Editor in Chief: **Forrest Shull**
Fraunhofer Center for Experimental Software Engineering, fshull@computer.org

A Brave New World of Testing?

An Interview with Google's James Whittaker

Forrest Shull

WHEN NEW TECHNOLOGIES take off, they make those of us in the profession scramble to keep up. Sometimes that translates into getting some training or grabbing a book or two on the latest methodology, tool, or notation. In other cases, it means that the paradigm changes in a truly significant way, and whole sets of job skills can suddenly become more in demand or obsolete.

So far, the increasing pervasiveness of cloud computing seems likely to fall into the second category. Some have argued that cloud computing isn't really new, representing a repackaging of technologies that have been around for some time. My own experience has been that the widespread adoption of this particular paradigm, and the increase in the amount of computing power that can be brought to bear as a result, are creating substantial changes to the usual way of doing business.

> "Bug-finding skills are less important now than they've ever been, because the developers are just so much closer to their users."

In their introduction, the guest editors have compiled a list of questions related to what our future, cloud-intensive world is going to look like—many of which I've heard myself from colleagues in government and commercial positions. The one that I hear most often is this: How should organizations leverage the power of this approach to improve testing and quality assurance of software? To get an answer, I turned to James Whittaker, an engineering director at Google, which has been at the forefront of leveraging the cloud. James is a noted expert and author on software testing, whose team has been managing Google's cloud computing testing. Some excerpts of our conversation:

What is it like right now, looking across cloud computing testing at Google? It sounds like a pretty major undertaking.

It is pretty dynamic. We just had a conference at Google where the theme was "Cloudy with a Chance of Tests." It had a two-pronged meaning: first, that the cloud itself changes testing—testing apps for the cloud is different than testing them for the desktop, for client-server, or even for the Web. Second, the cloud itself is changing test,

IEEE Software Mission Statement: To be the best source of reliable, useful, peer-reviewed information for leading software practitioners—the developers and managers who want to keep up with rapid technology change.

0740-7459/12/$31.00 © 2012 IEEE

ABOUT JAMES WHITTAKER

James Whittaker is an engineering director at Google, where he runs teams that build APIs and produce developer tools. He has been a long-time thought leader in software testing and has authored five books, including the *How to Break Software* series and the recently released *How Google Tests Software* (coauthored with Jason Arbon and Jeff Carollo, Addison-Wesley, 2012).

in the sense that a lot of test cases, test assets, and test features are accessible through the cloud that people don't have to have in-house anymore.

In one of your previous interviews, I came across a statement of yours that has become one of my favorite thought-provoking quotes. You said, "Anyone who says that testing is getting harder is doing it wrong." Could you expand on this a bit?

I do believe that testing is getting easier, and I think if you look at software, the quality is improving. You can't compare 1990s software with modern software and think that we've gotten worse!

Testing has had some impact on that, but really I think testing is getting far easier because of access to assets. You're not writing test cases that run in some local test lab and that have to be specifically configured for the machines in the test lab. You don't need specialized equipment just for running and monitoring the test lab. That's not the way people do it in the cloud.

In the cloud, all the machines automatically work together; there's monitoring software available, and one test case will run anywhere. There's not even a test lab. There's just a section of the datacenter that works for you from a testing point of view. You put a test case there and it runs. And all of the different scheduling software that any datacenter uses to schedule tasks can be used to schedule tests. So, a lot of the stuff that we used to have to write and customize for our test labs, we just don't need anymore.

When I started at Google, we had just one test lab, and now we've even closed that one. Literally all of our test cases are run in remote datacenters.

The other thing the cloud has done is brought us closer to our users. Think of Google Maps: it's really impossible to hire a group of testers to exhaustively test it. It's literally a piece of software of planetary proportions. If there's a bug in my address on Google Maps, I'm likely to be the only one who will find it. But the cloud also enables us to reach out to users who are early adopters to get better and richer bug feedback than we were ever able to do back in the client-server days, when once software got to the field it was very difficult to update and instrument. Now, it's easy to update a datacenter, it's easy to instrument a datacenter. If a customer finds a bug, it's easy for them to tell us about it, and it's easy for us to fix it and push that fix to all our users, by just refreshing a browser.

So the cloud really does change

IEEE Software

STAFF

Lead Editor
Dale C. Strok
dstrok@computer.org

Content Editor
Brian Brannon

Manager, Editorial Services
Jenny Stout

Editors
Camber Agrelius, Dennis Taylor, and Linda World

Publications Coordinator
software@computer.org

Production Editor/Webmaster
Jennie Zhu

Contributors
Alex Torres

Cover Artist
Eero Johannes

Director, Products & Services
Evan Butterfield

Senior Manager, Editorial Services
Lars Jentsch

Senior Business Development Manager
Sandra Brown

Membership Development Manager
Cecelia Huffman

Senior Advertising Coordinator
Marian Anderson
manderson@computer.org

things. It's a different model of development; it's a different model of testing; it's a different model of usage.

Regarding testers and the skill sets that they've traditionally been applying on the job, does the same skill set still apply? Or are people being asked to develop new skills to take advantage of all these cloud features?

That's a good question, and I've really been trying to drive this home when I lecture at testing conferences. The skill sets have changed. In the past, if you were a really good bug finder—really good at exploratory testing—that meant something, because some bugs were hard to ferret out. If you released them to the field, a lot of people would be affected by those bugs before you could push a fix. But it's easy to push fixes to the cloud, so the costs of those kinds of bugs have really gone down. One user can find it and report it, and we can fix it, and we really don't need all those exploratory testers after all.

The bottom line to me is that companies in the past have hired great testers to act like users. But users don't have to act. They know exactly what kind of scenarios they're going to run, they know exactly what their customer data is—so we don't need to worry about coming up with those scenarios and fabricating that data. Instead of finding the bugs, we need to figure out how to make the reporting of them absolutely seamless, so that when a bug does occur, it can be fixed in a matter of minutes or hours instead of in the weeks or months that would have been spent until the next service pack comes out.

Bug-finding skills are less important now than they've ever been, because the developers are just so much closer to their users.

So, if I can paraphrase what you've been saying, the cloud is changing the whole underlying economics of software development and software testing. It's easier and quicker for a company to try something, push it out to users, hear from the users what the problems are, and fix them, than it is to follow the traditional path of getting the requirements right up front, then getting the architecture right and nailed down, then getting the coding done well ….

Absolutely. By the time you do all that stuff, you're too late. Your competitor's beaten you to the market. On the cloud, you can really release and iterate—that's much more the development model of modern times.

But you have to be careful: Google's not pushing software out to its users saying, "Hey, is this any good? We're not sure!" There are a lot of intermediate steps. We have an internal process we call *dogfooding*, as in, if you're trying to sell dog food, you should eat your own product first to make sure it's okay. All our software is used internally first by Googlers before we push it out to the world. If you look at something like Google+, which we released last year, we used that internally among Googlers for many months before we released it. In that process of dogfooding Google+, we found far more bugs and far richer bugs than the test team associated with Google+.

The points you're making, about having representative users from the beginning who are able to use the product and help mature it, represents a much bigger paradigm shift than I had originally realized.

To me, that is just one of the most crucial things that companies absolutely have to get good at. In the past, if you found a bug in, say, your browser, you didn't know how to report it. You'd have to find some bug-reporting page on the vendor's site, and it would ask you what operating system you were using and what version of the browser you were using, and what other plug-ins you had installed…. But the machine knows all that stuff! So the idea is that once you crash, or once a user finds a bug, you just grab that machine state and send it back to the vendor so that they can understand the state the user was in exactly.

This seems like a very concrete model to use for functional testing. But does the same paradigm work if I'm worried about things like reliability, performance, or throughput?

Or better yet, security, privacy, and so on. I agree with you completely. I think the idea of paying top dollar for engineers to do functional testing really is an artifact of the 1990s and 2000s, and shouldn't be something that companies invest in heavily in the future. But things like security, privacy, and performance are very technical in nature. You don't do security testing without understanding a lot about protocols, machine states, or how the Web works; a lot of a priori knowledge is required. You can't replace that. So when I give advice to functional testers who say that I'm predicting the end of their job, specialization is one of the things I recommend. Specialization is crucially important.

In 2011, if I were a young tester looking for a specialization that I could really make my name in, I would most certainly be studying the technical aspects of user privacy and become an expert in privacy testing, because the world doesn't really have one. Every single application is going to handle privacy in a different way, and I think that's why we haven't figured it all out yet. Even in the cloud, which is a much more homogeneous environment, we haven't really solved it.

How does the simplistic testing model that we all learned in school—where you go first through unit testing, then integration testing, then system testing—adapt to the new paradigm?

We do integration testing, but we call it something different. People al-

ways say that Google just likes to change the names of things, but we did this one on purpose. We don't have to integrate it from environment to environment, but we do have to integrate it across developers. So developer A writes one module, developer B writes another module; to us, integration testing hits both developer A's anzsoftware that you simply do not have to run on the cloud: any sort of configuration test, and any sort of load testing, just isn't necessary in this new modern environment. Load is taken care of for you; if it slows down, new cells in the datacenter are spun off automatically.

When you hire new testers for your teams at Google, is there something in particular that you're looking for? You mentioned specialization as being important, but is there anything else that makes a good cloud tester versus just a good tester?

We certainly hire based on development skill as well—Google has traditionally hired folks with a computer science degree for either development or testing, and our test candidates are also asked development questions and coding questions in our interviews. We believe that those core skills are really crucial for doing this.

I really can't name a single commercial testing tool that Google has licenses to. For a lot of the stuff that we've done, because of this huge paradigm shift from client-server to cloud, the tools just aren't there. So the ability to write tools is really important to a modern tester. Those coding skills have really come to the forefront.

For folks who are trying to move legacy systems onto the cloud, does their development and testing process look a lot different from what they'd use when trying to do something more greenfield?

Google's been around since 1998, so we certainly have some of those old systems that needed to be ported. But a lot of the test infrastructure doesn't port at all and a lot of that stuff has to be absolutely, completely redeveloped. There are different protocols for the device or client to connect to the cloud; there are completely different input structures, so anything that was written on the old system won't run at all.

Really, you're often better off rethinking the application from the start, because there are so many new efficiencies. There are so many different types of tests that need to be run. And there are so many tests that you have to run on client-server software that you simply do not have to run on the cloud: any sort of configuration test, and any sort of load testing, just isn't necessary in this new modern environment. Load is taken care of for you; if it slows down, new cells in the datacenter are spun off automatically.

Where are things going in the future? Will abstractions allow developers and testers to worry about even fewer issues over time, or will there be new things that we do need to worry about as more and more people go on the cloud?

There are definitely some new things that we'll need to worry about. First and foremost, connecting to customers is going to be really important. As much as we have the server side of it down (instead of having a massively complex server, we just have this cloud that takes care of itself), there's still a lot of variation on the device/user side. If you look at the number of Android devices that are out there, and the number of operating systems and apps that people have configured onto them, that is still a hard testing problem.

The cloud actually makes that easier, too. Crowdsourcing companies are now connecting certain specific people with specific devices to people who are writing apps on those devices. So the idea of leveraging the crowd through the cloud is definitely something that hasn't been done before, and is a new phenomenon that we're watching really carefully here.

One thing is for sure, we're never going to settle on a single platform. Humankind doesn't seem to be capable of doing that, and I don't think it would be a good thing to eliminate competition among platforms. The Linux/Windows competition has always been healthy, and the same thing is happening in the mobile space now. So we're always going to have to develop for multiple platforms, and those platform owners are going to want to innovate as quickly as they can and they're not always going to be checking with you or each other on those innovations, so the developers are just going to have to be on their toes.

Learn More

My conversation with James touched on many more issues than I could note here. If you're interested in hearing more of the conversation we had, which ranged over additional issues such as cloud testing tools and handling privacy and robustness, then check out our half-hour audio interview at http://doi.ieeecomputersociety.org/10.1109/MS.2012.23.

More than anything else, my conversation with James made me aware again of the significant changes to the way we do business that accompany the cloud, and the new skills that are becoming important. Perhaps the best summary was James' comments that "People really need to take the cloud seriously and rethink testing from the ground up. There are a lot of sacred cows in testing that just go away with the transition to the cloud. Keeping an open mind and taking advantage of the efficiencies of the cloud are going to be really important." I certainly hope the remainder of this special issue on cloud computing will help give you useful food for thought in doing so.