# Is the software testing role relevant?

**Sriratna Josyula**

ratnaj@zillowgroup.com

**Jonathan Li On Wing**

jli@zillowgroup.com

## Abstract

Testing is one of the fields of Software Engineering that has been very rapidly evolving over time. The latest evolution is the departure of the traditional testing roles. Various companies like Microsoft that had previously a ratio of 1:2 Software Development Engineers in Test (SDETs aka Testers) to Software Development Engineers (SDEs aka Developers), have converted their SDETs to generic Software Engineers.

Until about 5 years ago, product testing involved some combination of manual and automated testing that ran into hundreds, if not thousands of test cases. Turnaround time for test, would be anywhere from few days to few weeks. Once shipped, if a product had a major bug, it would be part of a service pack or the next version, testing for which would again consume the same amount of time as before.

Fast forward to the current day - software is largely shipped as services, instead of as products - which makes it very easy for bugs to be fixed if found. This is one of the biggest reasons that has caused companies to take a very different approach towards testing.

As a result, we see shorter test passes, faster ship cycles and fewer resources dedicated to testing. But, that does not mean that software can be shipped without being properly tested.

This paper is an effort to shed light on what lies in the future for testing as a discipline and how to stay relevant in this field.

## Biography

*Sriratna Josyula is a Senior Software Test Lead at Zillow, in Seattle WA. She earned her Master's in Computer Science from University of Southern California. She has over 9 years of experience, working in various roles in test, starting from Validation Engineer to SDET to QA Manager, on products spread across the networking stack, from router firmware to advertising technologies.*

*Sriratna is passionate about all things related to quality. Her interests lie in improving software quality practices, project management and product management.*

*Jonathan Li On Wing is a Senior Engineering Manager at Zillow, in Seattle, WA. He earned his B.Sc. Cum Laude in Software Engineering from McGill University specializing in Artificial Intelligence and Requirements Analysis. He has 12 years of industry experience in various roles working at small and large companies such as Microsoft, Oracle, Expedia, Groupon, and SMART Technologies.*

*Jonathan's interests lie in Human-Computer Interaction, Automated Testing and Software Life Cycle Processes. He believes in bringing testing upstream and has been improving software testing processes in several companies. Jonathan sees himself as a customer advocate and enjoys challenging everything.*

# 1 Introduction & Motivation

Traditionally, the role of a person in test, whether it be a Software Tester, Software Test Engineer, Software Engineer in Test, or Software Development Engineer in Test was to sufficiently test a product and to “sign-off” that a piece of software was ready to be released. The term “Signing-off” came from other realms of engineering where an engineer had reviewed the specifications, verified that the project was compliant with any rules and regulations, verified that the project met the specifications, and to state that it was safe for usage.

This often led to a “throw over the wall” type of mentality, where Software developers would sometimes deliver sub-par quality software, just because they were not being held responsible for its quality. This was not only just a bad practice, but it also caused projects to often miss release deadlines because of time spent going back and forth between testing and development.

In the past decade, there has been a push to curb this mentality and make quality a team-oriented goal. There was a shift on accountability, where engineering teams were held accountable and not individual roles. However, if the engineering teams owned quality, why were there test roles? At that time, the thinking was that testers would still be the primary source of testing work -- both manual and automated -- but there would be an expectation that developers would help and that the throw over the wall mentality would disappear.

## 1.1 Evolving Role of the Tester

As the ownership of quality shifted towards the entire engineering team, the distinction between developers and testers got fuzzier. Companies started obscuring between the roles of developers and testers, and titles were merged. Teams went from having ratios closer to one tester per every two developers, to one tester per team.

Now, with the push towards Service Oriented Architecture and increased usage of telemetry the need for a specialist in manual and automated testing seems to be fading away. So, what does that mean to the testing role? Is it still relevant? Is it to disappear? What is the future? In this paper, we will discuss our thoughts of these questions.

First, we will look at what has led us to this point and study what some companies have done. We will then discuss what our belief of the next evolution of testing is, and how best to stay relevant. We conclude with a summary of our thoughts.

# 2 Evolving approach towards testing

Historically, especially when we consider waterfall style methodologies, verification was performed by “testers”. This was often the last step before the software was released. Typically, there would be a test plan written by testers and then that plan would be followed. These tests traditionally were manually executed, but later progressed to use more automation.

Nowadays, testing isn’t a specific step in most lifecycles. It is a part of the development process, and it isn’t as vigorous, detailed or specific. Testing tasks are shared across the team, and focus towards risky areas, likely scenarios, and happy paths.

## 2.1 Implications of shared testing tasks

Making testing a “shared” responsibility had effects on both the testing role itself and product development. For the role, some companies have started merging testing and development roles into one. In some cases, testers would need to interview for development roles or be laid off. This will further be expanded in Section 3.

The implications on product development is that *shallower testing* is performed; hard to think of unhappy paths, the bread and butter of a testers' job, are often ignored. James Bach and Michael Boulton contrasts *deep testing* versus *shallow testing* in their Rapid Software Testing courses as:

> Testing is deep to the degree that it has a probability of finding rare, subtle, or hidden problems that matter.
>
> Deep testing requires substantial skill, effort, preparation, time, or tooling, and reliably and comprehensively fulfills its mission.
>
> By contrast, shallow testing does not require much skill, effort, preparation, time, or tooling, and cannot reliably and comprehensively fulfill its mission. (Bach & Bolton, *Rapid Software Testing*, 2017)

## 2.2 Why is there a shift?

The main reasons why we see a shift in how companies approach testing is mainly because of (but not limited) to the following:

- Developing Software as a Service (SaaS) - building web services and applications instead of desktop applications.
- Shared Responsibility - quality is no longer just owned by the tester.
- Speed - moving fast and getting a product out faster to market is often prioritized over quality.
- Analytics and Telemetry - increased ability to monitor and find bugs in production.
- Partial Roll Outs - slowly dial users to use new functionality.
- Continuous Integration & Deployment - it is easier to ship or roll back changes.
- Microservice Architecture - moving away from monolithic applications.

### 2.2.1 Software as a Service (SaaS)

Developing desktop software applications typically involved long ship cycles and longer time to update (even more so in the period of installation via physical media). Additionally, one could not always guarantee that a user would install the update. Therefore, there was emphasis on the quality of every shipped version. With the shift to developing and deploying software as a service, software is very easily deployed, updated or rolled back. This meant that certain compromises could be made on quality, as long as some basic quality criteria were met.

### 2.2.2 Shared Responsibility

Many technology teams have moved to Lean and Agile methodologies, and this methodology challenges the need for separate development and test groups. They encourage having one team with a shared common goal. Everyone on the team is treated equally responsible and encouraged to take on any task.

### 2.2.3 Speed

Software is being built at a much faster pace than before, and companies want to get as many features out as possible, and as fast as possible. This does not necessarily mean that quality is not important. However, risk can be assessed, and a question can be raised as to how much testing is good enough. At a certain point, it will cost more to find a bug than to release the software as is.

### 2.2.4 Analytics and Telemetry

Tools and applications to help monitoring production code have been on the rise and becoming more sophisticated. Many tools now exist to help investigate production issues and to catch issues early on. With some such monitoring software even using advanced machine learning techniques, production issues can be identified even before they occur. Moreover, such analyses cannot only be done in production, but also in test environments. These tools help in making software more reliable and stable even before it is released to production.

#### 2.2.5 Partial Roll Out

A/B testing has been widely used to test features to test experiments and see which version performs well and which one doesn’t. This same framework can be used to limit user exposure to new features, thereby providing the advantage of releasing a new feature early, while also minimizing risk by not releasing it to 100% of a company’s user base. By having good metrics, monitors, and alerts as mentioned above, companies could release a feature to 1% of users and see if there are any errors, bugs, or issues. Gradually, they can dial up to 100% of users.

Additionally, this methodology has allowed companies to be able to cause brownouts where features can be quickly turned off while the engineering team fixes the bugs.

#### 2.2.6 Continuous Integration & Deployment (CI/CD)

With modern build processes, each commit an engineer does, can automatically build, launch a test environment, apply a battery of tests, and ship it to a set of environments (including production). Additionally, triggers could be set to automatically rollback to previous versions should bugs be found.

#### 2.2.7 Microservice Architecture

Moving to microservices architecture has helped in a few ways. First, by breaking products into small logical services makes testing features and functionality easier. That is, it is easier to test in isolation, and a microservice architecture’s goal is to isolate functionality into its own service.

Second, it is easier to ship a small web service to see how it performs and roll back if necessary. Third, based on our personal experiences, there seem to be less logical bugs in this type of architecture.

## 3 Case studies

We chose to include these companies in our Case Studies because they are our current/ former employers. *These comments and studies are based on our experiences working at these companies and from the references cited. These comments are not official responses from companies*.

The purpose of these case studies is to highlight the transformation that is happening to the traditional software testing role.

We will consider three companies that we have had experiences with, Zillow, Expedia, and Microsoft. Although we talk about three companies, we believe that these changes are happening at various other companies as well, based on articles we read and speaking with engineers.

### 3.1 Zillow

Zillow has seen its test discipline re-structure from being a heavily SDET owned operation towards a developer **owned**, but test owner **driven** testing.

Most teams at Zillow now have a test owner (Test Lead or an SDET and/ or a Test Engineer) who is responsible for the overall test quality of the team’s products.

Each test owner usually works with a team of about 8 developers. Test owner leads the testing effort, but responsibilities can vary between teams. Engineering teams decide what would be the best way to distribute test efforts among the team and proceed with the same for testing their features. For some teams, test owners take charge of test planning, execution, while developers write the test automation, for some others, test owners do the actual manual/automated testing, and for some other teams, test owners drive planning, test code reviews and releases. In most cases, developers own writing and maintaining test automation.

#### 3.1.1 Zillow's transformation of the testing role

Originally, Zillow had a Testing org (as well as Development and Product Management orgs). Testers did manual testing and wrote test automation. Zillow mostly hired SDETs and had an offshore manual testing team to do sitewide test passes.

As the company started growing, it re-structured orgs and as a result, broke into individual product teams, with each team having a Group Manager. Developers, Testers and Product Managers reported to the Group Manager. Most of these testers were SDETs and they wrote test automation, test plans, did some manual testing, and directed the offshore manual testers. SDETs spent most of their time doing manual testing and directing the manual testers. Management realized that Development teams were not taking ownership of quality or automated tests and were just pushing it over to the testers.

So, Zillow moved automated test responsibility to the developers, and switched to hiring test leads. Test leads would direct testing for the team, but Development teams would own the automated tests. Test Leads would also direct manual testing for offshore team. Even in this model, the role of the Test Lead varies by needs of their Development team. The role of test leads really is to make sure that their team is thinking about quality and doing the right thing for continuous improvement.

### 3.2 Expedia

At Expedia, testing role varied from one organization to another. In general, though, SDETs were the main drivers of quality. They owned test planning and the test automation platform. They built test frameworks and test automation themselves. They executed tests and logged most of the bugs. To be able to meet testing demands, developer to tester ratio was around 3 to 1.

Expedia has been going through a change in its approach to testing. There are groups that have already transitioned completely from having dedicated testers. In those teams, SDETs would need to change roles to be a developer, with a small focus on testing or find other teams that needed SDETs.

In teams that still have SDETs, it would be rare to have more than one per team. SDETs would focus on test architecture, worked alongside developers to write test automation, run tests, and mentored SDEs on how to write good tests.

### 3.3 Microsoft

At the time that we were at Microsoft (over 5 years ago), SDE:SDET ratio on the teams that we were on, was about 2:1. SDETs owned all the work related to test planning, test automation and maintenance.

However, from approximately about 3 years ago, Microsoft has moved to a “Combined Engineering” model, where SDETs no longer exist or exist only in very specialized teams.

The concept of Combined Engineering is that an engineer is responsible for his/ her feature end-to-end – from “from unit testing, to integration testing, to performance testing, to deployment, to monitoring live site” (Shah, 2017).

# 4 Future of software testing

In the future, this is what we envision the role of someone in Software Testing would be:

- To not just test in the traditional way, but to be an evangelist for quality.
- Coach and mentor your team on how to think about quality in the tasks they do.
- Focus on quality of customer experiences and not just on code quality.
- Risk assessment.
- Ask questions about the testability of a system.
- Push for quality upstream, right from planning for a new product or feature.
- Quality gates for when code is checked in.
- Unit test coverage
- Think of Test Driven Development
- Automate the “right” things!
- Excessive UI automation is never good.
- Learning new technologies and seeing how it can help them in their role and beyond.
- Analyze production issues to find issues or commonalities, and determine if it is a bug.

## 4.1 Evangelize the importance of Quality

In the never-ending quest to move fast to release software, quality often is forgotten. Since quality/ testing efforts are taken up towards later phases of a project (usually following product planning and development) it is common that there is not enough time between getting a final build ready to test and the product release date. This puts testers in a difficult situation - Should they be asking for more time to test? Should they fit in as much testing as possible within the time left? What if they find blocking issues that need fixing, and the product goes back into development, giving them even less time to test and sign-off?

This is where evangelizing for quality becomes important.

Help your team understand that quality is everyone’s responsibility and not just the tester’s. Coaching and mentoring your engineering team on quality, and how the team can improve, will become an important part of someone in a dedicated role for quality.

After all - a well-scoped and properly developed product is always better and faster to test, than otherwise!

## 4.2 Have a growth mindset!

Technologies for development and deployment of software are continuously evolving to become better and faster. To continue being relevant and thrive alongside this change, professionals in the field of quality need to have an awareness and an overall understanding of these technologies. Testing practices need to adapt based on technologies used to develop and deploy software. It is more important now than ever, to stay aware of upcoming technologies, and find out if there are any tools or services you can use to help with testing.

## 4.3 Drive a quality mindset!

Understanding the product and being involved in product and design discussions right from the beginning is key to successfully planning for test and delivering on the same. Someone in test should:

1. Work with product/ project manager to get product specs written and reviewed by the entire team. Ensure that the scope of the project is correctly defined and prevent “scope creep”.
2. Work with your engineering team to
    a. Assess risk

        i. What is the *most* critical feature in the product?
        ii. What would take the most time to test?
        iii. Which component is prone to having more bugs?
    b. Is the product testable? If not, understand what the reasons are, and plan for getting needed resources ahead of time to make testing possible before release.
    c. Define quality gates and acceptance criteria before the product moves over into test -
        i. Is Unit testing being done? This is the least expensive of all types of testing (remember the Testing Pyramid?) and extremely important.
        ii. Are test plans defined, and on time? Do they cover the needed product behavior as defined in the Product spec?
        iii. Ensure that test automation is a planned activity as part of release of the project.
    d. Be involved in the design and architecture
        i. Does the design make sense?
        ii. Are we using the right technologies?

## 4.4 Be a customer advocate

In many cases, a finished/ built product is delivered to a tester to test it out, but a professional in the field of quality (or testing), should have the right understanding about why a product is being built and, how to test it effectively.

- Why is our company/ team building this product?
    - What is the purpose of this specific product or feature?
- What customer problem does it solve?
    - What alternative approaches exist to solve this problem?
- Who are the types/ groups of users that will use this product?
    - How should I cater testing to ensure proper usability of this product?
- Do you believe that the finished product serves the customer as expected? Not just in terms coding requirements, but also in terms of user experience?
- How is the end user experience using this product (or feature)? How easy is it to use?

With all the emphasis given to and time spent improving software development processes and coding practices, it can be easy to overlook if a finished product or feature will serve the customer as expected. But remember – asking right questions at the right time during product development is key to a project's success. If a product isn't solving a user's need as expected, a thoroughly developed and well-tested product is of little use!

## 4.5 Remember that Internet of Things (IoT) is here

Looking at the current trend and the pace at which IoT is expanding, it is only fair to assume that it's here to stay. Testing software services end-to-end may include testing using devices of multiple operating systems, user interfaces, screen sizes, versions. Testing various aspects of a service, like accessibility, performance, scalability, usability, security, compliance, software upgrades can get much more complicated than before and will need careful planning and execution on behalf of testers.

## 4.6 Don't forget Security testing

As a direct result of having a diverse set of devices embedded in a service's workflow, there will be multiple places where data is exchanged between devices. Each of these interactions will involve devices that may use different security protocols to communicate, making it more challenging to test. Information security will be a big area for testers to focus on, not only to ensure that devices are communicating securely, but also to ensure client and customer data privacy.

### 4.7 Get comfortable with software that can write software

Not too distant in the future, test automation may not be written by testers or developers, but by software. What does this mean for testers? Should this be perceived as a threat to a tester's role or be considered as help? We think this can be a new opportunity for testers - this can be a tool that will help testing to move fast, without compromising on quality. Testers need to put in checks and balances in place for such software, to ensure that such software is testing the right things in the right way.

## 5 Conclusion

All evidence points towards a big shift in the testing role. More companies are re-thinking their approach to testing due to the reasons mentioned above (in Section 2) and joining the movement to get away from a traditional software-testing role. Does that mean that a career in testing is obsolete? No. Quality is important and relevant. So is delivering a high-quality customer experience for any product or service. This means that a testing role will still exist in most companies. However, it may not look like what it does right now. Focus of this role will shift towards "quality" rather than just "testing". We believe that testers will go from typical individual contributor – in-the-weeds – roles to more leadership and consulting type roles. This may look like specialized virtual teams that focus on software quality for an entire company or specialized roles in quality, for a set of teams. There may still be certain companies that need manual software testers with specific skills, but demand for such roles will continue to wane.

The expectation of writing automation or running manual tests will be shared, but where testers will excel is by being "quality stewards" - influencing and growing the team and organization to develop a quality mindset.

We believe that this transformation not only sets up a career in software testing for longer-term success but also is beneficial to product and company success.

## 6 References

Allanabana, Fazal, et al. 2016. "Product Integration Testing at the Speed of Netflix." Netflix Technology Blog. https://medium.com/netflix-techblog/product-integration-testing-at-the-speed-of-netflix-72e4117734a7 (accessed August 23, 2018)

Bolton, Michael. 2017. "Deeper (1): Verify and Challenge." http://www.developsense.com/blog/2017/03/deeper-testing-1-verify-and-challenge/ (accessed August 24, 2018)

Crowdsourced Testing. 2018. "Crowdsources Testing." Crowdsourced Testing. https://crowdsourcedtesting.com/ (accessed August 25, 2018)

ISO\IEC\IEEE. 2013. "The Role of Testing in Verification and Validation." *ISO\IEC\IEEE 29119-1:2013 Part 1: Concepts and definitions:* Section 5.1.1.

Page, Alan. 2016. "What's so special about specialists?" Testastic – The ramblings of a software quality philosophist. https://testastic.wordpress.com/2016/09/05/whats-so-special-about-specialists/ (accessed August 23, 2018)

Reddy, Jogannagary, et al. 2016.

Schaffer, André. 2018. "Testing of Microservices." Spotify. https://labs.spotify.com/2018/01/11/testing-of-microservices/ (accessed July 19, 2018)

Shah, Munil. 2017. “Evolving Test Practices at Microsoft.” Microsoft. https://docs.microsoft.com/en-us/azure/devops/learn/devops-at-microsoft/evolving-test-practices-microsoft. (accessed August 23, 2018)

Splunk. 2018. “Predictive Analytics dashboard.” Splunk Docs. https://docs.splunk.com/Documentation/ES/5.1.0/User/PredictiveAnalyticsdashboard (accessed July 18, 2018)

Thonangi, Uday. 2014. “Why did Microsoft lay off ‘Programmatic testers’?” https://www.linkedin.com/pulse/20140806183208-12100070-why-did-microsoft-lay-off-programmatic-testers/ (accessed August 23, 2018)

Ulrich, Adam. 2006. “What’s the difference between SDE and SDET at microsoft?” AdamU WebLog. https://blogs.msdn.microsoft.com/adamu/2006/01/27/whats-the-difference-between-sde-and-sdet-at-microsoft/ (accessed August 25, 2018)

Wikipedia. 2018. Crowdsourced testing. https://en.wikipedia.org/wiki/Crowdsourced_testing (accessed August 25, 2018)