Continuous Testing VS Test Automation

Md Rezaul Hasan

April 7, 2020

Contents

1 Introduction

In many IT organizations there is a "Wall of Confusion"[2] between development and operations. On the one hand , development wants change, but on the other, operations wants stability. Between them is a wall with a lot of confusion.

There has been a shift to break down this "Wall of Confusion." More teams are starting to work with a DevOps mindset places more emphasis on delivering quality in a service, with end-to-end responsibility.

To assure the quality of the service, testing was and will be an essential part of the process. With DevOps, changes are necessary to make sure testing really focuses on the quality in the delivered service.

Many testers today feel that there is too much overhead to maintain the automated tests and that it does not work that well these days. The tests just can not be automated and run the same way anymore.[12]

The main aim of this essay is to provide a clear understanding about how DevOps works in an organization to testing as continuous testing and automation testing. The objectives of the study are to identify the benefits of implementing DevOps in organizations where different tools is used for the challenges faced by organizations during DevOps adoption, to identify the solutions/ mitigation strategies, to overcome the challenges,the DevOps practices, and the problems faced by DevOps teams during continuous integration, deployment and testing.[12]

2 What is DevOps?

The term DevOps consist of nothing more than the contraction of the words:Development and Operations. DevOps started in 2009 with the DevOps Days, a conference organized by Patrick Dubois [11]. Although many have tried to come up with a definition for DevOps, there is not one universally accepted definition. The DevOps Agile Skills Association (DASA) has formulated six principles which cover most definitions of DevOps.

- Customer-Centric Action
- Create with the End in Mind
- End-to-End Responsibility
- Cross-Functional Autonomous Teams
- Continuous Improvement
- Automate Everything You Can

The five and six principles help to understand what DevOps really is about and mostly focus on effect of this in DevOps. In this essay I am going to more focus on last two principle.

3 Testing in DevOps

"The process consisting of all life cycle activities, both static and dynamic, concerned with planning, preparation and evaluation of software products and related work products to determine that they satisfy specified requirements to demonstrate that they are fit for purpose and to detect defects." [5]

This definition of software testing is known throughout the world. In DevOps these activities still need to be undertaken to verify quality of a product. Before DevOps, testing tended to be a phase in the SDLC which ended before the product was delivered to Operations. Testing, however, should not be a phase. It should be present from the start and should be a responsibility shared by the entire (delivery)team [9]. Done right, testing helps to "Build quality in" as how W. Edwards Deming described this [7]. The shared responsibility together with embedding it in the entire process should lead to a good quality product.

3.1 Automation

DevOps will not work without automation. Error-prone manual tasks can and should be replaced by automation. DevOps teams require fast feedback and automation is the way to get this to the team. It can speed up the existing processes and make sure the team receives feedback about the process as soon as possible. When automation is working, team members can focus on tasks which do require human intervention. Automation can play a role in the breakdown of the "Wall of Confusion." It could be possible that Development and Operations used their own set of tools for deploying and other processes. Within DevOps it is best if teams start using the same tools for the entire SDLC. This can be a way of bringing team members together and make the SDLC clear and coherent. Different skills present in the team can shape the automation to where it fits the needs of the entire team.

3.1.1 Automation Testing

In testing, more and more tests are being automated. Test engineers work more with testing tools and automation supporting their tests. It creates fast feedback loops which drives development design and release [9]. In DevOps you want to automate as much as possible. The testing quadrant diagram, as created by Brian Marick, was adopted by Lisa Crispin and Janet Gregory to create the Agile Testing Quadrants[6]. The quadrants show different kinds of tests where automation can play a role. Technology facing tests supporting the team, like unit tests, are automated.Business facing tests supporting the team, like functional tests, can be automated or done manual. These functional tests in DevOps should be automated, based on the DevOps principle "Automate everything you can" [1]. These functional tests are the tests that should be part of customer-centric testing as mentioned before.Technology facing tests critique to the project, however, require mostly tools and are therefore already automated.

The automation of the first three Agile Testing Quadrants should leave time and space for the last quadrant with business facing tests that critique to the product. These tests should be done manual and cannot be automated. With multiple skill sets in a DevOps team, it would benefit the team to perform these tests with the team during the time they saved with implemented automation.

The test pyramid can help the implementation of test automation. It makes a distinction between tests that can be executed fast on low levels (unit tests) and tests that are slower to execute (UI tests) [8]. The lower-level tests are most suitable for automation, which is why they are usually fast to execute. The test pyramid combines tests originally performed by developers (unit tests) and those performedby test engineers (service, UI tests). This is a testing strategy that will work with DevOps because it is a cross-functional strategy. Engineers in a DevOps team should share their knowledge and expertise to fully implement this test strategy. This strategy also helps teams making testing a shared responsibility within the team.

3.2 Continuous Testing

Automated testing can be part of a deployment pipeline and can be part of Continuous Integration, Delivery, or even Deployment. Deployment pipelines are the "automated implementation of your application's build, deploy, test and release process" [9]. The deployment pipelines are a way to empower teams to take control over their deliverable. A pipeline can help a team deploy their service and verify the quality in an automated way. In DevOps teams, it should empower all team members to deploy any version on any environment with the right controls in place. The pipeline can limit the complexity of deploying and testing a service. The knowledge difference between team members can be smaller when every team member could deploy and test a service with a push of a button.

Continuous testing can act as continuous process where every time the pipeline is started tests are being executed. Tests can act as go or no-go points in the pipeline to go to the next step in the process. It also gives the team up-to-date feedback on the quality of their service in different stages of development. Testing can also be used to test the automation. It can help understand if the automation executes the correct steps in a correct way. This includes the automation used for deploying services on different environments. With deployment testing, a team can take control of the infrastructure and check whether it is in the state where it should be. Testing will give teams control on their automation when they are relying much more on it.

4 Continuous Testing vs Test Automation

People might say that continuous testing and test automation are the same thing, but this is not true at all. Both testing strategies are test automation

based, but there are several ideas which differentiate the two strategies, and they can be divided into three different categories; risk, breadth and time.

- Risk
- Breadth
- Time

4.1 Risk

While test automation is all about testing user stories and functionalities to see if they are correctly implemented and work properly according to the specifications, continuous testing focuses on testing on a higher level. The meaning of the latter is that tests are written mainly with the purpose of testing whether the software is too risky to release or not. [3]

To explain what is meant by risk in this case one could use a grocery store example for instance. Before, a grocery store literally offered groceries. It had a location, products, commercials and customers. These days the majority of all grocery chains have their own website and application available for download, where customers can become a member, collect points, have access to recipes, search for products and get discounts, just to mention a few. An application could be bad in many ways. It could have a poor user interface, or not work properly, for instance taking a long time to load pages or not caching information so that the customer has to re-do actions. Having a bad application would be a huge risk to a company. Because of software failures they might lose customers to some other store with better software or better software ideas.

4.2 Breadth

OK so people talk. What differs continuous testing from test automation also is that small errors are taken much more seriously because of the damage it could make to a company's reputation. Small errors could for example be errors affecting the user experience in a bad way, like re-sizing or moving buttons or change the behaviour of them so that the user has to struggle a bit because of the change. It is not necessarily the customer him-/herself which is lost, but many others being lost because of brand damage on social media for instance.[3] People do not just tell their friend or their neighbour if something bad has happened anymore. Today there are endlessly many social media applications where people can say exactly what they want, and people usually do say what they want. That is why the tests need to be broad enough to notice when an application change impacts functionalities which users have come to rely on, without purpose [4].

4.3 Time

In the early beginning of test automation tests were created and ready to be run after the development phase. Now that more and more organizations are

adopting Agile and DevOps, the intervals between releases are becoming shorter and shorter. With shorter time between the releases it becomes even more important to obtain almost instant feedback from testing, to know whether your functionalities are implemented the right way or not. Otherwise the user stories might not be ready within the deadlines. And to obtain instant feedback the tests need to be run in parallel with the development.[3]

5 Monitoring

It is increasingly common to arrange monitoring from the start of software development. With an end-to-end responsibility in DevOps teams, monitoring is a way to get feedback from the service on production environments to the teams.Monitoring can vary from technical monitoring on server level: measuring CPU usage, memory, etc., to more functional monitoring: how many users are logged in etc. Functional monitoring gives to a certain degree insight into customer perception of the service. It can allow teams to track customers through their usage of the service.

It could be argued that monitoring can replace parts of testing. If a DevOps team has reached "the third way" [10], they gather feedback continuously and experiment with their service. Monitoring can help the team in the gathering feedback.The third way is when teams are mature enough to deliver new features fast and experiment to see what fits the needs of the customer.

Monitoring should be a more reactive way of getting feedback where testing is a more proactive approach. That is mainly because monitoring is focused on production environments and therefore a later step in the process. Testing can start at an early point in the process and gives teams more options to adapt to the outcome of testing. Monitoring can act as a way of testing when teams are able to adapt quick to the outcome of monitoring. If teams can create new features and deploy them fast,using, for instance, Continuous Deployment, teams can react fast. Monitoring can act as business facing tests critique of the product and would fit in the Agile testing quadrant[6]

6 Conclusion

The practice of testing in DevOps starts with the same foundations as testing in a nonDevOps environment would have. The end-to-end responsibility that a team has for a service means that quality requirements for both development and operations must be considered. Quality in the Operations section of the SDLC was usually not included but is now part of the scope of testing. The scope for testing in DevOps is on the entire functional service a team delivers. End-to-end testing therefore can take a different form. More specialized tests, like performance and security tests, will part of the new scope, although on a smaller scale. Due to the specialized nature of these tests, however, it is possible that these tests will be executed as part of a testing service outside the team.

DevOps teams should take ownership of these tests and should make sure they are executed. Functional tests in DevOps should focus more on the customer as part of customer-centric testing. This makes sure the quality a customer wants is put central in the work a team performs. In DevOps you want to automate everything you can. Automation in testing must be used where it can be used. With automation teams can get continuous feedback on their service and the steps they take to add new features to their service. Monitoring can be a valuable addition to testing and help to get quick feedback on the functional and technical levels on the service a team delivers. This feedback can be used to shape the service in a way it fits the customer.The role of a test engineer changes to a DevOps Engineer with a test expertise.The test expertise as part of the T-shaped model consists of knowledge and skills to implement and teach test strategies in a team. The test expertise should be able to connect Business and IT and different kinds of expertise in a team to get all the requirements for quality in a service. Responsibility for testing must be shared with the entire team. The engineer with test expertise can take a leading role in this and act as a coach.

References

[1] 6 principles of devops, 2017.

[2] Anand Srivatsav Amaradri and Swetha Bindu Nutalapati. Continuous integration, deployment and testing in devops environment, 2016.

[3] Wayne Ariola. Continuous testing vs. test automation: 3 key differences. june 2017.

[4] Wayne Ariola. Continuous testing defination: 14 key points., 2019.

[5] International Software Testing Qualifications Board. Certified tester foundation level syl-labus version 2018 version, 2018.

[6] Gregory J Crispin, L. Agile testing, 2008.

[7] W.E Deming. Out of the crisis. mit press, 2000.

[8] M.: Bliki Fowler. Test pyramid, n.d.

[9] Farley D Humble, J. Continuous delivery. addison-wesley, upper saddle river, nj, 2011.

[10] Behr K. Spafford G. Kim, G. The phoenix project: A novel about it, devops, and helpingyour business win. it revolution press, portland, or, 2013.

[11] Mezak S. The origins of devops: What's in a name?, 2018.

[12] A. Shafer. Agile infrastructure. speech presented at velocity conference, 2009.