How Google Tests Software: Organizational Structure

“for Freedom”

BOUN SWE-550 course

Serdar DALGIÇ
serdar@pardus.org.tr
Who is Serdar DALGIÇ?

- Pardus GNU/Linux Developer since 2007.
- TUBITAK BILGEM/UEKAE Employee since 2009 November.
- Turkish Linux Users Society Member since 2005.
- Boğaziçi University SWE student since 2009.

- How Google Tests Software - A Brief Interlude (22 Feb)
- How Google Tests Software - A Break for Q&A (4 May)

Focus Areas ----> Problem Domain
Organizational Structure

- Not an actual test organization!
- Tests within Engineering Productivity (Eng. Prod. is a F.A.)
- Eng. Prod. owns horizontal and vertical eng. disciplines.
- Test is the biggest!
Engineering Productivity

- Eng. Prod. is made of:
  1. Product Team
  2. Services Team
  3. Embedded Engineers
Product Team

• produces internal and open source productivity tools that are consumed by all walks of engineers across the company.

• They build and maintain code analyzers, IDEs, test case mgmt. systems, automated testing tools, build systems, source control systems, code review schedulers, bug databases....

• Tools that increase productivity!
Services Team

• provides expertise to Google product teams on a wide array of topics including tools, documentation, testing, release management, training and so forth.

• Every other F.A. has access to Eng. Prod. expertise.
Embedded Engineers

- effectively loaned out to Google product teams on an as-needed basis.
- Some might sit with the same product teams for years
- Others cycle through teams wherever they are needed most. Google encourages all its engineers to change product teams often to stay fresh, engaged and objective.
So Testers..

- report to Eng. Prod. Mngrs. but identify themselves as a part of the Product Team (like Search, Gmail or Chrome..)

- This separation of project and reporting structures have some advantages and disadvantages.
Advantages of the separation:

✓ To provide a pool for testers to share information
✓ Good ideas spread easily
✓ Whittaker says: "I like this strategy, 'cause it puts developers and testers on equal footing."
  • "..it allows us a many-to-one dev-to-test ratio."
  • "..Developers outnumber testers."
  • "..The better they are at testing the more they outnumber us. Product teams should be proud of a high ratio!"
Disadvantages of the separation:

✗ Testers are external source

✗ Product teams can't place too big a bet on them and must keep their quality house in order. (Yes, that's right: at Google it's the product teams that own quality, not testers.)

- Every developer is expected to do their own testing.
- The job of the tester is to make sure they have the automation infrastructure and enabling processes that support this self reliance.
- Testers enable developers to test.
The Problem - *cliché*

- "Developers can't test!!"
- Google's answer is to split the role:
- They solve this problem by having two types of testing roles at Google to solve two very different testing problems.
To Solve the Problem (*cliché*)

- Engineering roles that enable developers to do testing efficiently and effectively have to exist.

- Roles in which some engineers are responsible for making others more productive.
What are these roles?

- The SWE or Software Engineer
- The SET or Software Engineer in Test
- The TE or Test Engineer
The SWE or Software Eng.

• The Traditional Developer Role:
  – write functional code that ships to users.
  – create design documentation.
  – design data structures and overall architecture.
  – spend the vast majority of their time writing and reviewing code.
  – write a lot of test code for test driven design
  – SWEs own quality for everything they touch whether they wrote it, fixed it or modified it.
The SET or Software Eng. in Test

- Also a developer role except their focus is on testability.
  - review designs and look closely at code quality and risk.
  - refactor code to make it more testable.
  - write unit testing frameworks and automation.

- They are a partner in the SWE code base but are more concerned with increasing quality and test coverage than adding new features or increasing performance.
The TE or Test Eng.

- exact reverse of the SET. It is a role that puts testing first and development second:
  - spend a good deal of their time writing code in the form of automation scripts and code that drives usage scenarios and even mimics a user.
  - organize the testing work of SWEs and SETs,
  - interpret test results and drive test execution, particularly in the late stages of a project as the push toward release intensifies.

- They are the product experts, quality advisers and analyzers of risk.
Interaction btw SWE and SET

- SWEs own features and the quality of those features in isolation.
- SETs write code that allows SWEs to test their features. **Testing is done by SWEs.**
- SETs are there to ensure that features are testable and that the SWEs are actively involved in writing test cases.
- SETs primary focus is on the **developer**.
And then what does TE do?

- **User focused testing** is the job of the Google TE.
- TEs act as a double-check on the diligence of the devs.
  - Any obvious bugs are an indication that early cycle developer testing was inadequate or sloppy.
  - When such bugs are rare, TEs can turn to their primary task of ensuring that the software:
    - runs common user scenarios,
    - is performant and secure,
    - is internationalized and so forth...
Quality
Quality

• QUALITY is not Test!

• While it is true that quality cannot be tested in, it is equally evident that without testing it is impossible to develop anything of quality.

• Quality is achieved by putting development and testing into a blender and mixing them until one is indistinguishable from the other.

• Quality is a development issue, not a testing issue.
Quality

- Google's aim: merge development and testing so that you cannot do one without the other.

- “..Build a little and then test it. Build some more and test some more..”
A. Q. ?

serdar@pardus.org.tr
serdar.dalgic@linux.org.tr
facebook.com/maninsilence

Blog:

http://developer.pardus.org.tr/people/serdar/blog/