“The role of the scientific method in improving software testing”

by

Les Hatton

Professor of Forensic Software Engineering,
CISM, Kingston University, UK
l.hatton@kingston.ac.uk

Version 1.1: 28/Oct/2008

This presentation is available at http://www.leshatton.org/

A theme ...

“Good tests kill flawed theories; we remain alive to guess again”

“Science must begin with myths, and with the criticism of myths”

Karl Popper 1902-1994

Overview

- The transcendence of fashion over engineering
- The role of forensic engineering
- Some systems should not be tested

Fashion ➔ confusion

In the final year at Kingston 2008, third year projects used

- C, C#, C++, Java, Perl, PHP, MySQL, XML, HTML, XHTML, VB.Net on XP, Mac OS X, Linux, Vista with Eclipse, Netbeans, Ant, JWSDP, Glassfish, Linden script, DreamWeaver, Flash, Developer Studio.Net and a few others I can't recall.

This is to satisfy the *needs of industry* as if they had any idea what they wanted.

Fashion→ poor statistical reasoning ...

- Very few papers attempt to establish significance for their results using standard methods
- A typical result might read *A > B therefore A is (better / worse – substitute as appropriate) than B.*

… leads to poor control and planning …

In the USA …

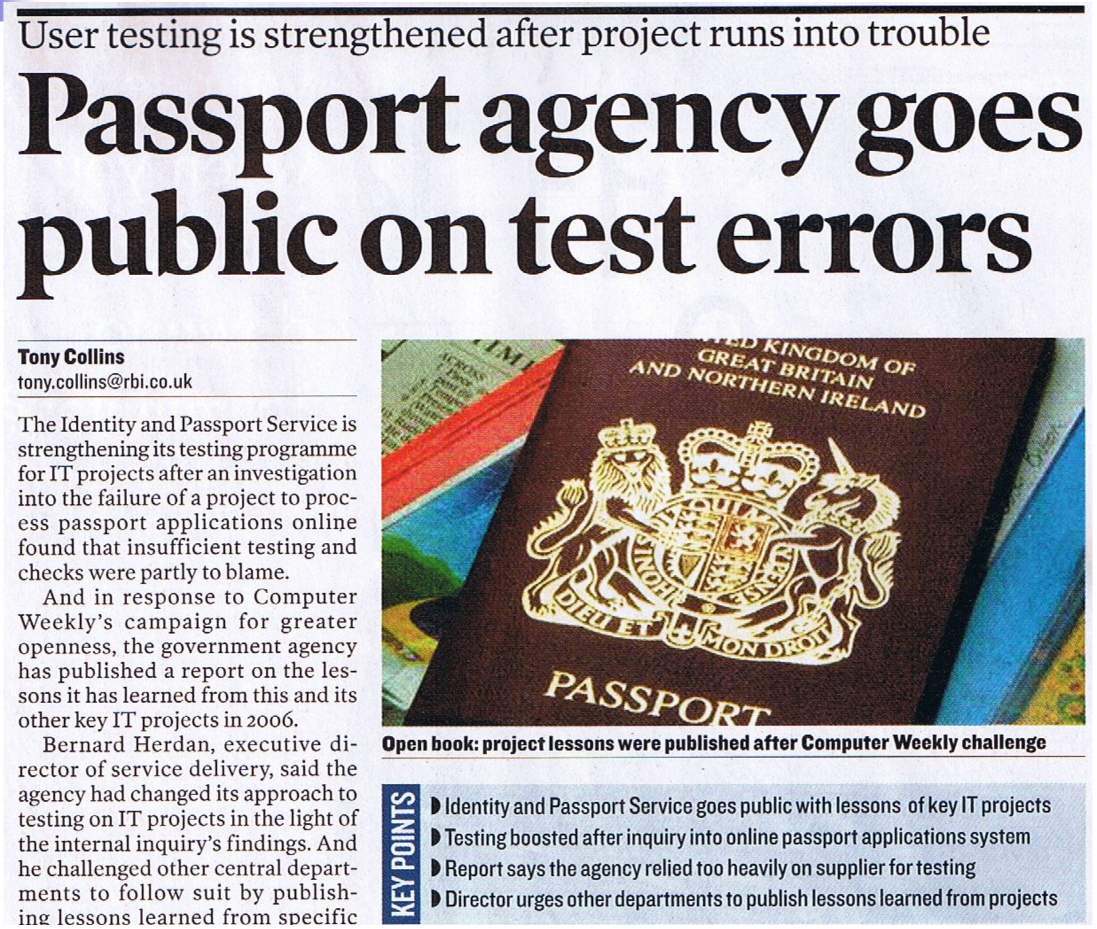
- 01/08/2008 US Office of Management and Budget have identified approximately 413 Government projects totalling at least $25 billion which are poorly planned, poorly performing or both.
  - http://blogs.spectrum.ieee.org/riskfactor/2008/08/

... and ...

A small selection from the UK:-

- 25/07/2008 BBC Radio Humberside system failure delays radio transmissions.
- 31/07/2006 Software failure hits 80 National Health Service Trusts
- 20/08/2006 Failure in software caused Western England homes to run out of water.
- 20/10/2005 Another Post Office systems failure delayed post
- 03/05/2005 Systems failure delayed ambulances in Western England
- 12/04/2005 Child Support Agency in crisis because of problems with new system
- 11/03/2005 Banking failures invalidate PIN cards
- 2000 (twice), 2002 (twice), 2004 and 2005. Air traffic control systems failures in South East England leading to many delays. The 2004 one took the whole country out.
- 22/12/2004 Post Office systems delayed pension payments
- 23/08/2004 Benefit system (again)
- 06/09/2003 Software failure disrupts British Airways flights world-wide
- 17/10/2001 More than 60% of records on Police Criminal Record Check found to be inaccurate
- 20/03/2001 GBP77m immigration service computer system scrapped
- 25/01/1999. Benefit system chaos due to software failure.

... too little, too late ...



User testing is strengthened after project runs into trouble

Passport agency goes public on test errors

Tony Collins
tony.collins@rbi.co.uk

The Identity and Passport Service is strengthening its testing programme for IT projects after an investigation into the failure of a project to process passport applications online found that insufficient testing and checks were partly to blame.

And in response to Computer Weekly's campaign for greater openness, the government agency has published a report on the lessons it has learned from this and its other key IT projects in 2006.

Bernard Herdan, executive director of service delivery, said the agency had changed its approach to testing on IT projects in the light of the internal inquiry's findings. And he challenged other central departments to follow suit by publishing lessons learned from specific

Open book: project lessons were published after Computer Weekly challenge

KEY POINTS
- Identity and Passport Service goes public with lessons of key IT projects
- Testing boosted after inquiry into online passport applications system
- Report says the agency relied too heavily on supplier for testing
- Director urges other departments to publish lessons learned from projects

Computer Weekly

17-Jan-2007

... and it goes on ...

11/08/2008

- Failures in new NHS computer system have meant hundreds of suspected cancer sufferers in London have their operations cancelled.
- People in contact with MRSA could not be contacted
- Many appointments lost.

http://www.bbc.co.uk/1/hi/england/7555077.htm

… and on …

Banks and payrolls …

- 08/07/2008. Westpac in Australia issued all payroll and direct debits twice.

  http://blogs.spectrum.ieee.org/riskfactor/2008/07/westpac_bank_glitch_causes_pro.html

- 11/08/2008. Swansea Council in Wales had to abandon a GBP 819,000 project with CAP Gemini after it increased to GBP 8,000,000 after continual problems.

  http://blogs.spectrum.ieee.org/riskfactor/2008/08/

... and is spreading to security ...

Event	# occurrences
Received messages	47,267
Rejected as non-compliant	- 32,501
Silently discarded as junk	- 13,722
Content filtered spam/scam	- 924
Trojans	- 22
Delivered (to 6 domains)	98 (0.2%)
Root dictionary attacks	160

Overview

- The transcendence of fashion over engineering
- <u>The role of forensic engineering</u>
- Some systems should not be tested

Forensic engineering

To improve a process, you must

- Define a measurable criterion for improvement
- Analyse the measurements regularly to understand what process changes will lead to improvement

For software testing, this means

- We have to have good objective measures. Since a successful test finds defects, *released defects* is an ideal candidate.
- We have to analyse all defects to link them to deficiencies in the test process, if any.

How good is good ?

A useful criterion

- Define a defect as a fault that has failed
- Define an executable line of code (XLOC) as any line of code which generates an executable statement
- Define asymptotic defect density as the *upper bound* of the total number of defects ever found in the product's entire life-cycle divided by the lines of code.

If your asymptotic defect density is < 1 defect per KXLOC (thousand executable lines of code), you are doing about as well as has ever been achieved.

Looking for patterns

- Where do we start ?
  - Aristotleans v. Babylonians: the role of measurement
  - Some examples
- Complicating factors
- Measurements and how to test them

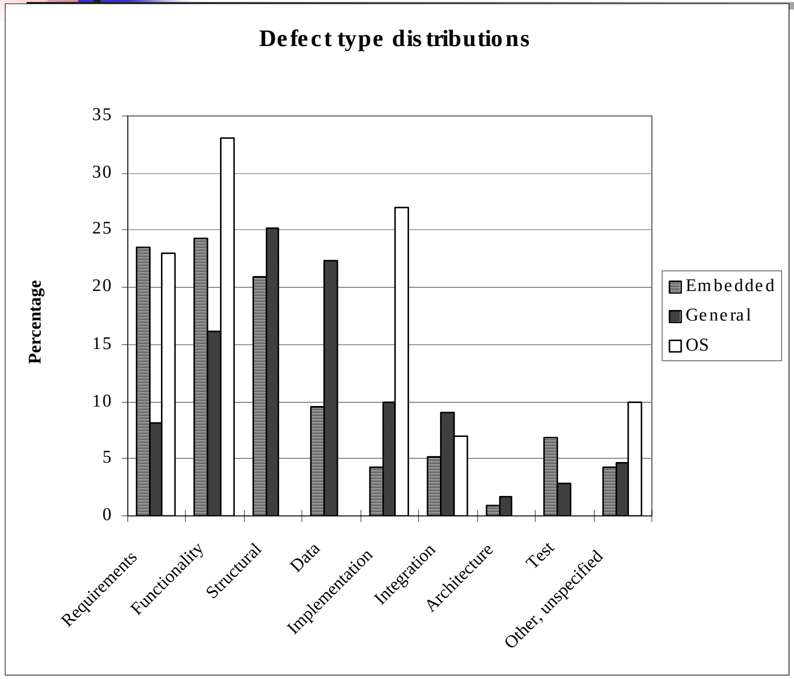
Aristotleans v. Babylonians

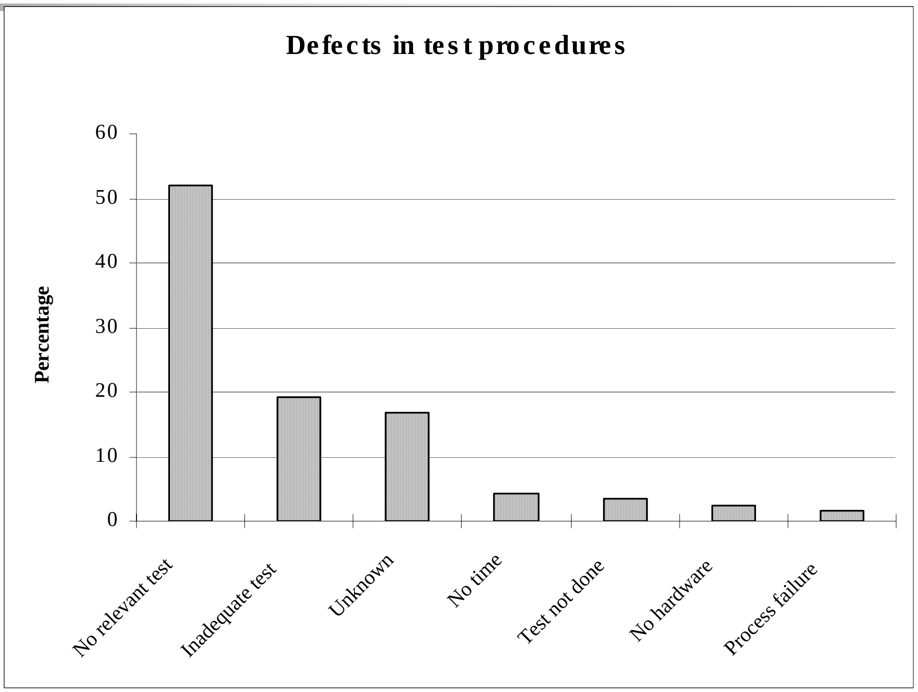
- Aristotleans ...
  - Decide everything by deep thought
- Babylonians ...
  - Learn the hard way by sticking their fingers in light sockets.

Development is an Aristotlean activity …

… Testing is a Babylonian activity

Typical defect profiles





Vinter and Poulson (1996)

OS supplier (2001)

Measure your own !

Extracting patterns

- Where do we start ?
- Complicating factors
  - We should test "systems" as well as software
  - Spreadsheets
  - The underlying architecture may not be very reliable
- Measurements and how to assess them

We test "systems" as well as software

- Systems inconveniently include human users
- Testing involves testing the system as a whole
  - This may include a mixture of computerisation and additional, poorly documented, time varying and somewhat erratic human behaviour which we attempt to assess with usability testing.

The system you test is often not the system which is used !

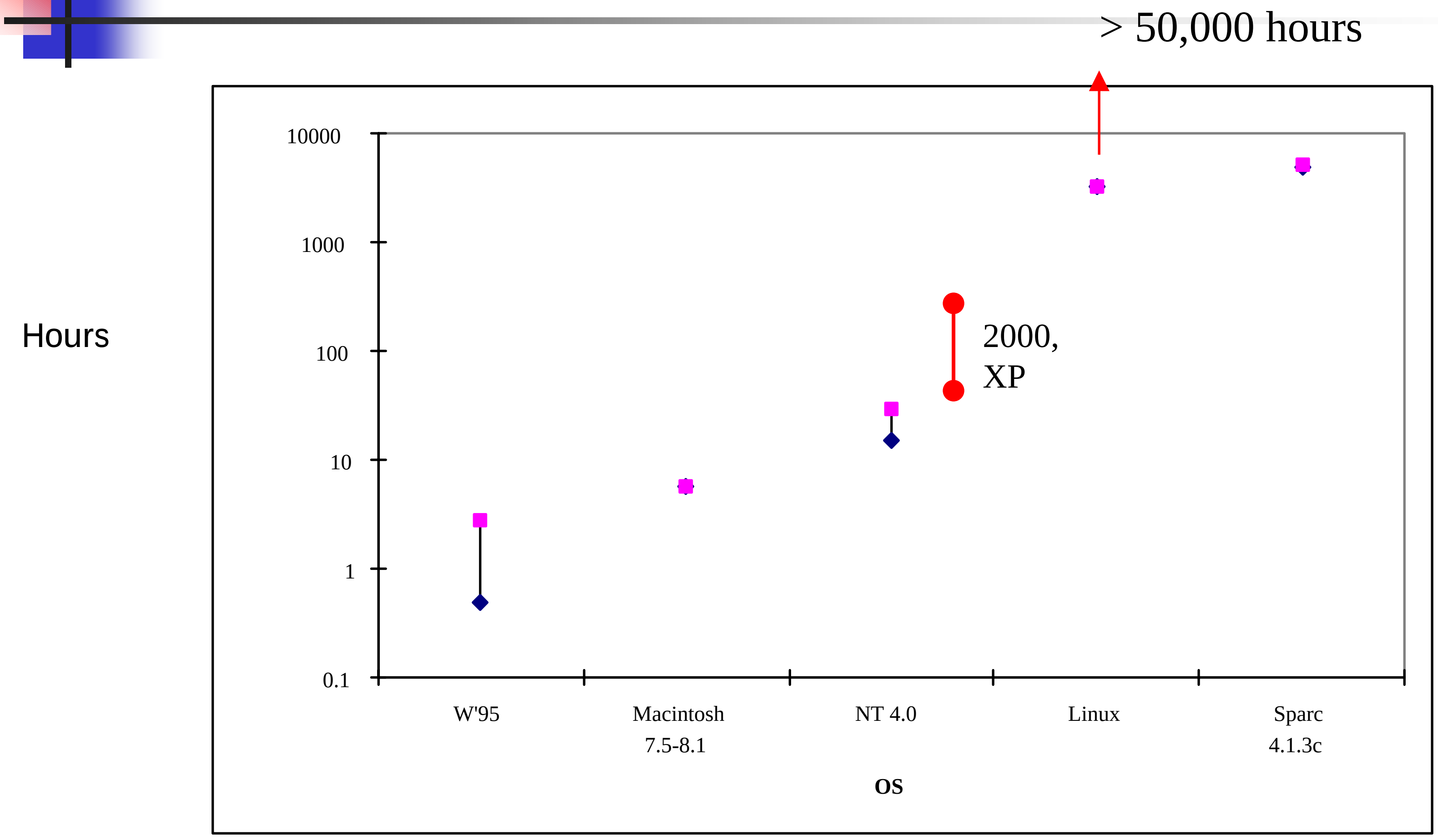
British Gas automated gas meter reading, August 2006

- Note the following
  - Gas Bill based on two estimates:
    - Human sub-system – They now require me to enter my own reading on their telephone entry system
    - Software sub-system – a telephone entry system attached to their billing database
  - My attempts ...
    - Enter my account number.
      - Accepted *without* repeating to check
    - Enter my gas bill reading
      - Repeats to verify and then says “does not agree with our records”
    - Enter same gas bill reading
      - Repeats to verify and accepts

Spreadsheets

- One of the great liberating applications of the 80s and 90s but one of the major headaches of the 21st century ...
  - Weird arithmetic
    - -x^2+1 != 1-x^2 (Allan Stevens 2005)
    - (4/3-1)*3-1 != ((4/3-1)*3-1) If you don't believe me
  - People keep data in them instead of in databases
    - This a major fly in the ointment in most companies because people cannot exchange data.
  - They are hard to test and consequently full of defects
    - 90% of all spreadsheets had errors which led to more than 5% error in the results. Ray Panko (University of Hawaii)
  - They are even harder to search for failure patterns

OS Reliability



> 400 years between failures reported by some users for Linux in 2006

Mean Time Between Failures of various operating systems

OS Reliability

24.5 million XP crashes per day

http://www.pcmag.com/article2/0,4149,1210067,00.asp

5% of Windows Computers crash more than twice a day

http://www.nytimes.com/2003/07/25/technology/25SOFT.html

IT / Technical Support • Networking
Support for International Students

Extracting patterns

- Where do we start ?
- Complicating factors
- Measurements and how to assess them
  - How not to present a result
  - How to present a result

Case History 1 – How NOT to present numerical results

The proportion of defects found by external users in this case history of a client server architecture is as follows, (Hatton (2007), IEEE Computer):-

Component	Proportion of defects found externally	Proportion of defects found internally
GUI Client	*57.2%*	42.8%
Computation Server	*39.1%*	61.9%

Tentative conclusion:- *External users are more sensitive to defects in GUI clients than in computational servers.*

No, we cannot say this reliably !

Case History 1 – doing it properly

Use the z-test for proportions and assume as a null hypothesis that the distribution of defects found externally by users in the GUI client (p_c)and the computation server (p_s) are in the same proportion.

Then ...

$$z = \frac{p_s - p_c}{\sqrt{\hat{p}\hat{q}\left\{\frac{1}{n_1} + \frac{1}{n_2}\right\}}} \sim N(0,1)$$

This gives z = -0.84 which is NOT significant.

We cannot reject the null hypothesis and *we cannot infer any such pattern*

Case History 1 – What can we infer ?

Be very careful to test results for significance before using them !

Sometimes, very useful and highly significant patterns emerge ...

Continued testing after delivery reduces the defect density the user sees by about half, providing we can update them regularly.

This result is statistically highly significant.

Case History 2 – Another example where intuition fails

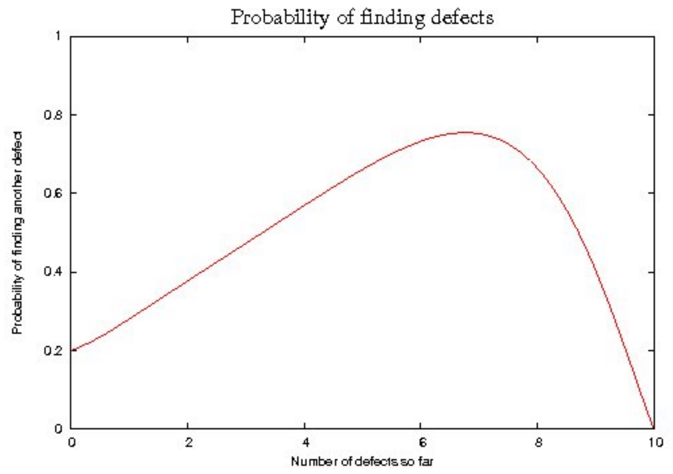
Code inspections are very widely believed to rely on checklists for their effectiveness. In a study of 107 teams, Hatton IEEE Software (2008) showed:-

Experiment phase	Mean defects found using checklists	Mean defects found without checklists
Phase 1 (2005) – 70 teams	*13.00*	11.09
Phase 2 (2006) – 37 teams	*7.18*	5.72

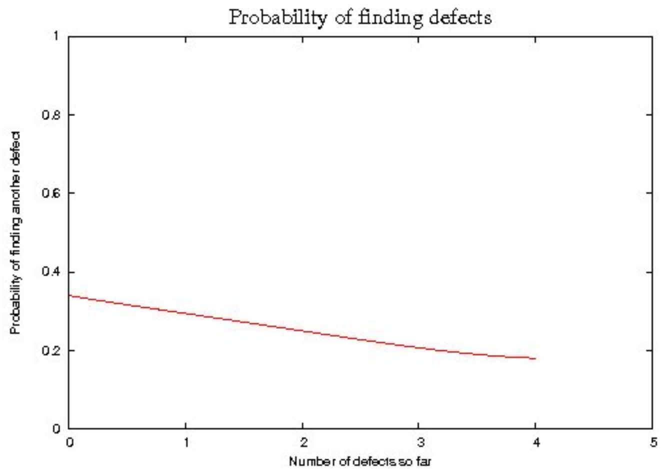
Conclusion:- *The differences are **not** statistically significant individually or collectively*

Some more useful empirical results

- Defects cluster when systems are in quasi-equilibrium (i.e. relatively immature) but not apparently when in equilibrium, (highly significant) (Hatton and Hopkins 2008).



NAG Fortran



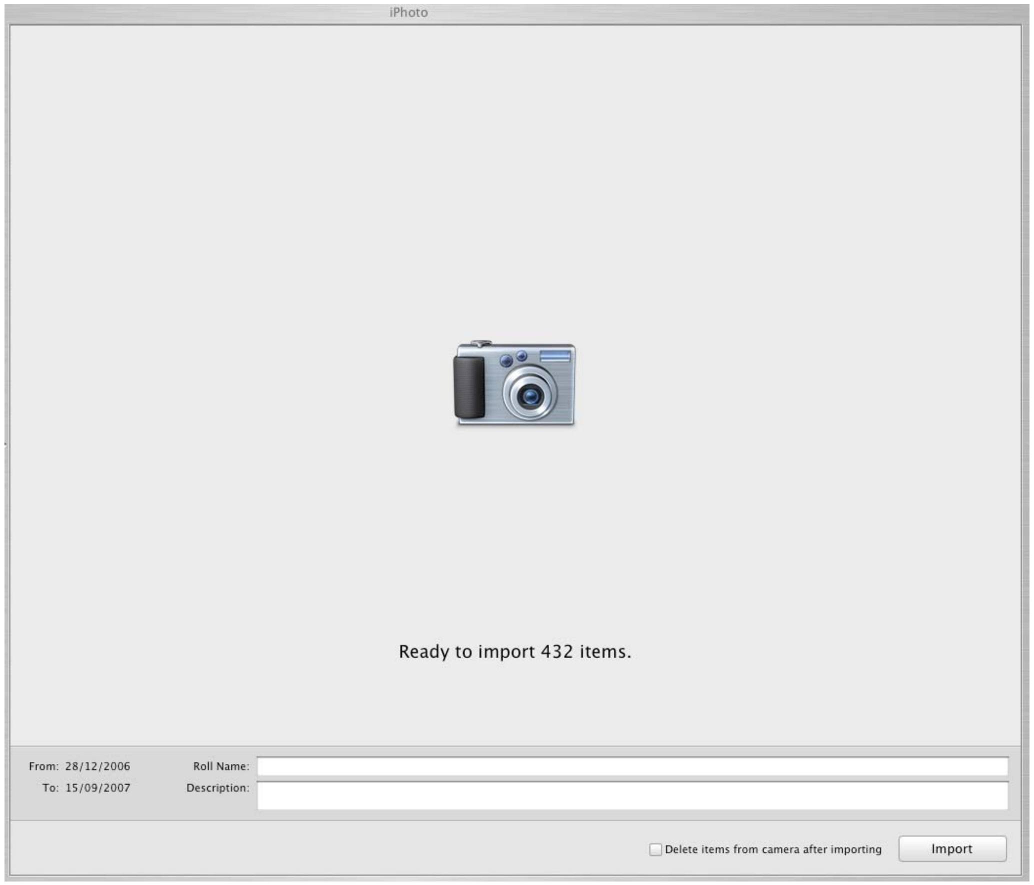
NAG C

Overview

- The transcendence of fashion over engineering
- The role of forensic engineering
- <u>Some systems should not be tested</u>

Testers need to remind developers what humans look like

iPhoto ™, ready for action



Ready to import 432 items.

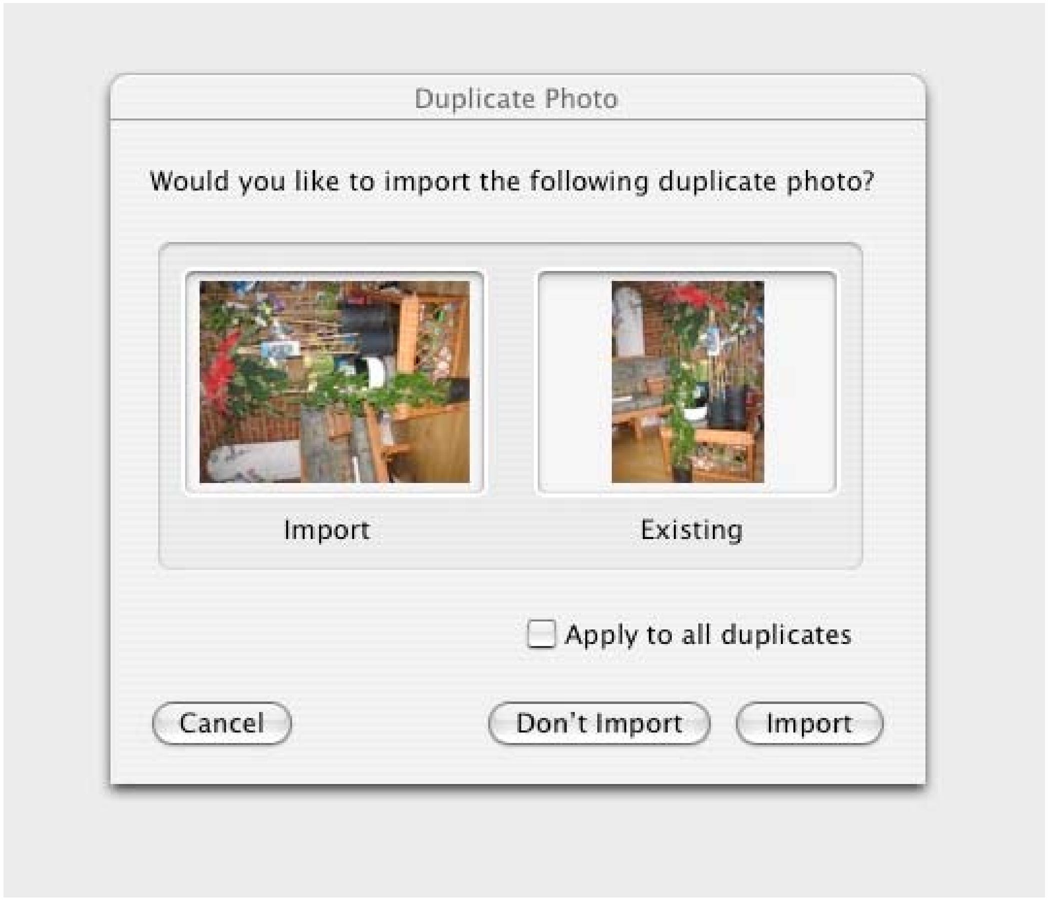
From: 28/12/2006
To: 15/09/2007

Roll Name:
Description:

Testers need to remind developers what humans look like

What does this mean ? Note the lack of grouping and causality.

You could test this against requirements but should you ?

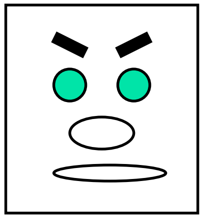


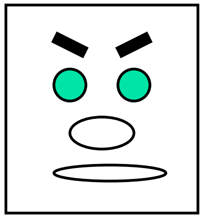
Would you like to import the following duplicate photo?

Testers need to remind developers what humans look like

This photo has already been imported to your computer

Be explicit …



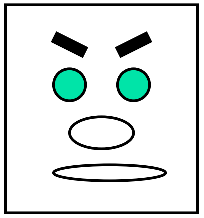


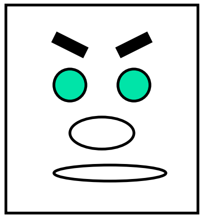
Would you …
• Like to import it again because you don’t trust me
• Like to skip it because you do trust me
• Like to import this and any other duplicates so I don’t have to ask again
• Like to skip this and any other duplicates so I don’t have to ask again
• Like to forget the whole thing and maybe practise the trombone

Testers need to remind developers what humans look like

This photo has already been imported to your computer

Or be causal





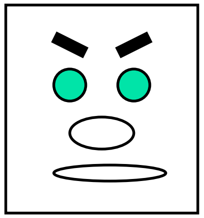
Would you …
- **Like to import it again because you don't trust me**
- **Like to skip it because you do trust me**
- **Like to forget the whole thing and maybe practise the trombone**

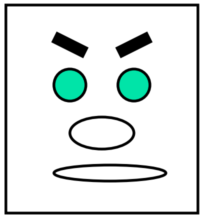
Followed by …

- **Do you want to (skip | accept) subsequent duplicates ?**
- **Do you want me to carry on asking ?**

Testers need to remind developers what humans look like

Would you like to import the following duplicate photo ?





Or lay it out properly …

- **Do not import**
- **Import**

- **Treat all subsequent duplicates the same way**

- **Forget the whole thing and maybe play the trombone**

Lessons for Testers

Keep careful defect data

Analyse it for failure patterns which have statistical significance

Use ONLY statistically significant results to improve your tests and your resource estimation. Be careful ! Differences between individuals are much larger than differences in technologies

Failure patterns contain vital information because they reflect the user's experience

Don't bother testing poorly laid out and non-causal interfaces – report your experiences back to the developer.

For more information and freely downloadable papers see:-

http://www.leshatton.org/, thanks.