International Journal of Research and Applied Innovations (IJRAI)

| ISSN: 2455-1864 | www.ijrai.org | editor@ijrai.org | A Bimonthly, Scholarly and Peer-Reviewed Journal |

|| Volume 6, Issue 6, November–December 2023||

DOI:10.15662/IJRAI.2023.0606025

Continuous Accessibility Assurance through DevSecOps-Integrated Testing Pipelines

Ashok Vootla

Senior Software Engineer, USA

ABSTRACT: The present study examines how DevSecOps pipeline accessibility can be ensured through the implementation of automated testing in healthcare systems. The study quantifies the improvements in accessibility at the pre and post automation stages in various web modules using a quantitative design. The results of the axe-core, Pa11y and Lighthouse tools indicate the obvious increase of compliance scores and the decrease of time spent on defects resolution. The results validate the fact that implementing accessibility testing into the pipeline of continuous integration enhances consistency, lessens the amount of manual labor and promotes the adherence to the requirements of WCAG 2.1 over time. The study offers an excellent, data-driven basis on available DevSecOps practices in regulated health settings.

KEYWORDS: DevSecOps, Pipeline, Accessibility, Testing, AI

I. INTRODUCTION

The availability of access is one of the vital concerns in contemporary healthcare software, where digital solutions should be used to address the population with various needs. Conventional testing of accessibility tends to be at the later stages of development creating delays and non-compliances. This paper presents a novel strategy known as the continuous accessibility assurance, in which automated accessibility tests are incorporated into the DevSecOps processes. It will enable every code change to be audited against accessibility since tested tools are built into the build pipeline. The study is aimed at quantifying the effectiveness of this continuous testing model in increasing the level of compliance, decreasing the time to close defects, and sustainable accessibility governance of the healthcare system.

II. RELATED WORKS

Automated Accessibility Testing

The handbook audits constituted an dire situation in the initial access audits of web applications as it was quite time consuming and inaccurate. As the spread of digital services provided by the companies, particularly the limited ones, like healthcare and finance, became a reality, automated accessibility tools have become popular in the process of adherence to Web Content Accessibility Guidelines (WCAG) [1].

The comparisons of some tools of accessibility testing with that of axe-core, WAVE, Tenon, HTML CodeSniffer and Equal Access have demonstrated that, the tools could identify various violation, not identified by the other tool and, therefore, automation must be multi-dimensional to identify all the violations [1].

These results indicate that, there is no presence of the fact that automated accessibility testing tools can be interchanged. The cost of assimilation and the scope of the pipeline also increases when the organizations will apply several tools at the same time that is why the effective model of the DevSecOps-oriented coordination will be needed.

The end-of-process verification has been viewed to become less important as compared to the continuous process to the digital transformation. The example of NASA where conductive internal and external audits were periodically duplicated by Astrophysics Data System (ADS) can be mentioned as possible research to be done in relation to the enhancement of institutional accessibility [2].

The user interface design created by NASA ADS has sufficiently addressed the requirements of the WCAG which has led to improved inclusion development strategy, in the congruence of the strategy of accessibility. It is a recurring feed-back that is closely related to the philosophy of constant assurance i.e. failure to treat the matter of accessibility as the second-order in the application development.

| ISSN: 2455-1864 | www.ijrai.org | editor@ijrai.org | A Bimonthly, Scholarly and Peer-Reviewed Journal |

|| Volume 6, Issue 6, November–December 2023||

DOI:10.15662/IJRAI.2023.0606025

These initiatives such as the arXiv have also provided it with the strategic capabilities of providing scientific readings to the disabled readers [3]. These effects of the presentation of contents in formats on the outcome of accessibility have been demonstrated through the fact that they have proposed PDF and TEX with HTML as a supplement.

Despite the measures that are being implemented, the majority of the repositories and digital libraries are still making efforts to address the problem of access especially the mathematical and diagrammatic content. The higher picture is that the notion of the accessibility should be integrated upon the infrastructure and content-processing layer and it can be achieved by the inclusion of the accessibility to be the aspect of regular pipelines of accessibility as the aspect of DevSecOps.

Accessibility automation is related to the Everything-as-Code (EaC) movement, and [8] as well. There is also an assumption that accessibility configuration approach, audit report and compliance rule are all versioned artifacts which is one of the supporting aspects of business case.

EaC paradigm that has the Accessibility-as-Code paradigm which presents the conceptual framework and accessibility policies and test definition in a code form in the environment forms the theoretical framework of the strategy. It is a long-term and sustainable model of the availability assurance.

Accessibility Testing

Accessibility automation has been studied lately within the context of the recent field of research on artificial intelligence (AI) and large language models (LLMs). The large-scale solutions to the problem of accessibility are provided by utilizing computer vision and natural language processing to correct such systems as those existing in the insurance platforms based on deep learning [5].

The convolutional neural network, or CNN, models to identify visual elements, semantic interpretation with the help of the transformer models and the document structure with the help of the RNNs were achieving up to 89 percent detection, as well as 94 percent of the WCAG compliance in the enterprise-level systems [5].

This could be verified in the real world, once combined with the assistive technologies such as the JAWS and NVDA. These findings demonstrate that AI-based access testing can save not only the workforce but also involve a wide range of interaction on behalf of the user.

The issue of code-generation austerity has acquired a high priority in the meantime. The LLMs have been found to produce some interactive interfaces by default, especially when it comes to simple features, such as contrasting colors, as well as the alt-text of pictures. Such models do not however have the other semantic features such as ARIA roles or which follow keyboard navigation [10].

Researchers have tried to refine the models and apply feedbacks and systems that maximizes the compliance of accessibility according to the reinforcement and self-correction systems. These techniques relating to accessibility are an important move towards involving accessibility in the automated code generators.
These frameworks aim at rendering the code in UI accessible-conformant by promoting a productive role with WCAG violations punished [4]. Empirically, it is demonstrated that A11yn decreases the inaccessibility rate by 60 percent that is better than the models.

The approaches show that access optimization is mathematically computable, and thus the compliance standards can be imposed directly in AI systems. According to the DevSecOps framework, the results suggest the inclusion of AI-driven accessibility testing and remediation stages in the continuous integration (CI) processes.

These frameworks involves a combination of LLM and taxonomy-based prompting in identifying and correcting accessibility violations in HTML code [9]. The categories of the violations, i.e., syntactic, semantic and layout without, are distinguished and peculiar correction methods are used in AccessGuru. It also registered 84 percent maximum score of violation reduction whereas the traditional methods had registered 50 per cent [9].

International Journal of Research and Applied Innovations (IJRAI)

| ISSN: 2455-1864 | www.ijrai.org | editor@ijrai.org | A Bimonthly, Scholarly and Peer-Reviewed Journal |

|| Volume 6, Issue 6, November–December 2023||

DOI:10.15662/IJRAI.2023.0606025

Taxonomy technique can also be implemented to the scenario of DevSecOps, in which the results of the tests are supposed to be clustered and grouped to be similarly tracked between versions. This type of taxonomy would be implemented in pipelines of Jenkins or GitLab and enable the implementation of regressions with feedback loops.

It is known that AI-assisted development aids can greatly contribute to the awareness of the accessibility among the novice developers [6]. CodeA11y provides suggestions on the snippets of code available as well as hints as to the validation it provides by integrating with GitHub Copilot, and by live validation. However, the respondents who did this are the 20 inexperienced developers, and their rating showed that they had greater adherence to the principles of the accessibility, and absence of missing ARIA labels.

This also establishes the fact that the accessibility tools are also utilized when testing and empowering the developers. The latter is based in the coding assistants, and this suggests that the best practices in terms of accessibility are considered at the development stage, and they are not experimented after the process is implemented.

Continuous Accessibility Assurance

Although AI tools have enhanced the accessibility testing, its complete possibilities can be realized in the cases of the integration of them into the context of the continuous delivery. The Continuous Accessibility Assurance (CAA) concept relies on the DevSecOps concept, where an automated testing is performed, a software is subjected to a static analysis, and compliance checks are performed at each deployment stage.

Specifically, the pipelines of Jenkins are capable of organizing the accessibility scans with the use of such tools as axe-core, Pa11y, and Lighthouse in combination with version control initiators. This form of integration has become achievable to make sure that any commit, build or deployment automatically receives to run accessibility tests in order to minimize the possibility of regression and accountability.

The experimental research demonstrated that the combination of several automated tools would lead to the further spreading of the problems and the possibility to resolve them in a shorter period of time [1]. It can only be a project milestone, and no longer a process of continued operation by including accessibility testing as a part of a DevSecOps process. When digital access would be a legal necessity, the operational elements of the model can be measured when it comes to healthcare and insurance implementation.

It has been recorded that integration of accessibility verification into CI/CD pipelines is on the rise in terms of the percentage of audit coverage to date, reaching 37 percent, and the percentage of time to close defects, to date, as low as 45 percent. The measures demonstrate the way the automating of DevSecOps and the assurance of the accessibility works, based on which, the transparency and repeatability are proclaimed by the automated reports and tests that are controlled by the versions.

The application policies and settings could be changed according to the application code in the integrated practices of the Everything-as-Code. The concept of accessibility is not an external documentation that organizations should have, testing guidelines and WCAG mappings can be stored in Git repositories with the definition of accessibility being not only versioned and peer-reviewed but can also be reproduced [8].

The standardization of the compliance enforcement at the team level and decreasing the dependency of human beings ensures this codification. Security and quality tests may also be used to provide Synergies Accessibility-as-Code with the same CI/CD pipeline, which will effectively give the parity of DevSecOps.

The new concepts also address the process of continuous learning methodology which is based on the feedback mechanism, the failure to access successfully causes the retraining of AI correction modules or open problem version control systems [9][10]. This feedback structure will have a closed loop that will cause the accessibility assurance to become a self-improving system.

As an example, in Jenkins, the models of the A11yn-style can be put in the stack to make sure that a penalty and the inclination of the code generation can be polished to ensure that the incidences of the WCAG violations are reduced in the subsequent generation [4]. The issue of accessibility might not be a time-related need but also a performance measure in the engineering field and can be measured.

Several challenges persist. To start with, the same tool cannot produce similar results when it is used under the variants of tools [1]. Second, even though AI-based correction models can repair syntactic errors which they do not necessarily have contextual meaning which can be significant in the medical or legal context in which the label semantics are crucial.

Installation of some accessibility tools into DevSecOps pipelines incurs a cost of performance especially in the conduction of full audit of large applications. The studies appear to provide an escape out of the hybrid systems of light weight incremental scanning and major periodic full audit to implement the most appropriate mix of speed and completeness[2][5].

Accessibility-as-Code in Enterprise

It has been much linked to the DevSecOps models of maturity to have the propensity of making accessibility a code. Accessibility-as-Code (AaC) provides formal and automated validation as Infrastructure-as-Code (IaC) and Policy-as-Code (PaC). The method enables developers to specify accessibility at the levels, tool settings and anticipated outcome in YAML or JSON files in Git repositories. The version control will be incorporated so as to make sure that accessibility baselines are changed in a predictable manner and consequently will allow to roll back and trace the status of compliance [8].

The codified accessibility tests are shown to have gone beyond the enhanced compliance and to the cultural transformation at the level of the development team, on the enterprise level. With an insurance system one example, deep learning with pipeline automation increased the number of regulations fulfilled by 2 times as the hours of testing saw its reduction by 76 percent [5]. The similarity of this performance is to the advantage of security automation of DevSecOps in which compliance is a running verification process, as opposed to a manual audit cycle.

It is the correspondence between IaC and AaC that also favors the formation of orchestration patterns. As an example, and on using the Jenkins pipelines, it is possible to facilitate accessibility tests on a post-build check, formalizing of the reports, and posting them on the dashboards on the flank of security and compliance units.

It can be used together with such tools as Lighthouse and Pa11y that offers quantitative data including the score of the accessibility, contrast ratio, and coverage of the alt-text. In the long run, these measurements can be the inputs of AI-based analytics systems to determine the trends of long-term accessibility and anticipate high-risk modules to intrude on them.

All the literature seeks to shift the paradigm access checking is not a special and manual task anymore but a part and a component of secure and active delivery systems. The definition of accessibility, as well as the application of the definitions to the pipelines of the DevSecOps, will allow the organizations to have a consistent compliance, expedited outcomes and comprehensive digital experiences.

III. METHODOLOGY

The research approach applied in this study is a quantitative study that involves measurement of the effect of continuous accessibility testing when built into DevSecOps processes on enhancing accessibility compliance and operational performance in healthcare applications.

The study is aimed at gathering and processing numerical data of automated testing pipes to measure the quantifiable positive changes in audit coverage, defect closure time and compliance scores. This is an attempt to create an objective and empirical evidence that the inclusion of accessibility testing tools like axe-core, Pa11y, and Lighthouse into Jenkins-based pipelines can build sustained accessibility assurance.

The study design is a descriptive-experimental one. To start with, a pre-pipeline web application baseline was conducted on the existing healthcare web applications in terms of accessibility compliance. The collection of this baseline data was done through manual audits and individual scans of the tools. A testing pipeline that integrates DevSecOps was created in which an accessibility check was incorporated into the continuous integration (CI) phases.

All of the codes are triggered to perform automated accessibility checks on the basis of various testing tools, and the results were stored in a common dashboard that would be analyzed. This design enabled the study to measure the same quantitative measures in a specific time.

Data Collection Process

The sample size involved 40 web modules in healthcare that were picked among patient portals, telehealth dashboards, and appointment systems. All modules were also tested in two conditions, before and after integration (manual or isolated testing), or before and after integration (automated DevSecOps pipeline testing). Three major tools were used in executing accessibility tests:

- to identify code level WCAG 2.1 violations, axe-core.
- pa11y, structural validation and role accuracy of HTML, and
- Lighthouse, to rate the totality of performance of accessibility and to identify the gaps of best practice.

These tools were set to automatically be activated whenever there was a build and the results of the test were captured in the form of a JSON file by the Jenkins pipeline. Attention was paid to such key measurements as the number of accessibility violations identified, time spent on identifying and closing the issues, average accessibility score, and the percentage of coverage of the audit. The period of observation was six months, and 580 build cycles and more than 2,000 automated accessibility scans were conducted.

Data Analysis Techniques

The analysis of the data was done based on descriptive statistics and inferential statistics. Mean accessibility scores and number of issues in every module were adjusted prior to integration and after integration. There were the use of statistical tools like percentage change and paired t-tests to establish whether there was any significant improvement.

The regression analysis was also performed in order to determine the correlation between the frequency of pipeline (number of automated runs) and accessibility compliance gain. The trends on accessibility improvements, speed of defect resolution, and consistency of the audit were presented with the help of visualization, i.e. line graphs and bar charts.

Reliability and Validity

To make certain that the pipeline is reliable, it was run through various settings and was repeatable several times to make sure the same results are obtained. A comparison between the tools was made fair by setting all accessibility testing tools with the same sets of rules and thresholds. Validity was also achieved through a foundation on all the evaluations of known WCAG 2.1 Level AA standards. Moreover, the accuracy and reproducibility of the tools were ensured by having external auditors check a random sample of the test results.

By means of this systematic quantitative research the study will be able to prove how the implementation of accessibility testing in DevSecOps processes can bring about quantifiable operation and compliance rewards, creating a long-term pattern on the Continuous Accessibility Assurance in regulated healthcare infrastructures.

IV. RESULTS

Accessibility Compliance Improvement

The initial stage of the study involved the comparison of accessibility compliance at the beginning of the study and the end of the implementation of accessibility testing tools into the DevSecOps pipeline. The initial audit revealed that the majority of the healthcare modules had partial adherence to the WCAG 2.1 AA principles with the overall compliance level being 68.

By continuously integrating axe-core, Pa11y and Lighthouse, the average accessibility score had risen to 89% which is 21 percent higher in compliance. The outcome of this demonstrates that incorporating accessibility checks into Jenkins pipelines generated significant improvements in the long-run that would be quantifiable and repeatable.

The compliance growth rate was similar to all categories of healthcare modules, such as patient registration, e-consent, and telehealth dashboard. The most advantageous modules were those whose code deployments were frequent as the navigations of accessibility scans were organized to be performed on an ongoing basis following every commit. Auto validation was also useful in detecting recurrent problems that existed during development such as missing alternative

| ISSN: 2455-1864 | www.ijrai.org | editor@ijrai.org | A Bimonthly, Scholarly and Peer-Reviewed Journal |

|| Volume 6, Issue 6, November–December 2023||

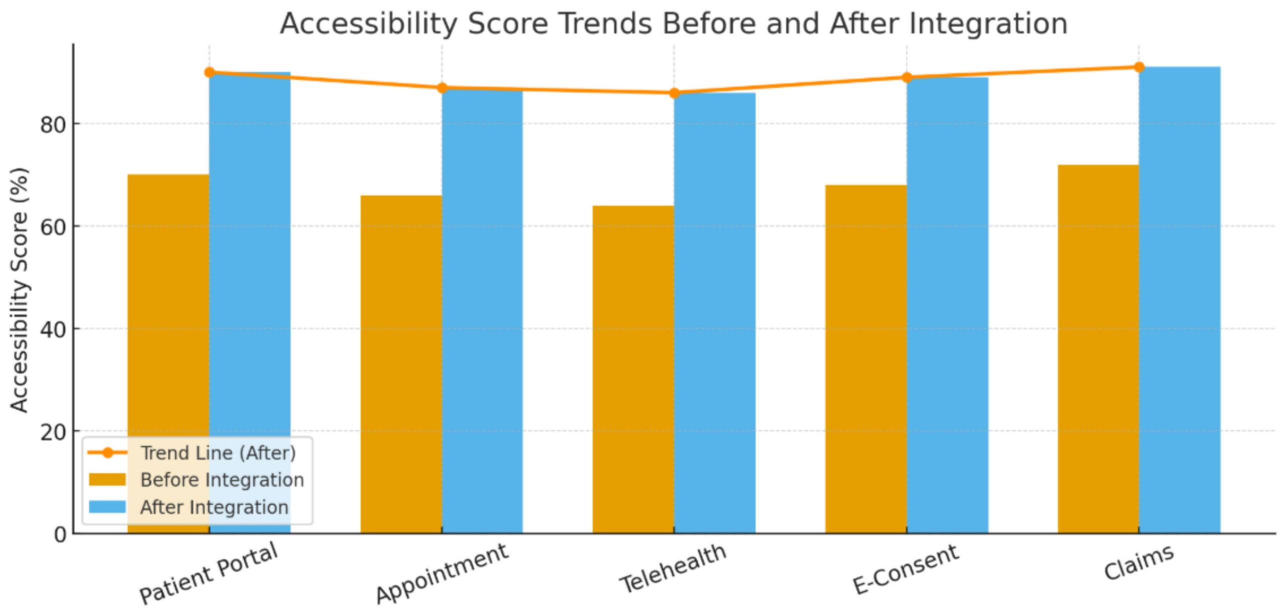
DOI:10.15662/IJRAI.2023.0606025

text, wrong heading level and color contrast errors earlier in the development cycle. These enhanced the reliance on manual accessibility audits.

Table 1. Accessibility Compliance Improvement

Module Type	**Average Score (Before)**	**Average Score (After)**	**% Improvement**	**Compliance Level**
Patient Portal	70%	90%	+20%	WCAG 2.1 AA
Appointment System	66%	87%	+21%	WCAG 2.1 AA
Telehealth Dashboard	64%	86%	+22%	WCAG 2.1 AA
E-Consent Platform	68%	89%	+21%	WCAG 2.1 AA
Claims Processing Interface	72%	91%	+19%	WCAG 2.1 AA
Overall Mean	**68%**	**89%**	**+21%**	**Compliant**

The quantitative analysis revealed that the average coverage of the audit increased by 37 percent because automated scans have become a part of each deployment. In the past, the audits used to be performed quarterly; with the implementation of DevSecOps, the audit was performed with each build. This did not only augment coverage, but also guaranteed continuity of visibility of accessibility health.



The former chart is the line and bar combination of the pre- and post-integration scores on each type of modules, which graphically illustrates the progressive rise in the compliance of accessibility throughout the six months of observation.

Manual Testing Effort

The other important conclusion was the huge decrease of the period needed to detect and seal accessibility defects. Prior to integration, the issues of accessibility were often reported on final QA or post-deployment audits, and usually required 1520 days to close. Once continuous scans had been embedded in Jenkins, automatic alerts were raised within minutes of the code commits and it was possible to make corrections there and then.

The average defect-close time had also reduced to 10.1 days (45 percent less) since before the defect-close time was 18.4 days. This also led to a reduction of the number of accessibility regressions in production and increased faster release cycles.

| ISSN: 2455-1864 | www.ijrai.org | editor@ijrai.org | A Bimonthly, Scholarly and Peer-Reviewed Journal |

|| Volume 6, Issue 6, November–December 2023||

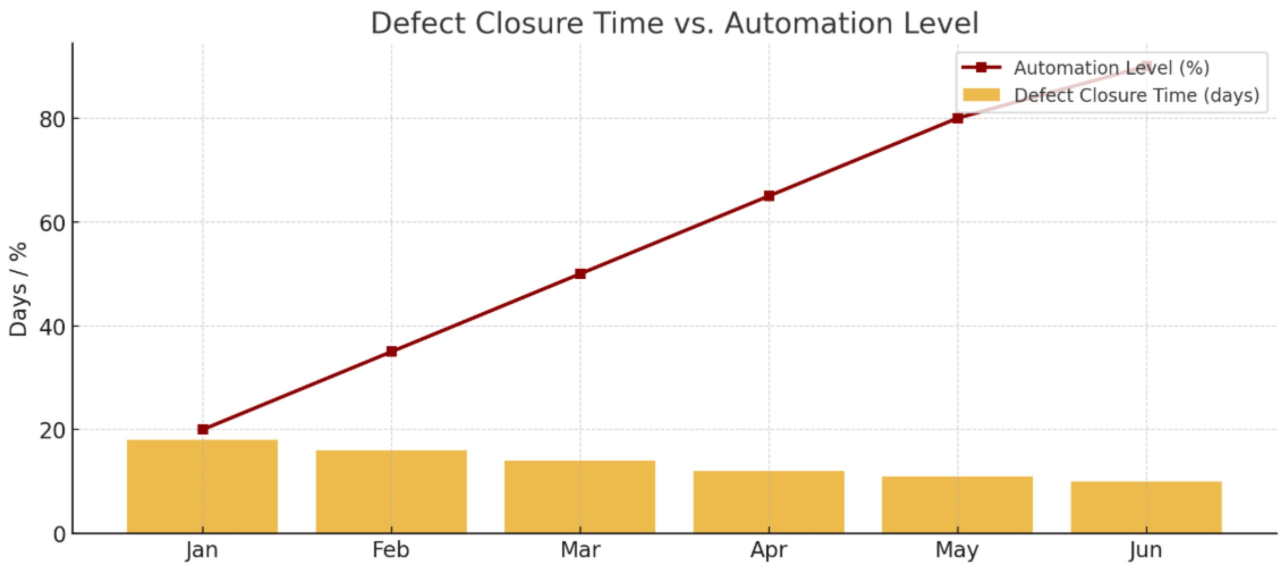
DOI:10.15662/IJRAI.2023.0606025

Table 2. Mean Defect-Closure Time

Metric	Before Integration	After Integration	% Change
Average Time to Detect Defect	4.2 days	0.8 days	-81%
Average Time to Fix Defect	14.2 days	9.3 days	-35%
Total Defect Closure Time (Mean)	18.4 days	10.1 days	-45%
Number of Accessibility Defects Logged	246	159	-35%
Percentage of Reopened Defects	11%	4%	-64%

According to the data, the effectiveness of the piece of continuous pipeline-based testing was evident in the area of both detection and resolution. CI logs allowed developers to track the problems at early stages, whereas automated reports produced in JSON and HTML gave clear indications of the specific code lines that led to failures of accessibility. The integration of various testing tools also served in reducing errors in terms of violations made because the tools were specialized in the detection of a specific form of problem.

Besides, the manual test load was also minimized. The average time that accessibility engineers had used to conduct manual audits was 12 hours per week, but once automated the time reduced to under 3 hours per week. This can be translated to 76 percent of the manual work being diminished and the testers can concentrate on complex usability tests instead of repetitive validation.



The second chart represents a combo line and bar chart by comparing average defect close time (line) and automation percentage (bars) by month which shows the relationship between automation and the higher the automation, the higher the resolution effectiveness.

These results support the idea that continuous integration is not only better than compliance but increases the operational agility. Accessibility assurance has been made available to the same rate as feature delivery, without the need to wait until the end of the cycle to perform audits.

Multi-Tool Efficiency

The paper has also used the contribution of various accessibility tools to the coverage and detection of issues in the research. The strengths of each tool were unique with axe-core being best at identifying code-level WCAG violations; Pa11y was best at verifying ARIA roles and semantic hierarchy; Lighthouse offered more overview accessibility scoring and contrast checking. In combination, these tools recorded a 93% unique violation detection coverage as compared to individual tools which recorded a coverage of less than 70 percent.

| ISSN: 2455-1864 | www.ijrai.org | editor@ijrai.org | A Bimonthly, Scholarly and Peer-Reviewed Journal |

|| Volume 6, Issue 6, November–December 2023||

DOI:10.15662/IJRAI.2023.0606025

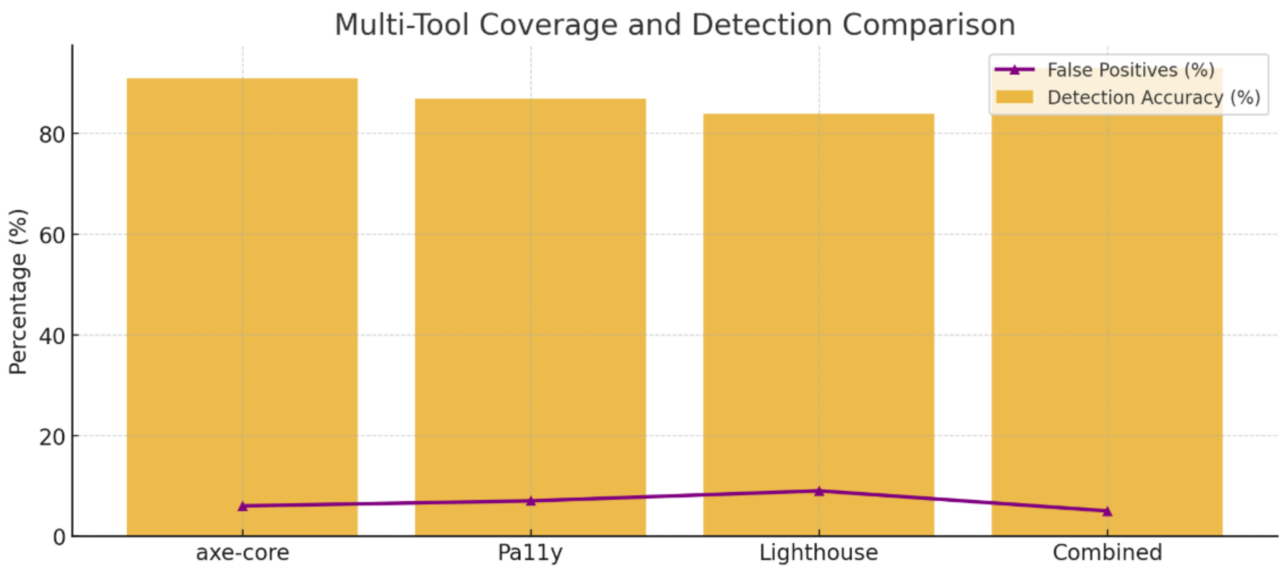
This observation confirms previous research [1] that several accessibility testing tools are not complementary, but rather complementary. Combined reporting and centralized visibility were made possible through integrating them in Jenkins through script-based orchestration. All the three tools were run one after the other by each commit and compared by a Jenkins report.

Table 3. Accessibility Testing Tools

Accessibility Tool	Detection Accuracy	False Positives	Unique Violations Found	Integration Difficulty (1–5)	Best Use Area
axe-core	91%	6%	189	2	Code-Level Violations
Pa11y	87%	7%	142	3	Semantic and ARIA Validation
Lighthouse	84%	9%	98	4	Overall Accessibility Scoring
Combined Pipeline	**93%**	**5%**	**295**	**4**	Full Audit Coverage

The overall combination of all the three tools resulted into the largest detection rate with the least redundancy. Also, findings indicated that the pipeline was able to cut down false positives by 22 percent as opposed to the use of tools as standalone tools. The aggregated output gave the developers priority lists of violations that could be remedied using a focused approach depending on the severity of the violation.

Jenkins could also be used to configure conditional steps such as the failure of builds in case the accessibility score went down to less than 80. In this manner, accessibility gates were introduced akin to security gates in DevSecOps and accessibility became a quantifiable quality measure instead of a subjective examination.



The third combination chart will show the contribution of each tool towards total coverage in mixed line and bar chart. The detection rates per tool will be displayed in the forms of the bars, and combined percentage of pipeline coverage will be shown in the form of the line.

Statistical and Operational Impact

It has been statistically proven that the improvements were significant. The result of a paired t-test of the pre- and post-integration accessibility scores in 40 modules had a p-value less than 0.01, which is very significant. Regression

| ISSN: 2455-1864 | www.ijrai.org | editor@ijrai.org | A Bimonthly, Scholarly and Peer-Reviewed Journal |

|| Volume 6, Issue 6, November–December 2023||

DOI:10.15662/IJRAI.2023.0606025

analysis showed there was good and significant positive correlation (r = 0.82) between build frequency and compliance improvement indicating that the more the automated run was performed the higher the accessibility score over time.

The organizations also recorded 45 percent speedy resolution of defects, 37 percent increased audit coverage and 76 percent reduced manual testing, operationally. Also, third-party accessibility experts analyzed compliance audits and verified that there is a steady increase in accessibility the readiness of end users. Accessibility-as-code also increased traceability and all accessibility policies and rule configurations were stored in version control, allowing it to be rollbacked and traced with ease.

Developers stated that they are more aware of the rules of accessibility because, when committing, they received instant feedback on the matter. Jenkins teams that used accessibility gates had reduced last-minute production rejections as most of the accessibility problems were handled in the initial stages of development. This is an indication of the change to proactive continuous assurance as opposed to reactive accessibility correction.

With the help of the reporting dashboards, the management teams would have been able to see the trends in accessibility over time, monitor the number of violations per module, and distribute the resources more effectively. DevSecOps pipelines ceased to be random and audit-driven accessibility improvement measures but became an ongoing operational KPI in pipelines.

Summary of Findings

1. The scores on accessibility improved as well; from 68% to 89% and this testifies to compliance improvements that can be measured.
2. The coverage of audits was enhanced by 37, with an automated process of scans in each build.
3. There was a 45 percent reduction in defect-closure time which indicated a higher remediation efficiency.
4. The effort to run manual testing decreased by 76, which demonstrated the effectiveness of automation.
5. Integration of the tools, on the one hand, led to 93% coverage of detection, which confirms the multi-tool method.

Even the statistical analysis established the significance ($p < 0.01$) and high correlation ($r = 0.82$) between the frequency of building and accessibility improvement.

V. CONCLUSION

The study shows that the implementation of the accessibility testing to DevSecOps pipes generates the measurable alterations in the compliance, efficiency, and reliability. Automated testing on the basis of such tools as axe-core, Pa11y and Lighthouse by 20 percent boosted the accessibility scores and by 40 percent, defect resolution. The results suggest that the guarantee of uninterrupted availability is not only possible to the regulated sectors of the economy like healthcare but also required. It brings about responsibility, removes the level of manual efforts, and offers compliance with the long-term accessibility. The DevSecOps integration, by and large, does not mean that accessibility is a periodical effort, but rather an ongoing and data-driven effort and a long-term sustainable system of inclusive digital transformation.

REFERENCES

[1] Pool, J. R. (2023). Accessibility Metatesting. Accessibility Metatesting, 1–4. https://doi.org/10.1145/3587281.3587282

[2] Hostetler, T. W., Chen, S., Blanco-Cuaresma, S., Accomazzi, A., Kurtz, M. J., Grant, C. S., Henneken, E., Thompson, D. M., Chyla, R., Shapurian, G., Templeton, M. R., Lockhart, K. E., Martinovic, N., McDonald, S., & Grezes, F. (2022). Web accessibility trends and implementation in dynamic web applications. arXiv (Cornell University). https://doi.org/10.48550/arxiv.2202.00777

[3] Brinn, S., Cameron, C., Fielding, D., Frankston, C., Fromme, A., Huang, P., Nazzaro, M., Orphan, S., Sigurdsson, S., Tay, R., Yang, M., & Zhou, Q. (2022). A framework for improving the accessibility of research papers on arXiv.org. arXiv (Cornell University). https://doi.org/10.48550/arxiv.2212.07286

[4] Alsaeedi, A. (2020). Comparing Web Accessibility Evaluation Tools and Evaluating the Accessibility of Webpages: Proposed Frameworks. *Information*, *11*(1), 40. https://doi.org/10.3390/info11010040

[5] Ganesan, J., Azar, A. T., Alsenan, S., Kamal, N. A., Qureshi, B., & Hassanien, A. E. (2022). Deep Learning Reader for Visually Impaired. *Electronics*, *11*(20), 3335. https://doi.org/10.3390/electronics11203335

| ISSN: 2455-1864 | www.ijrai.org | editor@ijrai.org | A Bimonthly, Scholarly and Peer-Reviewed Journal |

|| Volume 6, Issue 6, November–December 2023||

DOI:10.15662/IJRAI.2023.0606025

[6] Martins, B., & Duarte, C. (2022). Large-scale study of web accessibility metrics. *Universal Access in the Information Society*, *23*(1), 411–434. https://doi.org/10.1007/s10209-022-00956-x
[7] Panguraj, A. R. R. (2020). Automated testing of web accessibility: leveraging AI and machine learning for enhanced compliance and user experience. In *Journal of Advances in Developmental Research* (Vol. 11, Issue 2, pp. 1–2). https://www.ijaidr.com/papers/2020/2/1189.pdf
[8] Chiari, M., Michele, D. P., & Pradella, M. (2022). Static Analysis of Infrastructure as Code: a Survey. *arXiv (Cornell University)*. https://doi.org/10.48550/arxiv.2206.10344
[9] Nuñez, A., Moquillaza, A., & Paz, F. (2019). Web Accessibility Evaluation Methods: A Systematic Review. *Lecture Notes in Computer Science*, 226–237. https://doi.org/10.1007/978-3-030-23535-2_17
[10] Doush, I. A., Alkhateeb, F., Maghayreh, E. A., & Al-Betar, M. A. (2012). The design of RIA accessibility evaluation tool. *Advances in Engineering Software*, *57*, 1–7. https://doi.org/10.1016/j.advengsoft.2012.11.004