Estimating Latent Defects using Capture-Recapture: Lessons from Biology

Joseph R. Schofield

Sandia National Laboratories

P. O. Box 5800

MS 0661

Albuquerque, N.M. 87185

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.

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(R) ML 2, 3, and 4 practices.

A Quick Look Back & Update on Recent IFPUG / ISMA Presentations

2008	*Estimating Latent Defects Using Capture-Recapture: Lessons from Biology; Arlington, VA.; 2008 International Software Measurement and Analysis (ISMA) Conference; September 18, 2008* *Beyond Defect Removal: Latent Defect Estimation with Capture Recapture Method*; CrossTalk, August 2007 (reprinted in IFPUG's MetricViews, Winter 2008) *Latent Defect Estimation - Maturing Beyond Defect Removal using Capture-Recapture Method*; QAI QAAM Conference; September 10, 2008
2007	*'Manda, Panda, and the CMMI(R)*; Las Vegas, NV.; 2007; ISMA Conference; September 14, 2007
2006	*Defect Collection & Analysis – The Basis of Software Quality Improvement*; ISMA Conference, September, 2006 *Defect Management through the Personal Software Process^{SM}; CrossTalk, September 2003* *The Team Software Process^{SM} – Experiences from the Front Line*; Software Quality Forum; Arlington, Virginia, March; 2003 *Measuring Software Process Improvement - How to Avoid the Orange Barrels*; System Development, December 2001 *Usable Metrics for Software Improvement within the CMM*; Software Quality Forum 2000; Santa Fe, N.M.; April, 2000
2004	*Applying Lean Six Sigma to Software Engineering;* IFPUG Conference; September, 2004 *When Did Six Sigma Stop Being a Statistical Measure?; CrossTalk, April 2006* *Lean Six Sigma - Real Stories from Real Practitioners*; Albuquerque, N.M.; N.M. SPIN; August 2005 *Six Sigma & Software Engineering: Complement or Collision*; Albuquerque, N.M.; N.M. SPIN; August, 2004
2003	*Amplified Lessons from the Ant Hill – What Ants and Software Engineers Have in Common*; IFPUG Conference, Sept., 2003 *Lessons from the Ant Hill - What Ants and Software Engineers Have in Common*; Information Systems Management, Winter 2003
2002	*Counting KLOCs – Software Measurement's Ultimate Futility (I can't do this anymore, or who am I fooling?, or why not count ants?);* IFPUG Conference; September, 2002 *Lines of Code - Statistically Unreliable for Software Sizing*?; Computer Aid, Inc.; Webinar; October 14, 2008 *The Statistical Case Against the Case for using Lines of Code in Software Estimation;* 4th World Congress on Software Quality; Bethesda, MD.; September 17, 2008 *The Statistically Unreliable Nature of Lines of Code;* CrossTalk, April 2005 (Reprinted at least twice, cited by NIST *Metrics and Measures* http://samate.nist.gov/index.php/Metrics_and_Measures) *A Practical, Statistical, and Criminal Look at the Use of Lines of Code as a Software Sizing Measure*; N.M. SPIN; March, 2004

Why are we here?

- Recent defect horror stories
- Why testing will never eliminate defects
- How to remove defects early and reduce costs and rework
- What is Capture – Recapture Method
- Applying CRM – hands-on fun

Quick Hits on Capture-Recapture Method

- **CRM can be used to predict the number of estimated defects remaining in a product. This estimate can then be used to make quantified, data-driven decision on how to proceed with a software product.** *IFPUG MetricViews; Winter 2008*
- **Used by Jerry Weinberg to estimate remaining typos based on how many his reviewers found.** *Beyond Defect Removal: Latent Defect Estimation with Capture Recapture Method; CrossTalk, August 2007*
- **Try this: Something Fishy (statistics) from PBS Mathline.** *www.pbs.org/mathline*
- **A quickie primer – *Capture-recapture and Removal Methods*.** *www.ento.vt.edu~sharov/PopEcol/lec2/caprecap.html*
- **A more detailed look – *Life Cycle-Based Defect Removal with Capture Recapture Methods*.** *Computer Aid, Inc. webinar, April 22, 2008*
- **A quickie history – Capture-Recapture History.** *www.pitt.edu/~yuc2/cr/history.htm*

Quick Hits on Capture-Recapture Method

- **Used successfully for decades by population biologists to study fish and wildlife populations.**
- **Dates to 1894 and the tagging of fish by Petersen.**
- **Used in 1930 to estimate duck populations by Lincoln.**
- **1938 Schnabel uses the K-sample with CRM (Anderson Darling).**
- **Used more recently for epidemiology studies.**
- ***Capture-Recapture method: the gold standard for incidence and prevalence.*** *The New Zealand Medical Journal, June 20, 2003.*

BTW, 90 percent of the world's living organisms remain unclassified; *Spirit magazine, April, 2008.*

Can we (I) be honest?

- Software quality problems result from defective products and defective usage.
- Many root causes of poor product quality and poor usage exist.
- Our focus today is on the impact of software quality that results from defective product.
- 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 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)!
- Even experienced developers inject one defect per every ten instructions of code that are written.

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*

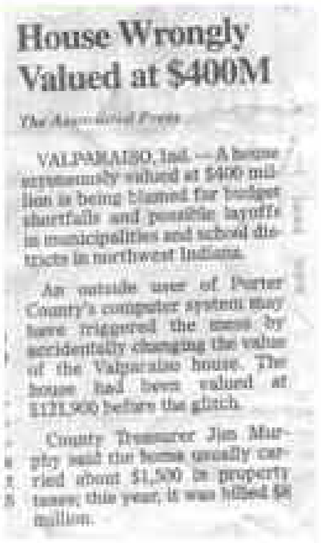
- MGM – Computer glitch slows MGM Mirage check-ins

  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

Sample defect types:

Completeness – Does it exist in its intended entirety?

- Are all the expected elements present?
- Is each element detailed to the level prescribed by the process?

Correctness – is it right?

- Does this item (process or information) do what was intended?
- If an element is wrong, treat it as incorrect

Consistency – is it used one way and the same way throughout the product and its interfaces?

- Is this process performed somewhere else and called something different? (Two programs or codes that perform the same function.)
- Is this information represented somewhere else or presented in some other way? (A date presented in different formats).
- Is an element duplicated in function or form, or if an element does not map to an ancestral artifact but should, treat it as inconsistent.

Conciseness – is it lean?

- Does it do more than intended?
- Is it larger than necessary?

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.

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

Okay, let's do this

- Because you realize defects are everywhere
- Because you recognize the need to eliminate defects
- Because you appreciate that latent defects can be estimated
- Because you came to grab something to take back
- Because some of us learn by doing . . .
- Because