Google

Testing 2.0: Enter the human

Presented at Yet another Conference (YaC), 1 October 2012
Moscow

Anthony F. Voellm
Google Cloud Security, Performance and Test Manager
voellm@google.com

Google

Testing 2.0 was first written up on the Google Testing Blog

http://googletesting.blogspot.com/2012/08/testing-20.html

Google

The hypothesis:

Today

One size fits all testing

- Do everything
- Do nothing
- Best guess
- Static code analysis

Tomorrow

The right amount of tests

- Skills / Knowledge
- Experience
- State of mind
- Behavior

Google

Overview

- Part 1 - The backdrop
- Part 2 - The big question
- Part 3 - The path forward

The backdrop

Google

All developers are are the same - right?

Google

Chevy Nova for a car is a great name in english. "No va" however in spanish means "no go."

Internationalization

Logical Bugs

BUGS

Reliability

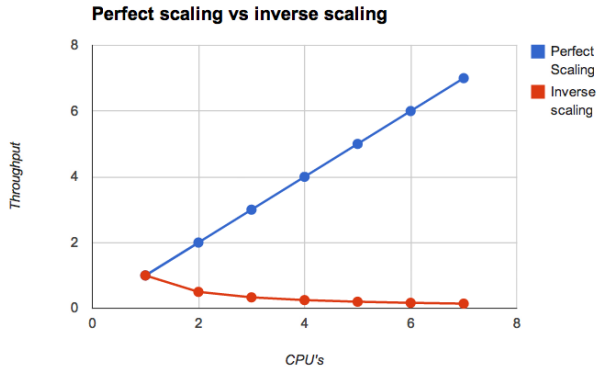
Accessibility bugs

Security bugs

Performance bugs

```
...
int x[10];

x[10] = -1;
...
```



Google

Where do bugs come from?

Humans...

The Google Brain - NYTimes article

One of the neurons in the artificial neural network, trained from still frames from unlabeled YouTube videos, learned to detect cats.

... in the future machines.

Google

How do bugs happen?

Fridays… :)

How do bugs happen?

uTest Blog - "... If you're **tired, angry or frustrated** for instance (like Patriots fans this morning) then you're almost guaranteed to make some careless mistakes. ..."

Google

How do bugs happen?

College mentee of mine -

"**Cut and paste** is wicked."

How do bugs happen?

"How do fixes become bugs" Paper in 2011 - "...the **bug-fixing process** can also introduce errors... Developers and reviewers for incorrect fixes usually **do not have enough knowledge...**"

How do bugs happen?

NIST Study - "Software is error-ridden in part because of its **growing complexity.** The size of software products is no longer measured in thousands of lines of code, but in millions. Software developers already spend approximately 80 percent of development costs on identifying and correcting defects, and yet few products of any type other than software are shipped with such high levels of errors."

Speaking of complexity...

Finding: Based on our samples, concurrency bugs are the most difficult (39%) to fix right. Among concurrency and memory bugs which were fixed incorrectly, the four most observed bug types are: data race, deadlock, buffer overflow and memory leak.

Implication: Developers and testers should be more cautious when fixing concurrency bugs. The allocation of fixing and testing resources could consider the types of bugs to be fixed.

From - "How do fixes become bugs" paper

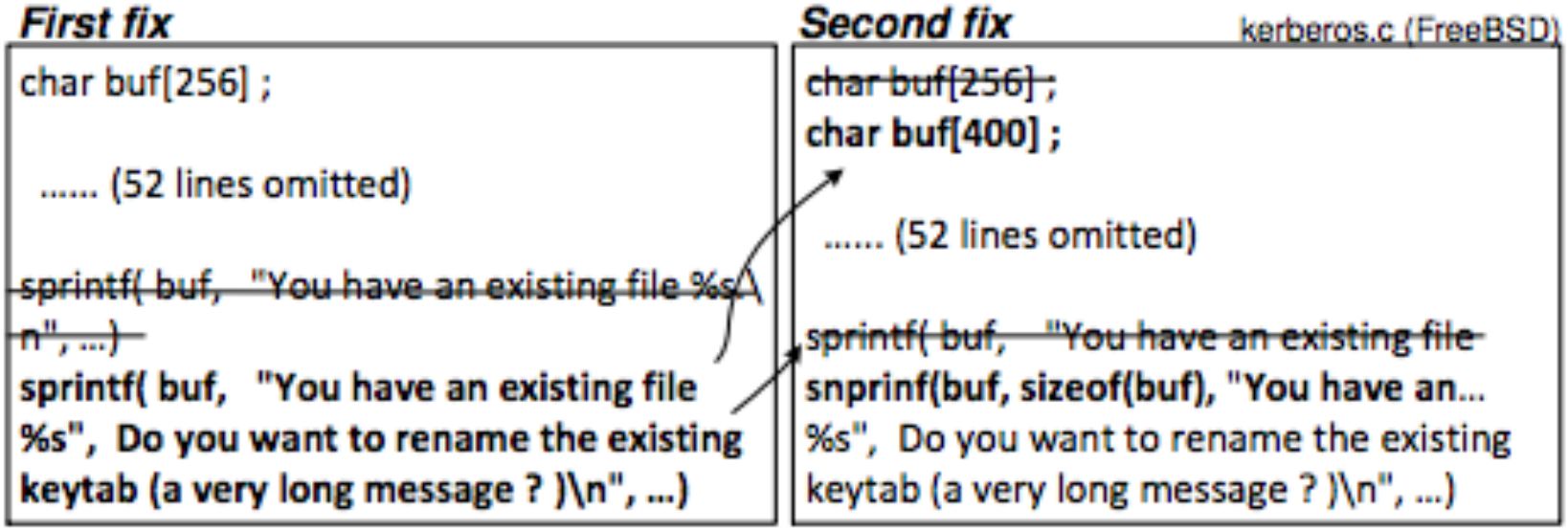
Google

The cost of bugs

- **Time**
  - 25%+ of developers time is fixing bugs
  - A 1 line fix takes 1+ hours of testing is common
- **Money**
  - ~$60 BIllion (9 zero's) to the US economy each year in 2002!
- **Reputation**
  - 10% Error rate on critical security fixes.

Google

Reputation … this might be the most important.



Years to develop and it only takes minutes to destroy.

The big question

One size fits all

With all the evidence that humans are the root cause of bugs *and* we all have different levels of skill.... Why do we all test the same?

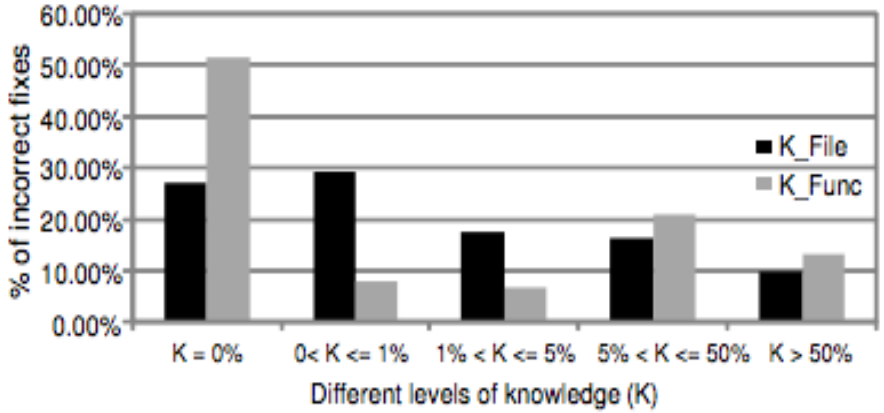


Figure 13: The distribution of incorrect fixes in different knowledge scales.

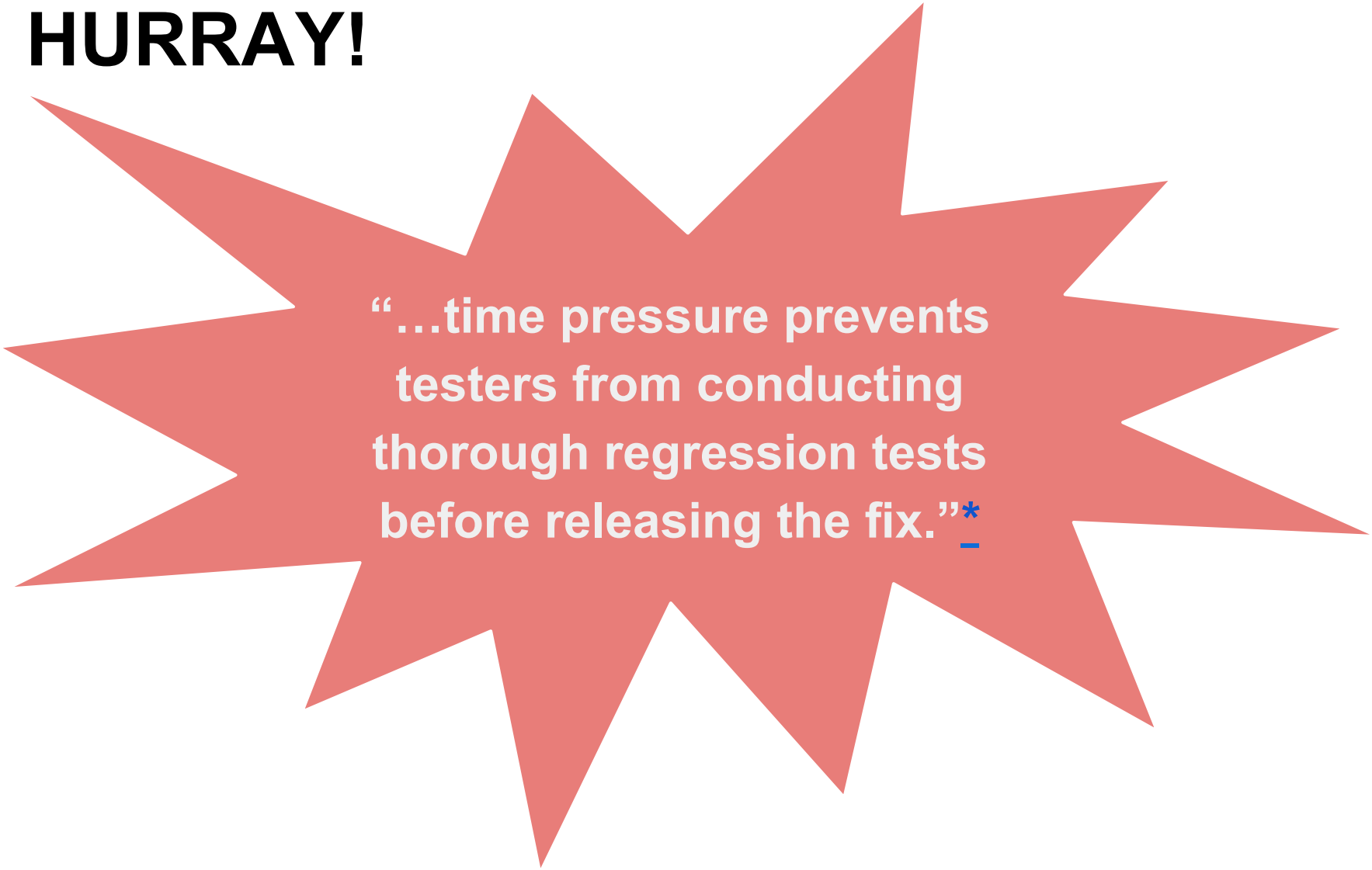
From - "How do fixes become bugs" paper

Types of testing

- All unit tests (15 minutes or less should be the target)
  - The most basic test with all layers stripped away.
- All Integration / System Tests (1 hour or less)
  - Uses multiple features together.
- All Performance Tests (8 hours or less)
  - Micro-benchmarks (fio, iperf, ...)
  - Industry benchmarks (SpecCPU, TPC-C, Hibernate, ...)
- All Reliability tests (days)
  - Longhaul
  - Leak detection tools
- All Security tests (weeks)
  - Smart Fuzzers
  - Static code analyzers like Coverity

Google

HURRAY!



Google

How do we choose what to run?

- Today:
  - Run everything
  - Run nothing
  - Selectively run based on complex change to test associations
  - Best guess based on developer caution

Google

How do we choose what to run?

- Tomorrow:
  - Run based on developer skill
  - Run based familiarity with code base
  - Run based on the complexity of the code
  - Run based on the type of bug being fixed
  - Run based on behavioral analysis of the code

Google

How to measure behavior and skill

VS

Code .....
Code .....
Code .....
Code .....
Code .....
Code .....
Code .....
Code .....
Code .....
Code .....
Code .....
Code .....
Code .....
Code .....

Freshman

Code .....
Code .....
Code .....
Code .....
Code .....
Code .....
Code .....
Code .....
Code .....
Code .....
Code .....
Code .....

Sophomore

Code .....
Code .....
Code .....
Code .....
Code .....
Code .....
Code .....
Code .....
Code .....
Code .....
Code .....

Junior

Code .....
Code .....
Code .....
Code .....
Code .....
Code .....
Code .....
Code .....
Code .....

Senior

Code .....

Guru

Measures

- What is the frequency of check ins by the developer?
- How often has this developer checked in a [severe] bug?
- Knowledge of the code reviewer.
- What is the size of the check in?
- Is the time of day of the check in unusable for the developer?

Measures

- Is the checkin in code the developer is "familiar" with?
- Does the developer write units tests?
- What percentage of the check in is covered by tests?
- Peer ranking on how people feel about your level of expertise.

Google

Measures

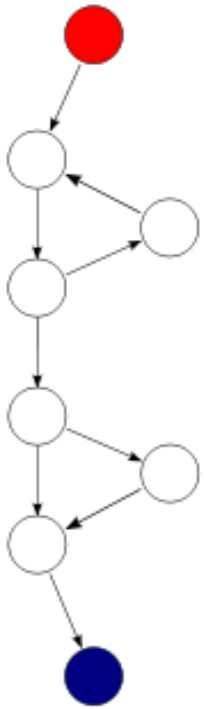
http://people.brandeis.edu/~sekuler/eegERP.html

Measures

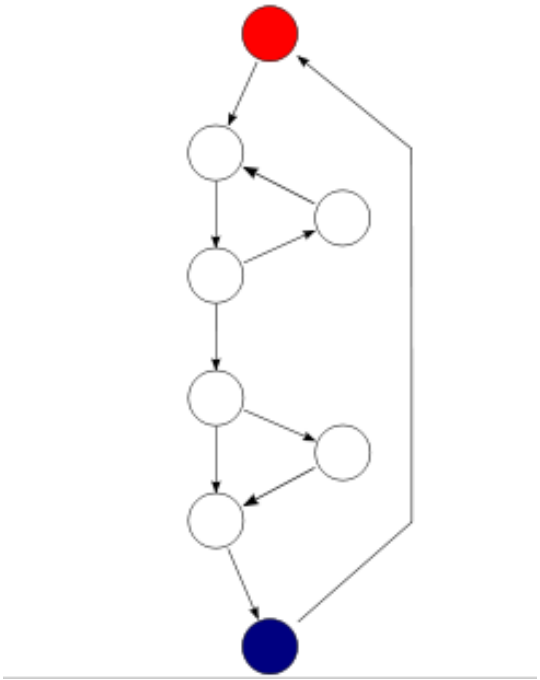
- Block count
- Number of system calls
- How layered is the code?
- Cyclomatic complexity
- ... you get the idea

Google

Cyclomatic complexity



$M = E - N + 2P$



$M = E - N + P$

E = the number of edges of the graph
N = the number of nodes of the graph
P = the number of connected components (exit nodes).

http://en.wikipedia.org/wiki/Cyclomatic_complexity

The path forward

Google

Use the human factor to ship faster...

- Create [AI] models that account for...
  - Experience
  - Knowledge of the code base
  - Code complexity
  - Measures of behavior
  - ... and much more
- Use the models as part of check in...
  - Developers with less risky profiles run less tests
  - Developers with higher risk profile run more tests
- Let automated systems running in parallel be the safety net.

Google

Let productivity soar!

Google

Thank you...

Special thanks to Yandex for inviting me to speak and in particular **Arkady Volozh** for approving and **Yelena Jetpyspayeva** for covering all the details.

Google

End - Questions?

Name: Anthony F. Voellm (aka Tony)
Contact: voellm@google.com
Blog: http://perfguy.blogspot.com
Twitter: @p3rfguy
G+ / FB

Appendix

Google

ABSTRACT:

Enter the human element. As you look at today's test systems, tools, and processes they are designed around the premise that all developers are created equal. Studies have shown developer error rates can vary widely and have a number of root causes - mind set of the developer at the time the code was written, experience level, amount of code in a check in, and much more. This talk is about putting the human element back into testing to drive greater efficiency and faster releases. As part of this talk we will look at Digital Code Forensics, Cyclomatic complexity, and other measures of code quality and how they might be used to determine what testing is needed. This is the bleeding edge... this is Testing 2.0.

Google

The message:

Not all developers have the same experience or skill level, and we can use this to improve the speed of development. Speed up the better developers, and slow down the less precise. We dont need a one size fits all policy, however we do need to base the decisions on data.

Google

References

- http://en.wikipedia.org/wiki/Software_bug#cite_note-1
- http://www.cs.unm.edu/~forrest/classes/readings/HowDoFixesBecomeBugs.pdf
- http://blog.utest.com/the-software-testing-mindset/2012/02/
- http://www.cse.buffalo.edu/~mikeb/Billions.pdf
- http://software-testing-zone.blogspot.com/2008/12/why-are-bugsdefects-in-software.html
- http://www.itbusinessedge.com/cm/community/features/guestopinions/blog/battling-software-defects-one-developer-at-a-time/?cs=39611
- http://istqbexamcertification.com/what-is-the-psychology-of-testing/
- http://ubuntuforums.org/archive/index.php/t-1582847.html
- http://sqa.stackexchange.com/questions/545/how-does-a-testers-perspective-toward-software-differ-from-a-developers
- http://software-testing-zone.blogspot.com/2009/04/software-testing-diplomacy-deal.html