International Journal of Core Engineering & Management
Volume-7, Issue-05, 2023 ISSN No: 2348-9510

THE IMPACT OF AUTOMATED TESTING AND DEVOPS ON SOFTWARE QUALITY: A REVIEW

Nihal Mohammed
Department of Computer Science,
Seattle University Seattle, US
nmohammed1@seattleu.edu

Abstract

The research presented here explores several facets of software quality and how DevOps and automated testing contribute to its enhancement. The paper discusses the necessity of ensuring software's dependability, fault tolerance, safety, and security, which has grown more challenging because of the complexity of software activities. To increase test plan coverage and improve program reliability, the paper references a study that introduces a novel approach to automated software testing that combines memory occupied monitoring and dynamic pseudo-random generation. The report also explores how cloud computing, team collaboration, deployment and test automation, and DevOps, and the popular CAMS methodology, have an impact on software quality. The findings demonstrate that DevOps can greatly raise the quality of software, particularly in terms of dependability, maintainability, and functional appropriateness. The results highlight the importance of software testing and DevOps procedures in creating high-quality software and point to the necessity for additional empirical and qualitative studies in this field.

Keywords: software quality, automated testing, DevOps, software architecture, reliability, fault tolerance, safety, security, CAMS model, deployment automation, continuous delivery.

I. INTRODUCTION

Software quality is a tricky domain that has caused challenges in terms of quantification and improvement for decades. While it comprises various features such as scalability, robustness, maintainability, performance, testability, and portability, the means to effectively measure and enhance software quality have often been elusive. Automation has always been quoted as one of the easier ways of measuring these features [5], and the recent advancements in automation have provided economically friendly ways to address these challenges and achieve higher software quality [1][2].

The main aim of this literature review is to explore different approaches for enhancing and maintaining software quality cost-efficiently. During our search for relevant papers, we discovered a prevalent categorization: automated testing and DevOps. Upon careful analysis of the gathered papers, we identified two key papers from each category. We have defined the following questions to help us achieve the aim of this literature review:

[RQ1] How does automated testing impact software quality?
The aim here is to analyse the findings from the selected research papers and evaluate the extent to which automated testing contributes to improving software quality.

[RQ2] How does DevOps impact software quality?
The aim here is to analyse the selected research papers and understand the role of DevOps in enhancing and maintaining software quality, considering factors such as collaboration, continuous integration and deployment, automation, and feedback loops.

[RQ3] Is there a gap in the software quality features that can be enhanced using automated testing? If yes, where does it fall short?
The aim here is to analyse the selected papers and identify the gaps in the advancements made in recent years to assess areas of improvement for assuring software quality.

This paper consists of six sections, with the current introduction being the first. In Section 2, we will provide a concise analysis of all the selected papers, outlining their main contributions and key findings. Section 3 will focus on synthesizing the collective insights from these papers, highlighting how they contribute to the overarching aim of this review, and answering the research questions defined. Subsequently, in Section 4, we will present an overview of the current state of the art in automated testing and DevOps. Finally, we will discuss the future directions for research in Section 5, followed by a comprehensive conclusion that summarizes the key findings of this paper.

II. OVERVIEW OF CURRENT STATE OF ART

Automated testing, DevOps principles, and the fusion of these methods to enhance software quality, reliability, and efficiency are the current state of the art in software testing and quality assurance. The following are the advancements we have observed in the current state of the art.

Reference [6] explored the future of software quality, with a focus on new trends and challenges. Some of the key topics discussed included the use of artificial intelligence (AI) for testing, the importance of security, and the need for continuous improvement.

Reference [7] proposes a new approach to classifying software testing work in DevOps environments. The authors argue that traditional approaches to testing are no longer sufficient in DevOps, and that a new approach is needed that takes into account the unique characteristics of DevOps.

Reference [8] reviews the literature on the perceived benefits of DevOps implementation. The authors found that DevOps can lead to a number of benefits, including improved software quality, increased customer satisfaction, and reduced costs.

Reference [9] proposes a new approach to assessing and generating higher quality unit test cases. The authors argue that traditional approaches to unit testing are not sufficient, and that a new approach is needed that takes into account the quality of the test cases themselves.

The research papers we have selected also make significant contributions to the field of software quality. For example, paper [1] discusses the use of automated testing to improve software quality. Paper [2] compares the use of different automated testing tools for mobile applications. Paper [3] discusses the use of DevOps to improve software quality. Paper [4] provides a systematic mapping of the literature on DevOps and software quality.

These research papers have had a significant impact on the field of software quality. They have helped to shape the way that software quality is thought about, measured, and improved. They have also led to the development of new tools and techniques for improving software quality. The current state of the art in software quality is constantly evolving. New research is being conducted all the time, and new tools and techniques are being developed. As a result, the field of software quality is constantly changing. This is a good thing, as it means that there are always new opportunities to improve the quality of software.

Here are some of the key trends in software quality that are likely to continue to develop in the future:

- The use of artificial intelligence (AI) for testing. AI is becoming increasingly sophisticated, and it is now being used for a variety of tasks in the software development process, including testing. AI-powered testing tools can automatically generate test cases, identify bugs, and even fix bugs.
- The importance of security. Security is becoming increasingly important as software systems become more complex and interconnected. Software developers need to be aware of the latest security threats and vulnerabilities, and they need to implement security best practices in their software development process.
- The need for continuous improvement. Software quality is not something that can be achieved once and for all. It is an ongoing process that requires continuous improvement. Software developers need to be constantly testing and improving their software, in order to ensure that it meets the needs of their users.

III. ANALYSIS

The analysis section aims to provide an in-depth evaluation of the research papers that have been reviewed. In this section, the research questions, methodology, and findings of the four papers (Appendix A) that were reviewed are summarized to gain a better understanding of our research topic.

Each of the papers reviewed offers a unique perspective on the topic and has used different methodologies to explore the issue at hand. Based on the analysis of each paper, we will draw meaningful conclusions about the topic and provide insights into the key factors that need to be considered when choosing between testing methodology.

A. Software automated testing: A solution to maximize the test plan coverage and to increase software reliability and quality in use [1]

The paper intends to address the growing relevance of software in complex systems and the need to assure its effective operation, fault tolerance, safety, and security. It has become difficult to maintain software quality because of the complexity of software activities frequently outpacing

technology developments. To maximize test plan coverage, raise software stability, and enhance quality in use, the research article suggests a novel method for automated software testing. The goal is to create a technique that combines quick automated tests for researching memory overflow and software defects. The study also aims to show the applicability of the suggested technique and specify the parameters of the software life cycle [1].

1) Research questions:

- What are the effectiveness and efficiency differences between dynamic automated testing and conventional manual testing?
- How does dynamic automated testing affect the dependability and usability quality of software?
- Can the suggested accelerated automated testing approach successfully find memory leaks and serious software bugs?
- Can the proposed method successfully detect memory overflow and software regression?

2) Methodology:

The study takes a black-box approach to software testing, in which the software under test is verified using a set of inputs with known expected outcomes based on functional specifications. A dynamic pseudo-random generation strategy that combines initial test parameters and statistical distributions of field data is used in the suggested test methodology to generate test cases. With this method, various test sequences that mimic real-world situations can be created and adjusted without altering the programming code. Any discrepancies between the test results and the anticipated results from the test cases are examined as potential flaws. The proposed approach also includes the simultaneous evaluation of used memory while running tests, using parameters like working set and private bytes to find memory leaks and overflows. The memory occupied monitoring step aids in spotting possible issues and calculating the Mean Time to Overflow (MTOF) metric, which shows how long it will be before the system runs out of memory resources [1].

3) Findings:

The findings of the research are related to the suggested test methodology. The technique of automated software testing that combines memory occupied monitoring and dynamic pseudo-random generation is successful in maximizing test plan coverage and boosting program reliability. The method offers useful data for quality in use evaluation and maintenance planning by replicating real operating settings and identifying memory leaks and memory overflow. The research uses an industrial application involving an automatic distribution gas station to show the generality of the suggested technique. The results emphasize the value of software testing as well as the potential time and money savings realized by automated test development [1].

B. A Comparative Analysis of Quality Assurance of Mobile Applications using Automated Testing Tools [2]

In this paper, the authors have done a comparative analysis for the different software quality factors.

1) Research questions:

- "What quality factors can be tested using the automated testing tools?"

International Journal of Core Engineering & Management
Volume-7, Issue-05, 2023 ISSN No: 2348-9510

- "Measurement of the overall trends of essential quality factors in the apps under test?"

2) Methodology:

Nine software quality factors have been tested including extensibility, maintainability, performance, scalability, robustness, usability, platform compatibility, testability, and correctness. These factors were specifically chosen because they are the most significant quality attributes. A brief description of these factors has been shown in Fig. 1.

All mobile testing tools including Dynodroid, Evodroid, FSM Droid, MobiGuitar, Renorax, Reran, Appium, Monkey Talk, UIAutomator were used for testing as they are industry level. Using the mentioned software testing tools, each tool was learnt and known which quality factor can be tested using which testing tool. The type of tests performed on each of the tools have been shown in Fig. 2.

3) Findings:

In this comparison of mobile testing tools, Dyno droid is praised for its ability to evaluate usability and discover bugs, which increases dependability and robustness. System testing is the major focus of Evodroid, which increases code coverage and promotes correctness, adaptability, and maintainability. With a focus on GUI testing, FSMDroid enhances performance while assessing usability, accessibility, and robustness. By monitoring concurrency issues and application accuracy, MobiGUITAR improves robustness and fault tolerance. Platform compatibility testing is done by Renorax to guarantee software quality on different platforms and support upcoming additions. Reran evaluates usability, performance, scalability, and security of complex systems using sensor inputs. Robotium evaluates complex GUI elements for usability, accuracy, and performance enhancement. Mobile web application testing for accuracy and user experience is the focus of Appium. Monkey Talk performs cross-platform testing for iOS and Android, checking platform compatibility. UIAutomator automates functional UI tests, ensuring correctness and usability across different devices. Fig. 3 shows the software quality features supported by each tool. Fig. 4 shows the frequency of software quality features that can be tested and enhanced by these testing tools.

Software Quality Factors	Description
Extensibility	Ability of software components to be added, modified and removed easily without badly effecting existing system. Flexibility is its category focused on ability of components to be added easily.
Maintainability	Maintainability is ability to make change for error corrections, supported by defined interfaces, documentations, comments in code.
Performance	Performance is related to acceptable response time.
Scalability	Ability to respond in an acceptable time in increased load or stress.
Robustness	Robustness is the ability of software to keep working and remain available in failure states by backup plans, data and hardware.
Usability	Usability is the ability of user to easily interact with the system using the user interface.
Platform compatibility	Software should run on several platforms like operating systems, browsers etc.
Testability	Testability refers to maximum and efficient code coverage by testing.
Correctness	Correctness is software should conform to with requirements or specifications.

Fig. 1. Software Quality factors along with their definitions [2]

International Journal of Core Engineering & Management
Volume-7, Issue-05, 2023 ISSN No: 2348-9510

Testing Tool	Testing Type	Platform
Dynodroid	Event driven testing	Android
Evodroid	System testing	Android
FSM Droid	GUI testing	Android
MobiGUITAR	GUI testing	Android
Renorax	Compatibility testing	C#, Python, VB.net
Reran	GUI, system, stress, and security testing	Android
Robotium	GUI, system, functional, and acceptance testing	Android
Appium	GUI and functional testing	Android, IOS
MonkeyTalk	Compatibility and functional testing	Android, IOS
UIAutomator	Functional and GUI testing	Android

Fig. 2. The testing tools along with their testing type and the platform supported [2]

Software Testing Tools	Software Quality Factors								
	Extensibility	Maintainability	Performance	Scalability	Robustness	Usability	Platform compatibility	Testability	Correctness
Dynodroid					✓	✓		✓	✓
Evodroid	✓	✓			✓			✓	✓
FSM Droid			✓		✓	✓		✓	
MobiGUITAR					✓			✓	✓
Renorax	✓	✓					✓		
Reran			✓	✓	✓	✓			✓
Robotium			✓			✓			✓
Appium						✓			✓
MonkeyTalk							✓		✓
UIAutomator						✓			✓

Fig. 3. Figure showing the software testing tools along with the software quality factors that they have tested in a mobile application [2]

Fig. 4. Bar Graph showing the frequency of testing tools that support software quality factor [2]

C. Improve Software Quality through practicing DevOps Automation [3]

This paper talks mainly about the effects of using DevOps on Software Quality by developing a research model with five hypotheses to identify the level of correlation between the two.

1) Research question:

The main question that was targeted by the Authors of this paper is:

- How does the use of DevOps affect the quality of software?

Before delving into the methodology. It is important to establish the definitions of a few terms. DevOps is a framework interspersed with cultural values, practices, and tools that help to gap the bridge between the Development and the Operations team during the Software Development Life Cycle to increase the speed of the software releases and improve the software quality.

It is also important to mention the CAMS model that is widely used as a set of baseline values to establish the DevOps culture [3]. CAMS stands for Culture, Automation, Measurement, and Sharing.

2) Methodology:

The research model in this paper consists of a conceptual framework that contains five hypotheses to analyze the effects of DevOps and CAMS on Software Quality.
The following are the hypotheses as mentioned in the paper [3].

- Hypothesis 1: Practice DevOps would improve the Quality of the Software
- Hypothesis 2: Culture of DevOps would improve the Quality of the Software
- Hypothesis 3: Automation in DevOps would improve the Quality of the Software
- Hypothesis 4: Sharing in DevOps would improve the Quality of the Software
- Hypothesis 5: Measurement in DevOps would improve the Quality of the Software

The five hypotheses were derived based on the main aim of this research, which was to identify the effects of the usage of DevOps on Software Quality.

To Qualitative and Quantitative validation of the different elements in the hypotheses, the indicators were identified and valued through an online questionnaire that was answered by professionals who had extensive DevOps experience from fields including Management, Development, Operations, and Quality Assurance [3].

3) Findings:

The result of the questionnaire was the degree of correlation between DevOps and CAMS to Software Quality, which has been summarized in a table given in the paper [3] as shown in Fig. 5 below.

International Journal of Core Engineering & Management
Volume-7, Issue-05, 2023 ISSN No: 2348-9510

		Quality of Software	Relationship
Practice DevOps	Pearson correlation	0.76**	
	Sig. (2-tailed)	0	Strong
Culture	Pearson correlation	0.741**	
	Sig. (2-tailed)	0	Strong
Automation	Pearson correlation	0.758**	
	Sig. (2-tailed)	0	Strong
Measurement	Pearson correlation	0.704**	
	Sig. (2-tailed)	0	Strong
Sharing	Pearson correlation	0.717**	
	Sig. (2-tailed)	0	Strong
**Correlation is significant at the 0.01 level (2-tailed)			

Fig. 5. Figure showing the Pearson's Correlation Coefficient between "Practice DevOps" and "Quality of Software" [3]

The degree of positive correlation of DevOps and CAMS to Software quality is quite strong ranging from 70% to 76%. After running multiple regression analyses were run, the relationship that has been shown in the table was quantified into an equation. [insert citation]
Software Quality = 1.409 + 0.176(C) +0.227(A) + 0.096(M)
+0.172 (S)
Here C stands for Culture, A for Automation, M for Measurement, and S for Sharing.
This concludes the answer to the main question behind this research.

D. DevOps and software quality: A systematic mapping [4]

1) Research questions:

The research questions that have been answered in this paper are:

- What are DevOps objectives which help towards ensuring software quality?
- Will the inclusion of automations in DevOps contribute to software quality?
- Can measurement in the DevOps lead to increased software quality?
- Can sharing in DevOps impact the software quality.
- Does DevOps culture have an impact on software quality?
- Does DevOps enable fast feedback which helps in software quality?
- Does DevOps practice bridge development of software and software quality assurance?
- How software architecture contributes to DevOps success and quality?
- Does continuous delivery in DevOps help in ensuring completion on time along with quality?
- How does DevOps impact usability, efficiency, maintainability, and portability in software?

International Journal of Core Engineering & Management
Volume-7, Issue-05, 2023 ISSN No: 2348-9510

2) Methodology:

The authors have selected papers based on the main inclusion criteria of having the aims of the study revolving around the above-mentioned research questions. These questions deal with the study's objectives, design, variables, research methodology, analysis, data sources, and results in terms of their validity and clarity, as well as their contribution to knowledge and compliance with the study's goal. Fig. 6 shows the final number of papers included after filtration.

Distribution of articles in database.

Database	Total outcome	First results	Final selection
Google Scholar	72	19	10
Web of Science	42	16	3
IEEE Xplore	57	9	4
Scopus	65	55	16
ACM	24	11	2

Fig. 6. Figure showing the distribution of articles in the database [4]

3) **Findings:**

Some of the fields that support DevOps' positive effects on software quality include deployment and test automation. Cloud computing and team cooperation. By using deployment automation and automatic testing there has been a significant improvement in the software quality. Factors like reliability, maintainability, and some functional suitability were also positively impacted when using DevOps. With the help of the DevOps team, they can have a clear goal and work towards it.

With the automation incorporated in DevOps, it can help high performers of the company to concentrate on the technical and innovative challenges and leave the mundane work to automation. Automation helps with swift delivery and automates the software delivery lifecycle; it also helps in business proliferation and improves accuracy.

DevOps facilitate feedback loops and enable the achievement of software quality, this improves the efficiency of the software and the collaboration between development and operations which is the focus of DevOps to improve the product and achieve continuous improvement. Practices like continuous integration and automation assist continuous delivery, enabling effective software delivery with high quality and shorter turnaround times. By integrating software development and quality control, DevOps approaches increase the dependability of applications. The CAMS (culture, automation, measurement, and sharing) factors have a big impact on the quality of software. Include operations professionals and invest in testing, especially test automation, to improve DevOps performance. Automation is a key success element.

The usage of DevOps in software development has expanded significantly, according to examination of the literature on the topic and its effects on software quality. Automation, sharing, and measurement are three DevOps tenets that are strongly correlated with successful software development. DevOps is also known for enabling quick feedback loops, which are essential for producing high-quality software. Additionally, there is a link between quality assurance and the DevOps software architecture.

The study emphasizes that DevOps helps to ensure software quality in a good way. In the context of DevOps, the research has mostly concentrated on automation, culture, continuous delivery, and quick feedback. DevOps is seen as more than just a theoretical idea; it can also be used in real-world applications for producing high-quality software. Although the area is expanding, the number of primary studies on DevOps and software quality is still small. Therefore, additional empirical study as well as qualitative research that is based on surveys is required to compare distinct circumstances in various businesses and nations.

IV. SYNTHESIS OF CORE PAPERS

The first pattern we observed among our papers is that each category – Automated Testing and DevOps – includes a main paper [1][3] that talks about the relationship it has with software quality and two papers that were more of a review style paper [2][4], albeit there is a significant difference between paper [2] and [4], which will be discussed in detail among other things in this section.

Paper [1] proposes accelerated automated testing. Just like hardware gets worn out over time, software can also become obsolete if proper care isn't taken. Unforeseen use cases that the software may face when it is deployed to its environment can cause dynamic defects. The proposed methodology has proven itself to increase the test plan coverage before deploying the product and studying the software regression and memory overflow through it. By incorporating the evaluation of memory overflow during regression testing, the paper aims to decrease testing costs, increase software reliability, and improve Mean Time to Overflow (MTOF) parameter. The MTOF parameter is used to estimate the failure of the system due to memory overflow.

One important observation that can be made from this paper is that it can estimate the MTOF before the product is even deployed, giving the developers some valuable feedback that can be used to increase the reliability and security of the product before deploying.

After having established how automation in testing can positively affect the quality of the Software. Paper [2] brings an interesting approach to a review paper. It evaluates some of the most prominent mobile application testing tools in the market and provides comprehensive statistics of all the main software quality features these tools can test and improve. The most frequently tested features are use ability, correctness, and robustness, while there is a considerable drop in the tools that can test extensibility, maintainability, scalability, and portability. This paper provided us with great insights to help answer RQ3.

Paper [3] talks about a popular DevOps methodology, CAMS – Culture, Automation, Measurements, and Sharing - and uses a conceptual research model to show how every attribute of CAMS positively affects software quality, rather than targeting the individual features in Software Quality. A Pearson's Correlation Coefficient of 0.76 is derived, to show that practicing DevOps has an extremely progressive relationship with software quality.

Paper [4] is a complete systematic mapping of DevOps on Software quality, it defines ten research questions to find answers for everything related to DevOps and how each attribute in DevOps affects the software quality attributes. One field of information that this paper brings in is marrying the concept of automation in DevOps. By automating various tasks and processes within the software development lifecycle, organizations can achieve higher efficiency, faster delivery,

International Journal of Core Engineering & Management
Volume-7, Issue-05, 2023 ISSN No: 2348-9510

and improved quality.

There can be different indicators that show how well the quality of a software is, like Efficiency, Scalability, etc. Although it would be ideal to have a one-size-fits-all situation, where the methods we discussed so far would satisfy all the required quality indicators. Although that is not the reality, Table 1 indicates all the indicators/features that were discussed in each of the papers we have included in this review. We have also added cost efficiency as one of the features as it is an integral part of our aim.

Table I. Software quality features discussed in papers

Feature	Paper 1	Paper 2	Paper 3	Paper 4
Cost efficiency	✓	✓		
Efficiency	✓	✓	✓	✓
Scalability		✓		
Reliability	✓		✓	
Functionality			✓	
Maintainability		✓	✓	✓
Usability		✓	✓	✓
Portability		✓	✓	✓
Extensibility		✓		
Correctness		✓		

The following are the answers we found to our research questions from analyzing the above papers in our review.

RQ1: How does automated testing impact software quality?
Automated Testing has a highly positive impact on software quality by conducting a multifold number of tests as opposed to manual testing in a fraction of the time it would take for the manual testing to be completed [1][2]. It can also be said from paper [1] that accelerated automated tests can find solutions to problems that wouldn't have been interpreted through manual testing. Incorporating accelerated automated tests before deploying can estimate the software regression, thereby increasing the software security, reliability, and customer satisfaction [1].

RQ2: How does DevOps impact software quality?
DevOps is a framework that advocates bridging the gap between Development and Operations to increase the quality of software. Online questionnaires with DevOps experts based on a conceptual research model studying the effects of Practicing DevOps on Software Quality by incorporating the CAMS methodology and software quality indicators as suggested by the ISO 9126 model concluded that they have a largely positive correlation quantified by Pearson's Correlation Coefficient [3]. Paper [4] explores how automation in DevOps further enhances software quality attributes. By automating tasks and processes throughout the software development lifecycle, organizations can achieve higher efficiency, faster delivery, and improved quality.

International Journal of Core Engineering & Management
Volume-7, Issue-05, 2023 ISSN No: 2348-9510

RQ3: Is there a gap in the software quality features that can be enhanced using automated testing? If yes, where does it fall short?

The findings of the paper [2] clearly point out that there is a trend among the features that can be tested using automated testing. According to the statistics collected among the tools evaluated in the paper [2], the feature that is highly tested is correctness with over 80% of the tools, and features such as extensibility, maintainability, platform compatibility and scalability have very low testability with 10-20% of the tools. There is a definite, substantial gap between the testable features of software quality among the papers discussed.

V. FUTURE DIRECTIONS AND CONCLUSION

The studies reported in this study demonstrated that automated testing, DevOps methods, and software quality are interrelated, with the potential to improve software development processes and products. It is crucial for researchers and practitioners to investigate cutting-edge methods and methodologies for addressing the issues connected with software testing and quality assurance as the need for dependable and high-quality software continues to rise. We discuss some potential future research directions and ways to further the state-of-the-art in this field in this section.

Artificial Intelligence and Machine Learning are Integrated in Automated Testing. The advancement of AI and ML techniques has the potential to change the automated testing industry. The use of these methods to enhance test case generation, test optimization, and defect detection can be explored in further research. Researchers might also investigate how AI and ML can be applied to improve how well automated testing tools can adapt to various programming languages, platforms, and domains.

Continuous Monitoring and Analysis of Software Quality Metrics. Software quality metrics can be continuously monitored and analyzed to provide useful information about the security, upkeep, and performance of software systems. Future research can concentrate on developing unique techniques for gathering, analyzing, and displaying software quality indicators in real-time, allowing development teams to identify problems swiftly and effectively.

Improving cooperation and Communication in DevOps. DevOps techniques place a strong emphasis on the value of cooperation and communication between teams working on development, operations, and quality control. To help distributed and multidisciplinary teams work together more effectively to assure software quality, future research can examine novel methods and technologies for enhancing communication and cooperation.

Dealing with Privacy and Security Issues in DevOps and Automated Testing. Security and privacy concerns are becoming more and more crucial as software systems get more intricate and interconnected. Future research might concentrate on integrating security and privacy issues into the automated testing and DevOps pipeline processes to make sure that software products adhere to the relevant security and privacy standards.

Examining the Effect of DevOps on Software design. The connection between DevOps techniques and software design needs more research. Future studies could look at how various architectural patterns and styles assist or obstruct the adoption of DevOps methods, as well as how these

practices can affect how software architecture evolves over time.

Automated Testing Tool Comparative Studies and Benchmarking. With so many automated testing tools on the market, it is important to conduct thorough comparisons and benchmarking studies to assess their respective merits and shortcomings. Such studies can be very helpful in advising practitioners on the best tools to use for their needs and can also point out areas that want improvement.

The field of automated testing and software quality assurance is thus ripe for innovation and investigation. Software quality and dependability will increase because of the integration of cutting-edge technologies, increased stakeholder collaboration, and a focus on security and privacy issues. To address the ever-evolving issues involved in software development and quality assurance, researchers and practitioners must keep collaborating to create fresh approaches and solutions. In doing so, they will guarantee that software systems continue to satisfy users' and organizations' needs in a digital environment that is becoming more intricate and linked.

REFERENCES

1. Marcantonio Catelani, Lorenzo Ciani, Valeria L. Scarano, Alessandro Bacioccola, Software automated testing: A solution to maximize the test plan coverage and to increase software reliability and quality in use, Compter Standards & Interfaces, Volume 33, Issue 2, 2011
2. Anjum, Haneen, Muhammad Imran Babar, Muhammad Jehanzeb, Maham Khan, Saima Chaudhry, Summiyah Sultana, Zainab Shahid, Furkh Zeshan, and Shahid Nazir Bhatti. "A comparative analysis of quality assurance of mobile applications using automated testing tools." International Journal of Advanced Computer Science and Applications 8, no. 7 (2017).
3. P. Perera, R. Silva and I. Perera, "Improve software quality through practicing DevOps," 2017 Seventeenth International Conference on Advances in ICT for Emerging Regions (ICTer), Colombo, Sri Lanka, 2017, pp. 1-6, doi: 10.1109/ICTER.2017.8257807.
4. Alok Mishra, Ziadoon Otaiwi, DevOps and software quality: A systematic mapping, Computer Science Review, Volume 38, 2020, 100308, ISSN 1574-0137, https://doi.org/10.1016/j.cosrev.2020.100308.
5. B. W. Boehm, J. R. Brown, and M. Lipow. 1976. Quantitative evaluation of software quality. In Proceedings of the 2nd international conference on Software Engineering (ICSE '76). IEEE Computer Society Press, Washington, DC, USA, 592–605.
6. Winkler, Dietmar, Stefan Biffl, and Johannes Bergsmann, eds. Software Quality. The Future of Systems-and Software Development: 8th International Conference, SWQD 2016, Vienna, Austria, January 18-21, 2016, Proceedings. Vol. 238. Springer, 2015.
7. Daiju Kato, Ayumu Shimizu, and Hiroshi Ishikawa. 2022. Quality Classification for Testing Work in DevOps. In Proceedings of the 14th International Conference on Management of Digital EcoSystems (MEDES '22). Association for Computing Machinery, New York, NY, USA, 156–162. https://doi.org/10.1145/3508397.3564840
8. Muthia Lazuardi, Teguh Raharjo, Bob Hardian, and Tiarma Simanungkalit. 2022. Perceived Benefits of DevOps Implementation in Organization: A Systematic Literature Review. In Proceedings of the 10th International Conference on Software and Information Engineering (ICSIE '21). Association for Computing Machinery, New York, NY, USA, 10–16.

https://doi.org/10.1145/3512716.3512718
9. Giovanni Grano. 2019. A new dimension of test quality: assessing and generating higher quality unit test cases. In Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA 2019). Association for Computing Machinery, New York, NY, USA, 419–423. https://doi.org/10.1145/3293882.3338984