iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 11 **Issue:** VII **Month of publication:** July 2023

DOI: https://doi.org/10.22214/ijraset.2023.54882

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 11 Issue VII Jul 2023- Available at www.ijraset.com

Optimizing Automation and Specialized Testing Techniques in DevOps

Sarthak Srivastava
Visa Inc

Abstract: DevOps encompasses a set of principles that hi ghlight the need for a different approach to testing. These principles emphasize placing the customer at the center of testing and employing more specialized tests. To ensure continuous feedback, automation should be leveraged to the fullest extent possible. All testing responsibilities are consolidated within a single team, requiring test engineers in DevOps to possess T-shaped skills that span across multiple areas of expertise.
Keywords: DevOps, Software Testing, Automation, Continuous testing, Continuous delivery

I. INTRODUCTION

In numerous IT organizations, there exists a barrier known as the "Wall of Confusion" that separates development and operations teams [1]. Development aims for change and innovation, while operations strive for stability and reliability. This wall represents the confusion and misalignment between these two entities. However, there has been a growing effort to break down this wall and foster a DevOps mindset. Teams are adopting a collaborative approach, bringing together engineers from different organizational silos to deliver value to customers which shows that the DevOps mindset emphasizes the importance of delivering high-quality services with end-to-end responsibility. With the shift towards DevOps, changes are being made to ensure that testing truly focuses on delivering quality in the services provided [1]. The emphasis is on comprehensive testing coverage to maintain high standards and optimize the customer experience.

II. WHAT IS DEVOPS?

DevOps is a methodology employed in the software development and IT industry that merges practices and tools to integrate and automate the work of software development (Dev) and IT operations (Ops), aiming to enhance and expedite the systems development life cycle. The inaugural DevOps Days conference was held in Ghent, Belgium in 2009, founded by Patrick Debois, a Belgian consultant, project manager, and agile practitioner[2].

DevOps aims to foster collaboration, coordination, and communication among previously segregated roles, such as development, IT operations, quality engineering, and security. By embracing a DevOps culture, practices, and tools, teams can enhance their confidence in the applications they build, better meet customer needs, and achieve business objectives more swiftly. It enables teams to consistently deliver value to customers by creating superior and more reliable products. DevOps greatly influences the application lifecycle, spanning the planning, development, delivery, and operations phases. DevOps emphasizes the involvement of all roles in each phase, promoting agility and visibility throughout the entire process.

Numerous organizations and platforms offer DevOps solutions and services. For instance, Microsoft Azure provides Azure DevOps, a collection of contemporary development services that empower teams to plan intelligently, collaborate effectively, and ship rapidly. It encompasses services like Azure Boards for work planning and tracking, Azure Pipelines for continuous integration and delivery, Azure Repos for version control, and more [3].

DevOps transcends a singular set of tools or processes; it represents a cultural transformation and a novel way of working. By breaking down the traditional barriers between development and operations, DevOps empowers teams to collaborate and operate efficiently throughout the complete application lifecycle. It places emphasis on automation, cross-team communication, and continuous improvement to achieve faster delivery of high-quality software. The following principles will help understand what DevOps is really about:

A. Collaboration

DevOps emphasizes collaboration and teamwork between development and operations teams. By breaking down silos and fostering communication, teams can work together effectively throughout the entire development and deployment cycle.

B. Continuous Integration and Delivery

DevOps promotes continuous integration and delivery practices. It involves automating the merging, testing, and deployment of code to enable rapid and frequent releases, reducing time-to-market [4].

C. Customer-Centric Action

DevOps encourages a customer-centric approach, where the development process revolves around the needs and requirements of the customers and end-users. Short feedback loops with real customers are established to ensure the products and services meet their expectations.

D. End-to-End Responsibility

DevOps teams take end-to-end responsibility for the software development lifecycle. Rather than separate teams focusing on individual phases, DevOps teams work together across planning, development, delivery, and operations [5].

E. Cross-Functional Autonomous Teams

DevOps promotes the formation of cross-functional teams that are empowered to make decisions and take ownership of the entire lifecycle of the applications they develop. This approach improves agility and enables faster decision-making.

F. Continuous Improvement

DevOps fosters a culture of continuous improvement, where teams regularly reflect on their processes and seek ways to enhance efficiency and quality. Continuous learning and adaptation are essential for driving ongoing improvements [6].

G. Automation

Automation is a fundamental principle in DevOps. It involves automating repetitive and manual tasks to increase efficiency, reduce errors, and enable faster and more reliable delivery of software [7].

III.TESTING IN DEVOPS

ASoftware testing encompasses all lifecycle activities, both static and dynamic, involved in planning, preparation, and evaluation of software products and related work products. Its purpose is to ensure that these products meet specified requirements, demonstrate fitness for purpose, and identify any defects present [8]. This definition of software testing is globally recognized. In the context of DevOps, these testing activities remain crucial to ensure the quality of a product. Unlike the traditional approach where testing was often treated as a separate phase in the software development life cycle (SDLC) that concluded before product delivery to Operations, DevOps emphasizes the integration of testing from the beginning of the development process [9]. In DevOps, testing is not confined to a single phase but is an ongoing responsibility shared by the entire delivery team. By incorporating testing right from the start and making it a shared responsibility, the aim is to build quality into the product, as described by W. Edwards Deming [10]. This approach, combined with its integration into the entire software development process will ultimately lead to the delivery of a high-quality product.

A. Scope

DevOps introduces a shift in scope, where teams are responsible for the end-to-end quality of their services. This encompasses activities aimed at ensuring the quality of their services, making it arguable that all test activities should be conducted within DevOps teams. The tests performed by the teams should be comprehensive end-to-end test cases that cover the entire scope of their services. However, this can be challenging when a team is accountable for a service that integrates with numerous other services. Business processes often involve multiple services, and end-to-end testing needs to encompass the services of multiple teams. The responsibility for conducting these end-to-end tests can become problematic since they may be owned by multiple teams. We employ contract-based testing, which involves creating a contract for a service and verifying if the service upholds that contract in order to resolve this issue. By imitating a customer or another service, these tests ensure that no contract breaches occur.

The functionality should be covered by the contracts and corresponding tests. However, this approach relies on the contracts aligning with the customer's needs and the absence of errors in the chain of contracts. It is important to have mechanisms in place to handle any discrepancies between contracts, and this is where end-to-end testing can help mitigate the associated risks[11].

In a microservices architecture, contract-based testing is a more suitable strategy since it involves multiple services. In contrast, in a monolithic architecture, it can be more challenging to apply contract-based testing due to difficulties in identifying individual services for testing.

To achieve end-to-end testing that covers multiple teams, small tests can be connected in a framework to cover the entire business process. Each team is responsible for a specific portion of the tests and should provide assistance to other teams to ensure the test chain functions properly. This approach allows for end-to-end testing within DevOps teams but requires extensive coordination and a testing framework that supports collaboration among multiple teams. This team would provide an end-to-end testing service for multiple teams, while the DevOps teams should establish criteria for the tests related to their specific services and take appropriate actions based on the test results. Nevertheless, regardless of the chosen approach, it is crucial to prioritize a customer-centric mindset when forming teams and establishing processes[12].

B. Customer-Centric Testing

The Agile Manifesto marked the beginning of a transformation in software development, with a strong emphasis on the process [4]. In his book "Specification by Example," Gojko Adzic highlights the distinction between "building the product right" and "building the right product" [13]. To achieve quality, it is essential to directly focus on the customer and their requirements [3].

In this context, the first principle of DASA DevOps, Customer-Centric Action, aligns with the need to prioritize the customer [3]. By adopting customer-centric testing, teams can shift their focus towards the customer and their expectations. The initial step in customer-centric testing involves collecting requirements, which then serve as the basis for defining test cases [8]. Specification by Example, also known as Behavior-Driven Development (BDD), provides a framework for dealing with requirements collaboratively [13]. It encourages a shared understanding among the business analyst, developer, and tester, similar to the concept of "The Three Amigos" in Agile development [14].

Specification by Example places the customer perspective at the forefront by utilizing formats such as Given-When-Then, commonly known as the Gherkin language [15]. This format allows for concise and behavior-oriented requirement descriptions, making it easier for both internal and external customers to understand. By leveraging multiple statements in the Gherkin format, a set of examples can be created to identify the desired functionality from a customer's perspective. These examples facilitate discussions between the customer and the DevOps team, fostering a more customer-centric approach [3].

To bridge the gap between the Given-When-Then format and actual tests, various test tools like Cucumber can be used. These tools enable engineers to create test cases using the Gherkin syntax [1]. Given, When, and Then act as keywords in these tools and can be associated with test code to execute the tests [17]. Additionally, these tools maintain the visibility of requirements, facilitating ongoing discussions and test execution reporting. It is important for teams to consider test management and the maintenance of glue code in Cucumber to ensure manageability and avoid excessive workloads [17]. It's worth noting that DevOps does not prescribe the mandatory implementation of Specification by Example or the use of Cucumber as the primary test tooling. Rather, these serve as examples of alternative testing approaches that prioritize a customer-centric mindset [3].

C. Specialized Testing

Specialized tests such as security and performance testing should also be conducted within a team. However, a challenge arises when the team lacks the knowledge and expertise in these areas. It is important to note that in DevOps, engineers are not expected to be superhumans [4]. Engineers may have gaps in their knowledge and skills, especially in specialized fields like security and performance testing. Nevertheless, being a T-shaped engineer in DevOps means constantly expanding one's knowledge and skills. It is possible for an engineer to acquire proficiency in both security and performance testing. Additionally, team members who possess operations expertise can contribute to understanding performance at the server level, while those with development expertise can help comprehend security frameworks and methods in code.

Another approach is to leverage testing services specifically focused on performance and security. However, teams should retain control over the testing process. In larger enterprise organizations, it is common for teams to rely on external security or performance testing services. Clear expectations and communication between the DevOps teams and testing service providers are crucial to ensure the desired testing outcomes are met.

Hybrid approaches are also possible, such as having an engineer with expertise in security/performance testing join the team for a short period or having specialized testing teams educate DevOps teams on the required knowledge. These approaches can help bridge the knowledge gaps within the team. They have a responsibility to maintain control over all types of testing required for their service.

IV. AUTOMATION

Automation is very important for the successful implementation of DevOps. It involves replacing error-prone manual tasks with automated processes. By embracing automation, DevOps teams can achieve fast feedback, enabling them to receive timely information about their processes. Automation accelerates existing workflows and ensures that team members can focus on tasks that require human intervention. Moreover, it helps break down the "Wall of Confusion" by encouraging teams to use the same set of tools throughout the Software Development Life Cycle (SDLC). This promotes collaboration, clarity, and coherence within the team. The diverse skills present in the team can shape the automation to align with the overall team requirements and goals.

Automation in DevOps streamlines software development, delivery, and management processes, leading to the rapid delivery of reliable software and reduced risk to the business. It addresses challenges such as manual processes, siloed work, handoffs, duplication of effort, security risks, and compliance issues. The COVID-19 pandemic has highlighted the importance of automation in DevOps pipelines, prompting increased interest and investment in automation among enterprises. By automating repetitive tasks and establishing standard templates, teams can achieve rapid application development, reduce production costs, improve collaboration, ensure regulatory compliance, and respond quickly to market changes.

A. Test Automation

Test engineers primarily work with testing tools and automation to support their tests. Automation in testing facilitates fast feedback loops, which drive the design and release of software development projects [9]. The Agile Testing Quadrants, initially introduced by Brian Marick and further refined by Lisa Crispin and Janet Gregory, illustrate different types of tests where automation can play a role. These quadrants distinguish between technology-facing tests that support the team (such as unit tests) and business-facing tests that can be automated or conducted manually (such as functional tests). In DevOps, it is recommended to automate functional tests, aligning with the principle of "Automate everything you can" [18]. These functional tests form an essential part of customer-centric testing. On the other hand, technology-facing tests that provide feedback on the project primarily rely on tools and are already automated. By automating the tests in the first three quadrants, teams can allocate time and resources to the last quadrant, which consists of business-facing tests critiquing the product. These tests are typically performed manually and cannot be automated. In a DevOps team, leveraging the diverse skill sets present allows for performing these tests collaboratively, utilizing the time saved through automation. The concept of the test pyramid aids in implementing test automation effectively, emphasizing the execution of fast, low-level tests (such as unit tests) that are suitable for automation. This approach combines tests traditionally conducted by developers (unit tests) and those performed by test engineers (service and UI tests). It aligns well with DevOps as a cross-functional strategy, necessitating knowledge sharing and expertise among team members to fully implement the test strategy. By adopting this strategy, teams can establish a shared responsibility for testing within the team [19].

B. Continuous Testing

Deployment pipelines are essentially the automated implementation of the build, deploy, test, and release process of an application [9]. These pipelines empower teams to have control over their deliverables and enable them to deploy services and ensure quality in an automated manner. In DevOps teams, the goal is to empower all team members to deploy any version on any environment with proper controls in place. By utilizing pipelines, the complexity associated with deploying and testing a service can be minimized. Moreover, when each team member can deploy and test a service with a simple action like pushing a button, the knowledge gap between team members becomes smaller. Continuous testing is an integral part of the pipeline, where tests are executed whenever the pipeline is initiated. Tests act as go or no-go points, determining whether the process can proceed to the next step. This continuous testing provides real-time feedback on the service's quality at different stages of development. Furthermore, testing can be used to verify the correctness of the automation itself, including the deployment automation across different environments. Through deployment testing, teams can take control of the infrastructure and verify its state. Testing enables teams to maintain control over their automation as they increasingly rely on it.

C. Monitoring

Monitoring serves as a means of obtaining feedback from production environments to the teams. Functional monitoring provides insights into how customers perceive the service and enables teams to track customer usage. It can be argued that monitoring can replace certain aspects of testing. When DevOps teams have embraced "the third way," which emphasizes continuous feedback and experimentation, monitoring becomes a valuable tool for gathering feedback. The third way signifies a level of maturity that allows teams to deliver new features quickly and experiment to meet customer needs.

Monitoring serves as a reactive approach to obtaining feedback, while testing takes a proactive approach. Monitoring primarily focuses on production environments, making it a later step in the process, whereas testing can commence early on and provides teams with more options to adapt based on the testing outcomes. Monitoring can act as a form of testing when teams can quickly adapt to the feedback it provides. With the ability to create and deploy new features rapidly, such as through Continuous Deployment practices, teams can react swiftly based on monitoring results. Monitoring can also serve as a means of critiquing the product from a business perspective, aligning with the Agile testing quadrant[7].

Monitoring plays a crucial role in DevOps, contributing to continuous delivery and enabling teams to gain insights into system state, customer experience, and key metrics. It encompasses both monitoring, which involves predefined metrics or logs, and observability, which focuses on exploring properties and patterns not predetermined. To excel in monitoring and observability, teams should ensure they have reporting mechanisms for overall system health, customer experience, and key business/system metrics. The role of monitoring in DevOps extends beyond ensuring high availability and also supports validated learning by tracking usage. Effective monitoring provides a holistic understanding of application behavior and user perception, contributing to a faster and higher-quality software delivery process[18].

In order to improve processes and achieve alignment, gathering feedback from team members is crucial. Feedback loops enable teams to streamline processes, minimize revisions, and simplify workflows. Effective team feedback fosters trust, collaboration, and alignment across the organization.

V. TEST ENGINEER ROLE

In DevOps, a test engineer plays a crucial role as an intermediary between the business and IT. They serve as a bridge, connecting business requirements to the technical implementation on the IT side. This connection facilitates an ongoing conversation between the business and IT teams, ensuring that the customer remains at the center of the action. As a conversation enabler, the test engineer can bridge the gap between development (Dev) and operations (Ops) by emphasizing the importance of quality, which serves as a common thread between the two domains. The test engineer actively participates in this conversation, facilitating the understanding that quality is a shared responsibility within the team. By leveraging their expertise, test engineers can contribute their insights on quality and ensure that different perspectives are combined into a cohesive test strategy.

In DevOps, the responsibility for preparing and executing test cases doesn't solely rest with the test engineers. T-shaped engineers with various areas of expertise should also be capable of performing these tasks. Test engineers can act as coaches, guiding their team members and helping them understand the intricacies of testing. It's important for the test engineer to share their test expertise within the team, fostering a culture of collaboration and knowledge sharing.

Overall, the role of a test engineer in DevOps goes beyond traditional testing practices. They play a critical part in facilitating communication, bridging gaps, and promoting quality throughout the software development lifecycle. By embracing the principles of DevOps and emphasizing the importance of quality, test engineers contribute to the success of the team and the delivery of high-quality software products.

A. T-Shaped, Test Shaped

A T-shaped test engineer should ensure that their own expertise aligns with the requirements of DevOps. Their test expertise should encompass the knowledge and skills necessary to gather accurate system requirements for testing purposes. These requirements serve as the foundation for creating and executing test cases using appropriate test techniques. Additionally, the test expertise should include an understanding of selecting and utilizing the right tools in the testing process. This expertise is consistent with the role of a test engineer outside the DevOps context.

Given the prevalence of automation in DevOps, a test engineer must possess technical skills and knowledge to implement testing within an automated framework. From Test Automation to Continuous Delivery, a test engineer should be able to integrate seamlessly with the team's automation practices. This typically involves having a basic understanding of coding and familiarity with the structure of programming languages. Programming knowledge aligns with the horizontal bar of the T-shaped model, complemented by vertical expertise in Test Automation and complementary test techniques.

Test engineers can also act as intermediaries between the Business and IT teams or different areas of expertise within a DevOps team. This role allows them to acquire knowledge from various team members, enabling their continuous growth. They should strive to expand their horizontal bar by gaining knowledge and insights from different parties in the team, thus enhancing their overall expertise.

B. Soft Skills

Although the current emphasis is on the technical skills required for a test engineer, it is equally important for them to possess strong soft skills. Soft skills are essential for initiating and maintaining effective communication between different parties. As an intermediary, a test engineer must have well-developed people skills in order to fulfill their role and take a leadership position.

Soft skills play a crucial role in facilitating collaboration and building rapport with team members, clients, and stakeholders. Effective communication skills are vital for presenting information clearly, whether through verbal communication in meetings or written communication in emails and documentation. Additionally, stress management and tolerance are valuable soft skills for test engineers, enabling them to effectively handle high-pressure situations and promote a supportive team environment.

Being proficient in public speaking and delivering presentations is also important for test engineers. They often need to present information to various audiences, including team members, clients, and management. Strong presentation skills allow test engineers to convey complex information concisely and confidently. These skills enable effective communication, stress management, public speaking, and overall leadership, which are essential for success in the role.

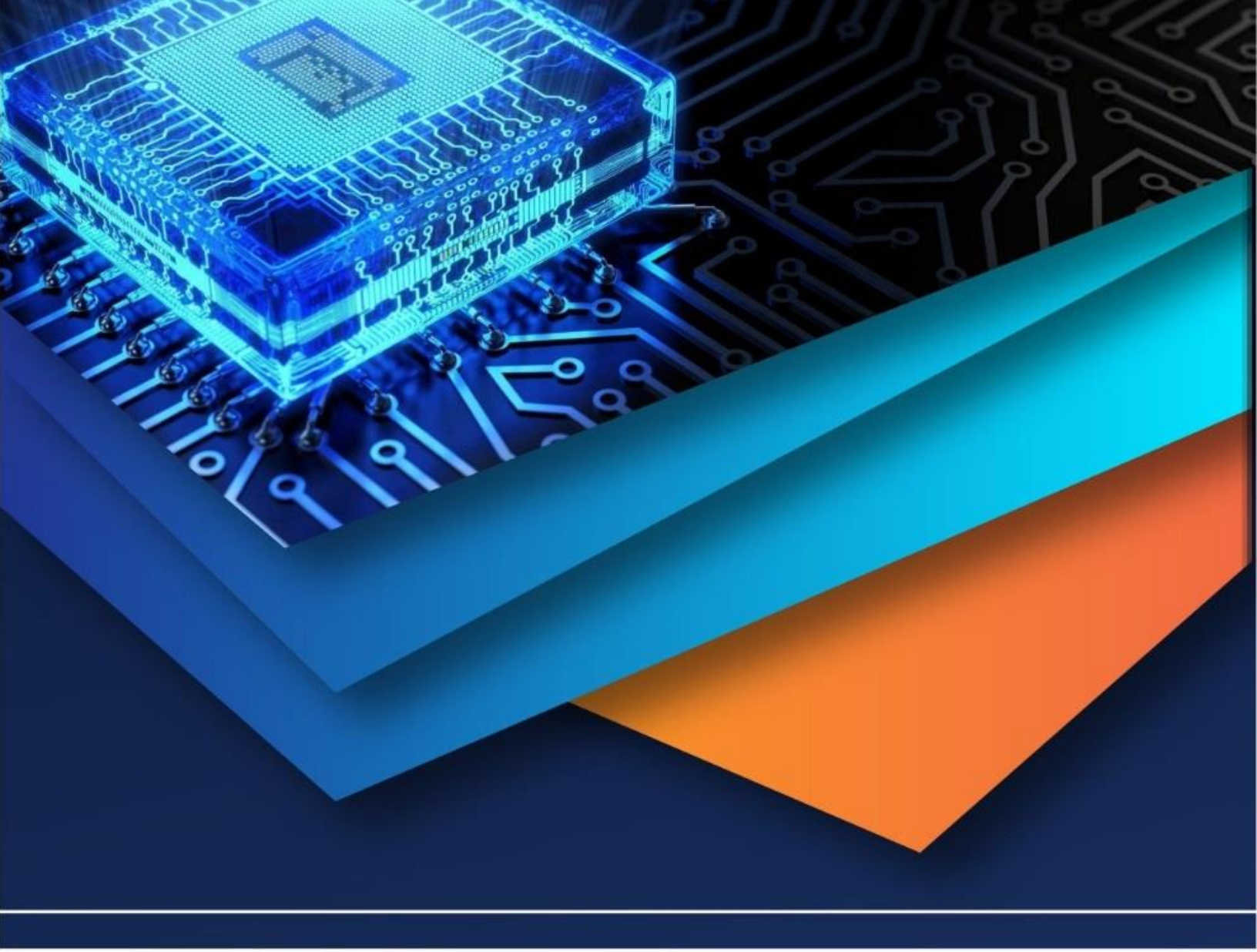
VI. CONCLUSION

In The practice of testing in DevOps follows the same foundational principles as testing in a non-DevOps environment. In DevOps, the entire team takes end-to-end responsibility for a service, which necessitates considering quality requirements for both development and operations. While quality in the Operations section of the software development life cycle (SDLC) was traditionally not included, it is now part of the testing scope in DevOps. Testing in DevOps focuses on the entire functional service delivered by a team, resulting in a different form of end-to-end testing. This expanded scope includes more specialized tests, such as performance and security tests, albeit on a smaller scale. Due to the specialized nature of these tests, they may be executed by a dedicated testing service outside the team. However, DevOps teams should take ownership of these tests and ensure their execution. In DevOps, functional tests emphasize customer-centric testing, ensuring that the quality desired by the customer is at the forefront of the team's work. Automation plays a significant role in DevOps, and it should be leveraged wherever possible in testing. Automation enables teams to receive continuous feedback on their service and the steps taken to add new features. Additionally, monitoring can complement testing by providing quick feedback on the functional and technical aspects of the delivered service. This feedback helps shape the service to meet customer needs effectively.

The role of a test engineer evolves into that of a DevOps Engineer with test expertise. This test expertise, as part of the T-shaped model, encompasses knowledge and skills to implement and teach test strategies within a team. The test expert should be capable of bridging the gap between Business and IT, as well as different areas of expertise within the team, to ensure all quality requirements are met. Responsibility for testing should be shared across the entire team, with the engineer possessing test expertise taking a leadership role and acting as a coach.

REFERENCES

[1] Shafer, A.: Agile infrastructure. Speech presented at Velocity Conference (2009)
[2] Mezak, S.: The origins of DevOps: What's in a name? https://devops.com/the-origins-ofdevops-whats-in-a-name/ (2018). Accessed 24 January 2018
[3] 6 Principles of DevOps. https://www.devopsagileskills.org/dasa-devops-principles/ (2017)
[4] Manifesto for Agile Software Development. https://agilemanifesto.org/ (2001)
[5] Herzberg, F., Mausner, B., Synderman, B.B.: The Motivation to Work. Wiley, New York (1959)
[6] Guest, D.: The hunt is on for the renaissance man of computing. The Independent (1991, September 17)
[7] Kim, G., Behr, K., Spafford, G.: The Phoenix Project: A Novel About IT, DevOps, and Helping Your Business Win. IT Revolution Press, Portland, OR (2013)
[8] International Software Testing Qualifications Board: Certified Tester Foundation Level Syllabus Version 2018 Version. https://www.istqb.org/downloads/send/51-ctfl2018/208-ctfl-2018- syllabus.html (2018)
[9] Humble, J., Farley, D.: Continuous Delivery. Addison-Wesley, Upper Saddle River, NJ (2011)
[10] Deming, W.E.: Out of the Crisis. MIT Press, Cambridge (2000)
[11] Aichernig, B.K.: Contract-based testing. In: Formal Methods at the Crossroads. From Panacea to Foundational Support, pp. 34–48. Springer, Berlin, Heidelberg (2003)
[12] Richardson, C.: What are microservices? https://microservices.io/ (2018)
[13] Adzic, G.: Specification by Example: How Successful Teams Deliver the Right Software, p. 3. Shelter Island, NY, Manning (2012)
[14] Dinwiddie, G.: The Three Amigos: All for One and One for All. Better Software. https://www.stickyminds.com/sites/default/files/magazine/file/2013/3971888.pdf (2011). Accessed November/December 2011
[15] North, D.: Behavior Modification. Better Software (2006, June 5).
[16] Gherkin Reference. https://cucumber.io/docs/gherkin/reference/ (n.d.)
[17] Cucumber. https://cucumber.io/ (n.d.)
[18] Crispin, L., Gregory, J.: Agile Testing. Addison-Wesley, Boston, MA (2008)
[19] Fowler, M.: Bliki: TestPyramid. https://martinfowler.com/bliki/TestPyramid.html (n.d.)



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY