Latent Defect Estimation – Maturing Beyond Defect Removal using Capture-Recapture Method

Software Assurance Workshop

Security-Enhanced Quality Assurance, Testing and Project Management

September 9th, 2008 QAAM - Baltimore, MD

Joe Schofield

Sandia National Laboratories

Albuquerque, N. M.

505 844-7977

jrschof@sandia.gov

Latent Defect Estimation – Maturing Beyond Defect Removal using Capture-Recapture Method

Joseph R. Schofield

Sandia National Laboratories

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company,
for the United States Department of Energy under contract DE-AC04-94AL85000.

About Sandia National Laboratories

Since 1949, Sandia National Laboratories has developed science-based technologies that support our national security. Today, the nearly 300 million Americans depend on Sandia's technology solutions to solve national and global threats to peace and freedom.

Sandia is a government-owned contractor operated (GOCO) facility. Sandia Corporation, a Lockheed Martin company, manages Sandia for the U.S. Department of Energy's National Nuclear Security Administration.

Abstract (an abbreviated summary of any in-depth analysis of a particular subject or discipline) *wikipedia*

- **Statistical sampling techniques for populations in biology can be easily applied to peer reviews and inspections to estimate latent defects in (software) products. In turn, these values can be used to quantify the quality of the process and to establish thresholds for repeating review and testing practices.**

- **Fifth graders have demonstrated competence in using Capture Recapture Method after a short introduction. "Participants" in this session will get hands-on experience in using CRM enabling them to help target effective defect-removal processes in their organizations. This approach can be used to support measurement-related CMMI® ML 2, 3, and 4 practices.**

What's the point?

This presentation deals with three challenges:

- our undiminished ability to generate product defects
- our deceptive reliance on testing to eliminate defects
- our inability to statistically predict undiscovered defects still embedded in our software

And history indicates:

- Software defects – still plenty abundant
- Software and product quality – still plenty to talk about
- Inspections / Peer Reviews – still underutilized
- Asking the tough questions – still plenty of non-answers
- Capture Recapture Method – still plenty (defects) to find

Beyond Scope for Today:

- Major versus minor defect classifications (and holy wars)
- Peer reviews versus inspections (and holy wars)
- Which statistical package to use to evaluate defect data (and holy wars)
- Defect classifications (and holy wars)
- How to conduct inspections (and holy wars)
- Roles on inspections / peer reviews

- How to write better test plans
- How to perform root cause analysis
- How to write review scripts

Contributors to the defect dilemma

- Software quality problems result from defective products and defective usage
- Many root causes of poor product quality and poor usage exist
- Software defects are injected by product developers
- Even trained and experienced developers inject defects
- Too often, a quality assurance group is assembled to remove defects from products
- Too often, a quality assurance group is chartered to develop comprehensive testing activities to reduce defects
- Many product defects exist in the requirements and design of the product; they cannot be removed during testing because they have become an accepted part of the product specification
- An increasing reliance solely on testing for defect removal will not address defects that emanate from requirements and design (but it will show lots of "activity" and require lots of resources)

Recent Examples of Defects

Marriott – Social security and credit card numbers of 200,000+ employees and customers missing

Ford – 70,000 employee and former employee social security numbers on a stolen computer

Sam's Club – 600 customer credit card data stolen in two weeks

Justice Department – posted social security numbers and personal data of persons involved in "cases" on its web site

More Recent Examples of Defects

TJ Maxx reported information from 45 million credit cards stolen. *informationweek; April 2, 2007*

TJX credit card thief ordered to pay ~ $600,000 and serve five years in prison. Original thieves have not been caught. About $3M is losses is known to have occurred from this crime. *informationweek; September 17, 2007*

TJX data breach may involve 94 million credit cards *USA Today; October 25, 2007*

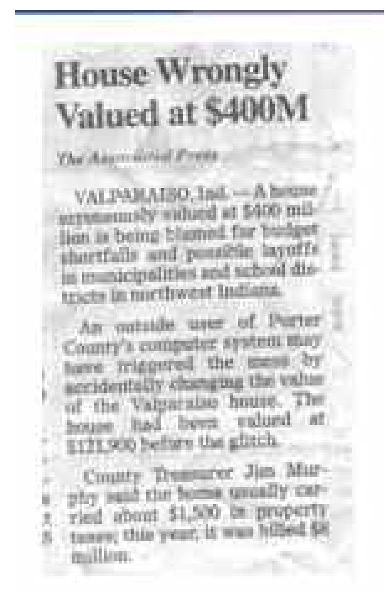
MGM – Computer glitch slows MGM Mirage check-ins

Workers resorted to manual check-in for thousands of guests

“glitch” hits seven hotels – five on the LV strip

“first time” this “bug” has surfaced

Las Vegas Review-Journal; October 24, 2007



House Wrongly Valued at $400M

The Associated Press

VALPARAISO, Ind. — A house erroneously valued at $400 million is being blamed for budget shortfalls and possible layoffs in municipalities and school districts in northwest Indiana.

An outside user of Porter County's computer system may have triggered the mess by accidentally changing the value of the Valparaiso house. The house had been valued at $121,900 before the glitch.

County Treasurer Jim Murphy said the home usually carried about $1,500 in property taxes; this year, it was billed $8 million.

And more . . .

Software defects cost the U.S. $59.6B a year[1]
38 percent of polled organizations have no SQA program[2]
Software technicians in Panama are charged with second degree murder after 27 patients received overdoses of gamma rays; 21 have died in 40 months[3]
BMW, DaimlerChrysler, Mitsubishi, and Volvo experience product malfunctions (engine stalls, gauges not illuminated, wiping intervals, wrong transmission gears) due to software[4]
In the year 2000, the nctimes placed the cost of one virus at $10B[5]
After more than two years of delay, the state Department of Labor's $13M million computer system to process unemployment insurance claims and checks still isn't fully off the ground[6]

1 Informationweek, *Behind the Numbers*, March 29, 2004; pg 94
2 CIO, *By the Numbers*, December 1, 2003, pg 28
3 Baseline – The Project Management Center, *We Did Nothing Wrong*, March 4, 2004
4 Informationweek, *Software Quality*, March 15, 2004; pg 56
5 www.nctimes.com/news/050600/d.html
6 Albuquerque Journal; *Computer A Real Labor For State*; 6/04
Reference: *Applying Lean Six Sigma to Software Engineering*; International Function Point Users Group; Schofield; September, 2004

Inspections – A response (almost 40 years old!)

- Developed by IBM in 1972 after three years of experimentation
- Referred to as a “Fagan inspection,” or “formal inspection”
- An expectation of formal inspection is to reduce rework (a lean six sigma source of “waste” / *muda*)
- Not intended as a substitute for testing
- Enhanced to include causal analysis activity for defect prevention (a CMMI® Maturity Level 5 Process Area)

Why Inspect Product?

- Eliminate the undesired
- Identify what's missing
- Determine if products fulfills intent
- Validate the verification process: value, efficiency, ROI
- Uncover process improvements
- Establish and sustain customer confidence

Assertions regarding defects

- The sooner a defect is detected (and removed) the lower the cost of repair and rework
- The later a defect is detected (and removed) the greater the consequence to cost and the impact to schedule
- Verification (by the supplier) and validation (by the customer) are the two means for identifying defects
- Defect discovery by the supplier is preferred
- Therefore, some verification (confirmed by defect injection and detection data) may be needed as part of the development (or modification) of each product artifact
- All stakeholders related to a product from upper management to the final builder are likely to inject defects. We all need to admit that we are recovering defect injectors
- Sources of defect removal include: personal reviews, inspections and peer reviews, testing, and customer change requests
- We need to collect data from all defect removal activities if we want to eliminate defects from products
- Defects found in testing evidence potential process or process execution failure; until resolved we can only guarantee more defects in the future

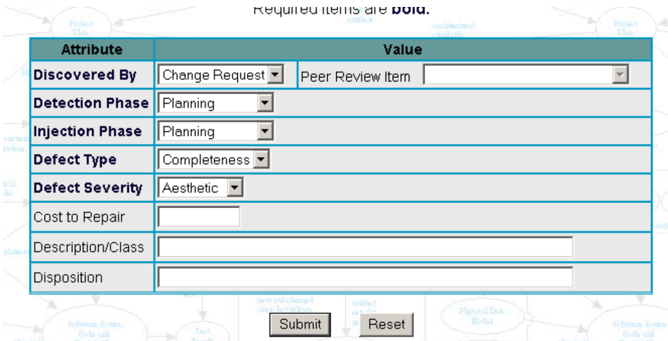
More assertions regarding defects

- Only ½ of the defects in a product are removed by testing; this limitation is not a reflection on the testing process.
- An organization's equivalent defect-related data is better than that of other organizations. The same is true of a project. The same is true for a person.
- Lessons learned from inspections, peer reviews, test results, and change requests should trigger needed process changes to eliminate the source of defects.
- Lessons learned from individuals should be shared with the team. Lessons learned with the team should be shared with the organization. The opposite flow exchanges should also occur: organization-to-team-to-individual.
- An inspection or peer review should be pre-requisite to the *completion* of the deliverable (in software engineering this is much more than the *code*).
- Inspections and peer reviews reduce the TCO of products.
- An inverse relationship exists between quality and defect density.

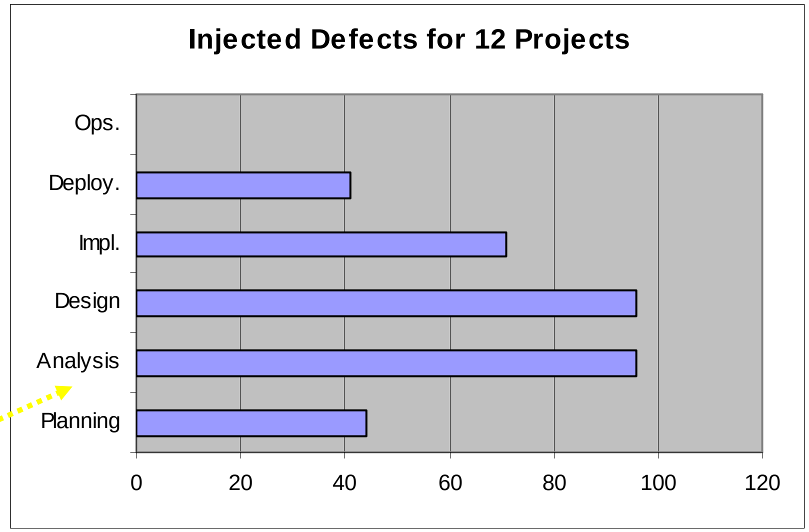
Getting to know your process

- In what work product (or sub-assemblies) do we inject the most defects?
- What is the estimate of how many defects are typically found in a product like this, using a review like this?
- In what verification activity do we detect the most defects?
- What is the average cost to repair a defect?
- What's the most we ever spent on rework related to a defect?
- What are the types of defects we are most likely to find by work product?
- What steps have been taken to eliminate the source of defects, and what was the measured result of that action?
- What training and organizational assets exist to assist new team members with verification activities?
- What is the return on investment for verification activities; that is, what does it cost to perform them and what would it cost if the product was released with those defects?
- How many more defects remain undetected in the product?

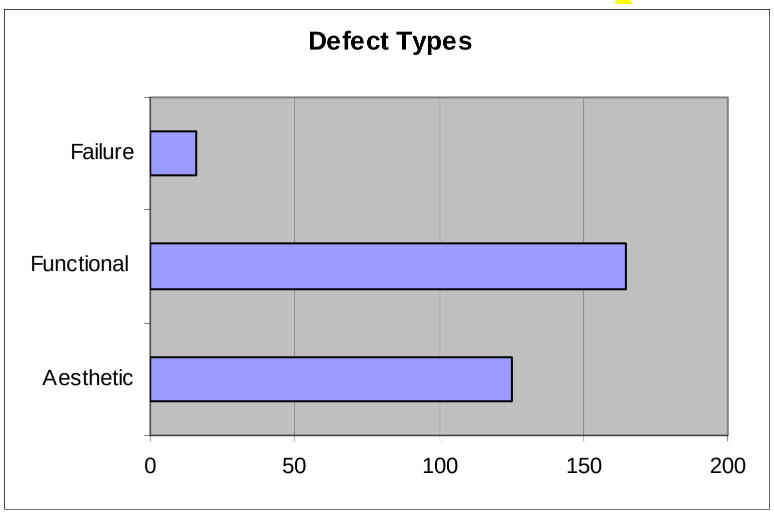
Some answers – measurement collection and analysis (GP 3.2, MA, VER, VAL)

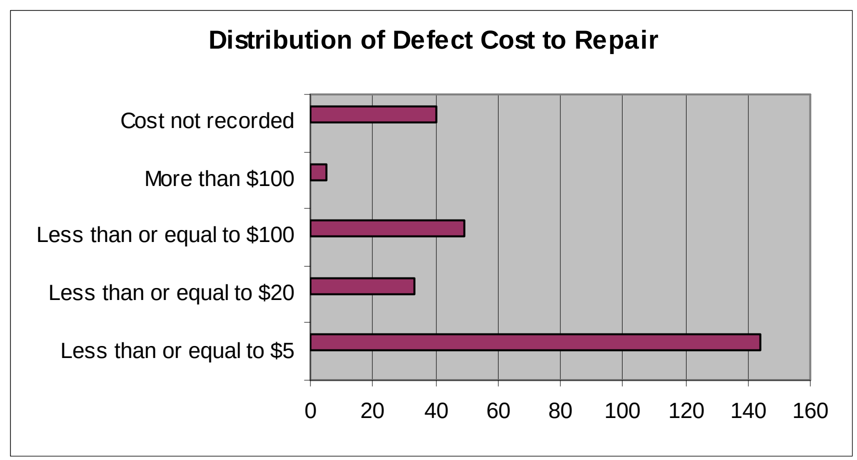


Measure / Record



Analyze





Some answers – measurement collection and analysis - (cont'd)

Modify or Input Defects

Defect summary by How and Where discovered

SILC Review Phase	Discovered By	Defect Type	Total Defects	Total Cost	Cost Per Defect
Planning	Change Request	Completeness	20	707	35.35
Planning	Peer Review	Completeness	91	3867	42.49
Planning	Peer Review	Consistency	21	667	31.76
Planning	Peer Review	Corrective	33	1481	44.88
Planning	Test Plan	Completeness	1	4	4.00
Analysis	Change Request	Completeness	6	180	30.00
Analysis	Change Request	Corrective	6	60	10.00
Analysis	Peer Review	Completeness	124	2900	23.39
Analysis	Peer Review	Consistency	103	1968	19.11
Analysis	Peer Review	Corrective	109	1890	17.34
Design	Change Request	Completeness	3	160	53.33
Design	Change Request	Corrective	4	170	42.50
Design	Peer Review	Completeness	265	7406	27.95
Design	Peer Review	Consistency	59	1313	22.25
Design	Peer Review	Corrective	162	2054	12.68
Design	Test Plan	Completeness	2	8	4.00
Design	Test Plan	Consistency	1	6	6.00
Design	Test Plan	Corrective	4	124	31.00
Implementation	Change Request	Completeness	2	80	40.00
Implementation	Change Request	Corrective	8	1337	167.13
Implementation	Peer Review	Completeness	63	2125	33.73
Implementation	Peer Review	Consistency	55	1909	34.71
Implementation	Peer Review	Corrective	76	2572	33.84
Implementation	Test Plan	Completeness	36	3801	105.58
Implementation	Test Plan	Consistency	15	1146	76.40
Implementation	Test Plan	Corrective	85	3151	37.07
Deployment	Change Request	Corrective	4	67	16.75
Deployment	Peer Review	Completeness	29	200	6.90
Deployment	Peer Review	Consistency	1	4	4.00
Deployment	Peer Review	Corrective	7	38	5.43
Operational	Change Request	Completeness	5	408	81.60
Operational	Change Request	Consistency	4	195	48.75
Operational	Change Request	Corrective	12	1215	101.25
Operational	Test Plan	Corrective	11	1319	119.91

Modify or Input Defects

Some answers – measurement collection and analysis - (cont'd)

Defect summary by work product

Artifact Reviewed	Total Defects	Total Cost	Cost Per Defect
External Interfaces Definition	94	1612	17.15
Information Model	57	525	9.21
Internal Components Definition	67	1507	22.49
Other: Business Rules	5	13	2.60
Other: Business Rules and Use Cases	33	122	3.70
Other: CSA Administrator Documentation	2	20	10.00
Other: CSA User Documentation	2	20	10.00
Other: EP Interim Solution: Source Code	13	308	23.69
Other: External Interfaces Definition - Admin	5	9	1.80
Other: External Interfaces Definition - Reporting	9	29	3.22
Other: External Interfaces Definition - Wizard	4	47	11.75
Other: Information Model. Data Dictionary	7	22	3.14
Other: Interim Solution Test Plan	6	159	26.50
Other: Internal Components Definition--Admin	1	1	1.00
Other: Internal Components Definition--Course	6	266	44.33
Other: RS2 User Process Model	5	70	14.00
Other: RS3 SW Requirements & Design Specification	1	50	50.00
Other: RS3 Software Source Code & Executables	5	610	122.00
Other: RS3 User Process Model	8	165	20.63
Other: RS5 User Process Model	4	14	3.50
Other: RS6 User Process Model	7	300	42.86
Other: RS7 Design	7	200	28.57
Other: RS7 Software Source Code & Executables	3	40	13.33
Other: RS7 User Process Model	8	135	16.88
Other: Software Requirements Specification	27	470	17.41
Other: Test Plan, Information Model, External Interfaces	25	134	5.36
Other: Use Case Diagrams & Textual Use Cases	1	2	2.00
Other: Use Case Model	3	7	2.33
Other: User Documentation	17	101	5.94
Project Plan	129	4005	31.05
Software Source Code & Executables	139	4561	32.81
Test Plan	209	3774	18.06
User Process Model	132	5641	42.73

For defect removal, Tom Glib reports some inspection efficiencies as high as 88 percent. Jones, *Software Quality*, pg 215

Some answers – measurement collection and analysis - (cont'd)

Phase Injected

Phase Detected

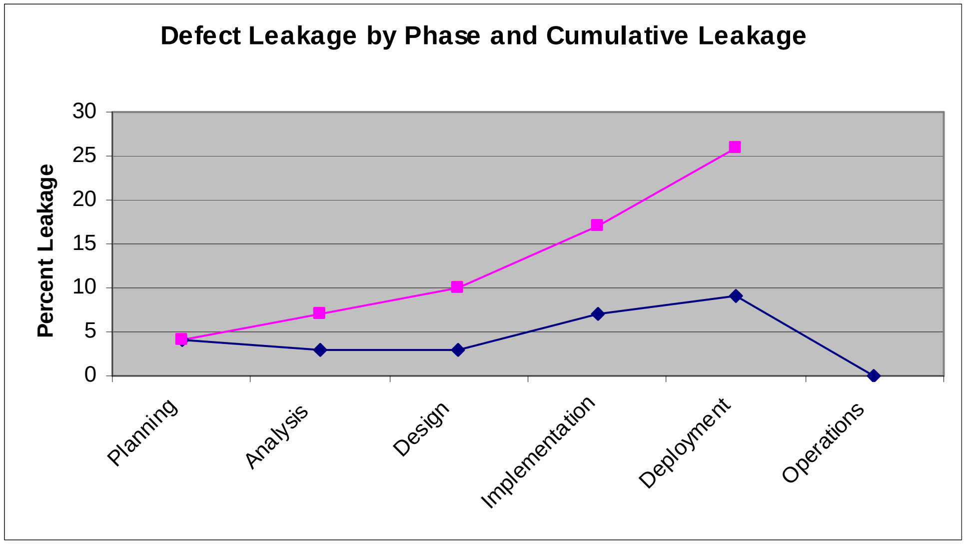
	Planning	Analysis	Design	Impl.	Deploy.	Ops.
Planning	109	4	8	8		
Analysis	1	290	2			
Design	3	9	476	2		
Imple.	1	1	13	296		
Deploy.				1	20	
Ops.			3	24	2	30
Total Injected	114	304	502	331	22	30
% leakage	4	3	3	7	9	

What does this association matrix REVEAL?

Some answers – measurement collection and analysis - (cont'd)

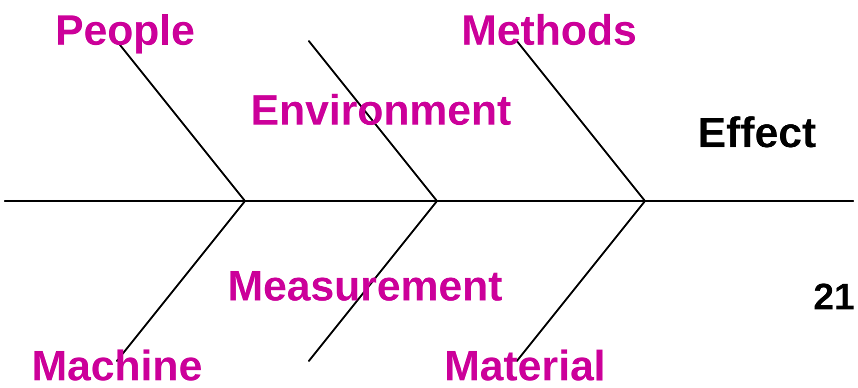
Given:

- **Peer Review is performed in Planning**
- **Peer Reviews are performed in Analysis**
- **Peer Reviews are performed in Design**

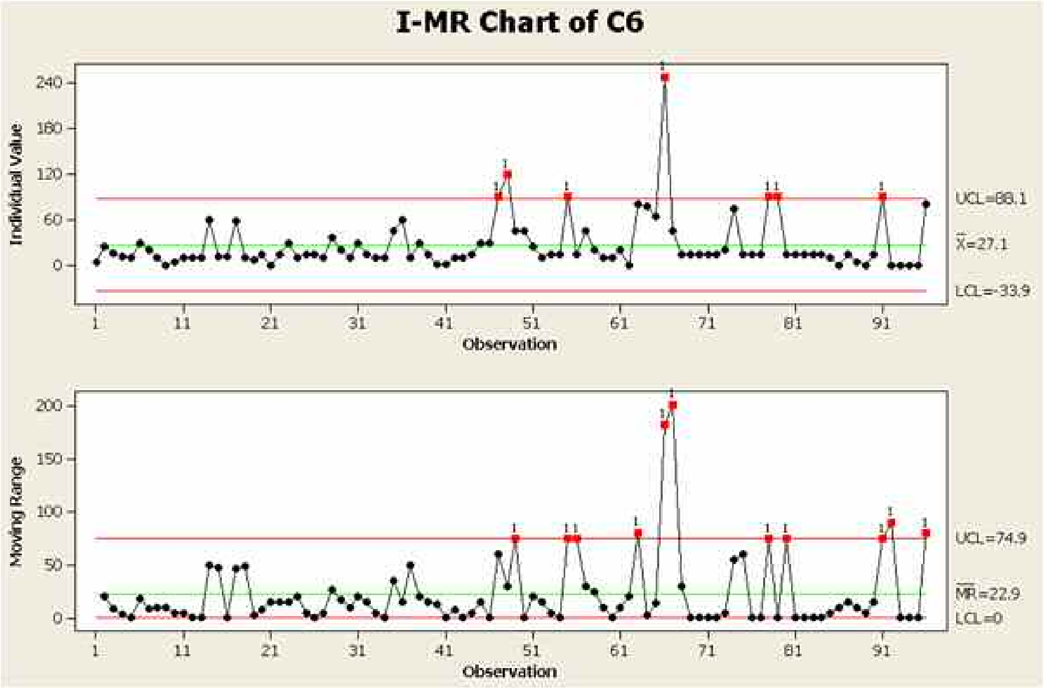


Look at Planning & Analysis

- **How is it that so many defects are removed in Implementation?**
- **Does the organization need more Peer Reviews in Planning & Analysis?**
- **How effective are Design Peer Reviews?**



Some answers – measurement collection and analysis / higher level maturity (cont'd)



Special (Assignable) Cause removal required at CMMI® Level 4

How well the process is performed

How many more defects remain undetected in the product?

Barry Boehm – requirements defects that made their way into the field could cost 50-200 times as much to correct as defects that were corrected close to the point of creation.[1] The U.S. space program had two high-profile failures in 1999 with software defects that cost hundreds of millions of dollars.

Capers Jones – reworking defective requirements, design, and code typically consumes 40 to 50 percent or more of the total cost of most software projects and is the single largest cost driver.[2]

Tom Gilb – half of all defects usually exist at design time[3], (confirmed by Jones's data).

Capers Jones – as a rule of thumb, every hour you spend on technical reviews upstream will reduce your total defect repair time from three to ten hours.[4]

O'Neill calculated the ROI for software inspections between four and eight to one.[5]

1. Boehm, Barry W. and Philip N. Papaccio. "Understanding and Controlling Software Costs," *IEEE Transactions on Software Engineering,* v. 14, no. 10, October 1988, pp. 1462-1477.
2. Jones, Capers. *Estimating Software Costs*, New York: McGraw-Hill, 1998.
3. Gilb, Tom. *Principles of Software Engineering Management*. Wokingham, England: Addison-Wesley, 1988.
4. Jones, Capers. *Assessment and Control of Software Risks*. Englewood Cliffs, N.J.: Yourdon Press, 1994.
5. O'Neill, Don; *National Software Quality Experiment: Results 1992 – 1999*: Software Technology Conference, Salt Lake City, 1995, 1996, 2000

An answer to the last question – *How many more defects remain in the product?* (Latent defect estimation)

Place a check mark in the intersecting cells for each defect found by each participant.
Count the defects that each engineer found (*Counts* for Engineer A, B, and C).
Column A: check and count all the defects found by the engineer who found the most unique defects. 5
Column B: check and count all of the defects found by all of the other engineers. 4
Column C: check and count the defects common to columns A and B. 2
The estimated number of defects in the product is AB/C. Round to the nearest integer. (5 * 4) / 2 = 10
The number of defects found in the inspection is A+B-C. 5 + 4 – 2 = 7
The estimated number of defects remaining is the estimated number of defects in the product minus the number found. (AB/C) – (A+B-C). 10 – 7 = 3

Use team “thresholds” to determine whether or not to repeat the Peer Review.

Defect No	Engineer Larry	Engineer Curly	Engineer Moe	“Column A”	“Column B”	“Column C”
1	√			√		
2	√			√		
3			√		√	
4	√	√		√	√	√
5	√			√		
6	√		√	√	√	√
7		√			√	
Counts	5	2	2	5	4	2

The capture-recapture method (CRM) has been used for decades by population biologists to accurately determine the number of organisms studied. LaPorte RE, McCarty DJ, Tull ES, Tajima N., Counting birds, bees, and NCDs. Lancet, 1992, 339, 494-5.
See also Introduction to the Team Software Process; Humphrey; 2000; pgs. 345 – 350

What if . . .

Two engineers find the most defects? (pick either for column A and complete the process)

Place a check mark in the intersecting cells for each defect found by each participant.
Count the defects that each engineer found (*Counts* for Engineer A, B, and C).
Column A: check and count all the defects found by the engineer who found the most unique defects. 5
Column B: check and count all of the defects found by all of the other engineers. 7
Column C: check and count the defects common to columns A and B. 3
The estimated number of defects in the product is AB/C. Round to the nearest integer. (5 * 7) / 3 = 12
The number of defects found in the inspection is A+B-C. 5 + 7 – 3 = 9
The estimated number of defects remaining is the estimated number of defects in the product minus the number found. (AB/C) – (A+B-C). 12 – 9 = 3

Defect No	Engineer Larry	Engineer Curly	Engineer Moe	"Column A"	"Column B"	"Column C"
1	√	√		√	√	√
2	√			√		
3		√	√		√	
4	√	√		√	√	√
5	√			√		
6	√	√	√	√	√	√
7		√			√	
Counts (L)	5	5	2	5	5	3
Counts (C)	5	5	2	5	6	4

What if . . .

Hardly any mutual defect finds?

Place a check mark in the intersecting cells for each defect found by each participant.
Count the defects that each engineer found (*Counts* for Engineer A, B, and C).
Column A: check and count all the defects found by the engineer who found the most unique defects. 4
Column B: check and count all of the defects found by all of the other engineers. 4
Column C: check and count the defects common to columns A and B. 1
The estimated number of defects in the product is AB/C. Round to the nearest integer. (4 *4) / 1 = 16
The number of defects found in the inspection is A+B-C. 4 + 4 – 1 = 7
The estimated number of defects remaining is the estimated number of defects in the product minus the number found. (AB/C) – (A+B-C). 16 – 7 = 9

Defect No	Engineer Larry	Engineer Curly	Engineer Moe	“Column A”	“Column B”	“Column C”
1	√			√		
2	√			√		
3		√			√	
4	√	√		√	√	√
5			√		√	
6	√			√		
7		√			√	
Counts (L)	4	3	1	4	4	1

Summary of key points:

Barry Boehm – requirements defects that made their way into the field could cost 50-200 times as much to correct as defects that were corrected close to the point of creation.[1] The U.S. space program had two high-profile failures in 1999 with software defects that cost hundreds of millions of dollars.

Capers Jones – reworking defective requirements, design, and code typically consumes 40 to 50 percent or more of the total cost of most software projects and is the single largest cost driver.[2]

Tom Gilb – half of all defects usually exist at design time[3], (confirmed by Jones's data).

Capers Jones – as a rule of thumb, every hour you spend on technical reviews upstream will reduce your total defect repair time from three to ten hours.[4]

O'Neill calculated the ROI for software inspections between four and eight to one.[5]

CMMI®-Enabled Practices with CRM

Measurement and Analysis

SG 1 Align Measurement and Analysis Activities

SP 1.1 Establish Measurement Objectives (reduce or eliminate defects)

SP 1.2 Specify Measures (estimated number of latent defects)

SP 1.3 Specify Data Collection and Storage Procedures (peer reviews)

SP 1.4 Specify Analysis Procedures

SG 2 Provide Measurement Results

SP 2.1 Collect Measurement Data

SP 2.2 Analyze Measurement Data

SP 2.3 Store Data and Results

SP 2.4 Communicate Results

Verification – VER

SG 1 Prepare for Verification

SP 1.1 Select Work Products for Verification

SP 1.2 Establish the Verification Environment

SP 1.3 Establish Verification Procedures and Criteria

SG 2 Perform Peer Reviews

SP 2.1 Prepare for Peer Reviews

SP 2.2 Conduct Peer Reviews

SP 2.3 Analyze Peer Review Data

SG 3 Verify Selected Work Products

SP 3.1 Perform Verification

SP 3.2 Analyze Verification Results

CMMI®-Enabled Practices with CRM

Organizational Process Performance

SG 1 Establish Performance Baseline and Models

SP 1.1 Select Processes

SP 1.2 Establish Process-Performance Measures

SP 1.3 Establish Quality and Process-Performance Objectives

SP 1.4 Establish Process-Performance Baselines

SP 1.5 Establish Process-Performance Models

Quantitative Project Management

SG 1 Quantitatively Manage the Project

SP 1.1 Establish the Project's Objectives

SP 1.2 Compose the Defined Process

SP 1.3 Select the Subprocesses that Will Be Statistically Managed

SP 1.4 Manage Project Performance

SG 2 Statistically Manage Subprocess Performance

SP 2.1 Select Measures and Analytic Techniques

SP 2.2 Apply Statistical Methods to Understand Variation

SP 2.3 Monitor Performance of the Selected Subprocesses

SP 2.4 Record Statistical Management Data

CMMI®-Enabled Practices with CRM

Causal Analysis and Resolution

SG 1 Determine Causes of Defects

SP 1.1 Select Defect Data for Analysis

SP 1.2 Analyze Causes

SG 2 Address Causes of Defects

SP 2.1 Implement the Action Proposals

SP 2.2 Evaluate the Effect of Changes

SP 2.3 Record Data

Generic Practices enabled by CRM

GP 3.2 Collect Improvement Information#

GP 4.1 Establish Quantitative Objectives for the Process#

GP 4.2 Stabilize Subprocess Performance#

CMMI® Process Areas, Goals, Practices, and more

CMMI® V1.2 Staged Representation
Level 5 – Optimizing
Causal Analysis & Resolution (CAR) [SUP]
Organizational Innovation & Deployment (OID) [PcM]
Level 4 – Quantitatively Managed
Organizational Process Performance (OPP) [PcM]
Quantitative Project Management (QPM) [PjM]
Level 3 – Defined
Decision Analysis & Resolution (DAR) [SUP]
Integrated Project Management (IPM) [PjM]
Organizational Process Definition (OPD) [PcM]
Organizational Process Focus (OPF) [PcM]
Organizational Training (OT) [PcM]
Product Integration (PI) [ENG]
Requirements Development (RD) [ENG]
Risk Management (RSKM) [PjM]
Technical Solution (TS) [ENG]
Validation (VAL) [ENG]
Verification (VER) [ENG]
Level 2 – Managed
Configuration Management (CM) [SUP]
Measurement & Analysis (MA) [SUP]
Product & Process Quality Assurance (PPQA) [SUP]
Project Monitoring & Control (PMC) [PjM]
Project Planning (PP) [PjM]
Requirements Management (REQM) [ENG]
Supplier Agreement Management (SAM) [PjM]
Level 1 – Initial

Categories
ENG - Engineering
SUP - Support
PcM - Process Management
PjM - Project Management

CMMI® Generic Practices	
GG5 Institutionalize a Optimizing Process	
GP5.1	Ensure Continuous Process Improvement
GP5.2	Correct Root Causes of Problems
GG4 Institutionalize Quantitatively Managed Process	
GP4.1	Establish Quantitative Objectives for the Process
GP4.2	Stabilize Subprocess Performance
GG3 Institutionalize a Defined Process	
GP3.1	Establish a Defined Process
GP3.2	Collect Improvement Information
GG2 Institutionalize a Managed Process	
GP2.1	Establish an Organizational Policy
GP2.2	Plan the Process
GP2.3	Provide Resources
GP2.4	Assign Responsibility
GP2.5	Train People
GP2.6	Manage Configurations
GP2.7	Identify and Involve Relevant Stakeholders
GP2.8	Monitor and Control the Process
GP2.9	Objectively Evaluate Adherence
GP2.10	Review Status with Higher Level Mgmt.
GG1 Achieve Specific Goals	
GP1.1	Perform Specific Practices