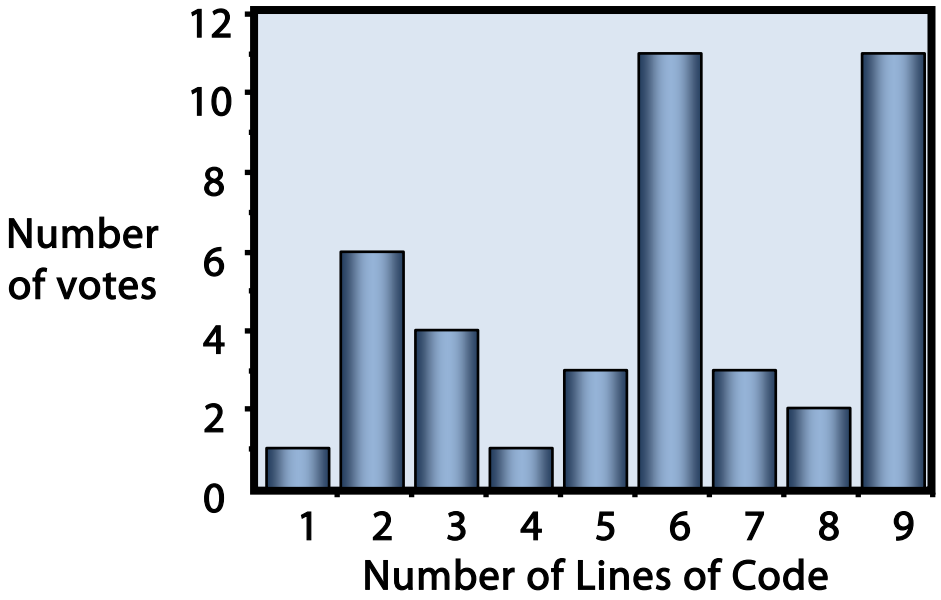
Requirements-based	Use Case Points, Story Points
Use Case Points have not become widely used and need more development. Story points are subjective to each team and are susceptible to several forms of bias.	

Functions	Function Points
Popular in IT. Several counting schemes (IFPUG, NESMA, Mark II, COSMIC, etc.). Manual counting is expensive and subjective—certified counters can differ by 10%. Automated FPs taking root.	

CISQ
Consortium for IT Software Quality

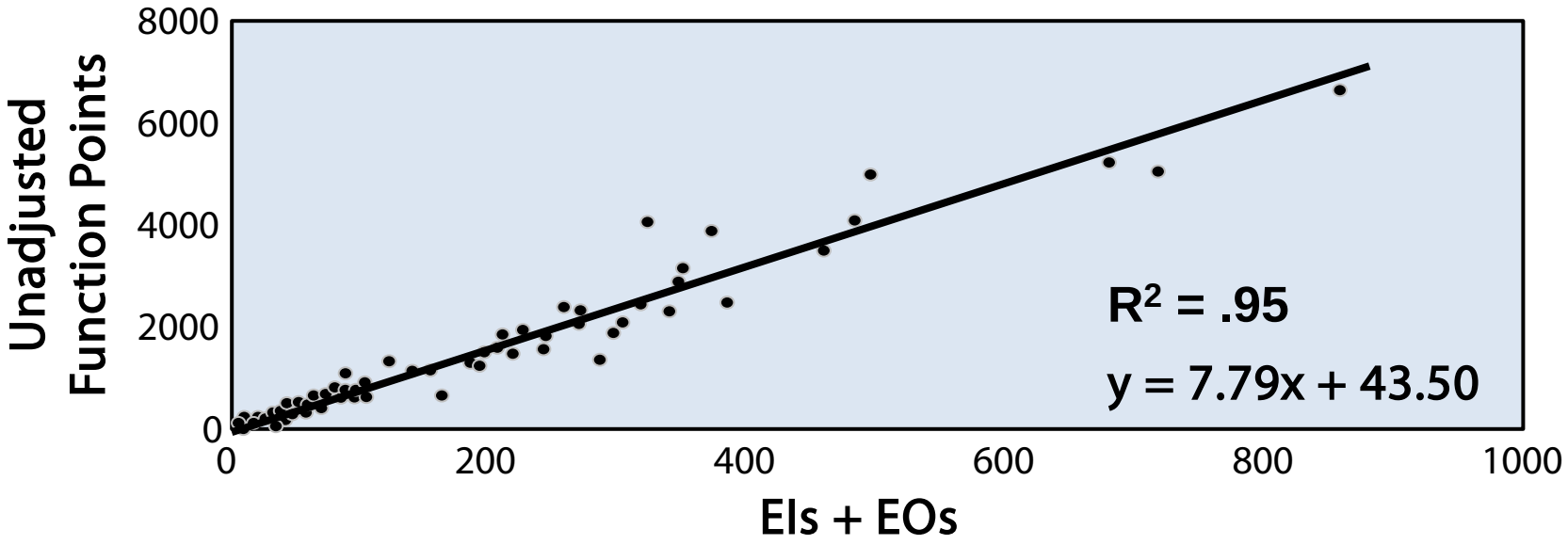
How Many Lines of Code ?

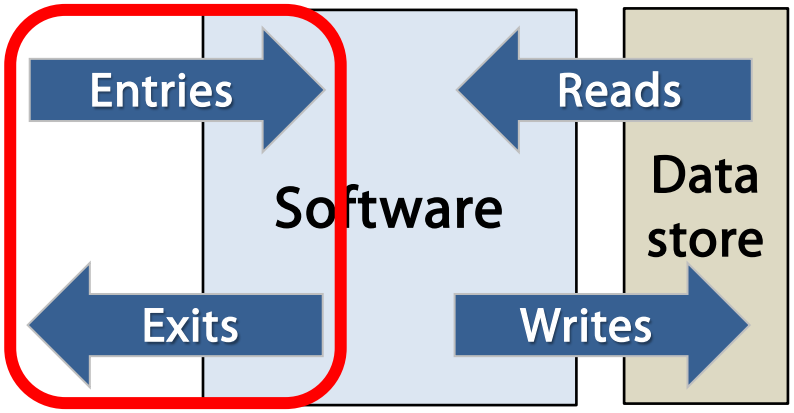
```
#define   LOWER 0       /* lower limit of table */
#define   UPPER 300     /* upper limit */
#define   STEP 20       /* step size */

main()    /* print a Fahrenheit-Celsius conversion table
*/
{
  int fahr;
  for(fahr=LOWER; fahr<=UPPER; fahr=fahr+STEP)
      printf("%4d %6.1f\n", fahr, (5.0/9.0)*(fahr-32));
}
```



B. Park (1992)

Function Point Estimation





Functional view of software

- Functional size can be estimated from external inputs and outputs
- Upfront functional analysis provides basis for good estimates
- Repository of FP data provides basis for calibrating estimates

Ebert & Dumke (2007). *Software Measurement*, p.188.

CISQ
Consortium for IT Software Quality

Effort — Weakest Measure

After the fact estimates

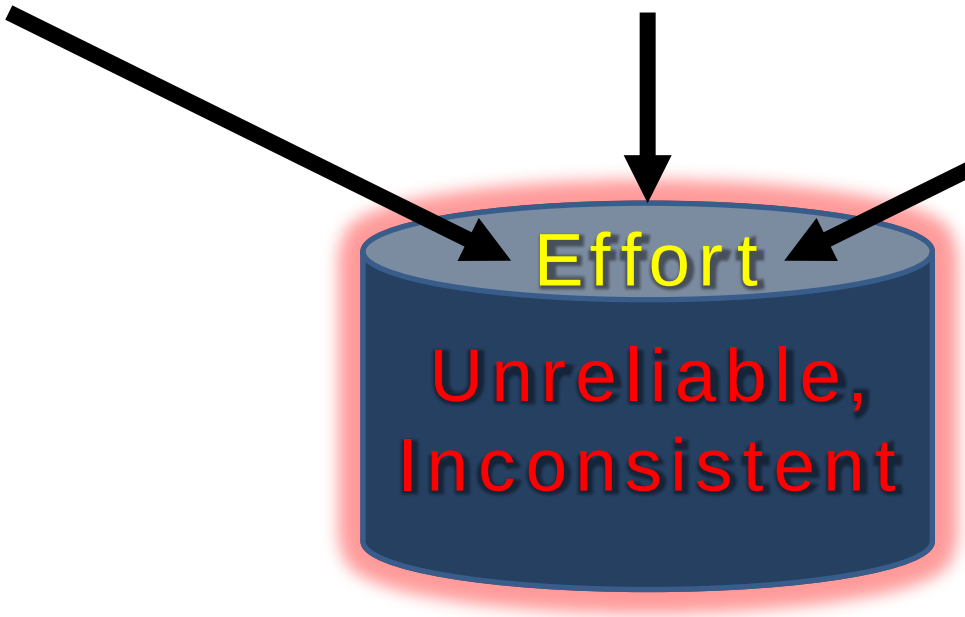
- Memory lapses
- Time-splicing
- Inconsistency

Under-reporting

- Contract issues
- HR issues
- Impressions

Lack of normalization

- Roles included
- Phases included
- Hours in P-Year

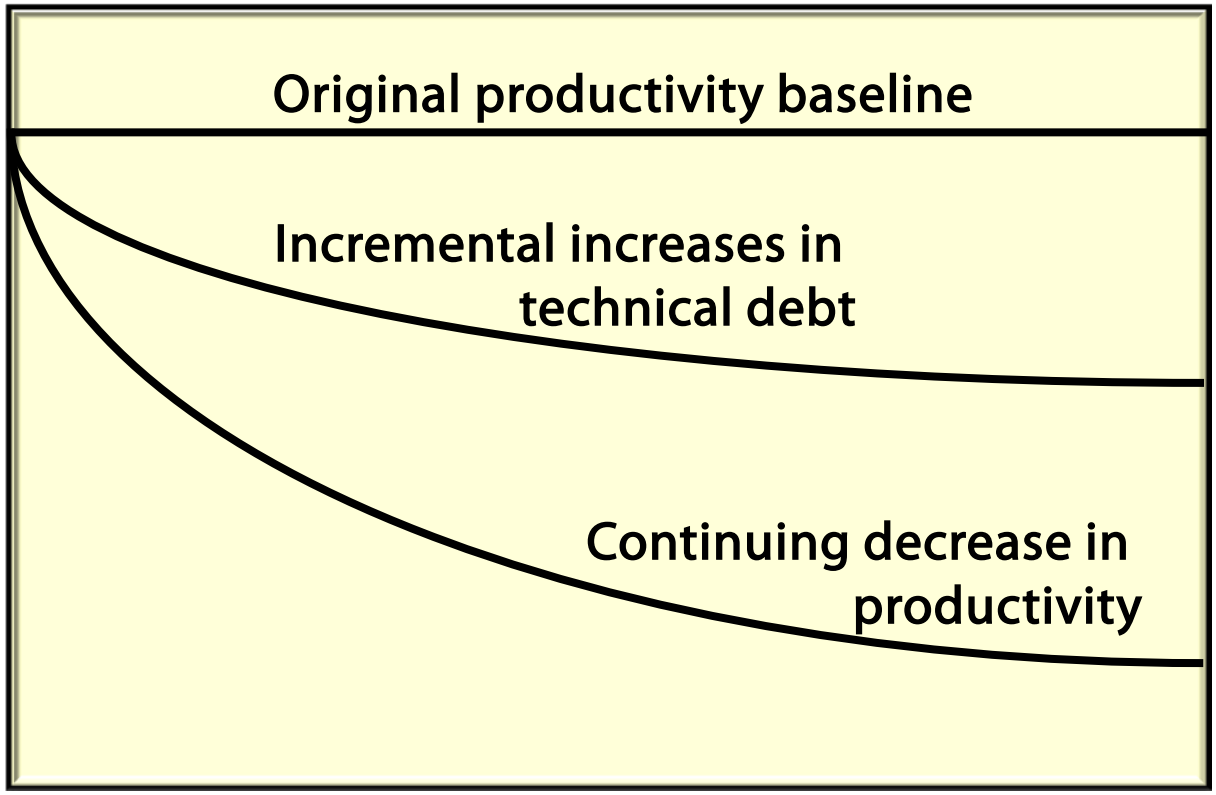


CISQ
Consortium for IT Software Quality

How Quality Affects Productivity

Assumption: Productivity is a stable number

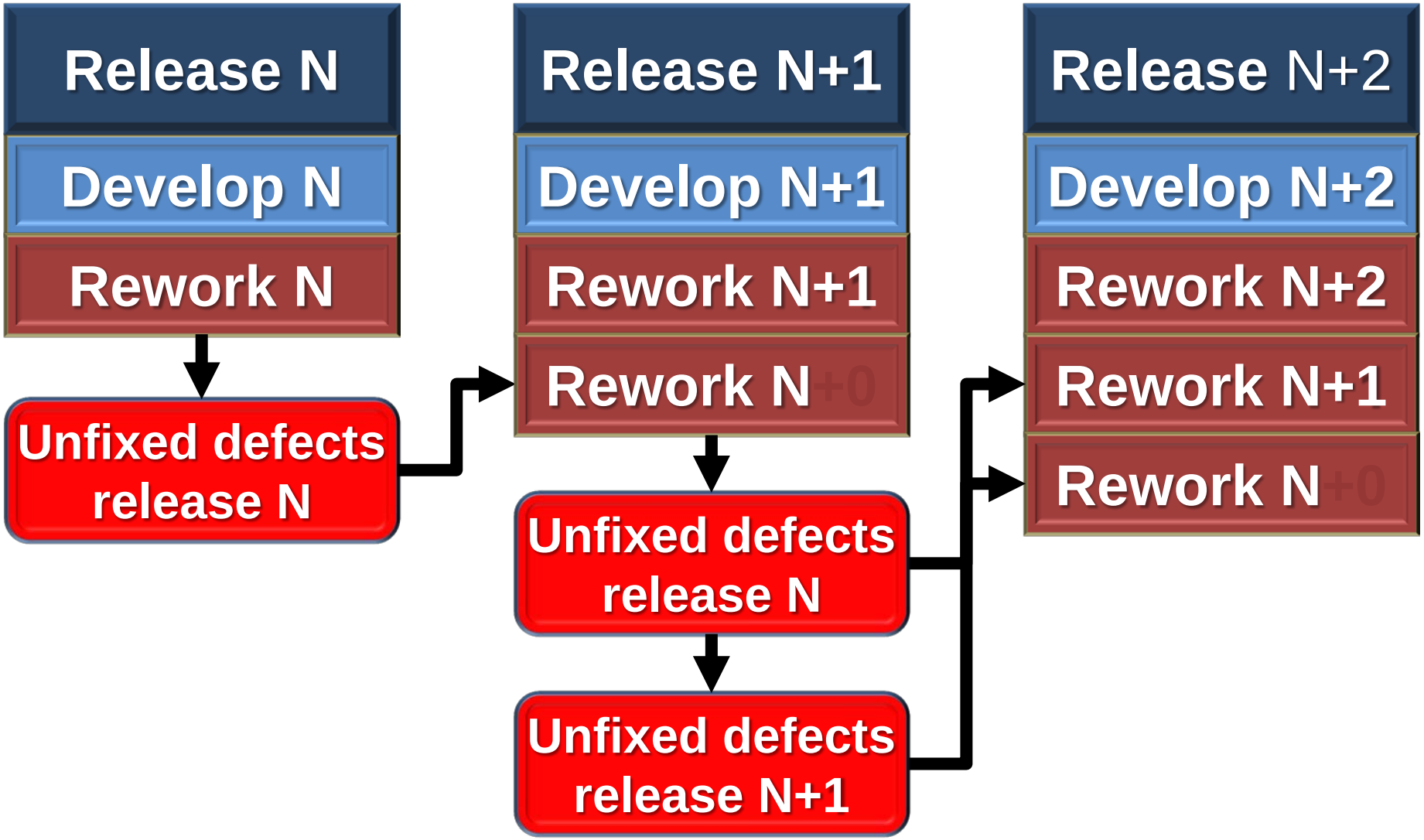
Reality: Productivity is unstable, tending to decline



Unless you take action !!!

CISQ
Consortium for IT Software Quality

Carry-forward Rework



CISQ
Consortium for IT Software Quality

Example of Quality Impact

Project A (Plodders)

- 20 developers, 3 months
- $120k per FTE
- 3 FPs per staff month
- 180 FPs delivered

- $3,333/FP cost

- 2 critical violations per FP
- $500 per fix
- Cost for 360 fixes = $180k
- Total Cost to Own = $780k

- $4,333/FP of TCO

Project B (Better, Faster, Cheaper)

- 20 developers, 3 months
- $120k per FTE
- 4 FPs per staff month
- 240 FPs delivered

- $2,500/FP cost

- 5 critical violations per FP
- $500 per fix
- Cost for 1200 fixes = $600k
- Total Cost to Own = $1,200k

- $5,000/FP of TCO

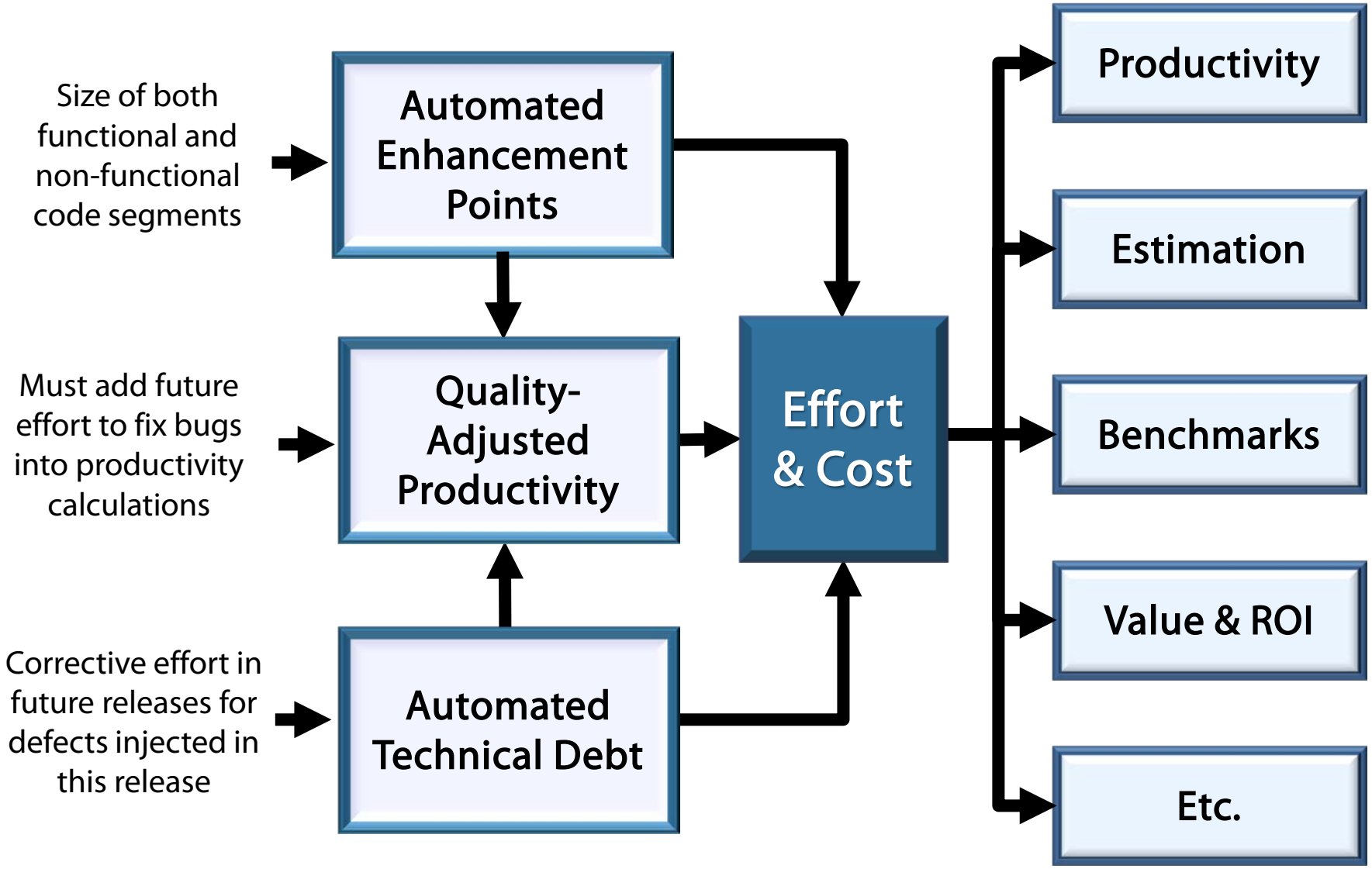
Project B is 25% more productive

However !!!

Project A is 13.4% more productive

CISQ
Consortium for IT Software Quality

Quality-Adjusted Productivity



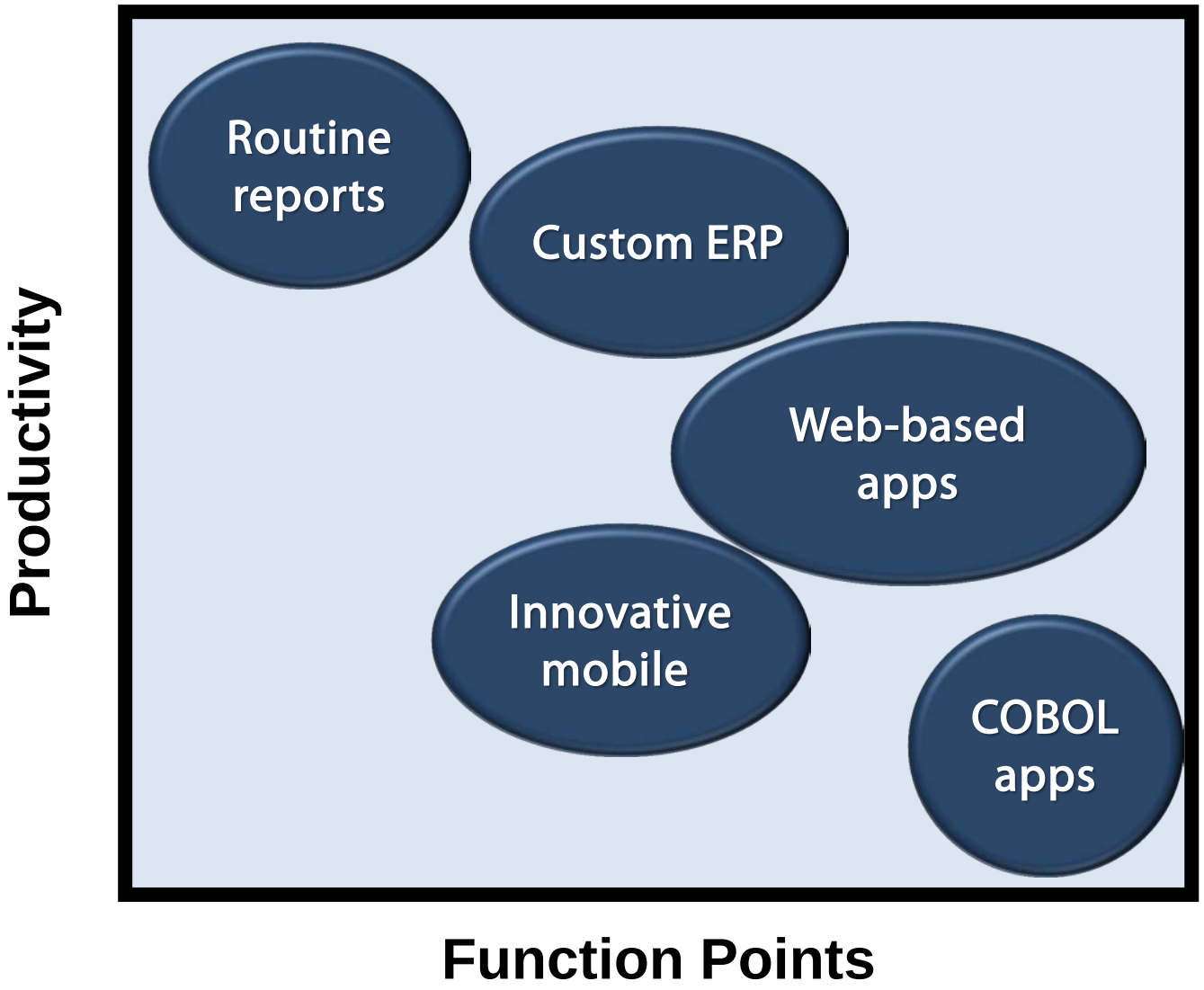
CISQ
Consortium for IT Software Quality

Best Practices for Analysis

1) Segment baselines
2) Beware sampling effects
3) Understand variation
4) Evaluate demographics
5) Investigate distributions
6) Account for maturity effects
7) Beware external data sets

CISQ
Consortium for IT Software Quality

1 — Evaluate Demographics



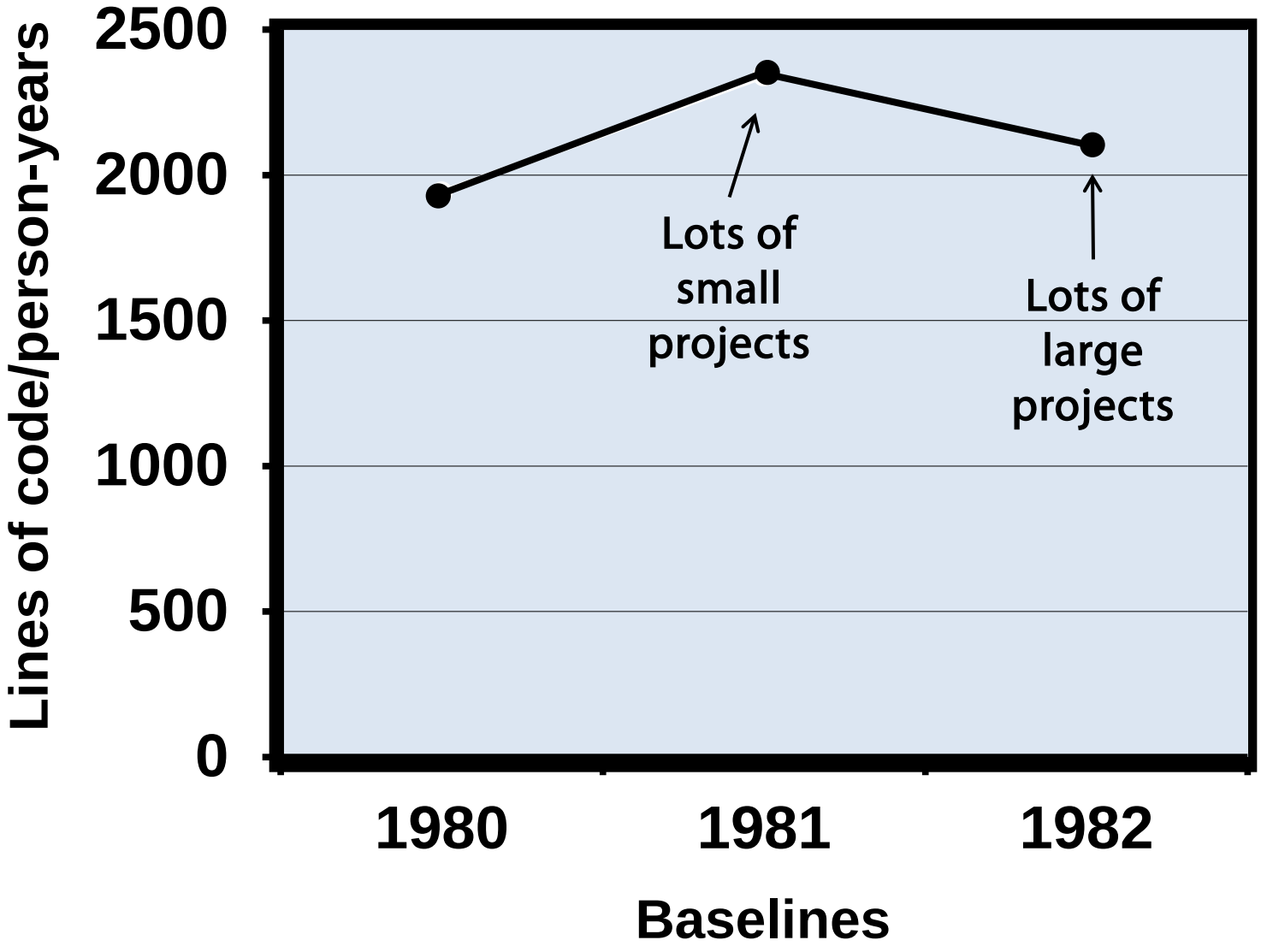
CISQ
Consortium for IT Software Quality

2 — Segment Baselines

Multiple baselines are usually the most valid

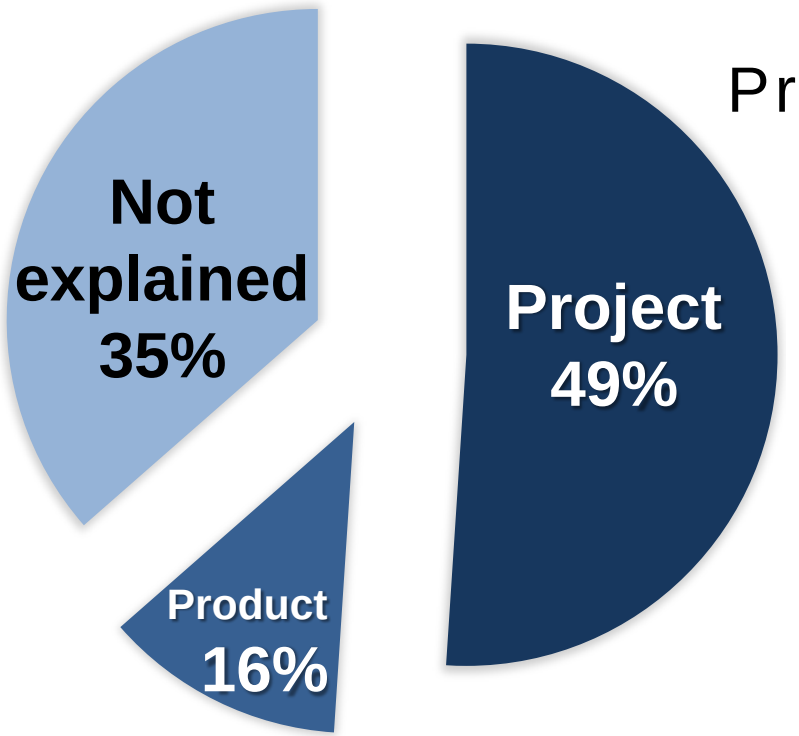
Year	Projects	Productivity
Total Corporate		
1981	28	2342
1980	21	1939

3 — Beware Sampling Effects



CISQ
Consortium for IT Software Quality

4 — Understand Variation



Project factors:

- concurrent development
- development computer
- requirements definition
- requirements churn
- programming practices
- personnel experience
- client experience
- client participation

Product factors:

- timing constraints
- memory utilization
- CPU occupancy
- resource constraints
- complexity
- product size

$R^2_{product} = .16$

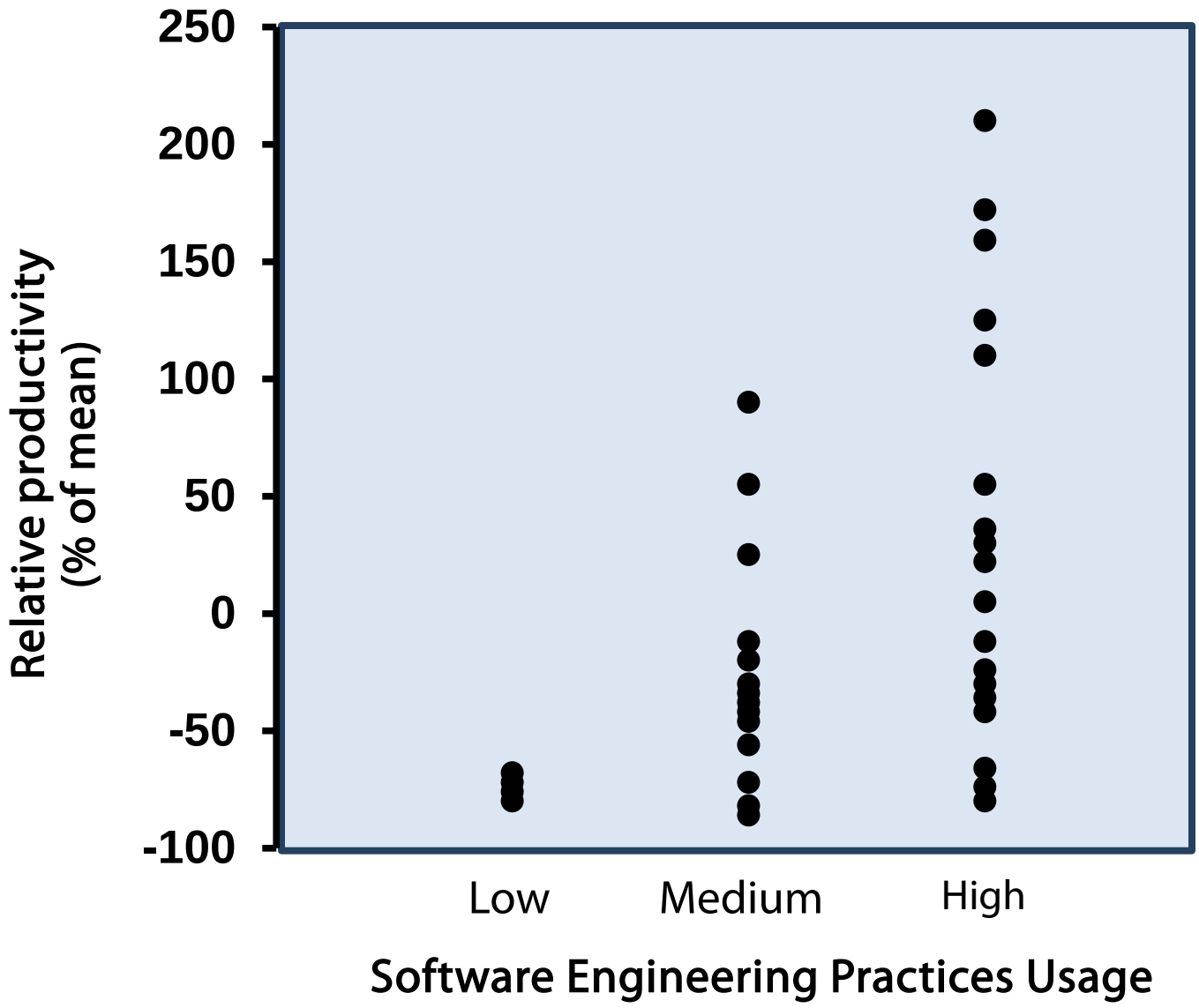
$R^2_{project} = .49$

$R^2_{model} = .65$

J. Vosburgh, B. Curtis, et al. (1984). *Proceedings of ICSE 1994.*

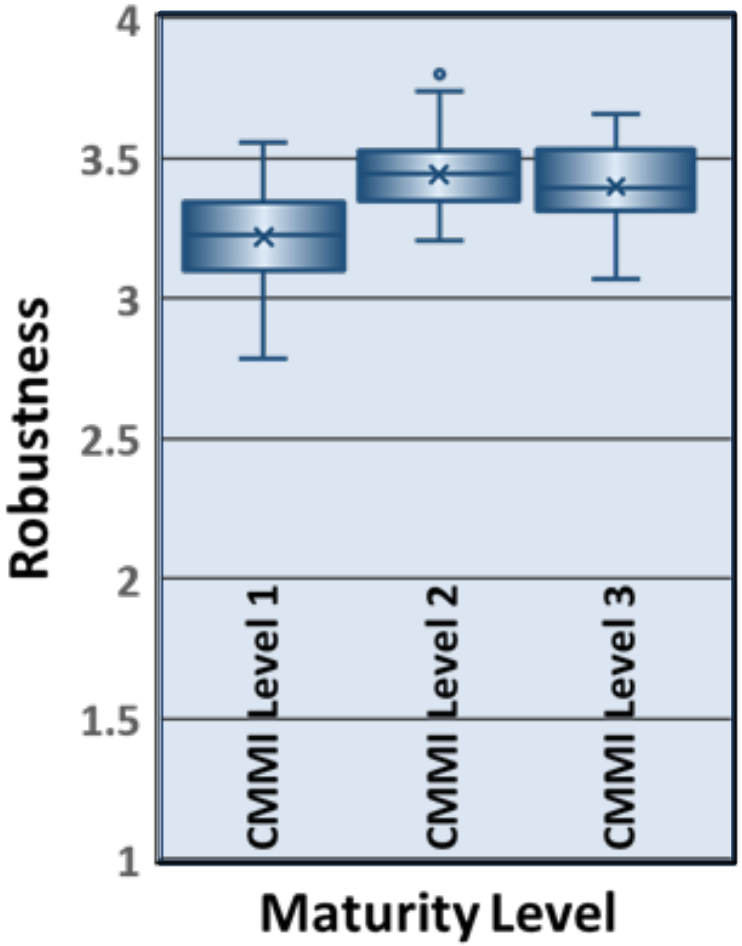
CISQ
Consortium for IT Software Quality

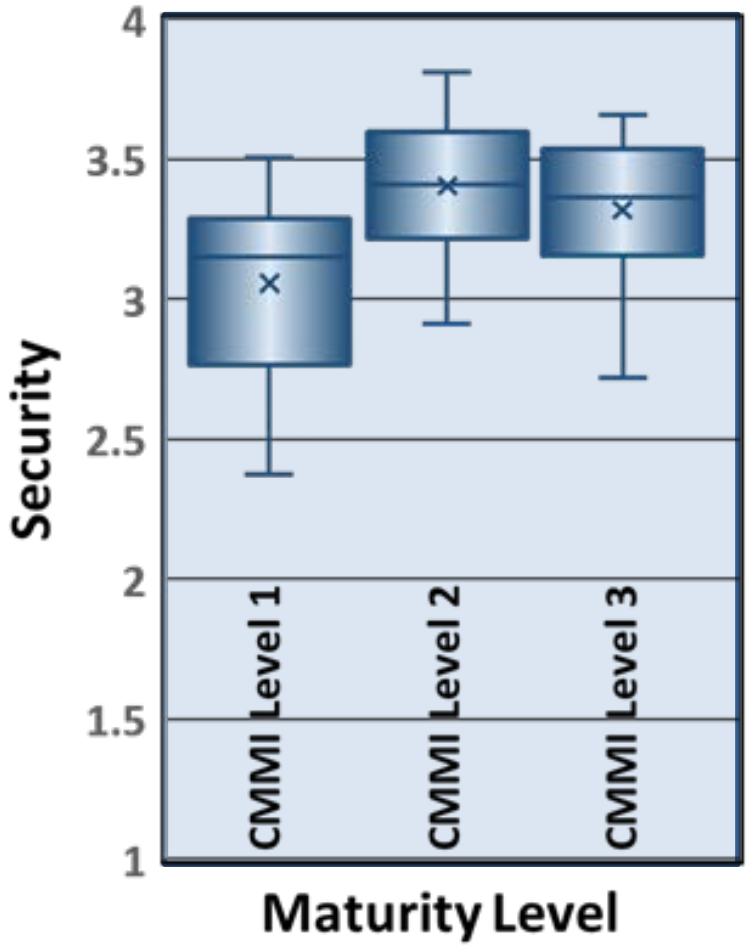
5 — Investigate Distributions



CISQ
Consortium for IT Software Quality

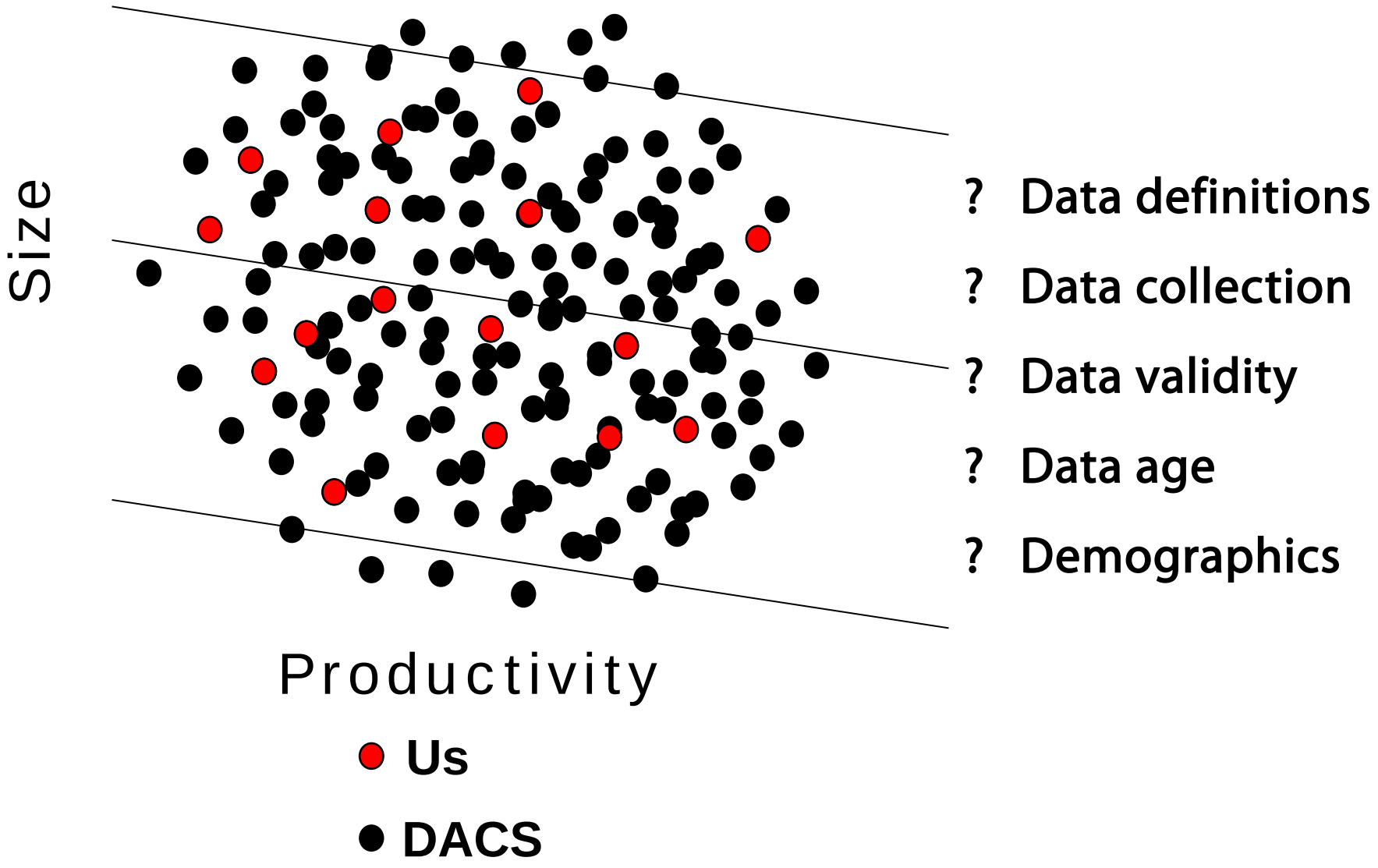
6 — Account for Maturity Effects





Which organization will be required to spend more effort on correcting software flaws?

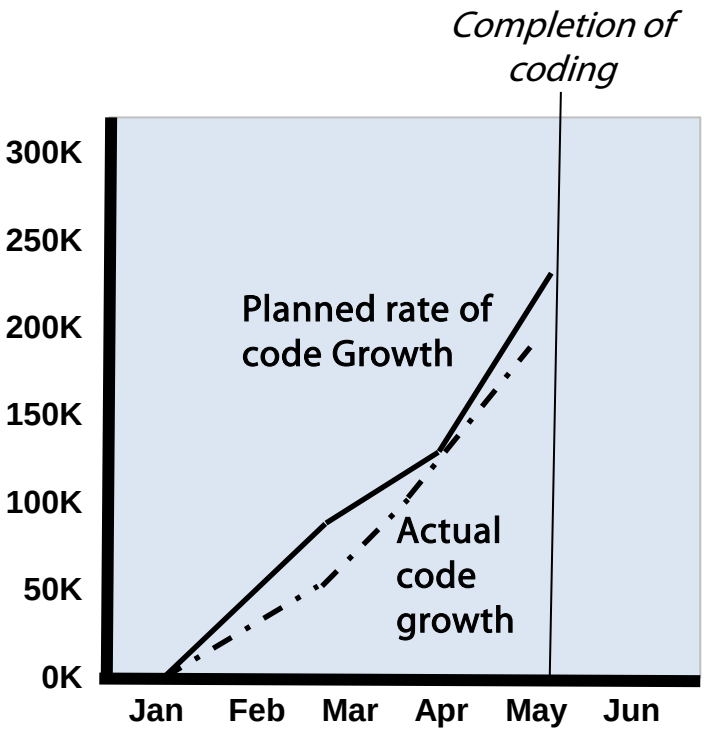
7 — Caution for External Data Sets



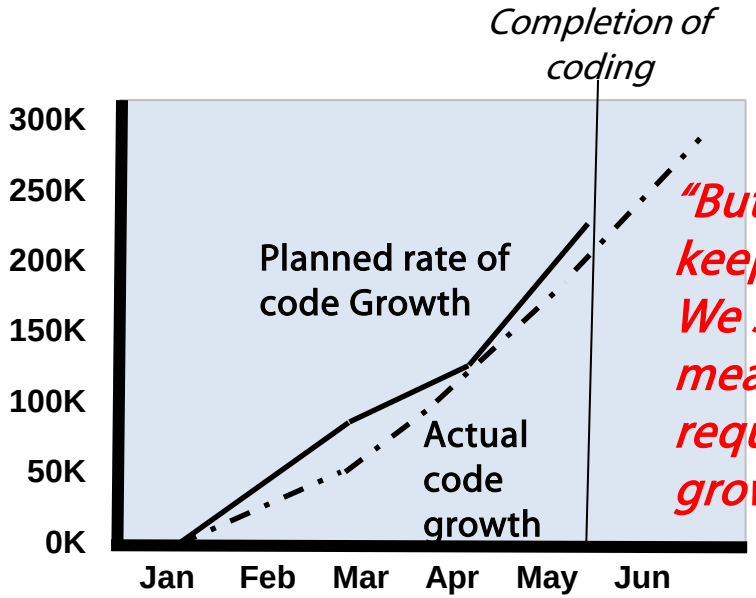
Section 2
Measuring to Manage Software Projects

1) Tracking traditional projects
2) Iterative and Agile measurement
3) Team Software Process measurement

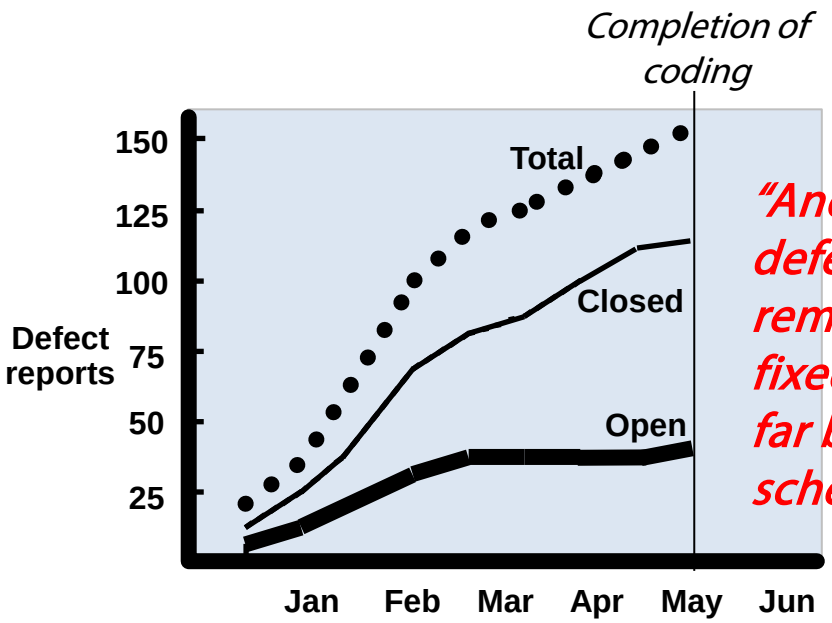
Measuring Project Progress



"The measures say we are on schedule"



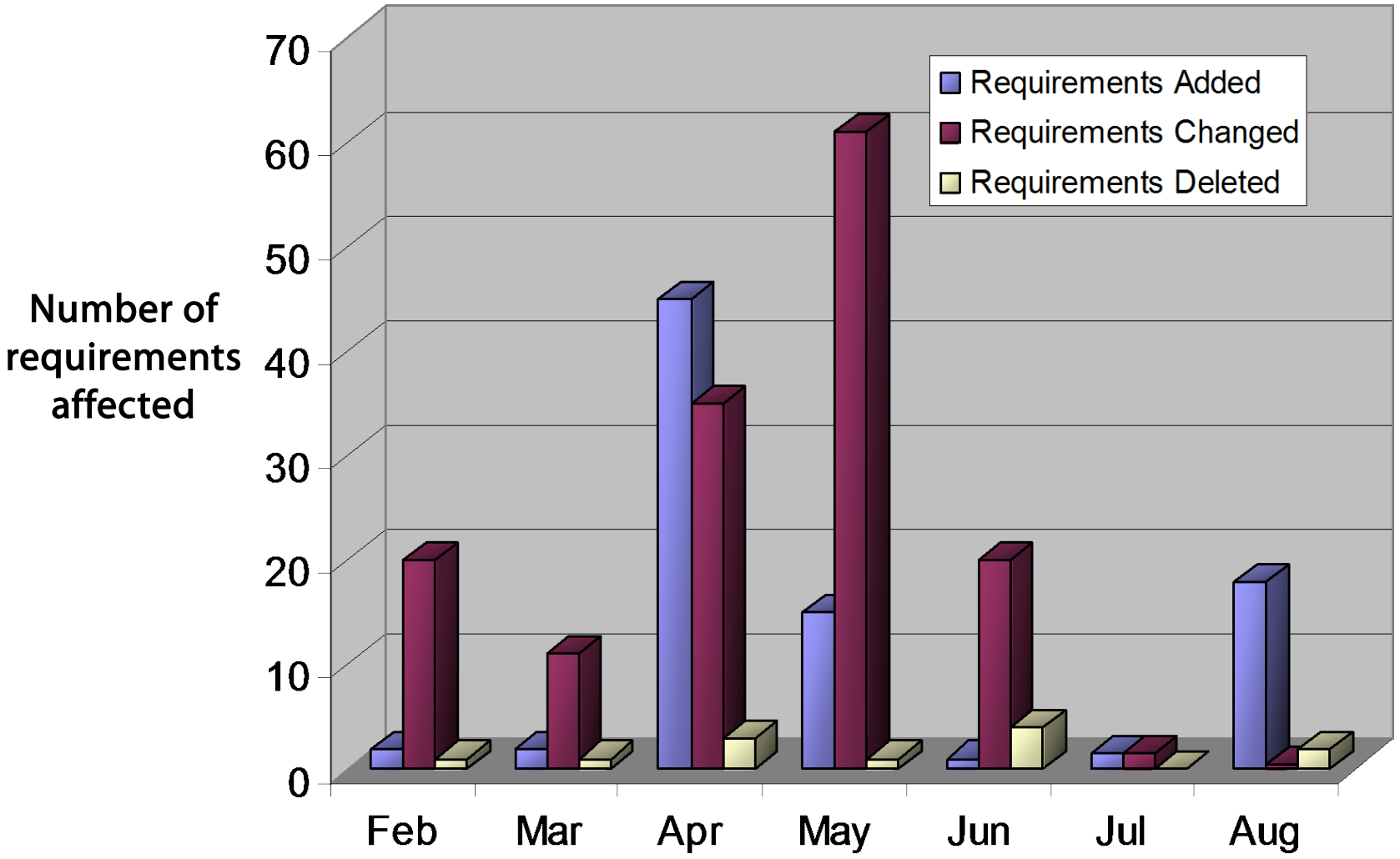
"But the code keeps growing. We should have measured the requirements growth"



"And all the defect reports remaining to be fixed will keep us far behind schedule"

CISQ
Consortium for IT Software Quality

Requirements Change Impact



CISQ
Consortium for IT Software Quality

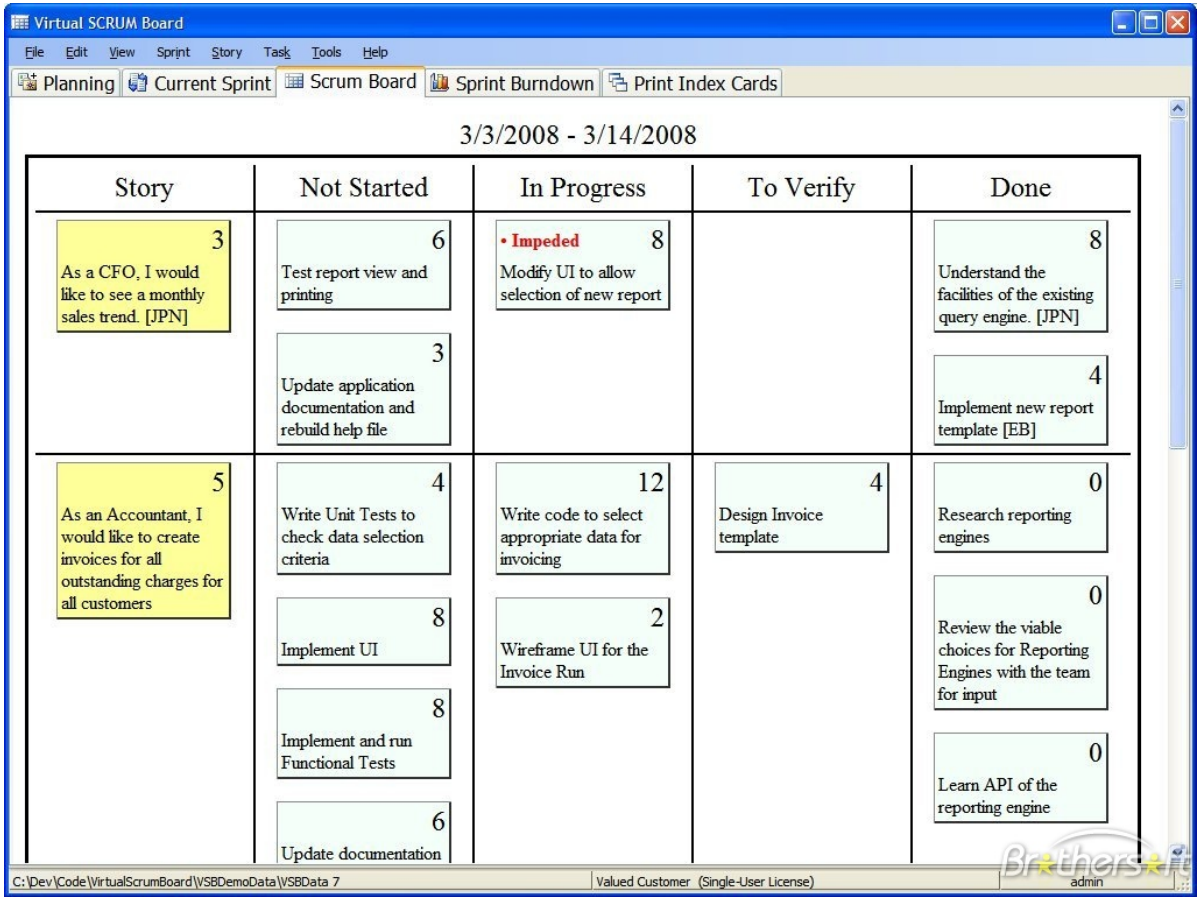
Phase Defect Removal Model

PHASE	Escapes Previous Phase / KSLOC	Defect Injection / KSLOC	Subtotal / KSLOC	Removal Effective-ness	Defect Removal / KSLOC	Escapes at phase exit / KSLOC
Rqmts	0	1.2	1.2	0%	0	1.2
Design	1.2	18.0	19.2	76%	14.6	4.6
Code & U.T.	4.6	15.4	20.0	71%	14.2	5.8
Inte-gration	5.8	0	5.8	67%	3.9	1.9
Sys. Test	1.9	0	1.9	58%	1.1	0.8
Opera-tion	0.8					

Kan (1999). *Metrics and Model in Software Quality Engineering.*

CISQ
Consortium for IT Software Quality

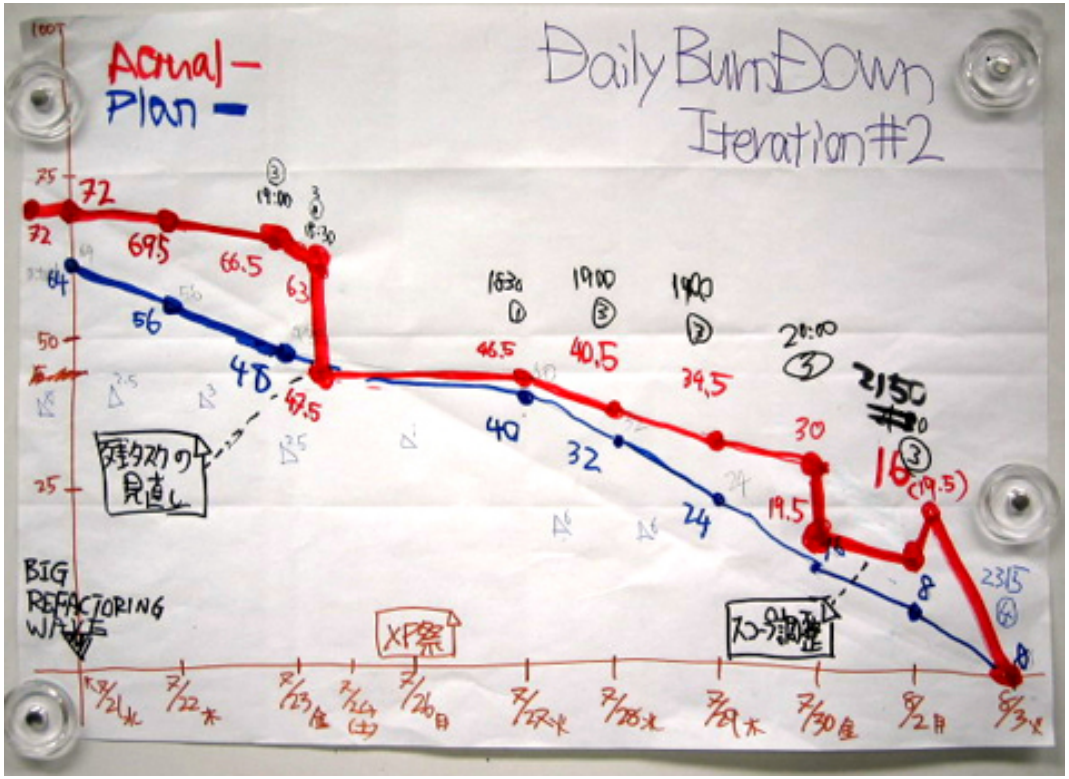
SCRUM Board



- Scrumboard displays the path to completion for each story by showing task status
- Story points are an estimate of days for each story
- Agile estimating is by feature (story) rather than by task
- ‘Done’ does not always mean all tasks are finished
- Kanban methods restrict the work in progress to a limited number of paths

CISQ
Consortium for IT Software Quality

Burndown Chart and Velocity

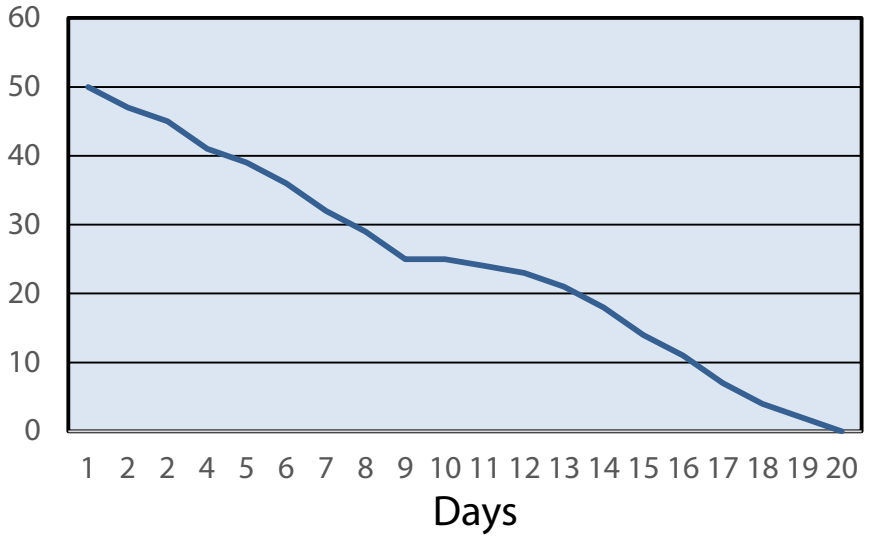


- Burndown chart displays story completion by day
- Burndown chart indicates actual versus estimated progress
- Velocity is the rate at which stories are completed
- Velocity indicates sustainable progress
- Velocity results provide one source of historical data for improving estimates
- Story points without the context of productivity factors are dangerous for estimating

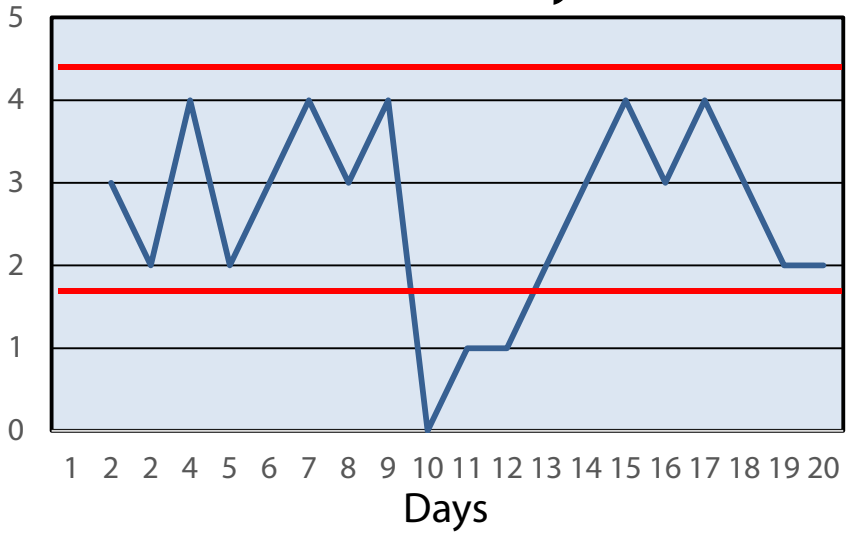
CISQ
Consortium for IT Software Quality

Analyzing Velocity

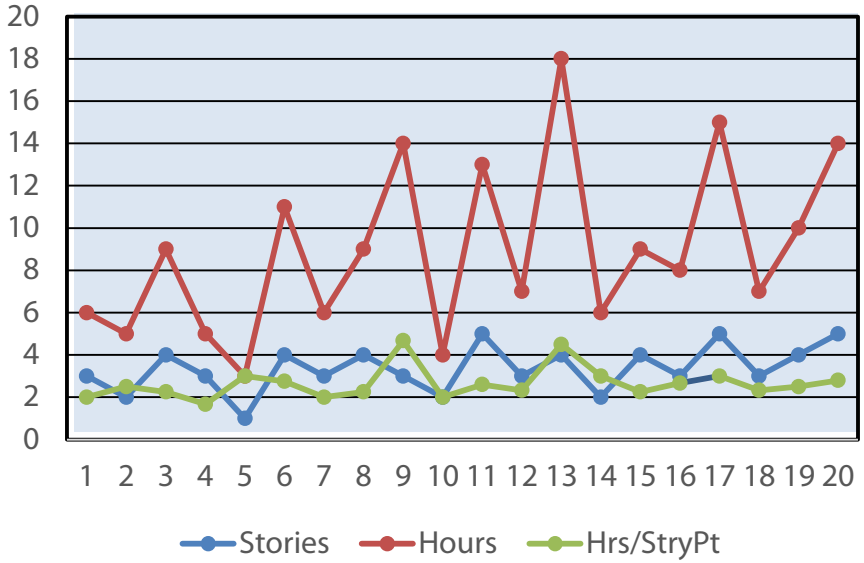
Burndown Chart



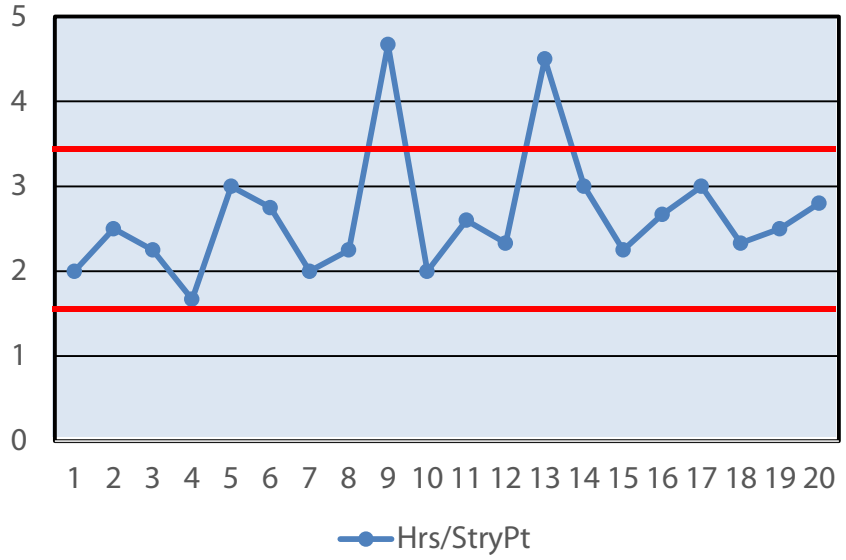
Team Velocity



Item Cycle Times

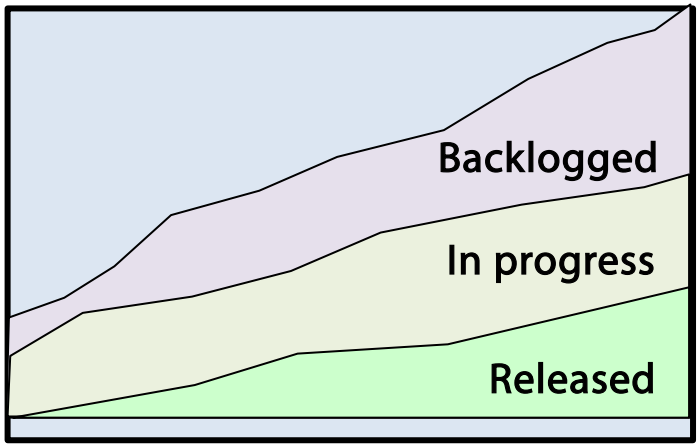


Story Points per Hour

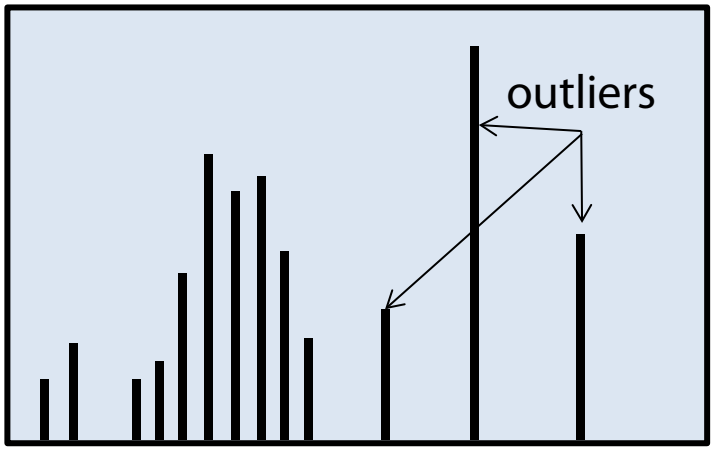


CISQ
Consortium for IT Software Quality

Measuring and Managing Flow



Cumulative Flow Diagram

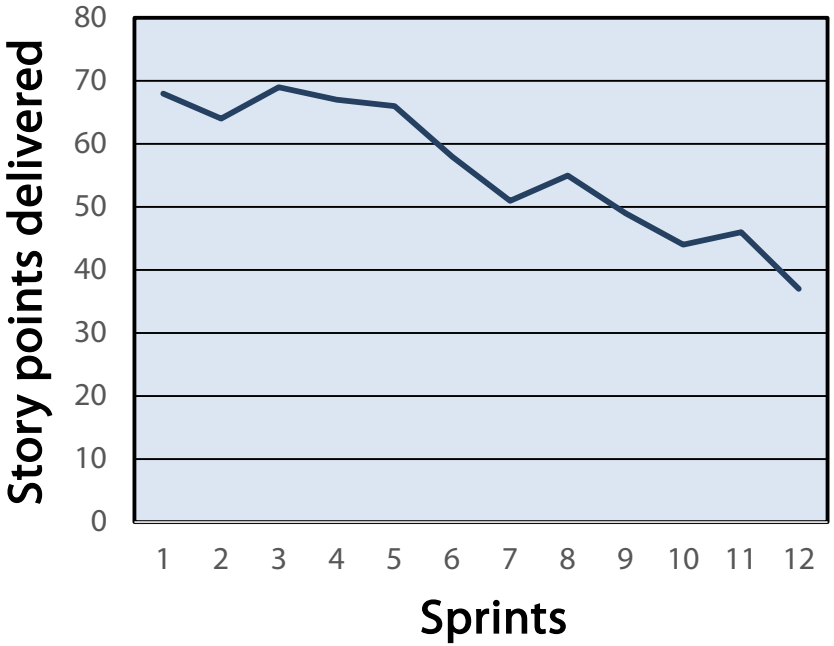


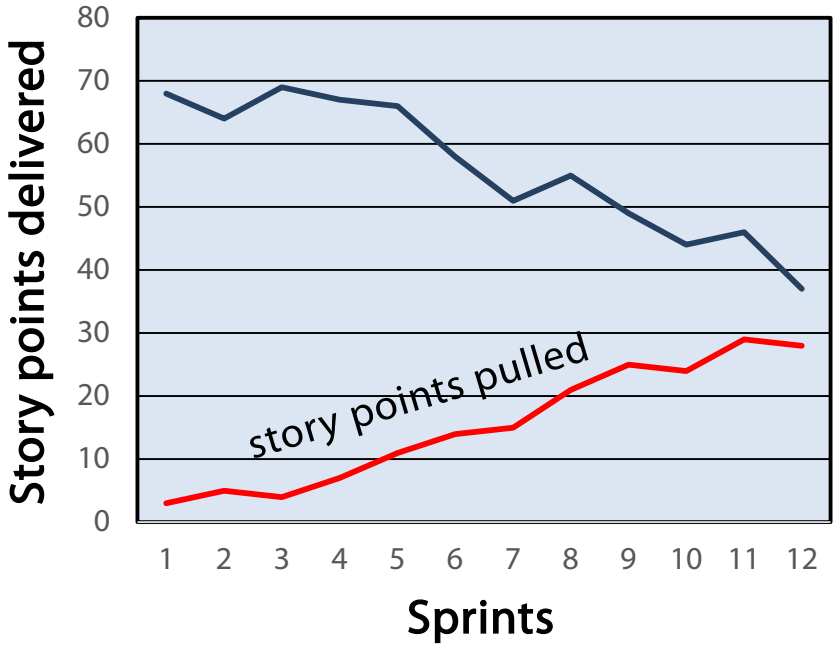
Days to delivery

Anderson (2010). *Kanban*.

CISQ
Consortium for IT Software Quality

Trends Over Sprints

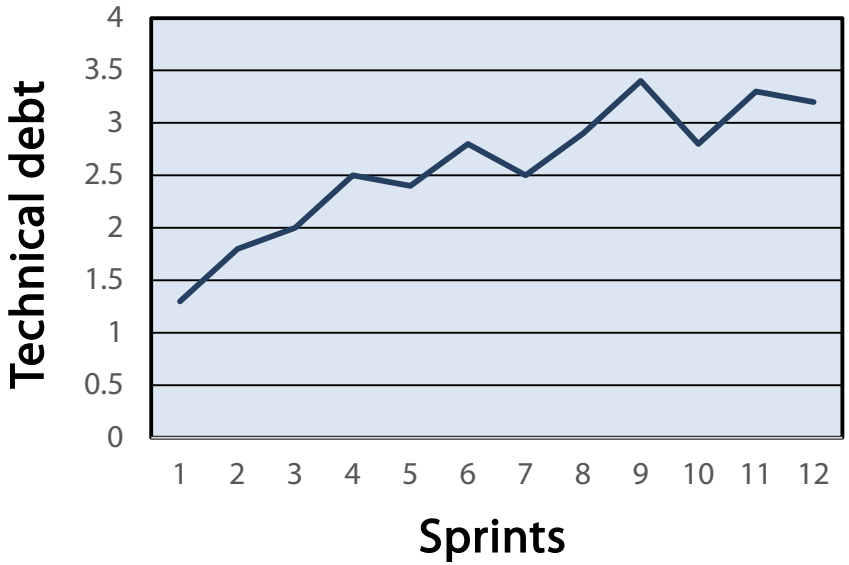




Track and correlate multiple measures across sprints

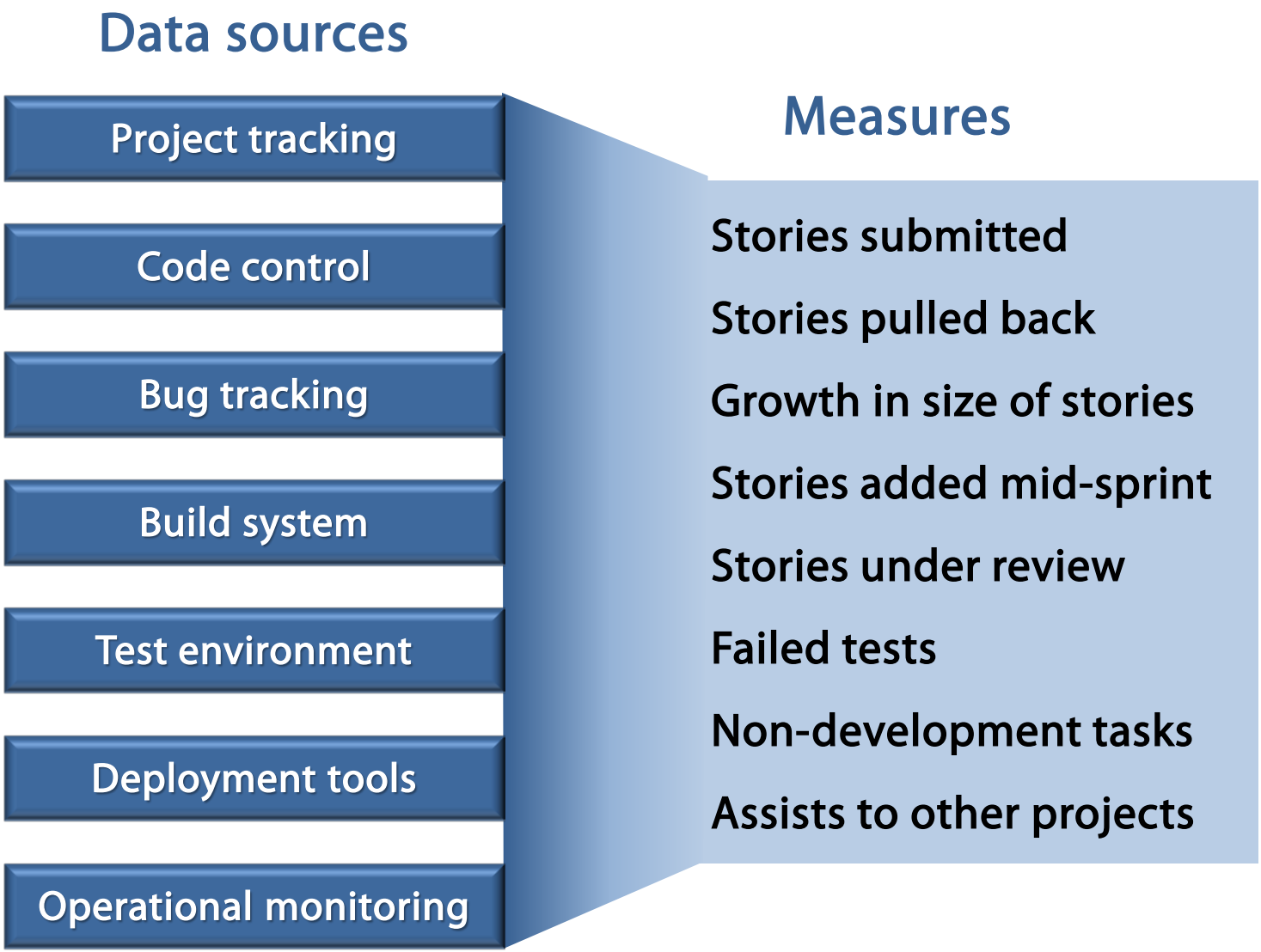
↓

Predictive models



CISQ
Consortium for IT Software Quality

Sources and Measures

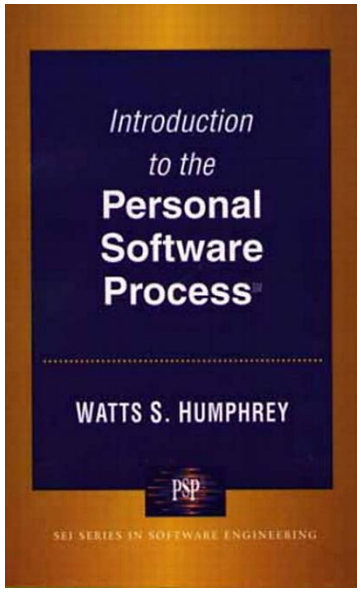


CISQ
Consortium for IT Software Quality

Recommended Books

CISQ
Consortium for IT Software Quality

Personal Software Process (PSP)



Cyclic Personal Process

PSP 2.1
Cyclic development

Personal Quality Management

PSP 2
Code reviews
Design reviews

PSP 2.1
Design templates

Personal Planning

PSP 1
Size estimating
Test report

PSP 1.1
Task planning
Schedule planning

Baseline Personal Process

PSP 0
Current process
Time recording
Defect recording
Defect type standards

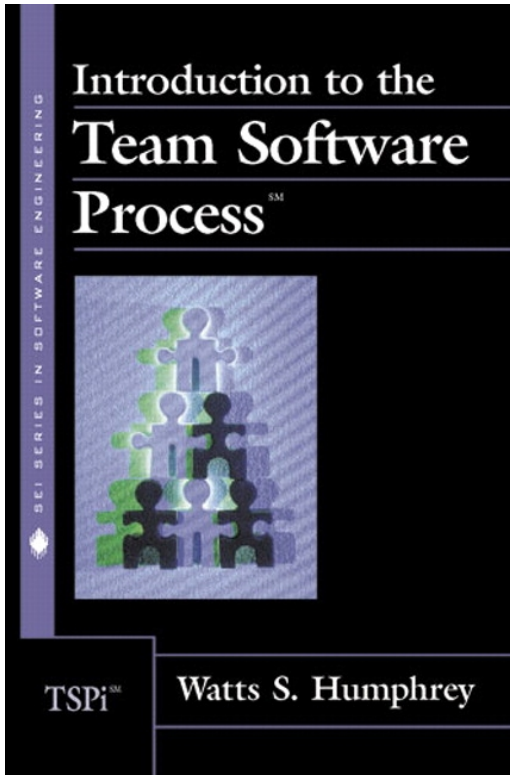
PSP 0.1
Coding standard
Size measurement
Process improvement proposal

Humphrey (1997). *Introduction to the Personal Software Process*.

CISQ
Consortium for IT Software Quality

Team Software Process (TSP)

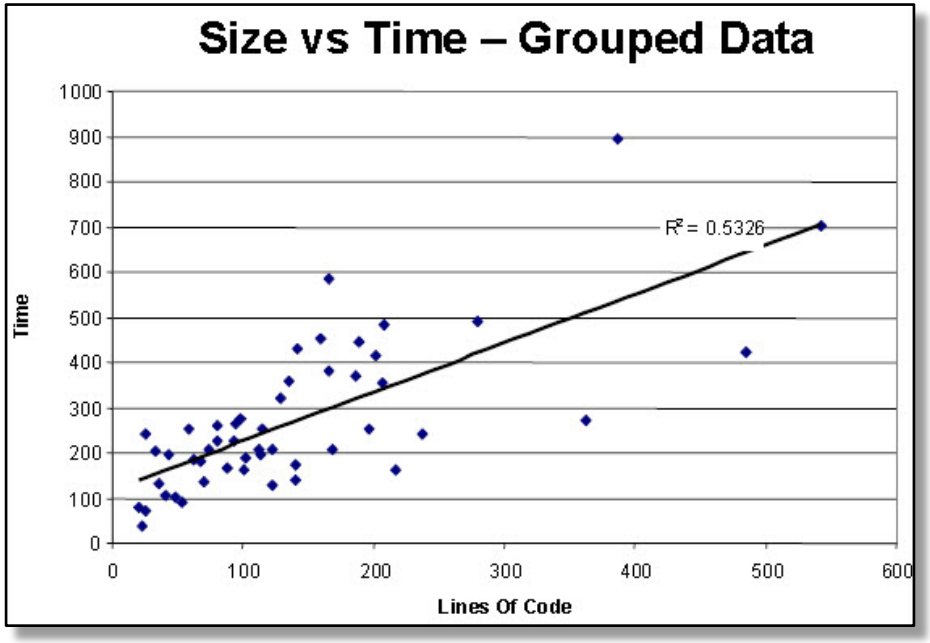
- **Built from personal processes of team members**
- **Well defined team roles**
- **Project launch workshops for planning**
- **Team measurement and tracking**
- **Post-mortems for improvement**
- **Team application of lean principles**



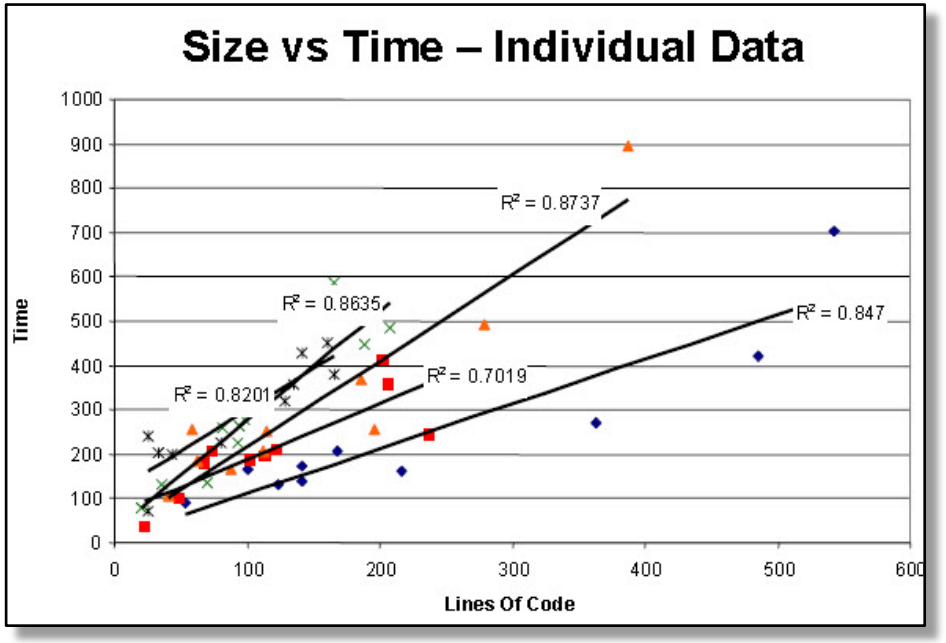
Humphrey (2000). *Introduction to the Team Software Process*.

CISQ
Consortium for IT Software Quality

Aggregated Personal Data Best



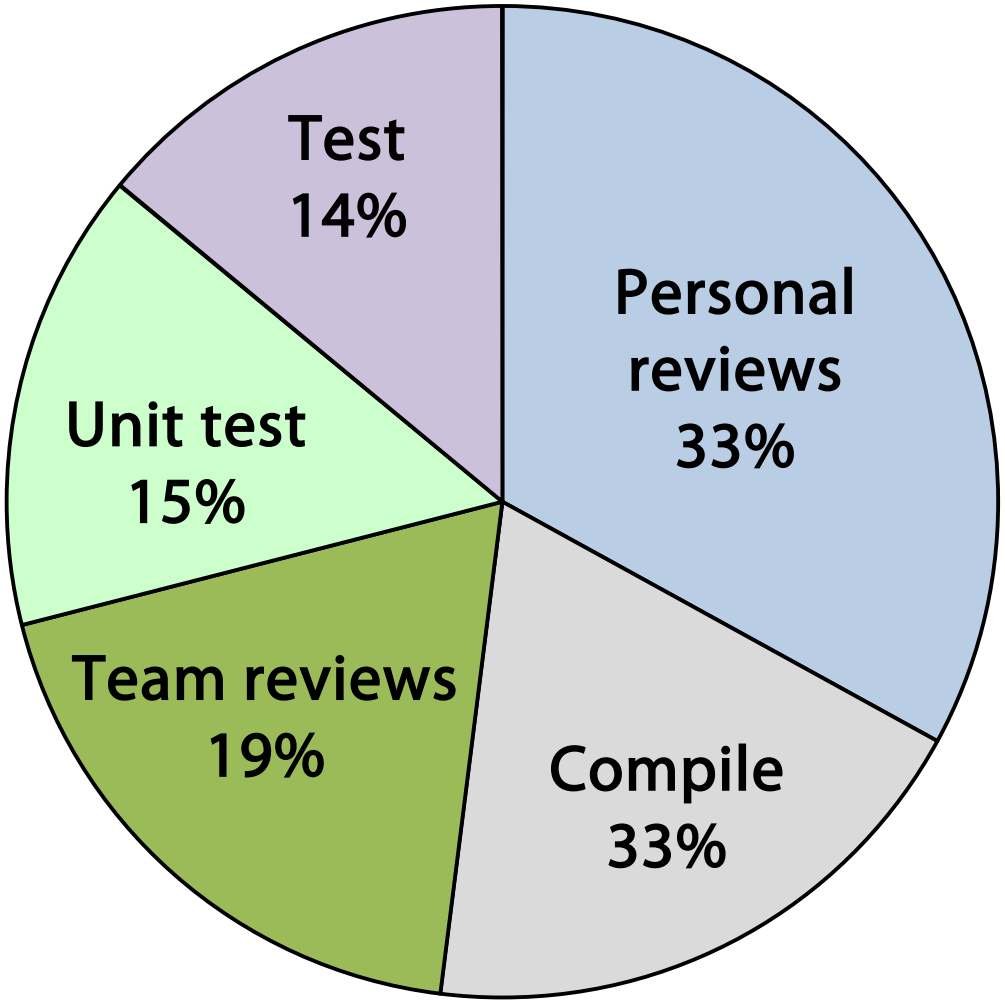
$R^2 = .53$



R^2s range from .70 to .87

The ability to predict the amount of time required to produce a piece of software is dramatically improved by estimating at the individual level first and then combining the estimates rather than developing a single team estimate averaged over individuals.

Defect Detection at Intuit



PSP/TSP shifts defect detection from the test phase back up into development

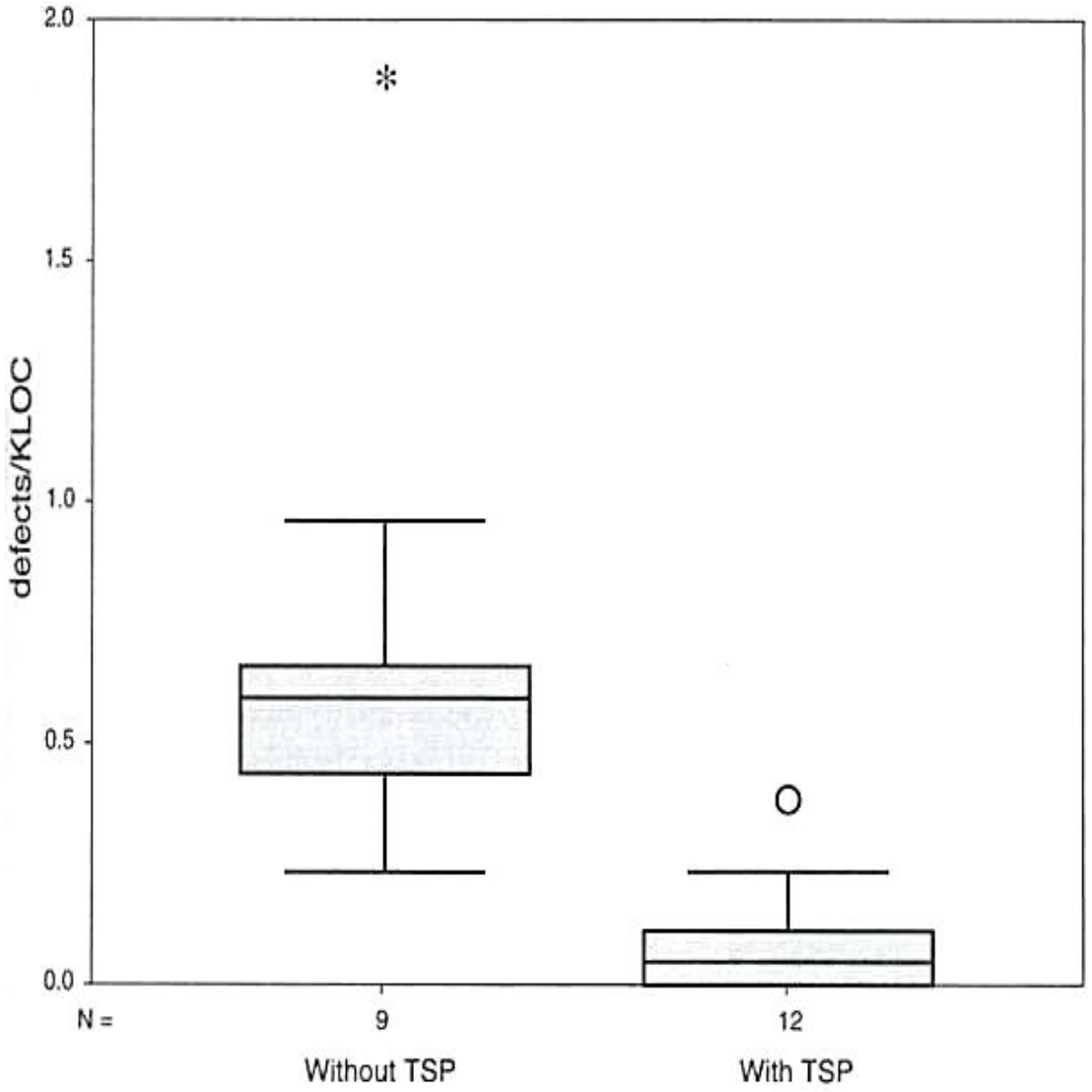
Consequently test changes from defect detection into correctness verification

Fagan (2005)

TSP Benefits

Dramatic reductions:

- Delivered defects
- Days per KLOC
- Schedule deviation
- Test defects/kloc
- Variation in results



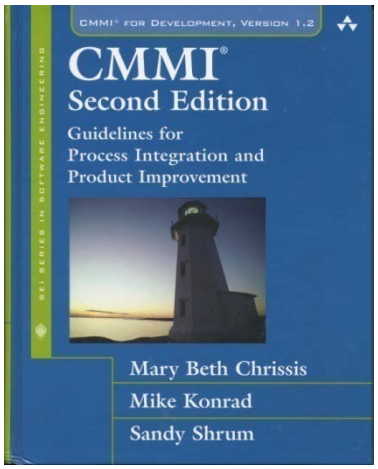
Humphrey (2006). *TSP: Leading a Development Team.*

Section 3
Measuring Software Improvement Programs

1) Improvement program methods
2) Maturity-based measurement
3) Improvement program results

CISQ
Consortium for IT Software Quality

CMMI Maturity Transitions



Level 5 Optimizing — **Innovation management**

Level 4 Quantitatively Managed — **Capability management**

Level 3 Defined — **Process management**

Level 2 Managed — **Project management**

Level 1 Initial — **Inconsistent management**

Innovative practices

Statistical practices

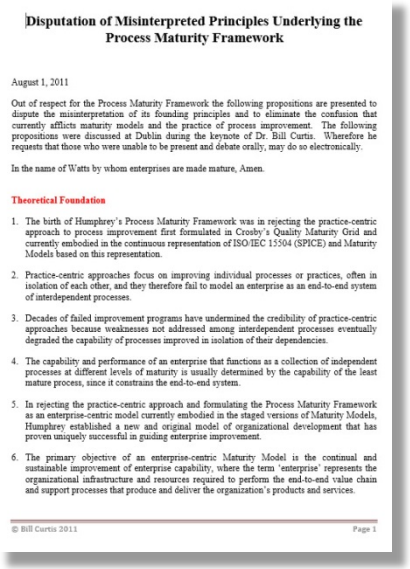
Standardized practices

Repeatable practices

Chrisis, Konrad, & Shrum (2005). *CMMI.*

CISQ
Consortium for IT Software Quality

Enhanced Maturity Framework



Disputation of Misinterpreted Principles Underlying the Process Maturity Framework

August 1, 2011

Out of respect for the Process Maturity Framework the following propositions are presented to dispute the misinterpretation of its founding principles and to eliminate the confusion that currently afflicts maturity models and the practice of process improvement. The following propositions were discussed at Dublin during the keynote of Dr. Bill Curtis. Wherefore he requests that those who were unable to be present and debate orally, may do so electronically.

In the name of Watts by whom enterprises are made mature, Amen.

Theoretical Foundation

1. The birth of Humphrey's Process Maturity Framework was in rejecting the practice-centric approach to process improvement first formulated in Crosby's Quality Maturity Grid and currently embodied in the continuous representation of ISO/IEC 15504 (SPICE) and Maturity Models based on this representation.
2. Practice-centric approaches focus on improving individual processes or practices, often in isolation of each other, and they therefore fail to model an enterprise as an end-to-end system of interdependent processes.
3. Decades of failed improvement programs have undermined the credibility of practice-centric approaches because weaknesses not addressed among interdependent processes eventually degraded the capability of processes improved in isolation of their dependencies.
4. The capability and performance of an enterprise that functions as a collection of independent processes at different levels of maturity is usually determined by the capability of the least mature process, since it constrains the end-to-end system.
5. In rejecting the practice-centric approach and formulating the Process Maturity Framework as an enterprise-centric model currently embodied in the staged versions of Maturity Models, Humphrey established a new and original model of organizational development that has proven uniquely successful in guiding enterprise improvement.
6. The primary objective of an enterprise-centric Maturity Model is the continual and sustainable improvement of enterprise capability, where the term 'enterprise' represents the organizational infrastructure and resources required to perform the end-to-end value chain and support processes that produce and deliver the organization's products and services.

© Bill Curtis 2011 Page 1

Level 1
Initial

Inconsistent management

Control baselines and commitments

Level 2
Stabilized

Work unit management

Standard integrated value chain

Level 3
Standardized

Organizational management

Optimize standardized process

Level 4
Optimized

Capability management

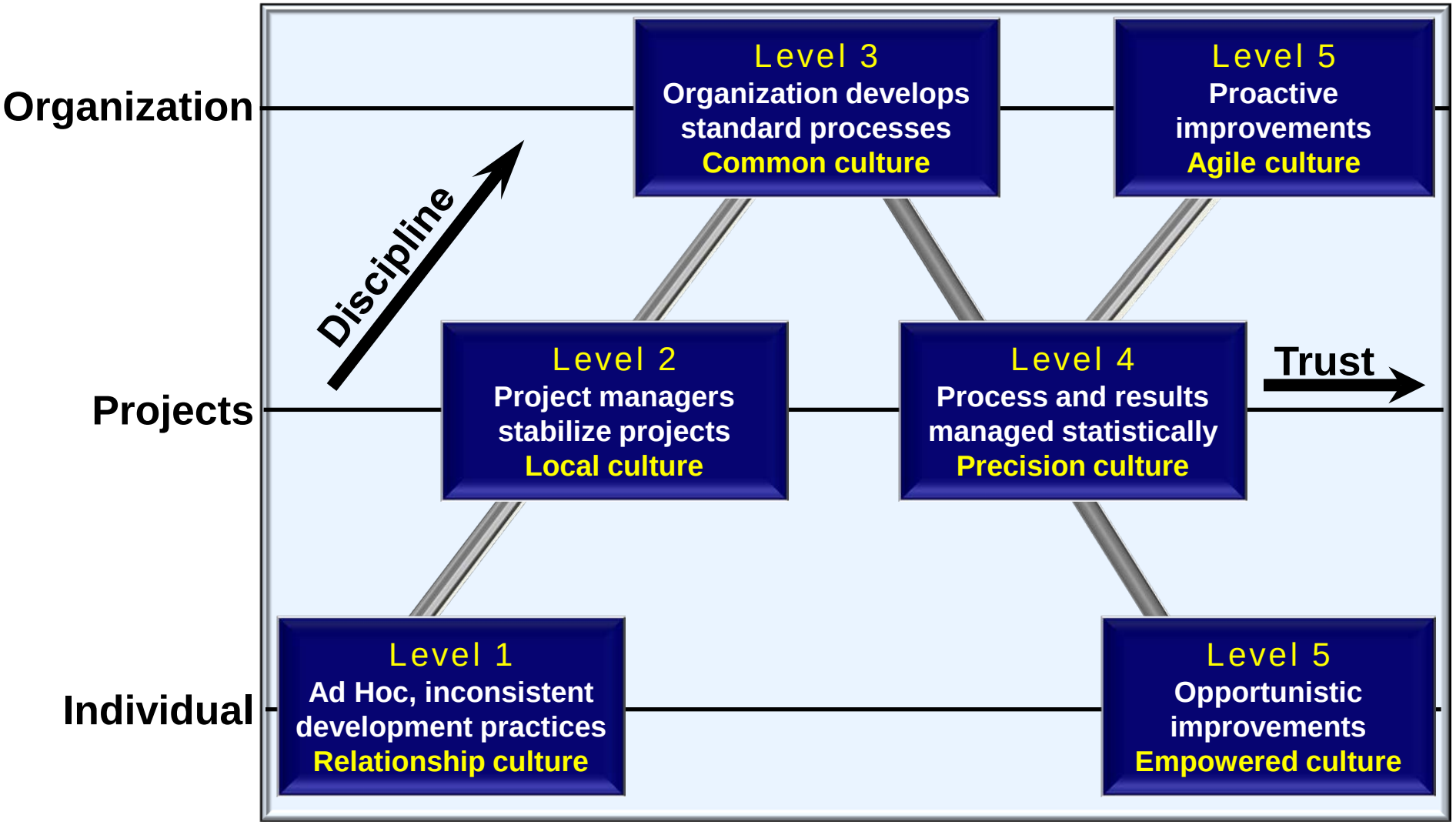
Innovate to new level of capability

Level 5
Innovating

Innovation management

CISQ
Consortium for IT Software Quality

Maturity Level Transformation



CISQ
Consortium for IT Software Quality

Measurement by Level

Measure-maturity mismatch slows improvement and creates dysfunction

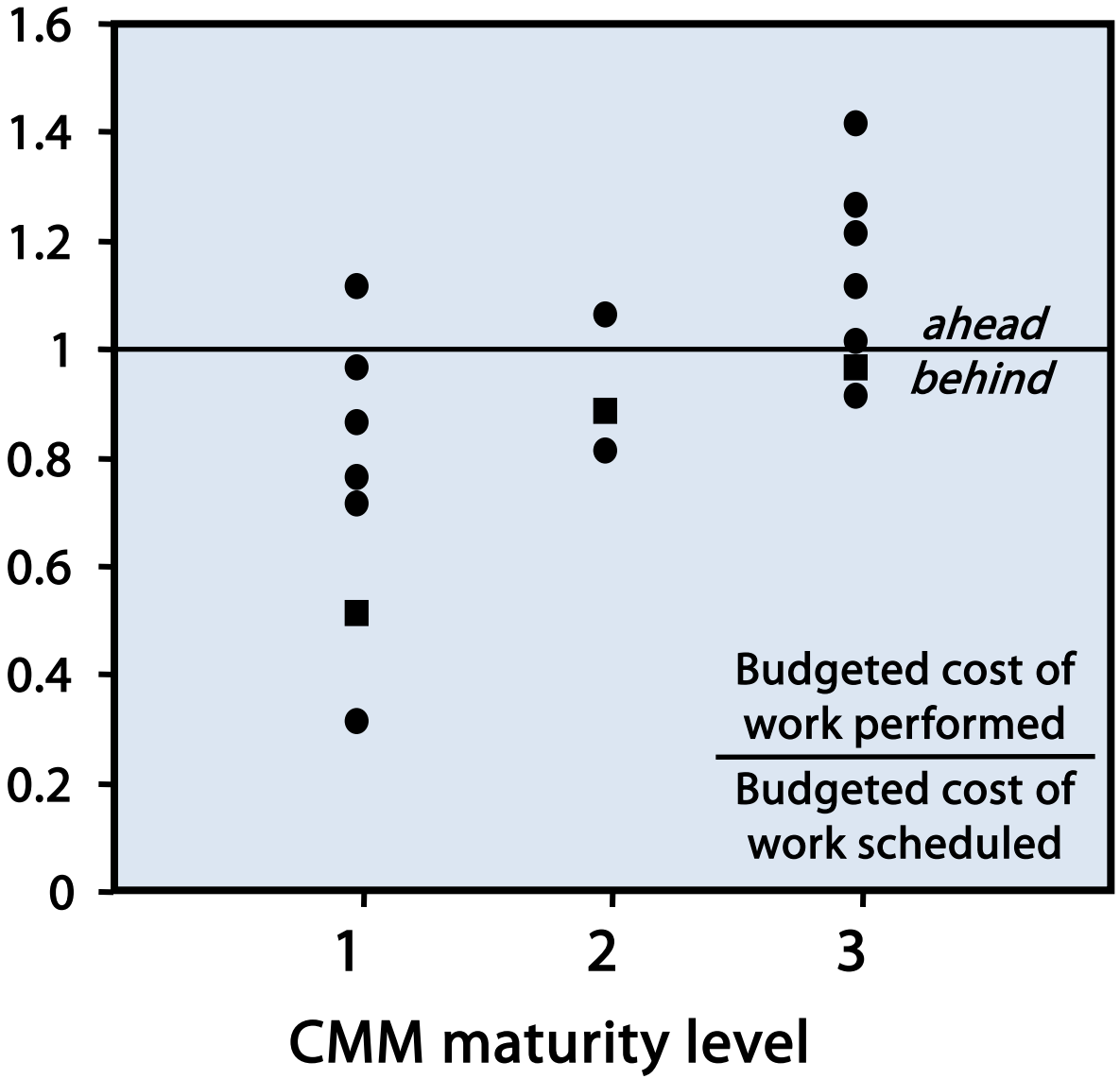
CISQ
Consortium for IT Software Quality

Average Improvement Results

Improvement benefit	Orgs	Annual median	Annual range
Productivity growth	4	35%	9% - 67%
Pre-test defect detection	3	22%	6% - 25%
Time to market	2	↓ 19%	15% - 23%
Field error reports	5	↓ 39%	10% - 94%
Return on investment	5	5.0:1	4:1 - 8.8:1

Schedule Performance

Schedule performance index

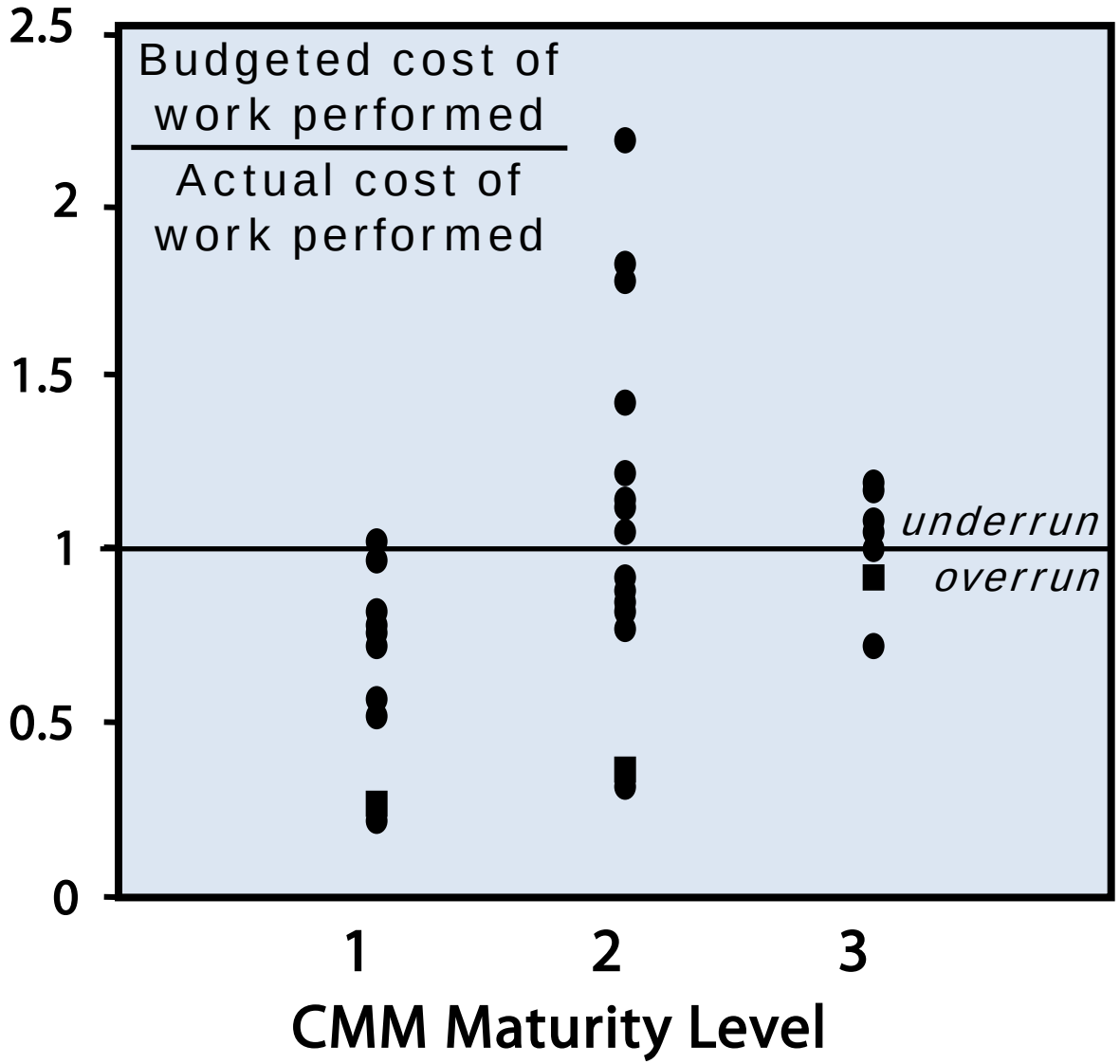


Air Force study of contractors

Flowe & Thordahl (1994)

Cost Performance

Cost performance index

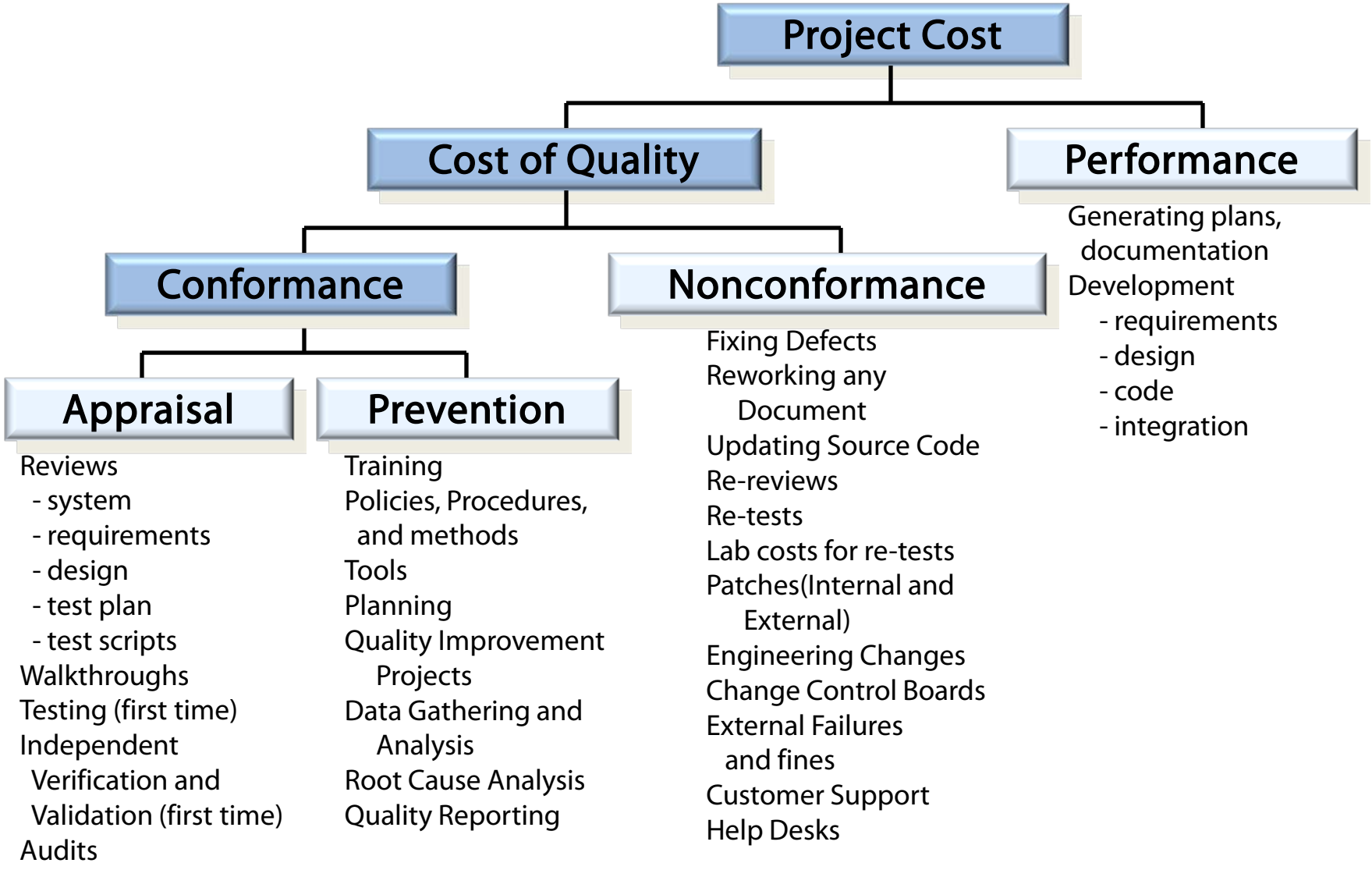


Air Force study of contractors

Flowe & Thordahl (1994)

CISQ
Consortium for IT Software Quality

Crosby's Cost of Quality Model



Crosby (1979). *Quality Is Free*.

CISQ
Consortium for IT Software Quality

Raytheon's Cost of Quality

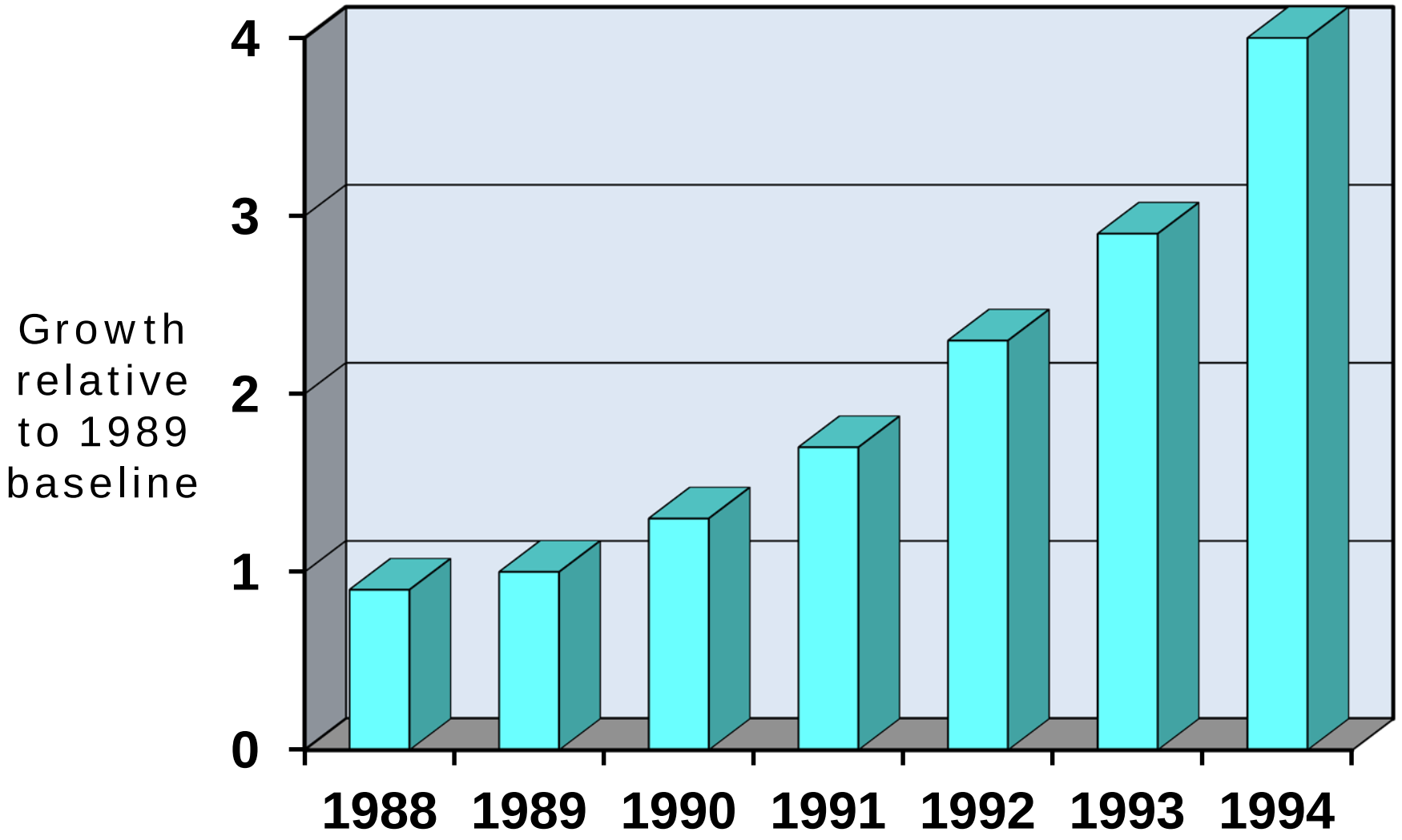
- Performance — ***cost of building it right first time***
- Nonconformance — ***cost of rework***
- Appraisal — ***cost of testing***
- Prevention — ***cost of preventing nonconformance***

Year	Level	Perform	Nonconf.	Appraise	Prevent
1988	1	34%	41%	15%	7%
1990	2	55%	18%	15%	12%
1992	3	66%	11%	{ 23%	}
1994	4	76%	6%	{ 18%	}

Dion (1993), Haley (1995)

CISQ
Consortium for IT Software Quality

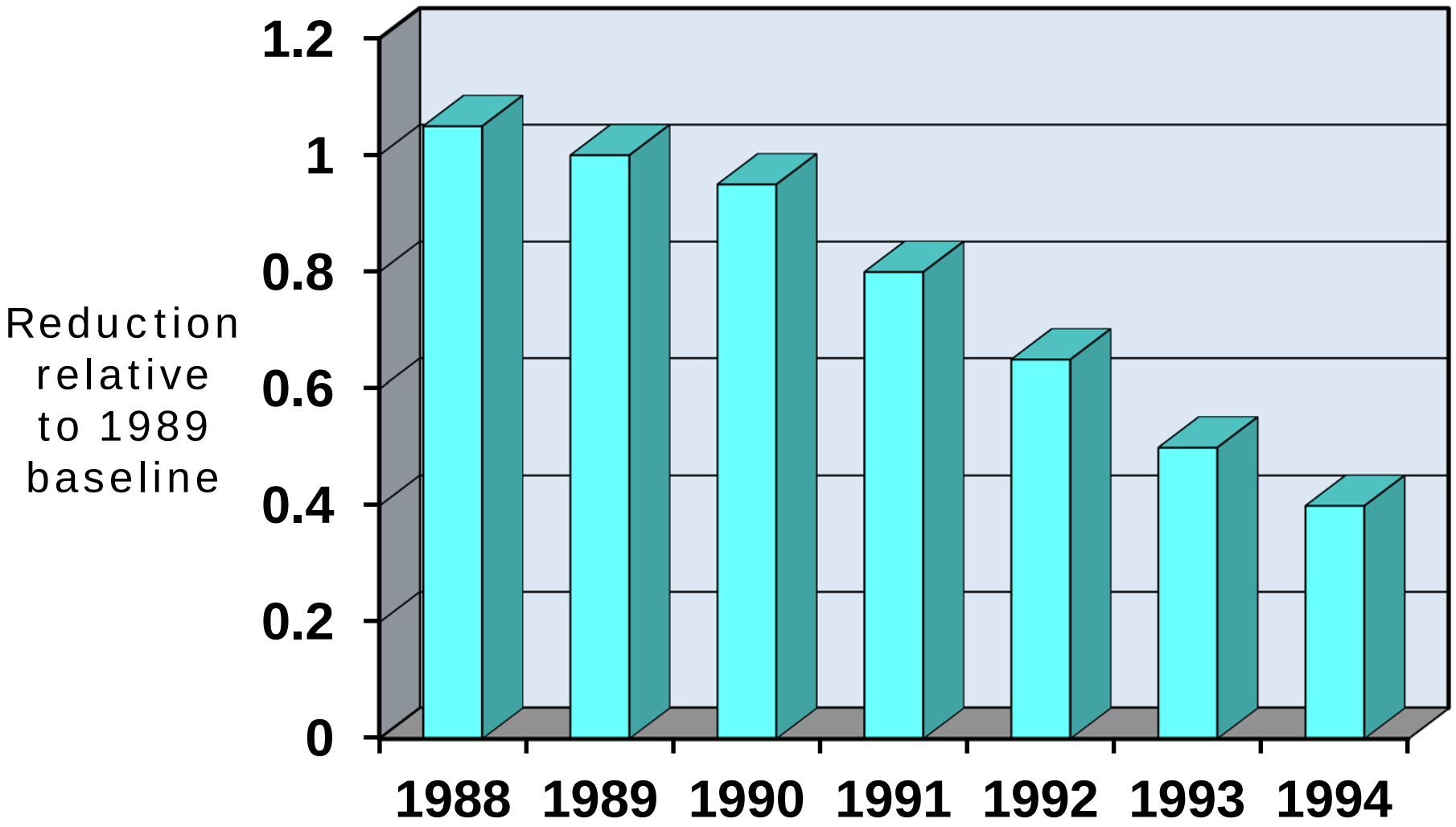
Raytheon's Productivity



Lydon (1995)

CISQ
Consortium for IT Software Quality

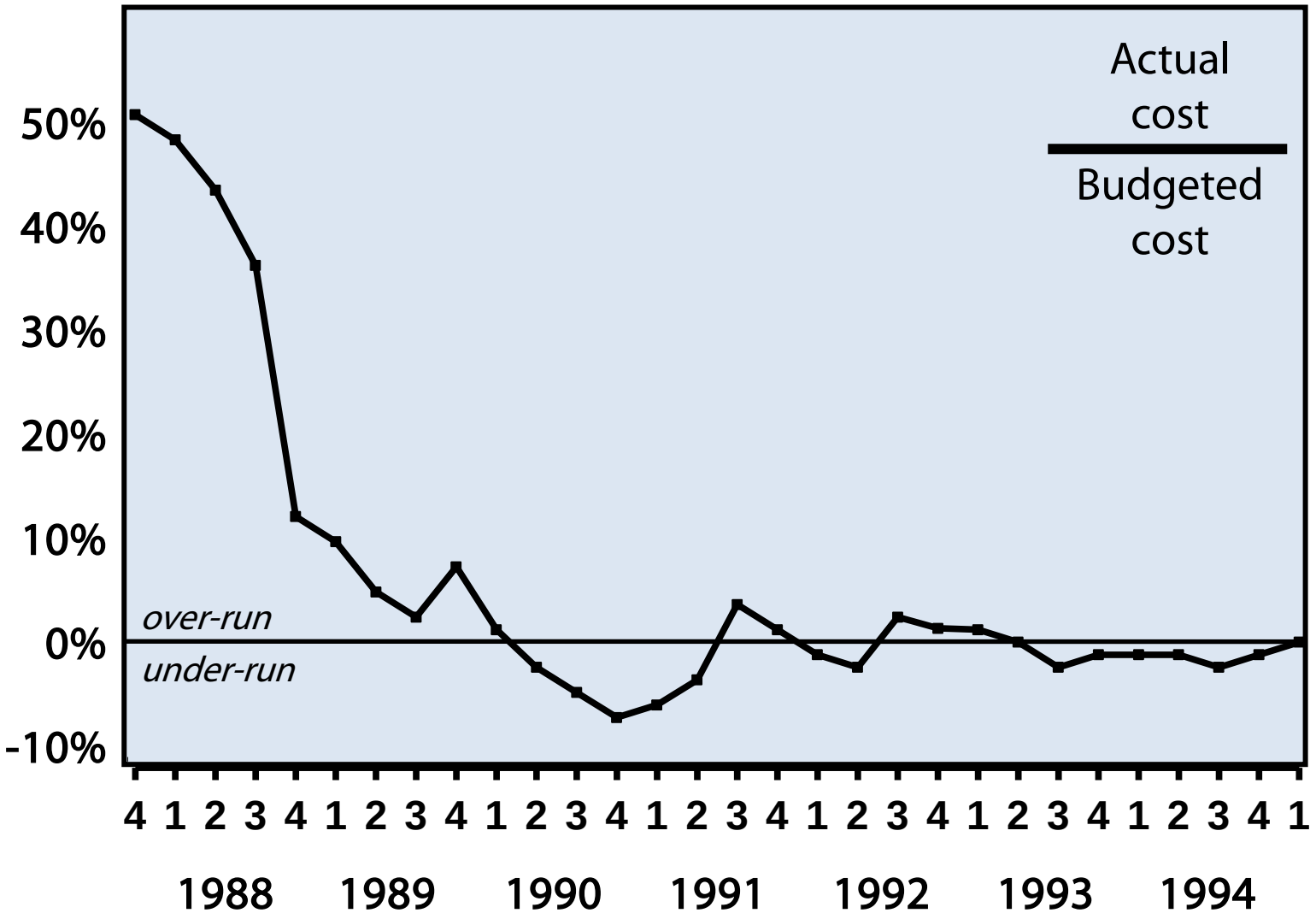
Raytheon's Cost Reduction



Lydon (1995)

CISQ
Consortium for IT Software Quality

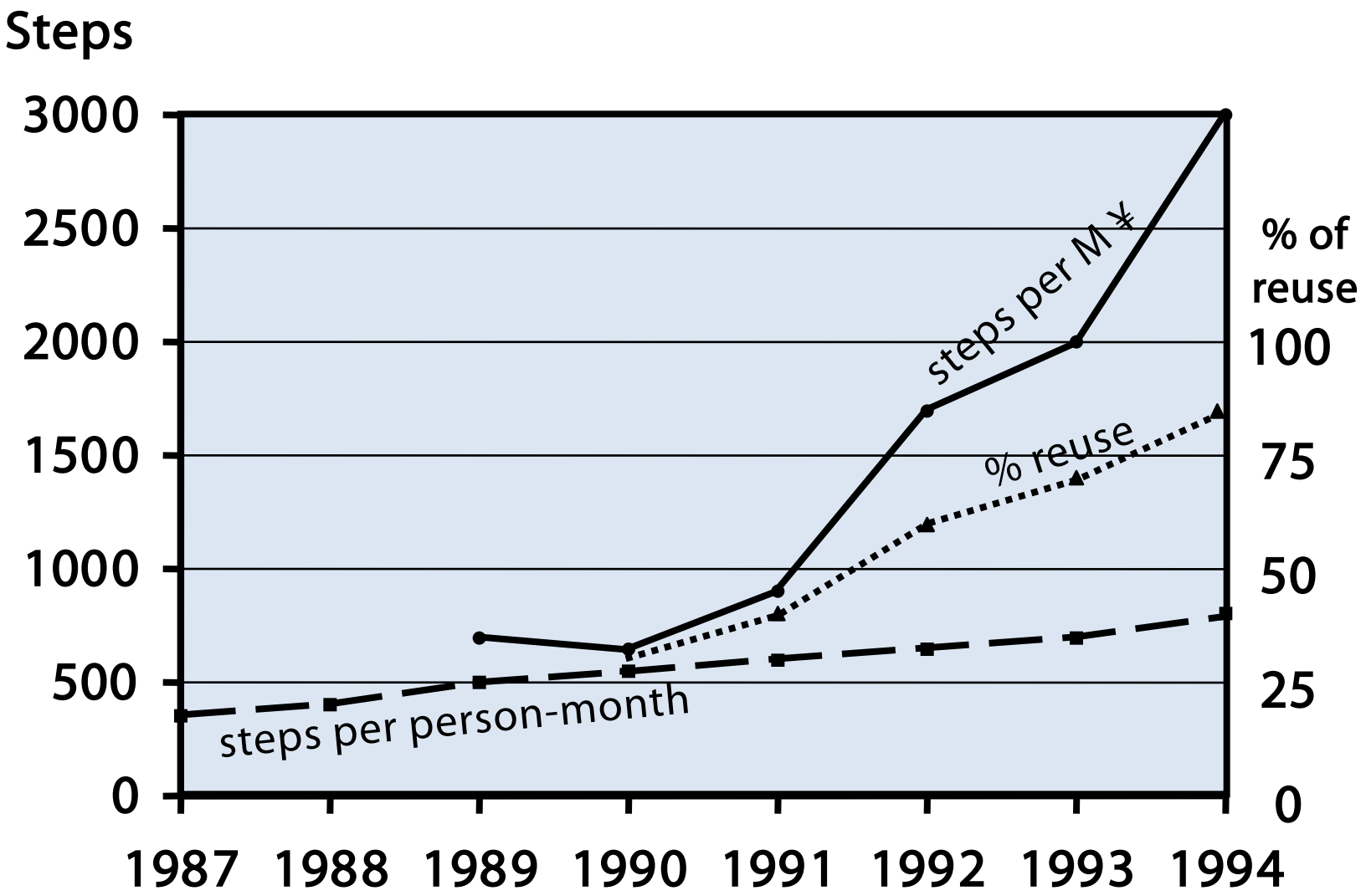
Raytheon's Cost Predictability



Haley (1995)

CISQ
Consortium for IT Software Quality

OMRON's Reuse gains



Sakamoto, et al. (1996)

Section 4
Measuring Quality and Technical Debt

1) Structural quality measurement
2) Technical debt
3) Consortium for IT Software Quality (CISQ)

CISQ
Consortium for IT Software Quality

What about the Product?

CMMI's primary assumption:

> **"The quality of a software system is governed by quality of the process used to develop it."**
>
> **- Watts Humphrey**

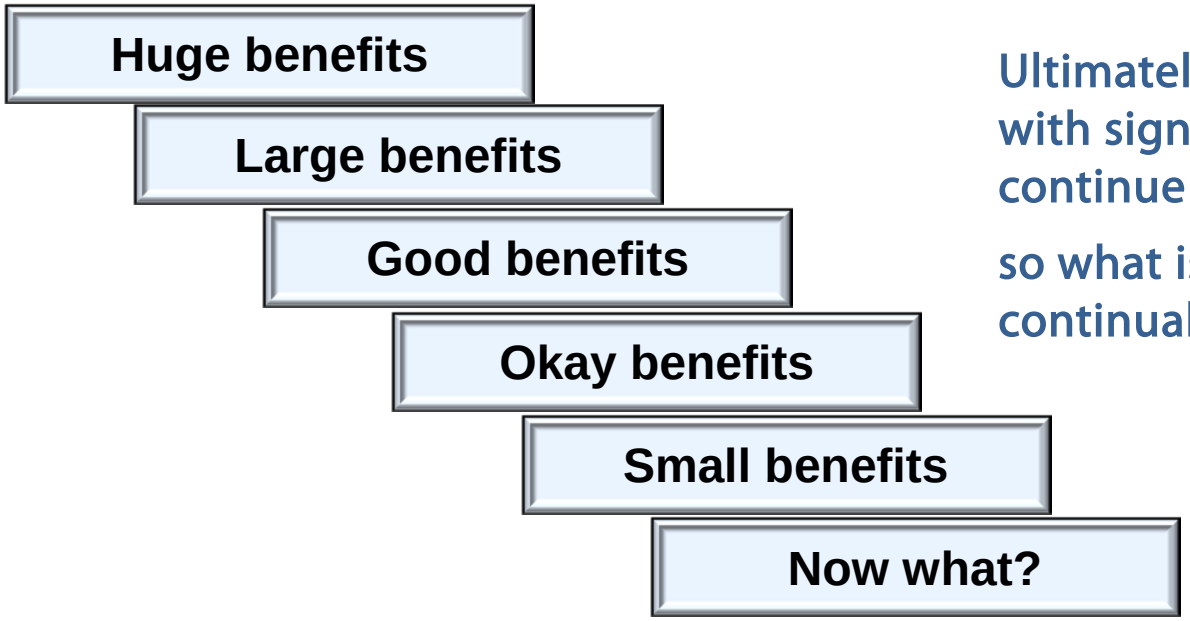
CMMI's product quality problem:

1. **Assessments do not verify product quality**
2. **Compliance focus undermines CMMI's primary assumption**
3. **Product quality focus at Level 4 in CMM was lost in quantitative process control in CMMI**
4. **The assumptions underlying control charts are violated by software data**

CISQ
Consortium for IT Software Quality

Six Sigma's Challenge

Six Sigma projects must have significant benefits



Ultimately we run out of projects with significant enough benefits to continue the program….

so what is the solution that allows continual improvement?

Process focus – **process improvement – *Six Sigma***

Product focus – **product improvement – *Design for 6σ***

Product focus supplements product focus to unlock even more value from software

Testing is Not Enough


review articles

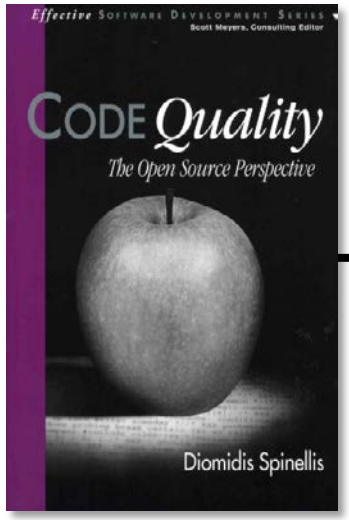
Who could fault an approach that offers greater credibility at reduced cost?

BY DANIEL JACKSON

A Direct Path to Dependable Software

"As higher levels of assurance are demanded…testing cannot deliver the level of confidence required at a reasonable cost."

"The correctness of the code is rarely the weakest link."



"…a failure to satisfy a non-functional requirement can be critical, even catastrophic…non-functional requirements are sometimes difficult to verify. We cannot write a test case to verify a system's reliability…The ability to associate code to non-functional properties can be a powerful weapon in a software engineer's arsenal."

Jackson, D. (2009), Spinellis, D. (2006)

CAST's Application Intelligence Platform

CISQ
Consortium for IT Software Quality

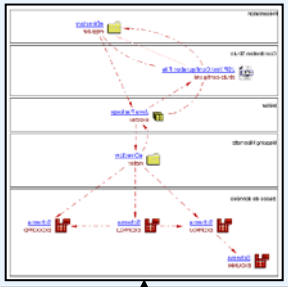
Language Parsers

- Oracle PL/SQL
- Sybase T-SQL
- SQL Server T-SQL
- IBM SQL/PSM
- C, C++, C#
- Pro C
- Cobol
- CICS
- Visual Basic
- VB.Net
- ASP.Net
- Java, J2EE
- JSP
- XML
- HTML
- Javascript
- VBScript
- PHP
- PowerBuilder
- Oracle Forms
- PeopleSoft
- SAP ABAP, Netweaver
- Tibco
- Business Objects
- Universal Analyzer for other languages

Application Analysis

Evaluation of 1200+ coding & architectural rules

Application meta-data



Detected Violations

- Expensive operation in loop
- Static vs. pooled connections
- Complex query on big table
- Large indices on big table

- Empty CATCH block
- Uncontrolled data access
- Poor memory management
- Opened resource not closed

- SQL injection
- Cross-site scripting
- Buffer overflow
- Uncontrolled format string

- Unstructured code
- Misuse of inheritance
- Lack of comments
- Violated naming convention

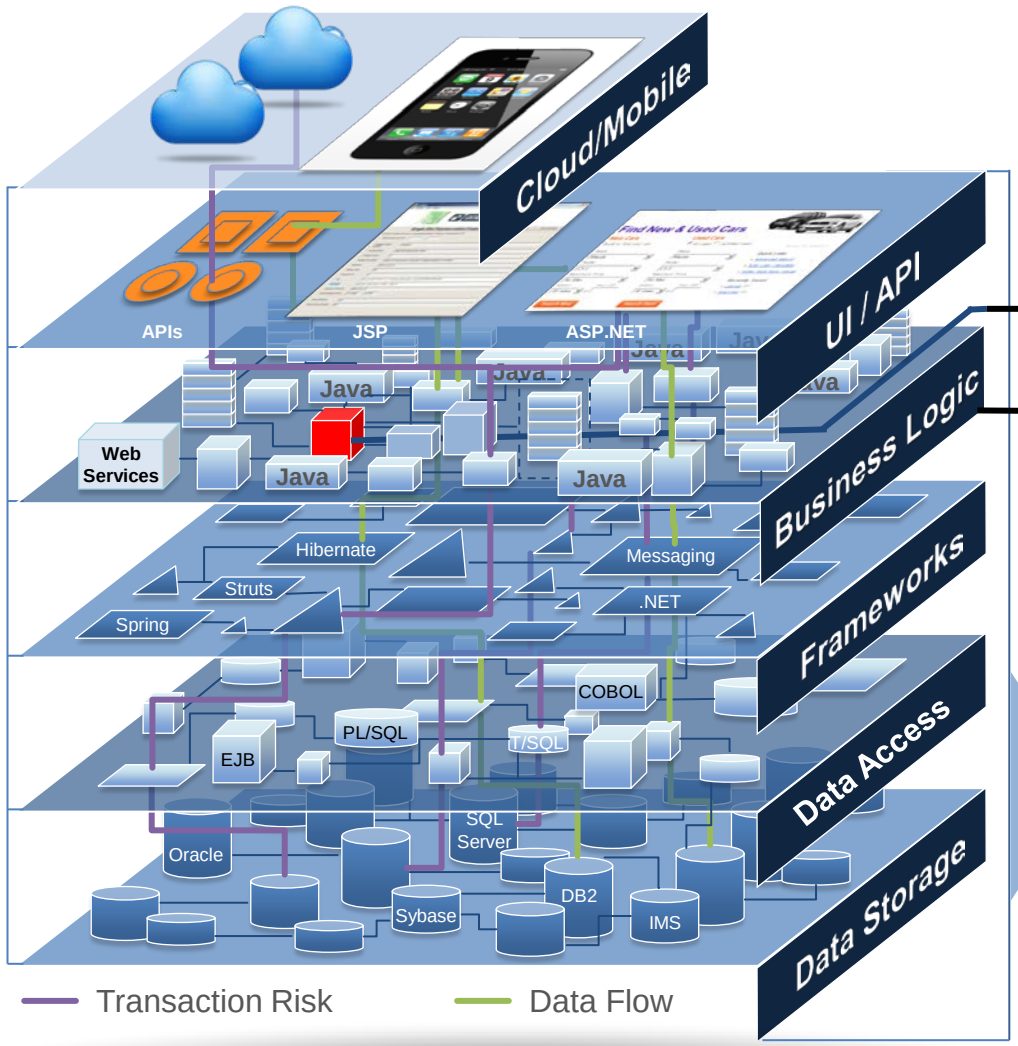
- Highly coupled component
- Duplicated code
- Index modified in loop
- High cyclomatic complexity

Quality Measurements

- Performance
- Robustness
- Security
- Transferability
- Changeability

CISQ
Consortium for IT Software Quality

Modern Apps are a Technology Stack



1 Unit Level

- Code style & layout
- Expression complexity
- Code documentation
- Class or program design
- Basic coding standards
- Developer level

2 Technology Level

- Single language/technology layer
- Intra-technology architecture
- Intra-layer dependencies
- Inter-program invocation
- Security vulnerabilities
- Development team level

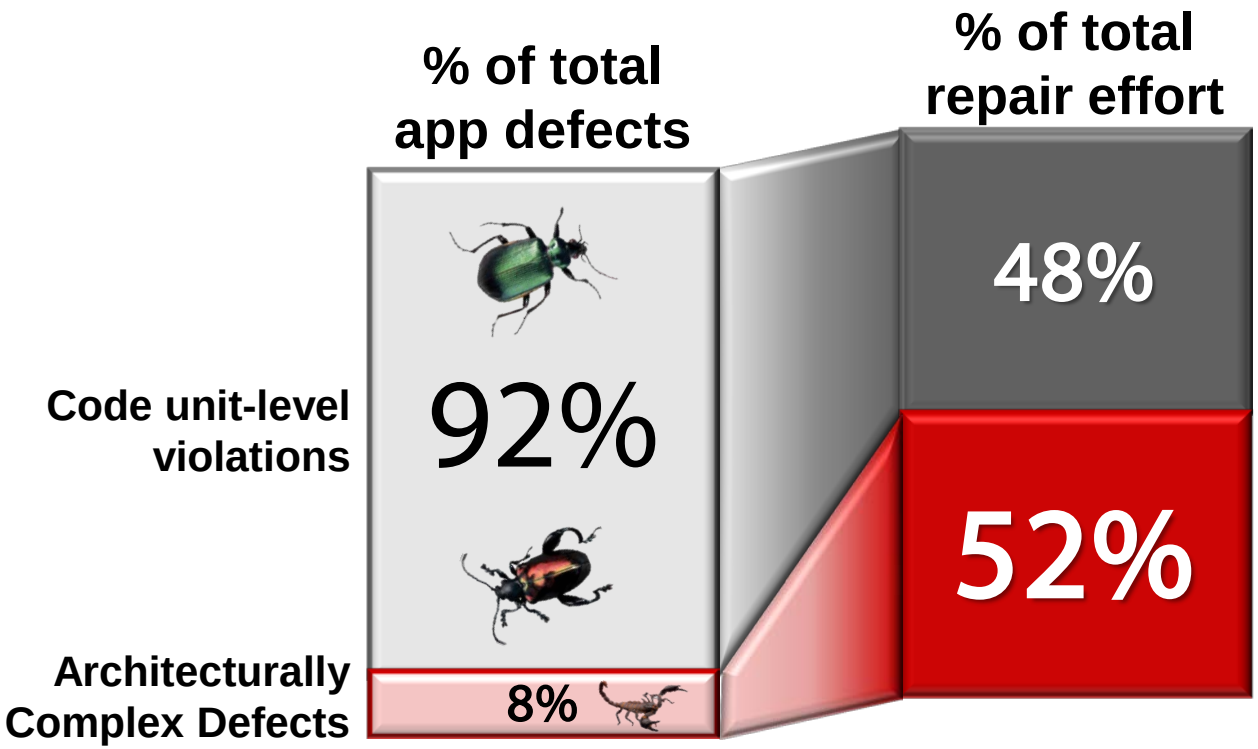
3 System Level

- Integration quality
- Architectural compliance
- Risk propagation
- Application security
- Resiliency checks
- Transaction integrity
- Function point
- Effort estimation
- Data access control
- SDK versioning
- Calibration across technologies
- IT organization level

CISQ
Consortium for IT Software Quality

Architecturally Complex Defects

Architecturally Complex Defect	A structural flaw involving interactions among multiple components that reside in different application layers

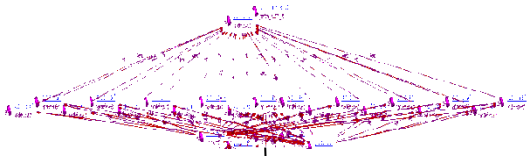


Most architecturally complex defects involve an architectural hotspot – a badly-built component that should be replaced

Propagated Risk

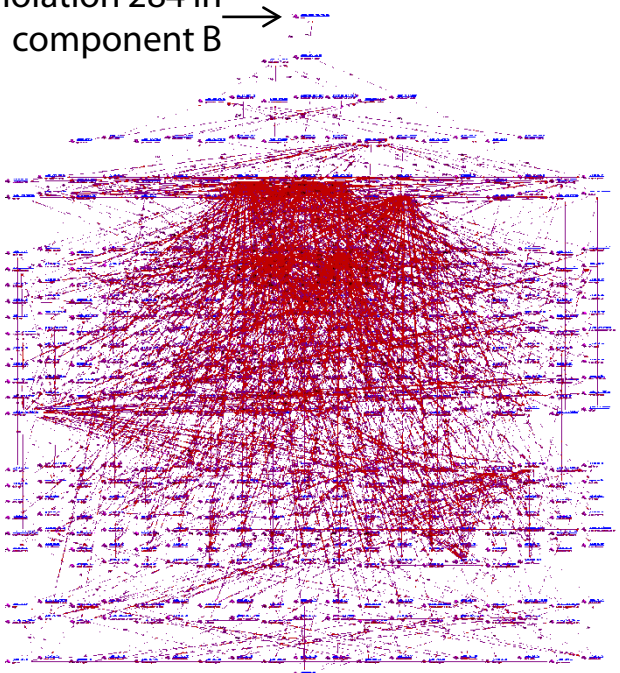
Propagated Risk Index	A ranking of the risk created by each violation based on the its severity and the breadth of its impact in the system

Violation 342 in component A →



The impact of violation 342 in Component A is only narrowly propagated in the system – it presents low risk

Violation 284 in component B →



The impact of violation 284 in Component B is very widely propagated throughout the system – it presents very high risk

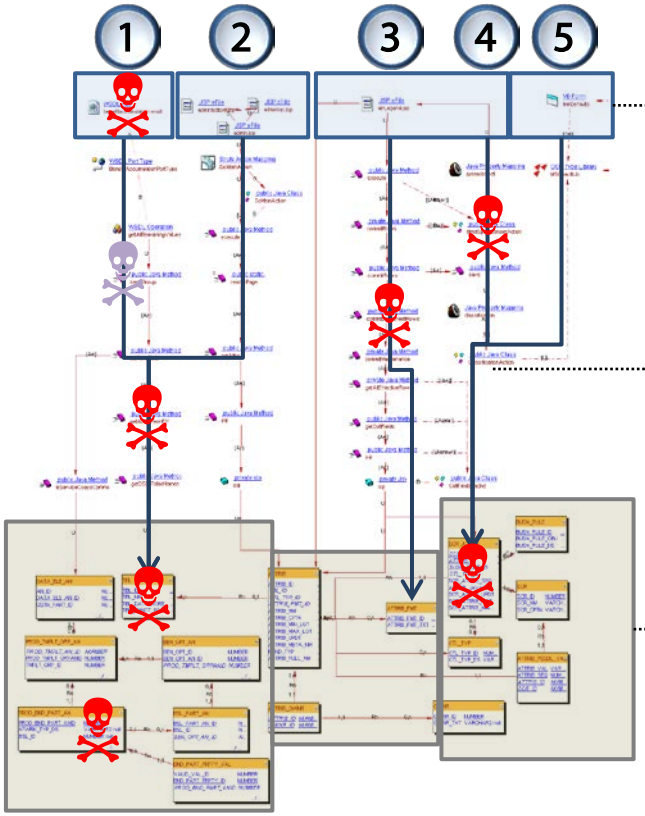
Remediating violation 284 in Component B will have the greater impact on reducing risk and cost

CISQ
Consortium for IT Software Quality

Transaction Risk

Transaction Risk Index	A ranking of transactions based on the risk created by the number and severity of violations along the transaction path

- Customers care most about the dependability of their transactions
- The risk of a transaction is determined by the number and severity of violations along its path
- Transactions can be ranked by risk to prioritize their remediation
- The Transaction Risk Index can be further tuned using the operational frequency of each transaction



Transaction 1 poses greatest risk

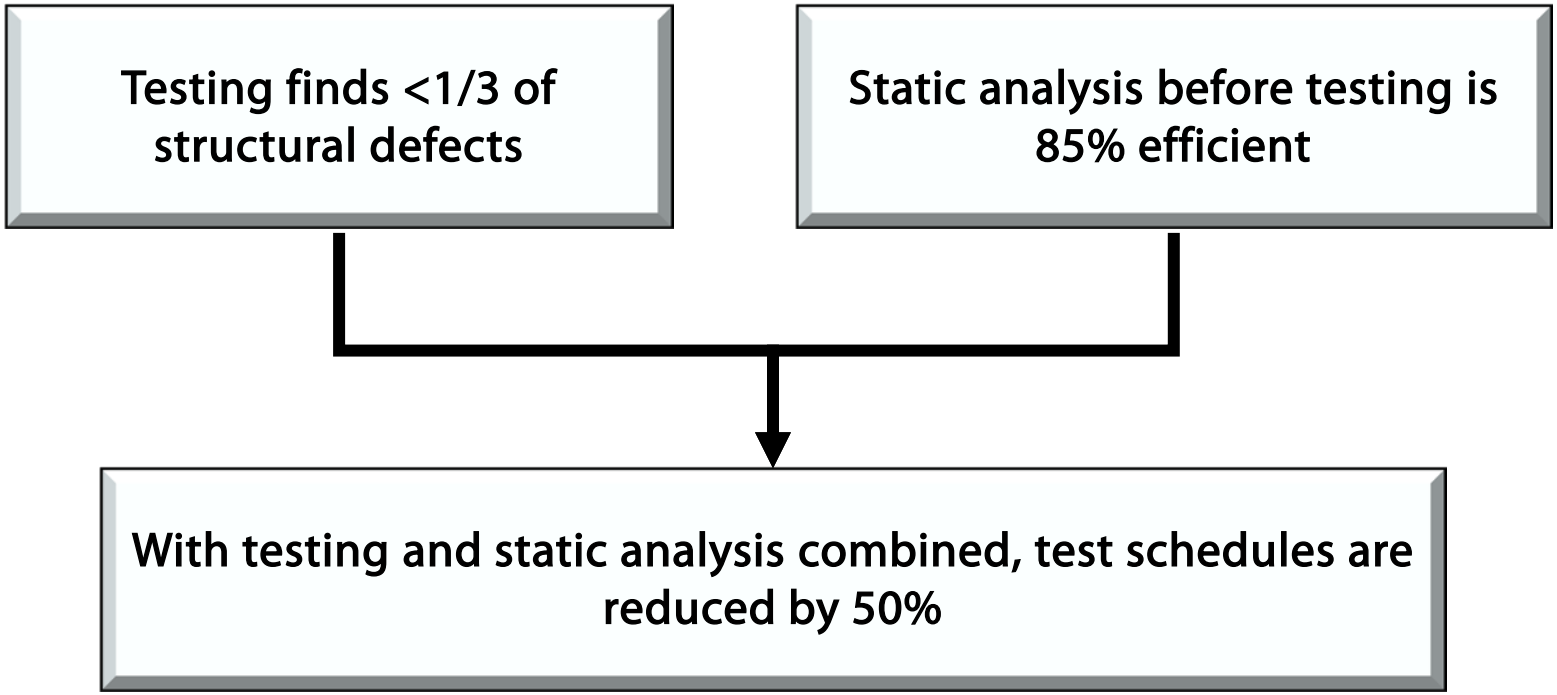
CISQ
Consortium for IT Software Quality

Risk-Based Prioritization

Violation	Health Factor	Severity	Propagated Risk Index	In a high risk transaction?	Frequency of path execution
#148	Security	9			
#371	Robustness	9			
#062	Performance	8			
#284	Robustness	7			
#016	Changeability	7			
#241	Security	5			
#173	Transferability	4			
#359	Transferability	3			

CISQ
Consortium for IT Software Quality

Static Analysis in US DoD

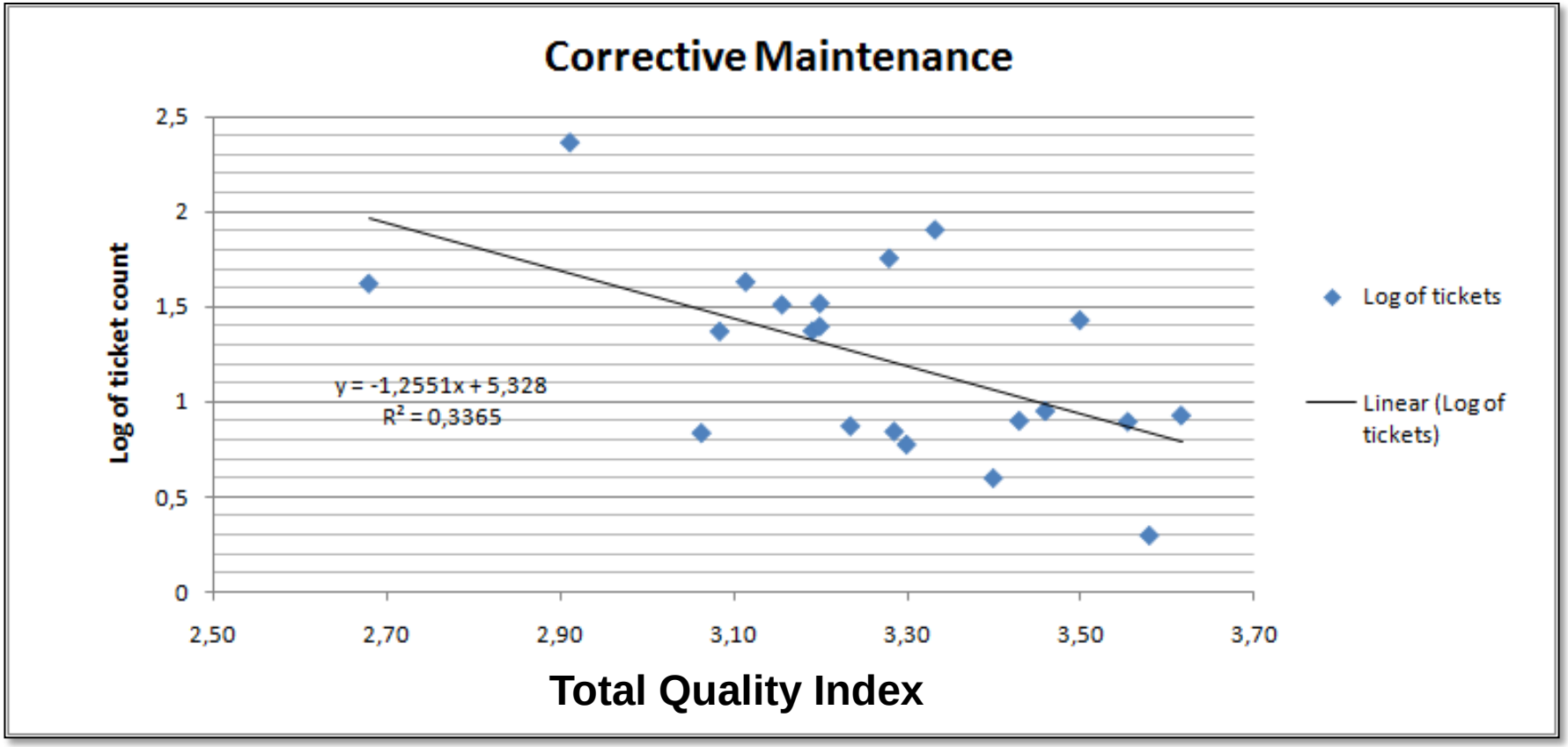


John Keane (2013).

CISQ
Consortium for IT Software Quality

Reducing Operational Incidents & Costs

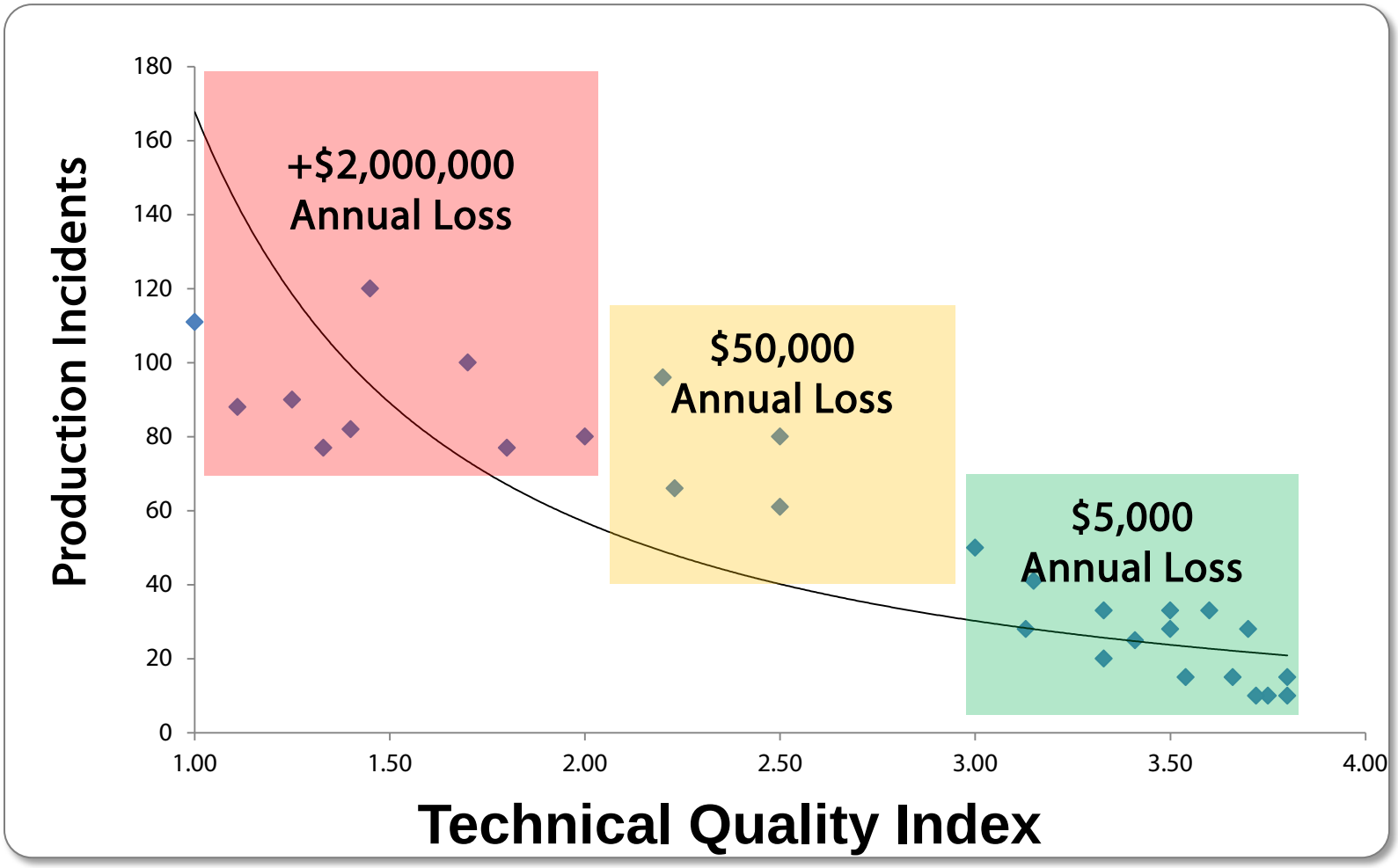
Study of structural quality measures and maintenance effort across 20 customers in a large global system integrator



TQI increase of .24 decreased corrective maintenance effort by 50%

CISQ
Consortium for IT Software Quality

Reducing Operational Losses



Large international investment bank
Business critical applications

CISQ
Consortium for IT Software Quality

Factors Affecting Structural Quality

Percentage of variation in structural quality scores accounted for by various factors

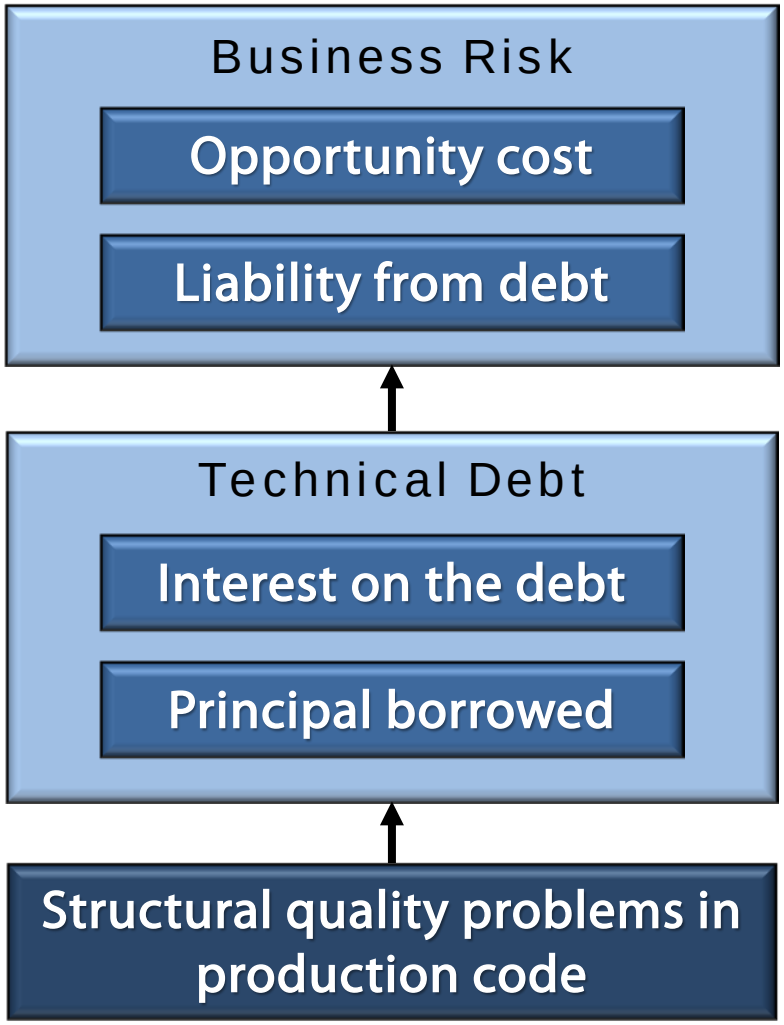
Factor	# apps	Robust	Security	Perform	Change	Transfer
Technology	1606	20%	3%	16%	26%	7%

2017 CRASH Report
Request from: info@castsoftware.com

CISQ
Consortium for IT Software Quality

The Technical Debt Metaphor

Technical Debt — the future cost of fixing defects remaining in code at release, a component of the cost of ownership



Opportunity cost—benefits that could have been achieved had resources been put on new capability rather than retiring technical debt

Liability—business costs related to outages, breaches, corrupted data, etc.

Interest—continuing IT costs attributable to the violations causing technical debt, i.e, higher maintenance costs, greater resource usage, etc.

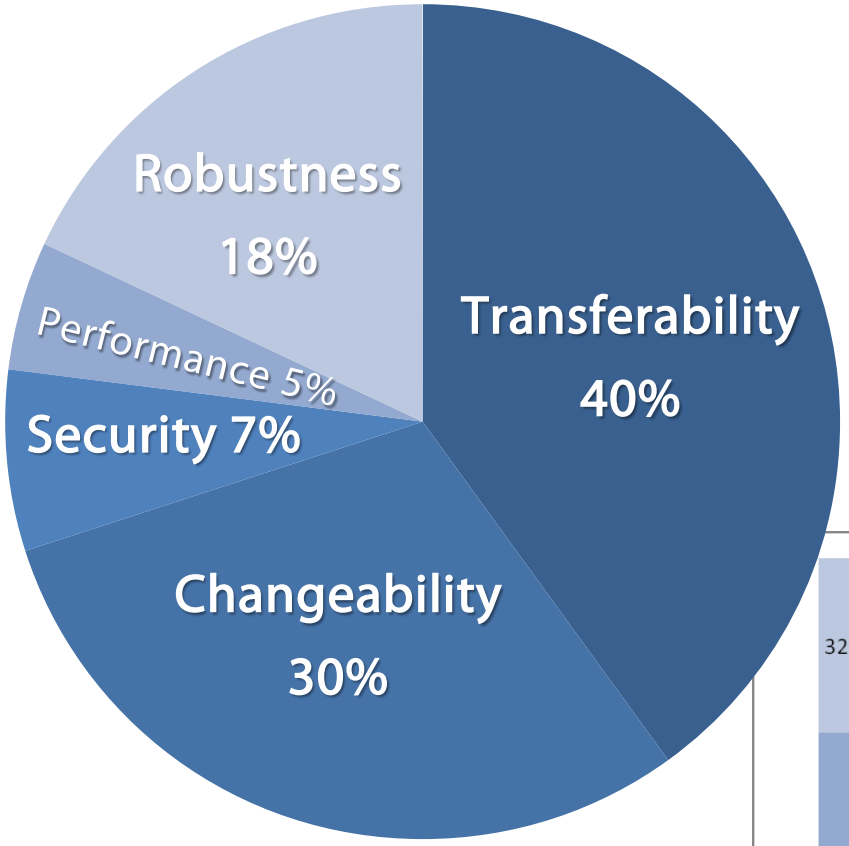
Principal—cost of fixing problems remaining in the code after release that must be remediated

Today's talk focuses on the principal

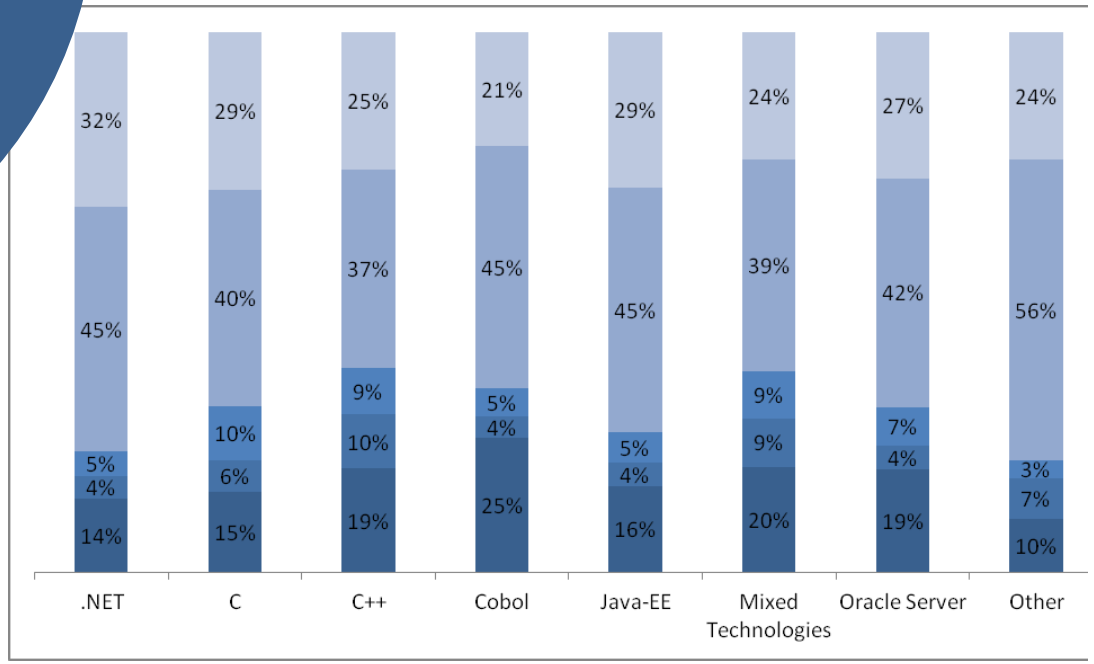
Curtis, et al. (2012). *IEEE Software*.

CISQ
Consortium for IT Software Quality

Tech. Debt by Quality Factor



- 70% of Technical Debt is in IT Cost
  (Transferability, Changeability)
- 30% of Technical Debt is in Business Risk
  (Robustness, Performance, Security)
- Health Factor proportions are mostly consistent across technologies



Curtis, et al. (2012). *IEEE Software*.

CISQ
Consortium for IT Software Quality

Join CISQ ! www.it-cisq.org

References

References — 1

Anderson, D.J. (2010). *Kanban*. Sequim, WA: Blue Hole Press.

Basili, V. Trendowicz, A., Kowalczyk, M., Heidrich, J., Seaman, C., Munch, J., & Rombach, D. (2014). *Aligning Organizations through Measurement*. Heidelberg: Springer.

Boehm, B. (1987). Increasing software productivity. *IEEE Computer*, 20 (9), 43-57.

Boehm, B. et al. (2000). *Software Cost Estimation with COCOMO II*. Prentice-Hall.

Chrissis, M.B., Konrad, M., & Shrum, S. (2011). *CMMI Third Edition: Guidelines for Process Integration and Product Improvement*. Reading, MA: Addison Wesley.

Cohn, M. (2006). *Agile Estimating and Planning*. Upper Saddle River, NJ: Prentice-Hall.

Crosby, P. (1979). Quality Is Free. New York: McGraw-Hill.

Curtis, B., Sapiddi, J., & Szynkarski, A. (2012). Estimating the principle of an application's technical debt. *IEEE Software, 29* (6), 34-42.

Dion, R. (1993). Process improvement and the corporate balance sheet. *IEEE Software*, 10 (4), 28-35.

Duncker, R. (1992). *Presentation at the 25th Annual Conference of the Singapore Computer Society*. Singapore: November, 1992.

Dyne, K. (1999). ESSI: Benefits from Continuous Software Improvements. In *Proceedings of ESEPG'99*. Milton Keyes, UK: ESPI Foundation.

Flowe, R.M. & Thordahl, J.B. (1994). *A Correlational Study of the SEI's Capability Maturity Model and Software Development Performance in DoD Contracts (AFIT/GSS/LAR/94D-2).* Dayton, OH: Air Force Institute of Technology, Wright Patterson Air Force Base.

References — 2

Haley, T., Ireland, B., Wojtaszek, E., Nash, D., & Dion, R. (1995). *Raytheon Electronic Systems Experience in Software Process Improvement: (Tech. Rep. CMU/SEI-95-TR-017).).* Pittsburgh: Software Engineering Institute, Carnegie Mellon University.

Ebert, C. & Dumke, R. (2007). *Software Measurement*. Berlin: Springer-Verlag.

Fagan, E. & Davis, N. (2005). TSP / PSP at Intuit. *Proceedings of the TSP 2005*. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University.

Fenton, N. (2006). New directions for Software Metrics. *CIO Symposium on Software Best Practices*.

Garmus, D. & Herron, D. (2001). *Function Point Analysis*. Boston: Addison-Wesley.

George, E. (2005). Project management with TSP. *Proceedings of the TSP 2005*. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University.

Herbsleb, J, Carleton, A., Rozum, J., Siegel, J., & Zubrow, D. (1994). *Benefits of CMM-Based Software Process Improvement: Initial Results (Tech. Rep. CMU/SEI-94-TR-13).* Pittsburgh: Software Engineering Institute, Carnegie Mellon University.

Humphrey, W. (1997). *Introduction to the Personal Software Process*. Boston: Addison Wesley.

Humphrey, W. (2000). *Introduction to the Team Software Process*. Boston: Addison Wesley.

Humphrey, W. (2006). *TSP: Leading a Development Team*. Boston: Addison Wesley.

Jackson, D. (2009). A direct path to dependable software. *Communications of the ACM, 52* (4).

Kan, S.H. (1995). *Metrics and Models in Software Quality Engineering*. Reading, MA: Addison-Wesley.

Keeni, B. (2000). *IEEE Software, 16* (4).

References — 2

Larman, C. (2004). *Agile and Iterative Development*. Boston: Addison Wesley.

Lydon, T. (1995). Productivity drivers: Process and capital. In *Proceedings of the 1995 SEPG Conference*. Pittsburgh: Software Engineering Institute, Carnegie Mellon University.

McGarry, J. et al. (2002). *Practical Software Measurement*. Reading, MA: Addison Wesley.

Paulk, M., Weber, C., Curtis, B., & Chrissis, M. (1995). *The Capability Maturity Model: Guidelines for Improving the Software Process*. Reading, MA: Addison-Wesley.

Peterman, W. (2000). *IEEE Software*, July/August issue.

Poppendieck, M. & Poppendieck, T. (2007). *Implementing Lean Software Deve*lopment. Boston: Addison-Wesley.

Sakamoto, K., Kishida, K., & Nakakoji, K. (1996). Cultural adaptation of the CMM. In Fuggetta, A. & Wolf, A. (Eds.), *Software Process*. Chichester, UK: Wiley, 137-154.

Sinha, M. (2006). Performance models – Enabling organizations prediction capability. *Proceedings of SEPG'06*. Pittsburgh: Software Engineering Institute, Carnegie Mellon University.

Schwaber, K. & Beedle, M. (2002). *Agile Software Development Using Scrum*.

Spinellis, D. (2006). *Code Quality.* Addison-Wesley

Van Epps, S. & Marshall, R. (2012). Using TSP at the end of the development cycle. *Proceedings of TSP 2012*. Pittsburgh: Software Engineering Institute, Carnegie Mellon University.

Vu. J.D. (1996). Software process improvement: A business case. In *Proceedings of the European SEPG Conference*. Milton Keynes, UK: European Software Process Improvement Foundation