Why testers are still important - a developer's perspective

Syed Ali - LeedsPHP - October 16th 2024

Want to see the slides up close?

www.bearthoughts.blog

Who am I?

- Senior full stack software engineer at Laka (bicycle insurance).
- Current tech stack is Node / Go / React.
- Near 13 year career in tech (Nov 18th 2011).
- Held multiple positions, across multiple industries.
- Lives in Hull.
- Really likes Tequila (feel free to buy me a shot)

What makes you qualified to talk about testers?

TL:DL; Nothing!

Nothing makes me qualified. I’ve been in a lot of different dev teams though, some with and some without dedicated testers. I have opinions. So indulge me. Or not - I won’t hold it against you if you’re in a bit of a food coma after the pizza!

So before we get into it, what is a tester?

A tester is a person who tests. Simple enough right?

My brother is a tester (and he will rip my throat out if I call him QA - he's a tester!), also called Syed Ali. We both have 4 names, and only one initial is different. So in the same way that we share 75% of our initials, I'm going to share some of his wisdom, and maybe borrow a little bit from one of his talks (don't worry - I'll sprinkle my own seasoning on it) - you can find the link to his talk via my blog - its at the top and I highly recommend watching it. I'm also very bias.

So before we get into testing and testers, let's ask and answer one simple question. WTF is the difference between testing and checking? (—New slide—) (Bonus points to anyone who got the reference I made just now as well).

So before we get into testing and testers, let’s ask and answer one simple question. WTF is the difference between testing and checking? (Bonus points to anyone who got the reference I made just now as well).

——Crowd participation——

A check is a very simple yay or nay. Do I have a nose? Yay. That’s a check. Am I wearing pink fluffy slippers? Nay - unfortunately. That’s a check,

To take that into the realms of code for a minute. Unit tests. Let’s say I have a calculator class and one of the methods is add that takes 2 arguments. A and B. My unit test asserts that the add method when called with 1 and 2, returns me the number 3. That’s a check.

Computers are brilliant things at running this code and giving you a yay or nay. So computers are great at checking. But what checks should we create? What do those checks mean? What information are we trying to find out with those checks? And regardless of all those checks being happy, is the product good? That there, everything around those checks, that’s testing.

The taller Syed Ali likes to see it as like an engineering console - kind of like this funky thing right here. Now imagine this is in a factory, and you've got a few labels from Conveyor Belt A to Z. Each with their own set of lights. Checks are constantly happening to see if the conveyor belt is good. If it is, you've got a green light. Any issues? Red light.

Testing happens when you dive into what those checks are, and finding what's gone wrong to make the light red.

So why is testing important?

- Verify that things work as you expect (obviously)!
- Verify that your platform is still stable.
- Verify that your platform is secure.
- Have confidence in the product you release.
- Instill confidence in your users and partners as the platform does not break for them.

In every industry, anything that's built needs to be tested. Concrete walls? It gets tested to see if it stands strong and upright. Laid flooring? It gets tested to see if its level. Put a car together? It gets tested on a track to make sure it goes, turns, and stops as you expect.

Testing isn't something that's limited to the software industry. In fact some terms came from other industries. Such as the smoke test - the smoke test was a test designed by plumbers to check pipework for leaks. Smoke would be blown into the pipes, and then the pipework would be inspected to see if any smoke was escaping.

In that same vain of plumbers testing pipework, we need to test our software. To carry on the plumbing and pipes metaphor, we need to inspect the pipework and make sure its fit for purpose. We need to test for leaks and make sure they're secure, so that external contaminants can't get in. We need to test that the pipework won't fall down when water is ran through it. We need to have the confidence that when its being used and we're not there to babysit it, that everything works as expected and we won't get a call at 3 in the morning telling us the building is flooded. And by that same token, we need to give the building owners confidence that they can go to sleep and won't wake up underwater.

Can’t developers just do it?

- Testing and feature development are different disciplines.
- Developers can write the code for the automated tests, but what tests are they writing?
- “Can’t see the wood for the trees”
- Split focus / Jack of all trades, master of none.

Developers can certainly write the code for automated tests. In fact developers should definitely be writing unit tests as part of the development process. But those tests are only valuable so far as to tell you that the single unit, in a controlled environment, responds as you expect. Those tests are a development tool more so than a testing tool.

Testing is a different discipline to development. Good testing is always looking at the bigger picture. Where as a developer I’m in the knitty gritty details of something, a tester is looking at how this works with everything else. A good tester is looking at how my feature is affected by things upstream of it, and how it affects things downstream. They’re not interested solely in just if this single unit works well, they want to see how it works in tandem with everything around it. They’re interested in everything that hasn’t been thought about. They’re interested in finding any exploits that exist because of the change. There will always be unknown unknowns, and testers are amazing at finding those.

And here’s the thing. The testing tools may have changed. We now have Behat, Selenium, Codeception, Cypress, Playwright, Ghost Inspector, and that’s just the few I could remember when I wrote these notes. The discipline of testing hasn’t. The idea of writing test packs hasn’t. The test packs have moved from printed tables to code thanks to the tools - but that still hasn’t changed the thinking behind those tests.

So short answer: no. I can’t just do it, while doing everything else you want from me. Go get a tester.

But my budget! I'd rather have another developer instead of a tester.

Has anyone here heard "green wave driving"?

——Crowd participation——

So I love driving. Be it driving fast or cruising along. I love it. One thing I really enjoy is thinking about traffic flow. Slowing down. Speeding up. Coming to a dead stop. All things that affect the flow of traffic.

I've driven at all hours. And one thing I enjoy doing during early morning or suoer late night drives is get all the green lights. I know now which traffic lights in Hull are radar and which aren't. One journey I've done a lot is from my house to the train station. I like to time my journeys. So its normally 16 minutes. Hitting the speed limit, and getting red lights.

Then I remembered something I heard on a podcast a while back about how in some cities where the speed limit was 45kph, people would drive at 40kph and would get green lights all the way. So I tried it. I drove at 22-24 all the way. I did the journey in 13 minutes with green lights all the way. 3 minutes off my journey by driving slower, but being able to stay consistent.

Now let's take those times, and apply it to development. Maybe to say a team of 5 developers. That's 3 minutes saved per developer - 15 minutes. And by my maths - we got 2 minutes extra over another developer, and you've got 5 quite relaxed developers to boot. We're better off than having 6 developers.

Adding another developer doesn't always mean things get done quicker. Sometimes the team needs a BA or product owner. Sometimes it needs a tester. Ultimately, you really need to look at what issues the team is running into, and then work to solve that. But also, have testers in your team!

Why testers are still important - a summary.

- Have a stable product.
- Find the issues before your users do.
- Overall increase in speed of development by avoiding regressions. A.K.A. Green wave development!
- Instill confidence in your release, your users, and your partners.
- Relevant to those on-call: decrease the likelihood of being woken up at 3am!

So there you have it. I believe testers are still very important and should be in every development team. This trend away from having testers is quite scary to me, and I hope the industry wakes up and sees its not a good thing to be taking away. Imagine taking away testers from the process of manufacturing… pacemakers. That's a scary thought. As software becomes even more ingrained in our daily lives… let's hope testers are always there in our teams, working with us to make sure things don't go wrong.

Time for questions / opinions / thoughts / anyone have a burning desire to just share?