

**“Recent case studies in testing:
Parallel Inspections and Testing after Delivery”**

by

Les Hatton

Professor of Forensic Software Engineering
CISM, University of Kingston
L.Hatton@kingston.ac.uk

Version 1.1: 21/Feb/2005

Overview



- Case History 1: Parallel Inspections
- Case History 2: Testing after delivery
- Conclusions

Case history 1: Parallel Inspections



- When do you stop testing ?
 - Subjective methods
 - Testers are bored
 - Test budget runs out
 - Testers feel its “OK”
 - Everybody is bored
 - Marketing “have a customer waiting”
 - Management makes death threats

None of these have any relationship to the product quality



Case history 1: Parallel Inspections

- When do you stop testing ?
 - Objective methods
 - Test plan is completed
 - Some form of coverage target is met
 - Minimum percentage of total defects are fixed
 - Desired level of reliability is reached

All of these have a relationship to the product quality, (although each makes some form of assumption).

Case history 1: Parallel Inspections

- Estimating the total number of defects by parallel inspection
 - Let $P(A)$ be probability of a defect being detected by person A in some product
 - Let $P(B)$ be probability of a defect being detected by person B in the same product.
 - *If* we assume independence:-

$$P(A \cap B) = P(A)P(B)$$

Case history 1: Parallel Inspections

- Estimating the total number of defects by parallel inspection
 - Suppose there are n defects altogether
 - Suppose person A finds a defects, person B finds b defects and q defects are found by both; then

$$\frac{q}{n} = \frac{a}{n} \frac{b}{n}$$

Case history 1: Parallel Inspections



- Estimating the total number of defects by parallel inspection
 - So, with a bit of re-arranging,

$$n = \frac{ab}{q}$$

Read this as n is an estimator for (ab/q) .

Case history 1: Parallel Inspections



■ Experiment

- Software component of 62 lines deliberately seeded with 26 ± 2 defects.
- Inspected for 30 minutes individually and 15 minutes as teams of two to identify common defects
- Run 3 times so far in India, Austria and Germany
- Artificial team of two compilers in full detection mode added.

Case history 1: Parallel Inspections



■ Results

- Data from India, 11 teams of 2 persons plus compiler 'team'
 - Trimmed mean = 24.4 (26)
 - Trimmed standard deviation = +/- 7.8 (+/- 2)(Trimmed by discarding outliers so robust)

Case history 1: Parallel Inspections



■ Results

- Data from Austria, 3 teams of 2 persons plus compiler 'team'
 - Mean = 28.09 (26)
 - Standard deviation = +/- 3.74 (+/- 2)
- (Not statistically robust)

Case history 1: Parallel Inspections



■ Results

- Data from Germany, 4 teams of 2 persons plus compiler 'team'
 - Mean = 21.81 (26)
 - Standard deviation = +/- 10.29 (+/- 2)
- (Not statistically robust)



Case history 1: Parallel Inspections

■ Conclusions

- Although the relationship assumes independence, if there are sufficient test teams, robust estimation is pretty good.
- Even with non robust estimation, the ‘ball park’ estimate is not bad compared with conventional ways of estimating the total number of defects.

Case history 1: Parallel Inspections



■ Caveats

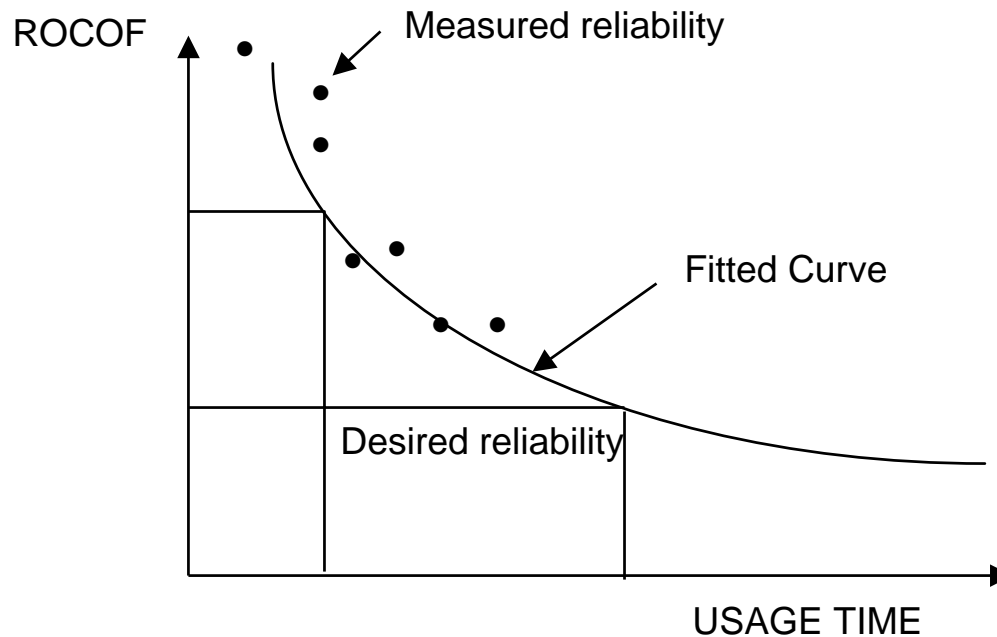
- Independence threatened if checklists used (although no obvious signs of this in the experiments, some of which used checklists)
- Problems when very experienced person teamed with very inexperienced person resolvable only by trimming.

Overview



- Case History 1: Parallel Inspections
- Case History 2: Testing after delivery
- Conclusions

Case history 2: Testing after Delivery



- The rationale is to gain availability without compromising reliability

Case history 2: Testing after Delivery



- Question: How do you test after delivery ?
- Answer:
 - Make sure you can update software easily. This means Internet download only – no hard media.
 - Keep on testing as if you had not released it and release new versions as you find defects.

How much additional improvement should we expect to get ?

Case history 2: Testing after Delivery

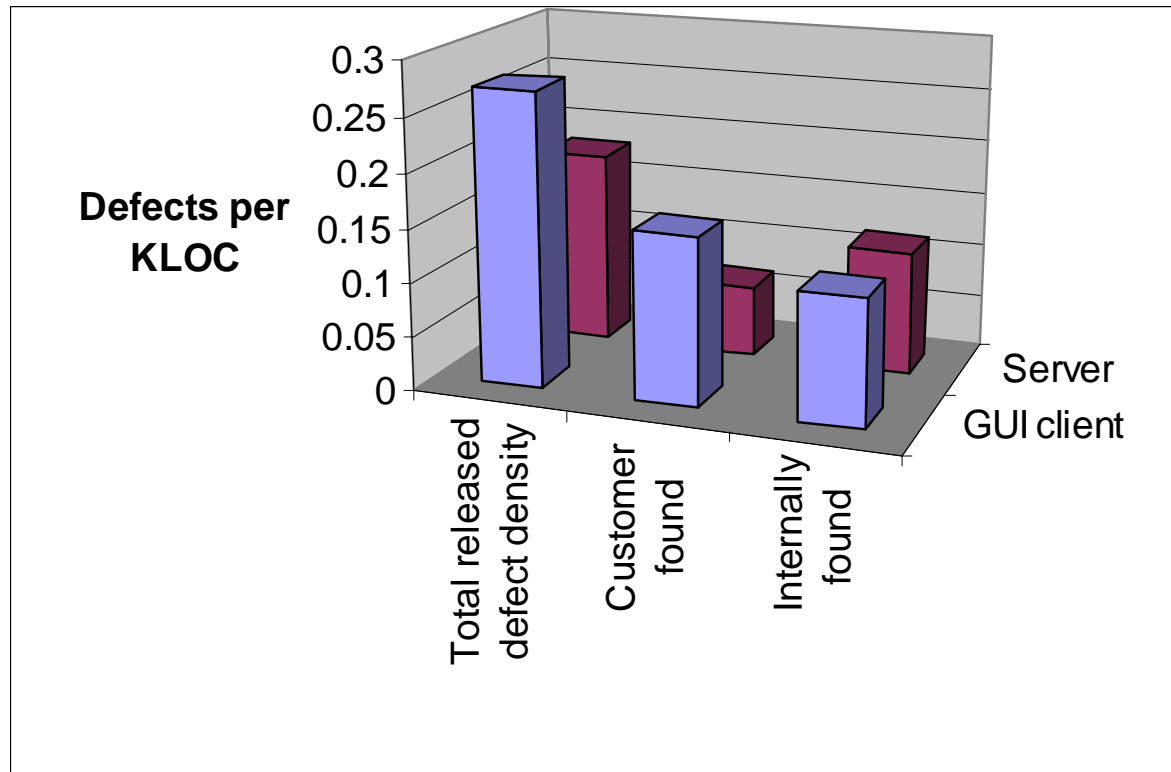


■ Experiment

- Two client / server products around 120,000 LOC each.
- All defects stored pre and post delivery and tagged with the discoverer.
- Two hypotheses tested:-
 1. Is any improvement achieved by Testing after Delivery significant and how much should we expect ?
 2. Are users more sensitive to stylistic defects (GUI) or serious defects (server) ?

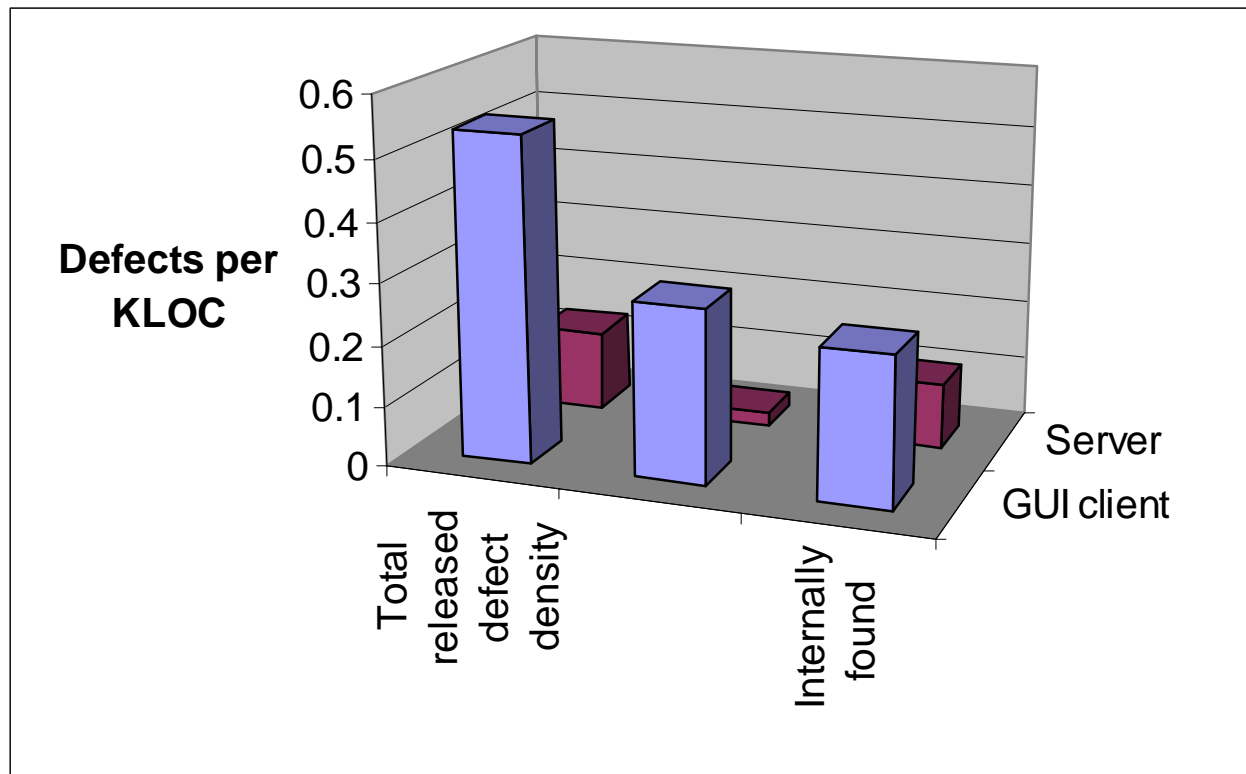
Case history 2: Testing after Delivery

■ Data – Product 1



Case history 2: Testing after Delivery

■ Data – Product 2



Case history 2: Testing after Delivery



- Results: Hypothesis 1

- In both cases, *more than half the defects found after release* were found by internal testing and corrections released before any customer encountered them.

(Significance not yet calculated).

Case history 2: Testing after Delivery

■ Results: Hypothesis 2

- The z-test for proportions was used.
 - Let p_s be the proportion of defects found by users in the server and p_c be the proportion of defects found by users in the GUI client and $q_x = 1 - p_x$. Let n_s and n_c be total number of defects found in server and client.

Then:-

$$z = \frac{p_s - p_c}{\sqrt{\hat{p}\hat{q}\left(\frac{1}{n_s} + \frac{1}{n_c}\right)}} \sim N(0,1)$$

Case history 2: Testing after Delivery



- Results: Hypothesis 2
 - Users are more sensitive to stylistic defects in the GUI than in substantive defects in the server at the 2-3% confidence level in both products
 - Note that in both cases, the really important data is produced by the server.

Overview



- Case History 1: Parallel Inspections
- Case History 2: Testing after delivery
- Conclusions



Conclusions: Parallel Inspections

- Parallel inspections appear to be a useful tool for estimating the total number of defects in a sub-system or system with the following caveats:
 - Teams should be reasonably balanced in experience
 - Trimming outliers controlled non-independence effects well in the experiments where this was possible.

Conclusions:

Testing after Delivery

- Provided updates can be delivered quickly and efficiently by Internet download, the defect density the customer sees can be reduced by around 50% if testing continues after delivery.
- Customers are more sensitive to simple GUI defects than important server defects so adjust your test priorities accordingly.

Reference site



For more information, downloadable papers and software, see:-

<http://www.leshatton.org/>

l.hatton@kingston.ac.uk