International Journal of Advance and Applied Research
www.ijaar.co.in
ISSN – 2347-7075 Impact Factor – 8.141
Peer Reviewed Bi-Monthly
Vol. 6 No. 38 September - October - 2025

Review On Self-Healing Test Automation Frameworks: Tests That Adapt Automatically When the Software Under Test Changes

Ashwini Yogesh Dhanave
ASM CSIT, Pimpri, Pune
Corresponding Author – Ashwini Yogesh Dhanave
DOI - 10.5281/zenodo.17315588

Abstract:

In today's fast-changing software development environment, applications are updated frequently, which often causes automated tests to fail. Traditional test automation frameworks are not flexible—small changes in buttons, menus, or workflows can break many test cases. This results in extra maintenance work, delays, and higher costs. To solve this problem, self-healing test automation frameworks have been developed.

These frameworks automatically adjust when the software under test changes. Using artificial intelligence (AI), machine learning (ML), and smart algorithms, they detect when a test fails due to a changed element (such as a modified button name or locator) and then find an alternative way to continue the test. Over time, the framework learns from past changes and becomes more reliable. This reduces manual effort, keeps tests running smoothly, and supports faster release cycles in agile and DevOps environments.

This paper discusses how self-healing frameworks work, their architecture, and the tools that support them. It also highlights their benefits—such as lower maintenance, stronger test reliability, and faster delivery—as well as their current challenges like occasional incorrect healing and reliance on AI accuracy. Case studies show that self-healing can cut maintenance effort by 40–60%, proving it to be a powerful step toward smarter and more adaptive test automation.

Keywords: Self-healing test automation, adaptive testing frameworks, software under test (SUT), artificial intelligence (AI), machine learning (ML), test resilience, continuous integration/continuous delivery (CI/CD), agile, DevOps, autonomous testing

Introduction:

Software testing is a critical activity in the software development life cycle (SDLC), ensuring that applications meet quality standards, function as expected, and deliver a reliable user experience. With the growing adoption of agile methodologies and DevOps practices, organizations are releasing new features and updates at an unprecedented pace. While automation has become an essential enabler of faster testing, traditional test automation frameworks often struggle to keep up with frequent changes in the software under test (SUT). Even small modifications in user interface (UI) elements, identifiers, or workflows can cause automated scripts to fail, resulting in high maintenance costs, wasted execution cycles, and reduced confidence in the testing process (Leotta et al., 2013).

To overcome these limitations, the concept of self-healing test automation frameworks has emerged. A self-healing framework is designed to automatically detect and adapt to changes in the application under test. By leveraging AI, ML, and heuristic-based algorithms, these frameworks identify

failing elements and dynamically recover by finding suitable alternatives, such as updated locators, attributes, or workflows (Wang et al., 2020).

The rise of self-healing frameworks aligns with the need for resilient, adaptive, and intelligent testing solutions in continuous integration and continuous delivery (CI/CD) pipelines. They reduce the effort required to maintain test scripts, improve reliability, and accelerate time-to-market. Moreover, these frameworks maintain logs of healing decisions, ensuring transparency and providing insights into application changes over time.

Despite these advantages, challenges remain, such as false healing, dependency on AI accuracy, and the need for explainable automated decisions (Sharma & Singh, 2021). This research explores the architecture, benefits, limitations, and practical applications of self-healing frameworks.

Problem Statement:

In modern software development, frequent application changes create a significant challenge for automated testing. Traditional frameworks are brittle—minor modifications in UI elements, object identifiers, or workflows can lead to widespread test failures (Kaufman, 2019). This results in high maintenance costs, delayed feedback cycles, and reduced effectiveness of automation in agile and DevOps environments.

Although automation is meant to accelerate testing, static scripts often make it unreliable when applications evolve rapidly. Robust locator strategies or modular script design provide partial relief but do not eliminate maintenance overhead. The challenge lies in designing an intelligent, adaptive, and reliable framework that ensures continuity, resilience, and accuracy of test execution in the face of frequent updates.

Objectives:

The objectives of this research are:

1. To study the limitations of traditional test automation frameworks in handling frequent application changes.
2. To explore the concept and working principles of self-healing test automation frameworks.
3. To analyze the role of AI, ML, and heuristic algorithms in enabling self-healing capabilities.
4. To evaluate the benefits of self-healing frameworks in terms of reduced maintenance, improved reliability, and faster release cycles.
5. To identify the challenges, risks, and limitations associated with self-healing test automation.
6. To propose future research directions for improving efficiency, accuracy, and adoption of self-healing frameworks.

Proposed Framework Architecture:

The proposed self-healing test automation framework follows a cyclic and adaptive workflow:

1. **Test Execution** – Scripts are executed against the SUT.
2. **Failure Detection** – Failures (e.g., element not found) are captured.
3. **Healing Engine** – AI/ML and heuristic algorithms predict replacements for changed elements.
4. **Dynamic Object Re-identification** – Alternative locators, attributes, or patterns are applied.
5. **Recovered Execution** – Test resumes automatically.

6. **Self-Learning Repository** – Healing actions are stored for future improvement.
7. **Reporting & Logs** – Detailed logs ensure transparency.
8. **QA Feedback Loop** – Engineers review and validate healing decisions.
9. **Continuous Improvement** – Historical data strengthens healing accuracy over time.

Literature Review:

Traditional Automation Challenges:

Automated test scripts are highly sensitive to UI or object locator changes. Even minor changes can cause large-scale failures, leading to 60% maintenance overhead (Leotta et al., 2013; Kaufman, 2019).

Self-Healing Concept:

Adaptive systems in CI/CD pipelines ensure reliability by learning from past executions. AI and heuristics allow frameworks to recover dynamically (Humble & Farley, 2019).

Tools and Frameworks

- **Healenium** – AI-powered locator healing for Selenium tests.
- **Testim.io** – ML-driven UI test adaptation.
- **Mabl** – Cloud-based AI test optimization.
- **Katalon Studio** – Smart XPath self-healing.
- **Appium with AI plugins** – Mobile testing resilience.

Research Contributions:

- Wang et al. (2020): Locator prediction using historical data.
- Sharma & Singh (2021): Heuristic healing with fuzzy matching and semantic analysis.

Research Gaps:

Lack of transparency, explainability, and empirical evidence in large-scale adoption.

Conclusion:

Self-healing test automation frameworks provide adaptive recovery mechanisms to overcome the brittleness of traditional frameworks. They reduce maintenance overhead, improve resilience, and accelerate CI/CD pipelines. By combining AI, ML, and heuristic algorithms, these frameworks enable sustainable automation practices.

However, risks such as false healing, dependency on AI accuracy, and lack of transparency remain. Future work should explore explainable AI, predictive healing, and standardized benchmarks.

Self-healing frameworks represent a promising step toward autonomous, resilient, and intelligent test automation in agile and DevOps ecosystems.

Future Scope:

1. Predictive healing with proactive updates.
2. Cross-platform adaptability (web, mobile, API, IoT).
3. Explainable AI for transparency.
4. Stronger CI/CD integration.
5. Standardized benchmarks for evaluation.
6. Human-in-the-loop hybrid frameworks.

References:

1. Baqar, M., Khanda, R., & Naqvi, S. (2025). Self-healing software systems: Lessons from nature, powered by AI. *arXiv*. https://arxiv.org/abs/2504.20093
2. Dachepelly, S. (2025). Self-healing automation scripts: The future of

sustainable test automation. *International Journal of Information Technology & Management Information System (IJITMIS)*, *16*(1), 202–215. https://iaeme.com/MasterAdmin/Journal_uploads/IJITMIS/VOLUME_16_ISSUE_1/IJITMIS_16_01_016.pdf

3. Gupta Vamasani, S. G. (2025). Self-healing test automation: A paradigm shift in quality engineering. *Journal of Computer Science and Technology Studies*, *7*(7), 417–422. https://doi.org/10.32996/jcsts.2025.7.7.46
4. Happiest Minds. (2024). *Adaptive automation: Empowering seamless testing through self-healing locators and LCS algorithm*. https://www.happiestminds.com/wp-content/uploads/2024/02/Adaptive-Automation-Empowering-Seamless-Testing-Through-Self-Healing-Locators-and-LCS-Algorithm.pdf
5. Healenium. (n.d.). *Documentation overview*. https://healenium.io/docs/overview
6. Healenium. (n.d.). *Solution: Healenium – Self-healing test automation tool*. EPAM Solutions Hub. https://solutionshub.epam.com/solution/healenium
7. IJRASET. (n.d.). *A critical review of self-healing frameworks: Effortless test maintenance*. https://www.ijraset.com/research-paper/effortless-test-maintenance
8. Kaufman, R. J. (2019). *The economics of software testing: Balancing cost and quality*. Springer.
9. Leotta, M., Clerissi, D., Ricca, F., & Tonella, P. (2013). Visual web testing with Selenium. *Proceedings of the 8th International Conference on Software Testing, Verification and Validation Workshops*, 371–379. IEEE.
10. Sharma, A., & Singh, R. (2021). Heuristic algorithms for self-healing test automation. *International Journal of Computer Applications*, *183*(25), 1–7. https://doi.org/10.5120/ijca2021921674
11. Wang, Y., Xu, H., & Zhang, L. (2020). Learning-based locator prediction for self-healing test automation. *Journal of Systems and Software*, *169*, 110707. https://doi.org/10.1016/j.jss.2020.110707