



Ten things we know to be true

... about Testing

Mark Striebeck
Engineering Manager

Ten things we know to be true



1. Focus on the user and all else will follow.
2. It's best to do one thing really, really well.
3. Fast is better than slow.
4. Democracy on the web works.
5. You don't need to be at your desk to need an answer.
6. You can make money without doing evil.
7. There is always more information out there.
8. The need for information crosses all borders.
9. You can be serious without a suit.

We first wrote these “10 things” several years ago. From time to time we revisit this list to see if it still holds true. We hope it does-and you can hold us to that.
(September 2009)

Development Culture at Google

Google

The need for information crosses all borders



- One, open source code repository
 - Code Reviews
 - Open Tools
 - No binary dependencies (build everything from HEAD)
-

Democracy on the web works



- Very little top-down management and standards
- Delegate responsibility downwards – (almost) all of it!

How to drive change?

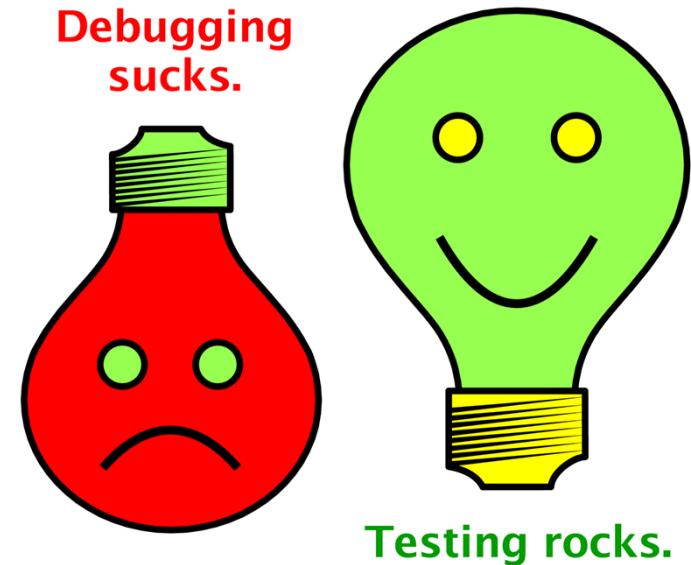
Democracy on the web works



- Very little top-down management and standards
- Delegate responsibility downwards – (almost) all of it!

How to drive change?

- Grouplets (20% time)
- Sponsored/Supported by upper management



Where to start?

Google

2006



- Google grows fast (product, people, source code, dependencies)
 - QA is doing mostly manual testing
 - Engineers can write automated tests, but adoption isn't great, coverage is spotty
 - But integration/release problems are showing up
-

It's better to do one thing really, really well



- What to focus on (end-to-end testing, unit testing, latency testing, UI testing ...)?
- Decided to focus on
 - What we feel passionate about
 - Where we have experience
 - What can make a difference
 - What we can affect

Unit / Developer testing

You can be serious without a suit



First, focus on getting the message out

- Fixits
- Tech Talks (M. Feathers, J. Kerievsky, G. Meszaros, E. Evans...)

and ...

Fixit Participants Worldwide - Engineering Fixit - Developer Testing: March 8, 2007 - Mozilla

https://testing-fixit.corp.google.com/?act=stat

Testing Grouplet | What's a Fixit? | What's a Grouplet? | Home | Contacts | Help

Google Engineering Fixit - Developer Testing: March 8, 2007

The Testing Grouplet will sponsor an engineering-wide Tech Talk. Objective: *More and better developer productivity*

This Testing Fixit follows the Testapalooza conference, to take presentations and workshops.

Fixit Participants Worldwide

535 participants in 29 locations:



Atlanta	2
Bangalore	1
Beijing	16
Belo Horizonte	15
Boulder	6
Cambridge	8
Dublin	6
Groebenzell	1
Haifa	1
Hyderabad	1
Irvine	14
Kirkland	32
Krakow	3
London	10
Montreal	3
Moscow	2
Mountain View	283
New York	38
Phonix	11



You can be serious without a suit



... and of course: Testing on the Toilet



Testing on the Toilet

Time is Random

Episode 30

When a method's inputs can't be clearly identified, how can it be well tested? Consider this method in Java:

```
/** Return a date object representing the start of the next minute from now */
public Date nextMinuteFromNow() {
    long nowAsMillis = System.currentTimeMillis();
    Date then = new Date(nowAsMillis + 60000);
    then.setSeconds(0);
    then.setMilliseconds(0);
    return then;
}
```

There are two barriers to effectively testing this method:

1. There is no easy way to test corner cases; you're at the mercy of the system clock to supply input conditions.
2. When `nextMinuteFromNow()` returns, the time has changed. This means the test will not be an assertion, it will be a guess, and may generate low-frequency, hard-to-reproduce failures... **flakiness!** Class loading and garbage collection pauses, for example, can influence this.

Is `System.currentTimeMillis()` starting to look a bit like a random number provider? It should! The current time is yet another source of *non-determinism*; the results of `nextMinuteFromNow()` cannot be easily determined from its inputs. Fortunately, this is easy to solve: make the *current time* an input parameter which you can control.

```
public Date minuteAfter(Date now) {
    Date then = new Date(now.getTime() + 60000);
    then.setSeconds(0);
    then.setMilliseconds(0);
    return then;
}

// Retain original functionality
@Deprecated public Date nextMinuteFromNow() {
    return minuteAfter(new Date(System.currentTimeMillis()));
}
```

Writing tests for `minuteAfter()` is a much easier task than writing tests for `nextMinuteFromNow()`:

```
public void testMinuteAfter () {
    Date now = stringToDate("2012-12-22 11:59:59.999PM");
    Date then = minuteAfter(now);
    assertEquals("2012-12-23 12:00:00.000AM", dateToString(then));
}
```

This is just one way to solve this problem. Dependency Injection and mutable Singletons can also be used.

[Deterministic algorithm](#)
Discusses the nature of deterministic algorithms, what introduces non-determinism, and times when determinism is undesirable.
http://en.wikipedia.org/wiki/Deterministic_algorithm

[Testing Legacy Code Codelab](#)
Who has time to read an entire book anymore? Now you can get all the benefits of the Feathers book in Codelab form. A bargain at twice the price.
<http://wiki/Codelab/TestingLegacyCode>

Info, feedback, and discussion: <http://tott>



You can be serious without a suit



... and of course: Testing on the Toilet

Forbes, January 10, 2007

Google is No. 1: Search and enjoy The people are brilliant. The perks are epic. But can Google's founders build a culture that doesn't depend on the stock price?

I'm sitting on a heated toilet in my pajamas. I'm in engineering building 40 at Google on "pajama day," and directly in front of me, attached to the inside door of the toilet stall, is a one-page flier, printed on plain white paper, titled "Testing on the Toilet, Episode 21." The document, which is designed to prod the brains of engineers who test software code, explores such subjects as "lode coverage" and reminds engineers that even biobreaks need not interrupt their work.

Presuming that ones's stay here isn't sufficient to process that lesson, the sheet provides a link to two internal Web sites, <http://tott/> (for Testing on the Toilet) and <http://botw/> (Bug of the Week). Not being a software engineer, I understand little of what I'm reading. Yet it reminds me of the first two sentences of the now famous founders' letter Page and Brin distributed to prospective Google shareholders before the company's 2004 IPO: "Google is not a conventional company. We do not intend to become one." Mission accomplished.

San Jose Mercury News, October 22, 2007

"Tech Notebook: Google embraces workers in unlikely places"
By Vindu Goel, Scott Duke Harris and Dean Takahashi

We were at Google recently, working on a story, and after consuming a little too much of that free Google water, we felt the need to visit the restroom. We were surprised to find that Googlers can't escape the Mountain View company's warm embrace even in the loo. Not only are the toilet seats heated, but the company also strategically posts tips on testing software code. Engineers can ponder them while they're, ahem, doing their business. ...
"We'll be putting episodes on this blog on a regular basis and providing PDFs so you can print them out and put them up in your own bathrooms, hallways, kitchens, moon bases, secret underground fortresses, billionaire founders' Priuses, wherever," the company said.

You can be serious without a suit



... and of course: Testing on the Toilet

Forbes, January 10, 2007

Google is No. 1: Search and enjoy The people are brilliant. The perks are epic. But can Google's founders build a culture that doesn't depend on the stock price?

I'm sitting on a heated toilet in my pajamas. I'm in engineering building 40 at Google on "pajama day," and directly in front of me, attached to the inside door of the toilet stall, is a one-page flier, printed on plain white paper, titled "Testing on the Toilet, Episode 21." The document, which is designed to prod the brains of engineers who test software code, "lode coverage" and reminds engineers not interrupt their work.

Presuming that ones's stay here isn't s lesson, the sheet provides a link to tw sites, <http://tott/> (for Testing on the T of the Week). Not being a software e of what I'm reading. Yet it reminds m of the now famous founders' letter Pa prospective Google shareholders bef "Google is not a conventional compa become one." Mission accomplished.

San Jose Mercury News, October 22, 2007

"Tech Notebook: Google embraces workers in unlikely places"
By Vindu Goel, Scott Duke Harrisand Dean Takahashi

We were at Google recently, working on a story, and after consuming a little too much of that free Google water, we felt the need to visit the restroom. We were surprised to find that Googlers can't escape the Mountain View company's warm embrace even in the loo. Not only are the toilet seats heated, but the company also strategically posts tips on testing software code. Engineers can ponder them while they're, ahem, doing their business. ...

Washington Post, October 21, 2006

"Building a 'Googley' Workforce Corporate Culture Breeds Innovation"
By Sara Kehaulani Goo Washington Post Staff Writer Saturday, October 21, 2006; Page D01

MOUNTAIN VIEW, Calif. -- To understand the corporate culture at Google Inc., take a look at the toilets. Every bathroom stall on the company campus holds a Japanese high-tech commode with a heated seat. If a flush is not enough, a wireless button on the door activates a bidet and drying. Yet even while they are being pampered with high-tech toiletry, Google employees are encouraged to make good use of their downtime: A flier tacked inside each stall bears the title, "Testing on the Toilet, Testing code that uses databases." It features a geek quiz that changes every few weeks and asks technical questions about testing programming code for bugs. The toilets reflect Google's general philosophy of work: Generous, quirky perks keep employees happy and thinking in unconventional ways, helping Google innovate as it rapidly expands into new lines of business.

g PDFs so you can print them
bases, secret underground

Problem solved?

Google

Focus on the user and all else will follow



After all these events and education, engineers knew the value of unit testing

... yet nothing material changed!

“Well, they don't get it!”

Focus on the user and all else will follow



After all these events and education, engineers knew the value of unit testing

... yet nothing material changed!

“Well, they don't get it!” “What did we do wrong?”

Focus on the user and all else will follow



After all these events and education, engineers knew the value of unit testing

... yet nothing material changed!

“Well, they don't get it!” “What did we do wrong?”

“Why did engineers not do what they knew is the right thing to do?”

2 main issues:

- Where to start?
 - Legacy code
-

Focus on the user and all else will follow



Test Certified

- Small steps to get teams started
 1. Setup Tools – get familiar with testing
 2. Learn techniques and technologies that enable your team to write tests for all new code and refactor code accordingly
 3. Improve test suite and get all code under test

4. ...



Focus on the user and all else will follow



Test Mercenaries

- Internal consulting group that embedded itself into teams to
 - Refactor code to make it testable
 - Educate and train team
 - Start writing tests
 - Drive development process changes

Now problem solved?

Google

Fast is better than slow



If it takes 30+ minutes to build and run your tests...
... you just don't bother

Fast is better than slow



If it takes 30+ minutes to build and run your tests...
... you just don't bother

- Parallelized all builds and test runs into the cloud – build+test time went down to minutes
- Company-wide continuous integration system (“Run all affected tests at each change!”)
- Presubmit system that runs all tests *before* submitting

Today:

- 10,000+ developers across 40+ offices around the world
- 20+ changes per minute; 50% code base changes every month
- More than 50 million test cases executed every day

There is always more information out there



Needed a BIG test storage system

- Created an open system with a web UI, an API and a query interface
 - Integrated into all build/test tools
 - Result: EVERY build/test run is logged
-

There is always more information out there



Needed a BIG test storage system

- Created an open system with a web UI, an API and a query interface
- Integrated into all build/test tools
- Result: EVERY build/test run is logged

Today:

- Engineers send URL's to build runs instead of copy/paste build logs
 - Used by many Dashboard and other systems
 - Lots of quality analysis (bug trends, flakiness, code quality...) - per team, but also Google-wide
-

Today

Google

Great just isn't great enough



Could vastly increase release frequency (“push on green”)

Some teams were asked to release less often because other teams/systems can't deal with the high churn!

Main bottleneck: Time to release and to analyze if a release causes issues.

You don't need to be at your desk to need an answer



And now do all this again – for mobile!

Observations

Google



Changing the culture takes a LOOOOOOONG time

You need the right change agents (passionate, knowledgeable, respected...)

Bottom-up is harder than top down – but it sticks!

Don't fall into the “They just don't get it” trap!

Focus on the next important thing!



Changing the culture takes a LOOOOOOONG time

You need the right change agents (passionate, knowledgeable, respected...)

Bottom-up is harder than top down – but it sticks!

Don't fall into the “They just don't get it” trap!

Focus on the next important thing!

... and seriously: Don't take yourself too serious!

Lean...

Google



Is Google a lean company?

What could Google learn from lean?

What could lean learn from Google?
