International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 10 Issue: 02 | Feb - 2026 SJIF Rating: 8.659 ISSN: 2582-3930

Review on Automated Software Quality Assurance Using CI/CD Tools

Tanay D. Jaybhaye[1], Amol D. Wakhare[2]

[1]*Student, Department of CSE, Deogiri Institute of Engineering and Management Studies, Chh. Sambhajinagar, Maharashtra, India*

[2]*Asst. Professor, Department of CSE, Deogiri Institute of Engineering and Management Studies, Chh. Sambhajinagar, Maharashtra, India*

[1]tanayjaybhaye1561@gmail.com, [2]amolwakhare@dietms.org

Abstract- Modern software development requires rapid delivery while maintaining high quality standards. Traditional Software Quality Assurance (SQA) approaches relied heavily on manual testing and late defect detection, resulting in increased costs and delayed releases. With the adoption of Agile and DevOps practices, Continuous Integration and Continuous Deployment (CI/CD) pipelines have emerged as essential mechanisms for automating software development and quality assurance processes. This review paper examines existing techniques, tools, and practices used for automated software quality assurance in CI/CD environments. It discusses the role of automated testing, static code analysis, security scanning, and quality gate mechanisms in maintaining software reliability, maintainability, and security. The study also reviews commonly used CI/CD tools and quality assessment platforms that support continuous quality monitoring. The paper identifies major benefits of automated quality assurance, including early defect detection, faster feedback cycles, improved collaboration, and reduced manual effort. However, it also highlights limitations such as fragmented quality reports, dependency on fixed threshold rules, and lack of holistic quality evaluation methods. The review concludes that integrated and automated quality evaluation frameworks are necessary to support objective and consistent release decision-making in modern DevOps environments.

Key Words: CI/CD Pipeline, Software Quality Assurance, DevOps, Automated Testing, Quality Gates, Static Code Analysis.

1. INTRODUCTION

In modern software engineering, rapid development cycles, distributed teams, and continuous delivery expectations have fundamentally transformed how software systems are built and maintained. Traditional Software Quality Assurance (SQA) methodologies—characterized by manual testing phases conducted after development—are no longer sufficient to meet the demands of agile, scalable, and high-availability systems. As organizations increasingly adopt DevOps practices, automation has become central to ensuring reliability, security, and performance without slowing down deployment pipelines. This paradigm shift has led to the emergence of Automated Software Quality Assurance (ASQA) integrated within Continuous Integration/Continuous Deployment (CI/CD) environments.

Continuous Integration (CI) and Continuous Deployment (CD) represent foundational DevOps practices that aim to reduce integration issues, accelerate software delivery, and improve system stability. CI enables developers to frequently merge code changes into a shared repository where automated builds and tests are triggered. CD extends this process by automating deployment to staging or production environments. When integrated effectively, CI/CD pipelines serve as automated quality gates that continuously validate software artifacts before release.

Automated Software Quality Assurance refers to the systematic use of tools, scripts, and pipelines to automatically validate software functionality, performance, security, and maintainability. Unlike traditional QA approaches that rely heavily on manual intervention, ASQA integrates testing, static analysis, code quality checks, vulnerability scanning, and compliance validation directly into the CI/CD workflow. This shift from reactive defect detection to proactive quality enforcement significantly reduces technical debt, minimizes production failures, and improves overall software resilience.

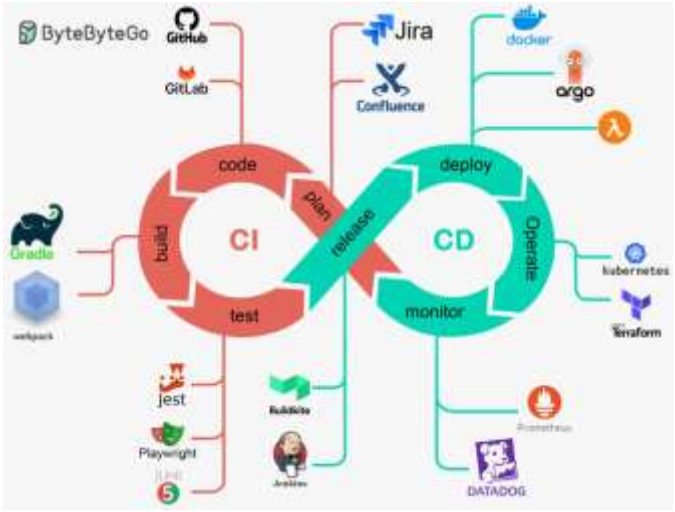

Fig.1: CI/CD Process Overview

The need for automation in quality assurance is driven by several industry trends. First, agile development methodologies promote iterative releases with shorter sprint cycles, often delivering updates weekly or even daily. Second, micro services architectures introduce distributed systems complexity, increasing the risk of integration failures. Third, cloud-native deployments and containerization technologies require continuous monitoring and validation of infrastructure configurations. Under such conditions, manual testing becomes a bottleneck, whereas automated QA embedded within CI/CD pipelines enables continuous validation at scale.

Despite the widespread adoption of automated CI/CD pipelines and quality analysis tools, current software quality evaluation practices remain fragmented. Most existing approaches assess quality using isolated metrics such as code coverage, defect counts, or vulnerability severity levels. These metrics are typically evaluated independently using predefined threshold-based quality gates, which do not capture the multidimensional nature of software quality.

As a result, development teams often struggle to interpret multiple tool-specific reports, leading to subjective, inconsistent, and time-consuming release decision-making processes. Furthermore, existing CI/CD quality gate mechanisms lack integrated frameworks capable of aggregating diverse quality metrics into a unified and objective assessment model.

The absence of a comprehensive quality scoring approach creates a significant research gap in automated software quality governance, highlighting the need for unified multi-criteria evaluation frameworks capable of providing holistic quality assessment and supporting automated release decisions in CI/CD environments.

2. LITERATURE SURVEY

1. Continuous Integration and DevOps Automation

Recent research highlights that Continuous Integration and Continuous Deployment (CI/CD) play a major role in modern software quality assurance. DevSecOps practices integrated within CI/CD pipelines enable automated security testing and early vulnerability detection, significantly improving system reliability and compliance [1]. Studies also show that automation in CI/CD pipelines improves developer productivity, reduces manual effort, and shortens release cycles [2].

Systematic literature reviews indicate that CI adoption improves defect detection, collaboration, and development transparency [3]. Organizations using automated pipelines achieve faster delivery cycles and improved software reliability [4]. Modern DevOps environments rely heavily on CI/CD automation to maintain continuous quality monitoring throughout the software lifecycle [5]. Earlier foundational research established that automated pipelines reduce defect density and improve deployment reliability [6].

2. Static Code Analysis and Software Quality Measurement

Recent studies emphasize the role of continuous quality assurance practices that integrate static code analysis with automated testing and monitoring tools in CI/CD pipelines [7]. These integrations significantly reduce defect leakage into production systems and improve overall software quality.

Static analysis tools such as SonarQube analyze source code to detect bugs, vulnerabilities, and maintainability issues without executing the program [8]. Integrating static analysis into CI pipelines improves defect detection and reduces maintenance costs [9], [10]. However, these tools generate independent metrics rather than a unified quality score, making overall quality interpretation difficult.

3. Code Coverage and Testing Quality Metrics

Recent research highlights the importance of shift-left testing approaches, where testing activities are performed early in development to reduce defect fixing costs and improve software reliability [11]. Integrating automated testing frameworks into CI/CD pipelines significantly improves code coverage, defect detection rates, and release speed [12].

Code coverage metrics are widely used to evaluate testing effectiveness. Tools such as JaCoCo measure the percentage of executed code during testing [13]. Although coverage improves defect detection, it alone cannot guarantee software quality, and effective evaluation requires integration with other quality indicators such as reliability and maintainability [14], [15].

4. Software Security Assessment in CI/CD Pipelines

Recent research shows that DevSecOps practices integrate automated security scanning directly into CI/CD pipelines to detect vulnerabilities early and ensure compliance with security standards [1]. Early integration of security assessment significantly reduces remediation costs and improves software reliability [16].

Dependency vulnerability scanning tools developed by OWASP help identify risks in third-party libraries [17]. However, most pipelines still treat security metrics independently rather than integrating them into comprehensive quality evaluation models.

5. Multi-Criteria Software Quality Evaluation Models

Recent research emphasizes that combining multiple quality indicators into a unified evaluation model provides more accurate software quality assessment [18]. Traditional frameworks such as ISO/IEC 25010 define software quality characteristics including reliability, maintainability, and security [19].

Earlier research proposed multi-criteria frameworks for integrating software metrics into unified evaluation models [20], [21], but these models lack practical implementation within automated CI/CD environments.

6. Dynamic Quality Gates in DevOps Environments

Recent studies suggest that dynamic quality gates based on weighted evaluation models provide realistic readiness assessment and support objective release decisions [22]. Automated pipelines also improve developer productivity and deployment performance [2].

Traditional quality gates rely on static threshold rules to determine software readiness [23], which often leads to inconsistent and subjective release decisions.

3 LIMITATIONS OF EXISTING APPROACHES

Despite significant research advancements, several limitations remain:

- Quality tools provide isolated metrics rather than unified evaluation
- Release decisions rely on static threshold rules
- Lack of integration between testing, security, and maintainability metrics
- Absence of automated quantitative quality scoring models
- Dependence on manual interpretation of reports

These limitations highlight the need for integrated automated quality evaluation systems capable of generating a single quantitative quality score.

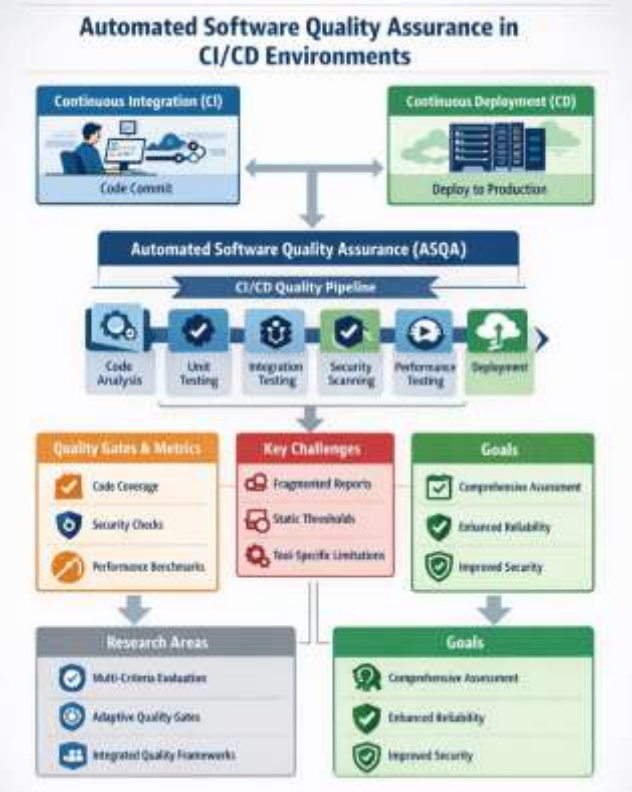

Fig.2: ASQA Overview in CI/CD Environments

3. RESEARCH GAP IDENTIFIED

Based on the literature review, several research gaps are identified. Existing CI/CD pipelines rely on independent tools such as static analyzers, coverage tools, and vulnerability scanners, but there is no integrated framework capable of combining their outputs into a single comprehensive quality assessment. Current quality gates evaluate readiness using isolated metrics rather than overall software quality.

Most systems depend on static threshold rules that fail to capture trade-offs among quality attributes. Although theoretical models define multiple quality characteristics, they are rarely implemented using mathematical evaluation techniques in automated pipelines. Security assessment is also typically

performed independently and not integrated into holistic quality decision frameworks.

These limitations highlight the need for an automated, integrated, and quantitative quality evaluation system capable of generating objective release decisions based on multiple quality indicators.

4. PROBLEM STATEMENT

In modern software development environments, Continuous Integration and Continuous Deployment (CI/CD) pipelines are widely adopted to automate the processes of building, testing, and deployment. These pipelines integrate multiple software quality assessment tools, including static code analyzers, code coverage tools, and security vulnerability scanners, to evaluate various aspects of software quality.

Despite their widespread use, existing CI/CD quality evaluation approaches remain fragmented. Each tool generates independent and isolated reports related to specific quality attributes such as testing effectiveness, maintainability, reliability, and security risks. Consequently, development teams are required to manually interpret multiple quality metrics to assess overall software readiness. This manual evaluation process is time-consuming, subjective, and often leads to inconsistent and unreliable release decisions.

Furthermore, traditional quality gate mechanisms rely on fixed threshold values for individual metrics, such as minimum code coverage levels or allowable vulnerability counts. These static rule-based approaches fail to capture the multidimensional nature of software quality, where multiple quality attributes collectively influence overall software readiness.

Another significant limitation is the absence of integrated frameworks capable of automatically aggregating diverse quality metrics into a unified evaluation model. The lack of an automated quantitative quality scoring system restricts organizations from performing objective, data-driven release decision-making in fast-paced DevOps environments.

Therefore, there is a critical need for an integrated and automated software quality evaluation framework that can collect heterogeneous quality metrics, combine them using a unified scoring model, and support consistent and objective release decisions within CI/CD pipelines.

5. CONCLUSION

This review paper examined existing research on automated software quality evaluation in CI/CD environments. It analyzed key areas including CI/CD automation, static analysis, testing metrics, security assessment, and dynamic quality gate models.

The study identified significant limitations in current approaches, particularly the lack of integration between quality tools and the absence of unified quality scoring frameworks. Most existing systems rely on isolated metrics and static threshold rules, leading to subjective and inconsistent release decisions.

The findings emphasize the need for integrated, automated, and data-driven quality decision systems capable of combining multiple software quality indicators into a single comprehensive evaluation model. Such systems can improve release reliability, reduce manual effort, and support effective quality governance in modern DevOps environments.

REFERENCES

[1] A. M. Ferreira, M. A. Brito, and J. de Lima, “Software Quality in an Automotive Project: Continuous Inspection,” *Empirical Software Engineering*, Springer, 2024.

[2] K. Ramdass and S. Jain, “The Role of DevSecOps in Continuous Security Integration in CI/CD Pipelines,” Journal of Quantum Science and Technology, vol. 6, no. 2, pp. 45–58, 2025.

[3] S. Chittala, “Impact of CI/CD Automation on Developer Productivity and Software Quality,” Journal of Software Engineering Applications, vol. 17, no. 1, pp. 12–24, 2024.

[4] S. Soares, R. Oliveira, and M. Ribeiro, “Adoption of Continuous Integration and its Impact on Software Quality,” Information and Software Technology, vol. 134, 2021.

[5] D. Lopez and J. Garcia, “Quality Improvement using Automated CI/CD Pipelines in DevOps,” Journal of Software Engineering Research, vol. 11, no. 3, pp. 201–215, 2023.

[6] Y. Zhang, H. Li, and X. Wang, “Automated Quality Assurance in DevOps-Based CI/CD Pipelines,” Journal of Systems and Software, vol. 188, 2022.

[7] N. Forsgren, J. Humble, and G. Kim, Accelerate: The Science of Lean Software and DevOps, IT Revolution Press, 2018.

[8] E. Rahman and L. Williams, "Continuous Quality Assurance in DevOps," IEEE Software, vol. 38, no. 5, pp. 72–79, 2021.
[9] SonarSource, "SonarQube Documentation," 2020. [Online]. Available: https://docs.sonarqube.org
[10] J. Bell, "Static Code Analysis for Software Quality Improvement," IEEE Computer, vol. 51, no. 4, pp. 80–85, 2018.
[10] D. Spinellis, Code Quality: The Open Source Perspective, Addison-Wesley, 2006.
[11] R. Kumar and A. Singh, "Shift-Left Testing in CI/CD Environments," International Journal of Software Engineering and Applications, vol. 14, no. 2, pp. 33–45, 2023.
[12] J. Wang, L. Chen, and Y. Liu, "Automated Testing in CI/CD Pipelines," IEEE Software, vol. 39, no. 3, pp. 55–63, 2022.
[13] JaCoCo Team, "JaCoCo Documentation." [Online]. Available: https://www.jacoco.org
[14] S. McConnell, Code Complete: A Practical Handbook of Software Construction, Microsoft Press, 2004.
[15] T. Mens and T. Tourwé, "A Survey of Software Refactoring," IEEE Transactions on Software Engineering, vol. 30, no. 2, pp. 126–139, 2004.
[16] R. Pressman and B. Maxim, Software Engineering: A Practitioner's Approach, McGraw-Hill, 2015.
[17] OWASP Foundation, "OWASP Dependency Check Documentation." [Online]. Available: https://owasp.org
[18] N. Fenton and J. Bieman, Software Metrics: A Rigorous and Practical Approach, CRC Press, 2014.
[19] ISO/IEC 25010, Systems and Software Quality Requirements and Evaluation (SQuaRE) — Software Quality Model, ISO, 2011.
[20] M. Lanza and R. Marinescu, Object-Oriented Metrics in Practice, Springer, 2006.
[21] Sharma and S. Gupta, "Automated Software Quality Monitoring using CI/CD Pipelines," International Journal of Software Engineering and Applications, vol. 12, no. 3, pp. 45–56, 2021.
[22] H. Yang, K. Zhou, and F. Liu, "Testing Maturity and Deployment Performance in CI/CD," Journal of Systems Engineering, vol. 27, no. 4, pp. 301–312, 2022.
[23] L. Bass, I. Weber, and L. Zhu, DevOps: A Software Architect's Perspective, Addison-Wesley, 2015.