The American Journal of Engineering and Technology
ISSN 2689-0984 | Open Access

TYPE Original Research
PAGE NO. 222-229
DOI 10.37547/tajet/Volume07Issue06-23

Integrating AI Tools Into the Continuous Testing Process.

Vladyslav Korol
Software Developer Engineer in Test Penn Entertainment
Miami, USA

OPEN ACCESS

SUBMITED 18 April 2025
ACCEPTED 24 May 2025
PUBLISHED 30 June 2025
VOLUME Vol.07 Issue 06 2025

CITATION
Vladyslav Korol. (2025). Integrating AI Tools Into the Continuous Testing Process. The American Journal of Engineering and Technology, 7(06), 222–229. https://doi.org/10.37547/tajet/Volume07Issue06-23

Abstract: This paper utilizes AI tools to enhance the ongoing test cycle in a DevOps environment, thereby creating Metabase. This data architecture is robust and scalable, supporting a highly responsive release process. The project is vital since the releases have become more frequent; standard automation has already reached its limit, increasing the costs of maintaining scripts and, consequently, resulting in a significantly higher total cost due to the discovery of many more defects. The novelty of this work is grounded in an approach to choosing and applying AI tools through comparative analysis over available commercial and open-source platforms, supported by content analysis of empirical use cases and quantitative assessments from industry reports. Herein, a methodology is presented that consists of architectural solutions for data lake organization, continuous model training scenarios, and ML endpoints integrated into the CI/CD pipeline, which hosts Predictive Test Selection, as well as self-healing and test-case prioritization mechanisms. It narrates the creation of the prompt-engineer position and the connections between QA/ML experts and organizational facets. This paper discusses the application of clear AI measures for risk assessments. The final calls shown here indicate that intelligent automation enables reducing a regression set to a barely necessary size while maintaining 99.9% bug detection and minimizing false alert failures. This, in turn, leads to improvements in MTTR and TCO quality. TestPilot and FlakeFlagger verify Meta's practices; furthermore, it is anticipated that forecasts will retain a CAGR of 20.9% in the global AI testing market growth. Solution maturity, encompassing both SaaS and on-premises models, offers a flexible choice to regulated industries. Metabase architecture is shown in which raw and processed data are kept separately to ensure timely model retraining as well as to minimize computational costs. This article will be helpful for software architects, QA managers, DevOps teams, and ML engineering specialists involved in building scalable and resilient testing architectures.

Keywords: artificial intelligence, continuous testing, DevOps, self-healing, Predictive Test Selection, data lake, ML integration, software quality, Metabase.

INTRODUCTION

The transition to a DevOps model has made delivery speed the primary competitive factor, with continuous testing serving as its built-in safeguard. Today, 83% of developers already participate in DevOps practices, and every build undergoes a chain of automated checks to prevent defects from leaking into production at high release velocities [1]. Traditional automation no longer copes with the increased frequency of changes. Such routine tasks consume up to half of the entire automation budget, turning the test suite into a bottleneck of the pipeline and forcing teams either to slow down or to release code with reduced coverage. This imbalance between development speed and test inertia not only lengthens feedback loops but also increases the total cost of defects detected late.

It is precisely at these points that a demand for smart tools arises. The implementation of AI-oriented platforms yields savings in the overall testing budget and adds efficiency to test runs through more precise test-case selection and extended coverage. Self-healing mechanisms, capable of automatically correcting selectors or API routes, reduce script maintenance costs, freeing engineers' resources for more valuable tasks. The case studies on regression testing with AI frameworks demonstrated a reduction in cycle duration of up to 30%, directly accelerating release delivery without compromising quality [2]. This, in turn, gives one the ability to leverage artificial intelligence not as a trendy extra, but as the natural and upcoming way in the evolution of continuous testing, which would maintain the balance between DevOps pipeline velocity and product reliability.

MATERIALS AND METHODOLOGY

The study of AI tool assimilation for ongoing testing was conducted through a review of 19 works, comprising three academic papers [4–6], market reports [1, 3, 19], and technical overviews [13-16]. Studies on Predictive Test Selection [4] and LLM-driven test generation [5] form the theoretical base. The research on self-healing approaches to reducing flaky failure complements the framework [6]. Industry reports have shown that traditional automation loses effectiveness as release frequency increases [1, 3] and that the AI-testing market is rapidly developing [19]. Additional data were provided on the functional capabilities of market leaders—Applitools, Testim, Functionize, Mabl, and Tricentis [8–12]—while the TestRail report [13] offered quantitative evaluations of tool usage.

Methodologically, the work includes four components. First, a comparative analysis of platforms: evaluation of the depth of the AI layer (self-healing, generation, visual regression) and licensing schemes (SaaS vs. on-premises) for Applitools, Mabl, Testim, Functionize, Tricentis, as well as open-source projects Retecs and DeepOrder [8–12, 14]. Second, a Grey Literature Review [7] allowed the identification of the prevalence of self-healing and test-prioritization patterns among commercial solutions. Third, a content analysis of empirical cases reveals the effectiveness of Predictive Test Selection at Meta [4], the median coverage of 70.2% by LLM-generated tests at TestPilot [5], and the reduction of false failures using FlakeFlagger [6]. Fourth, an analysis of the numbers in the industry reports: how often test-runs are done and changing amounts of tool shares from TestRail [13], predictions for how well the AI-testing market will do [19], and a look at GPU infrastructure costs for keeping up training and inference [15–16, 18].

The technical portion of the study is supported by architectural overviews of data lakes [14] and continuous model training methodologies [15, 16]. Scenarios for integrating AI modules into CI/CD are considered, including sending a hash and test list to an ML endpoint, ranking a subset for execution, and self-healing requests upon UI step failures [15]. Risk assessment accounted for explainable AI metrics [17] and economic factors (growth in GPU-instance expenses) [18]. Organizational aspects—such as the roles of prompt engineers and interactions between QA and ML engineers—are described based on pilot implementation practices and the monitoring of key metrics (MTTR, code-to-production velocity, and the share of defects reaching production) [17].

Thus, the combination of platform comparative analysis, grey-source review, content analysis of cases, and quantitative industry data has enabled the development of a concise yet comprehensive methodology for implementing AI tools in continuous testing, encompassing technology selection, organizational preparation, and technical integration.

RESULTS AND DISCUSSION

The transition from mechanical automation to intelligent automation is explained by two interrelated factors. First, companies have already exhausted cost savings on scripts: 68% of organizations in the recent World Quality Report 2024 stated that without generative AI further expansion of automation becomes economically unfeasible, whereas implementation of GenAI promises to compensate for the growth of the regression suite through intelligent selection and repair of tests [3]. Second, the test data itself has become a rich source of defect correlations, and machine learning models can extract patterns that are not readily available to manual analysis.

The qualitative difference between automating routine steps and intelligent automation is evident at the task level. Suppose the former forces the machine to repeat pre-described scenarios. In that case, the latter tasks AI with decision-making: which part of the suite to run, how to fix a broken selector, or which code fragments appear potentially defective. DevOps business metrics measure the practical effect. Thus, the Predictive Test Selection strategy deployed in the Meta monorepository reduced the volume of executed tests to one-third of the original, preserving 99.9% defect coverage and halving infrastructure costs, which directly accelerated the code-to-production cycle [4]. Simultaneously, large language model–driven test generation demonstrates a median coverage of 70.2% of code lines in the TestPilot project, outperforming classical heuristics with no copy-paste from existing tests [5].

This is also accomplished by lowering false-positive breakages. The FlakeFlagger tool increased the precision of identifying flakes in 16 out of 23 industrial projects; consequently, teams could detect actual defects caused by environmental instability at a faster rate [6]. All these impacts together form a new Total Cost of Ownership (TCO): the part of the budget that was previously allocated for script upkeep and long runs is now redistributed to feature creation.

The particular situations of AI use in constant testing encompass the entire journey of ensuring quality is achieved. With the rise of intelligent automation, large language models examine how features are signed or user records and create unit as well as complete end-to-end (E2E) tests. This progress has been particularly significant in recent efforts, such as GPTTestGen and TestPilot. Then comes self-healing: models check the current DOM against the baseline one and automatically pick new locators. According to a multi-year review [7], self-healing is the most common AI pattern among one hundred commercial tools.

Next is prioritization: reinforcement learning algorithms, such as Retecs or stochastic classifiers from the Meta approach, select a minimal yet bug-rich subset of the regression suite, which allows for keeping run duration within minutes even with a sharp increase in code. Predictive testing fills remaining gaps: an analysis of coverage metrics and failure history forms a map of hotspots, where additional checks are directed even before code review. The fight against flaky tests is built on clustering logs and identifying unstable steps. Generative analytics automatically produces an executive summary for management, thereby reducing the cognitive load required for interpreting results.

Thus, AI shifts focus from manual script support to semantic risk management. The combination of intelligent generation, self-healing, prediction, and prioritization simultaneously accelerates release flow. It increases the reliability of feedback, thus directly improving speed-to-release, reducing Mean Time To Resolution (MTTR), and optimizing the total cost of quality ownership.

The global market for AI-oriented testing tools has already moved beyond the niche novelty phase and entered the saturation phase. A meta-analysis [7] counted over one hundred commercial and academic solutions, with Applitools, Testim, Functionize, AccelQ, and Mabl being the five most mentioned in the industry. According to [19], the global AI-enabled testing market size was estimated at $856.7 million in 2024 and is projected to increase from $1,010.9 million in 2025 to $3,824.0 million by 2032, showing a compound annual growth rate (CAGR) of 20.9% during the forecast period. North America dominated the global AI-enabled testing market, accounting for a 35.27% share in 2024, as shown in Fig. 1.

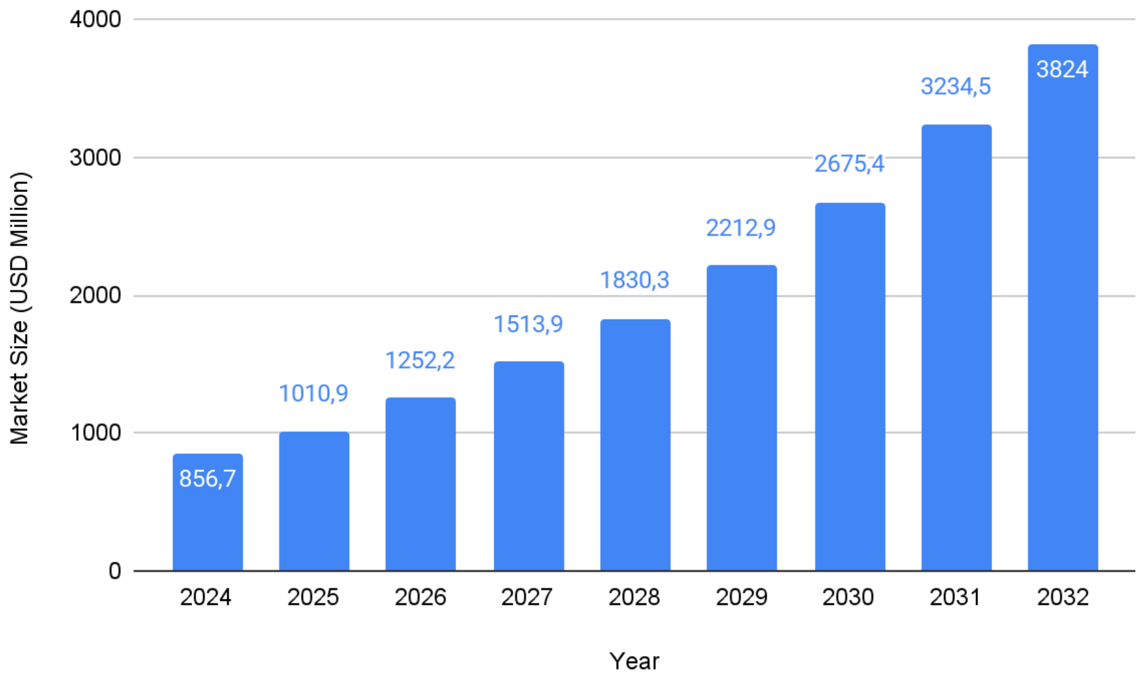

Fig. 1. Projected Market Size of AI-Enabled Testing Tools, 2024–2032 (USD Million) [19]

Consulting confirms the same trend: Gartner forecasts that the share of enterprises using AI add-ons to testing pipelines will grow from 15% in 2023 to 80% by 2027 [8], and the latest Forrester Wave evaluated fifteen vendors, categorizing them into leaders, strong performers, and niche players [9]. These figures demonstrate that platform selection has ceased to be an innovative experiment and has become a matter of technology strategy.

Visual and cognitive testing products lead the commercial SaaS segment. Applitools, specializing in Visual AI, achieved an approximate annual revenue of USD 35 million and expanded its distribution through AWS Marketplace, indicating a mature client base and a cloud-optimized business [10]. Mabl, in turn, operates on a full-pipeline-as-a-service model and has raised a total of $76.1 million in venture capital, including investments from GV and CRV, demonstrating institutional investor confidence in the platform's long-term growth [11]. Testim and Functionize compete in the same price range, focusing on self-healing and test generation. Both services were included in the Forrester report's Strong Performer category, balancing their relatively lower revenue with high technical scores [9]. Tricentis, occupying the enterprise segment, is noted by Gartner as a representative vendor in the 2024 AI automation guide, and its Tosca platform remains the only one in the review offering end-to-end codeless automation for ERP landscapes, which is critical for large enterprises that are not ready to rewrite tests manually with every SAP release [12].

Technically, these solutions differ in the depth of their AI layer and licensing scheme. Applitools, Mabl, and Testim are sold on a SaaS subscription with monthly billing for parallel sessions. Functionize and Tricentis complement the subscription with volume licensing of connectors to private clouds, which is particularly important for the banking and government sectors. In terms of functional coverage, the visual engines of Applitools/Functionize lead in detecting UI regression, whereas Mabl and Testim are stronger in generative recording of scenarios and automatic locator remapping. All five platforms support REST API for embedding predictions into existing CI/CD scripts. Still, only Tricentis and Functionize provide an on-premises variant, which increases their maturity for regulated industries.

The open segment is formed by research projects that more often serve as intelligent inserts rather than fully functional systems. DeepOrder, built on deep recurrent networks, demonstrated better early bug detection efficiency compared to industrial heuristics. Both tools are distributed under MIT-type licenses, require self-training on run history, and currently lack commercial SLAs, which limits their use by teams with insufficient ML competencies.

Comparing maturity, one can consider SaaS platforms ready for plug-and-play implementation, as they offer certified integrations, user support, and key performance indicators (KPIs) for reporting. Open-source solutions offer finer algorithm configuration and no licensing costs, but transfer operational risks to the team. The decision between them ultimately comes down to the ratio of the three metrics above: release speed, incident resolution time, and total cost of ownership. In places where cost predictability and regulatory reporting are critical, Applitools, Tricentis, or Functionize lead; in areas where flexibility and experimentation are more vital, Retecs or DeepOrder give the team control over model and data.

The safe operation of AI parts begins with obtaining the correct data for each attempt, including details about the change, setup of the surroundings, start-up events, step-by-step notes, coverage counts, and time measurements. According to TestRail's 2023 report, automated tests are run more than one hundred times per day by 62% of teams, generating tens of gigabytes of logs every week. Therefore, information must be presented in a simple form, along with smaller, concise pieces for quick review [13]. Compared to 2021, fewer teams were in the 0–100 range (42% to 38%) while more teams were in the 1,001–10,000 range (17% to 20%), an overall movement towards heavier automated testing as shown in Fig.2.

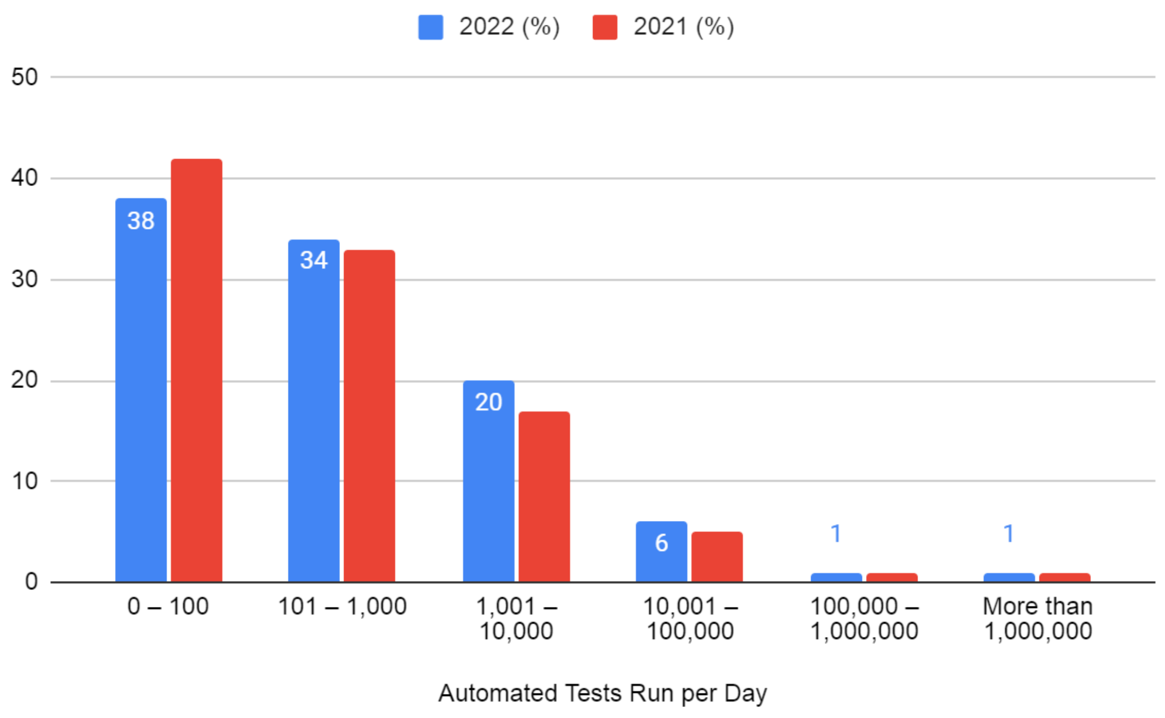

Fig. 2. On average, how many automated tests does your team run per day? [13]

According to Fig. 3, Selenium remains the most popular automation tool (approximately 41% in 2022, although it was 45% in 2020), whereas the JavaScript-oriented Cypress grew from 12% in 2020 to 19% in 2022. Java-based tools (JUnit, TestNG) remain at roughly the same level of about 16–17%, while the mobile solution Appium, conversely, gradually loses share (from 21% in 2020 and 2021 to 15% in 2022), as does the behavior-driven BDD framework Cucumber (from 18–20% in 2020–2021 to 14% in 2022). Cloud services gain notable prominence (BrowserStack – 11% in 2022), and new players emerge: Playwright entered the sample for the first time at 8%. Approximately one-fifth of respondents in 2022 indicated 'Don't Know', which suggests a particular gap in awareness of specific tools within teams.

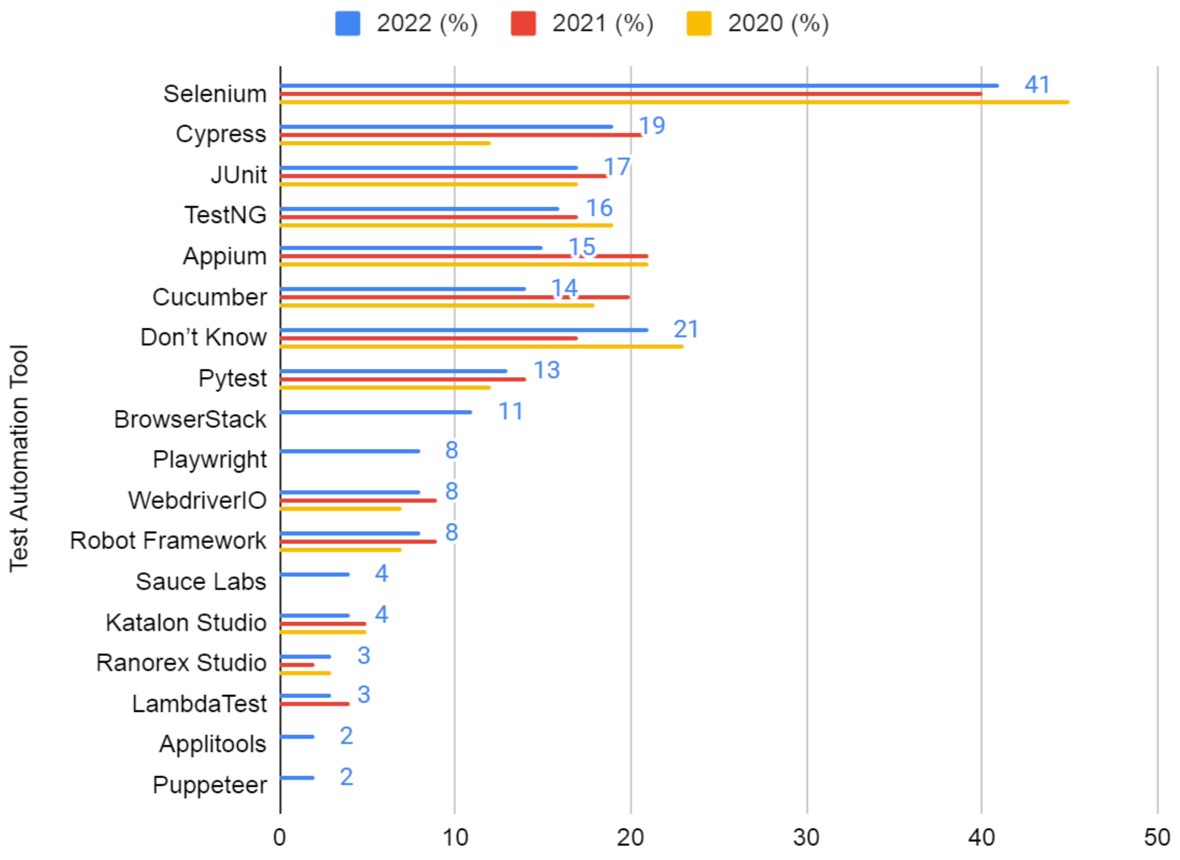

Fig. 3. What test automation tools, suites, or frameworks do you use? [13]

A suitable destination is the enterprise data lake, which is a repository: an object-based storage system that is S3 compatible and able to accept binary artifact archives and structured events. Informed by a systematic review of data-lake architectures, it is confirmed that keeping separate the raw and prepared views reduces latency between new test data arriving and being ready for analytics, as complete dataset copies are not needed [14].

The ML service has been developed as a standalone microservice, consisting of two parts. A background trainer retrieves new examples from the lake layer and updates models, while the inference part provides REST and gRPC interfaces for CI agents. The practice of continuous training is described in Springer Data Engineering research, where active, proactive training updates weights incrementally with new data and reduces parameter transfer volume by two to four orders of magnitude without loss of quality, thus making retraining within a day technically and economically feasible [15].

Integration into the CI/CD pipeline itself seems minimalist. After the artifact build, the pipeline script sends the change hash and the list of available tests to the ML endpoint and receives a ranked subset in return. Upon initiation of UI scenarios, the driver wrapper, upon each step failure, makes a secondary self-healing request that returns alternative selectors. The final mandatory element is a closed feedback loop. A reporting script publishes actual results (success, error type, duration) back to the lake, and the retraining scheduler monitors data volume and staleness to launch incremental model updates before concept drift occurs. Modern reviews of online learning emphasize that this strategy preserves metric stability amid code and environment changes while minimizing the computational costs of full retraining [16].

The motive for actual tests is a preexisting agreement: the majority of firms utilize or deploy generative AI for QA jobs, yet admit that its worth must be demonstrated with numerical indicators. Therefore, the pilot begins with a narrow, well-isolated microservice where the change flow is relatively stable and historical runs are easily collectible. In choosing success metrics, DORA-like indicators are used, including mean time from commit to green build, mean time to resolution (MTTR), and the share of defects caught before production. The baseline for comparison is formed from existing logs. After calibration, the model is transferred from the lab to night builds, and coverage is gradually expanded: first, the regression of critical user flows, then the service layer, and finally component tests.

Process evolution also requires organizational restructuring. The testing team assumes the role of prompt engineers, describing the domain logic for the generative model, whereas ML engineers ensure continuous training and data quality. Change management is based on transparent visualization of model impact: a board with daily sparklines of MTTR and machine-time savings is shown to all participants. It serves as a trigger for further suite expansion.

Simultaneously, a risk control system is built. The main technological vulnerability is model output opacity: without explanations, an engineer does not know why a test was excluded or healed. Explainable AI research recommends using interpretability and trust metrics as mandatory complements to classical AUC/Accuracy metrics. The Pandora framework demonstrates that hybrid human-machine defect analysis accelerates the localization of systemic failures by nearly three times [17]. Lastly, the infrastructure costs: Datadog notes a 40% year-over-year increase in organizational spending on cloud GPU instances used for inference and model training for test analytics, so when scaling, planning for spot or on-premises modes is critical [18]. Therefore, a well-scoped pilot, gradual rollout, specialist cross-training, and steady monitoring of metrics make it possible to implement AI tools without compromising process stability. When these practices are followed, corroborated by industry research, the economic effect – reducing MTTR, accelerating release, and lowering TCO – becomes reproducible, and risks shift from the undefined zone to being managed.

CONCLUSION

The study demonstrates that integrating AI tools into the continuous testing process is not merely a fashionable overlay but an almost inevitable step in the evolution of DevOps ecosystems. Transition from simple repetition of pre-described scenarios to intelligent management of the test suite and self-diagnosis of infrastructure establishes a qualitatively new paradigm in QA. The use of machine learning algorithms for log clustering, identification of flaky failures, and creation of a map of hot spots for testing enables a rapid response to changes in the codebase and minimizes false-positive triggers. This, in turn, not only reduces mean time to recovery (MTTR) but also increases trust in test results, making feedback genuinely timely and reliable.

Moreover, the presented market research indicates that AI-oriented tools have evolved beyond niche solutions into a mature commercial ecosystem. Industry leaders—from Applitools and Testim to Tricentis and Functionize—offer various deployment models (SaaS and on-premises), allowing platform selection according to specific regulatory and infrastructural requirements. Considerable growth in investment and market expansion forecasts already confirms that the choice of an AI platform has become an integral part of a company's technological strategy, and above all, a key factor in achieving a competitive advantage for those who want to balance release speed with product stability. This approach, supported by explainable AI practices under organizational restructuring (which acquires prompt engineering skills where QA and ML engineers work collaboratively), minimizes both technical and human risks related to implementation.

Thus, integration of AI tools into the continuous testing process represents an evolution from mechanical automation to risk management at a semantic level. Comprehensive application of intelligent test generation, self-healing, prioritization, and predictive selection establishes feedback speed and quality previously unattainable with traditional approaches. When described practices are followed and appropriate infrastructure for data storage and processing is established, the economic benefits—accelerated release, reduced Mean Time To Resolution (MTTR), and optimized total cost of ownership—become reproducible and manageable.

REFERENCES

1. "State of CI/CD Report 2024: The Evolution of

Software Delivery Performance," CD Foundation. https://cd.foundation/state-of-cicd-2024/ (accessed May 01, 2025).

2. M. S. Bari, A. Sarkar, and S. A. M. Islam, "Ai-augmented Self-healing Automation Frameworks: Revolutionizing Qa Testing with Adaptive and Resilient Automation," Advanced International Journal of Multidisciplinary Research, vol. 2, no. 6, Dec. 2024, doi: https://doi.org/10.62127/aijmr.2024.v02i06.1118.
3. Open Text Corporation, "World Quality Report 2024 shows 68% of Organizations Now Utilizing Gen AI to Advance Quality Engineering," PR Newswire, Oct. 22, 2024. https://www.prnewswire.com/news-releases/world-quality-report-2024-shows-68-of-organizations-now-utilizing-gen-ai-to-advance--quality-engineering-302282709.html (accessed May 03, 2025).
4. M. Machalica, A. Samylkin, M. Porth, and S. Chandra, "Predictive Test Selection," arXiv, Jan. 2018, doi: https://doi.org/10.48550/arxiv.1810.05286.
5. M. Schäfer, S. Nadi, A. Eghbali, and F. Tip, "An Empirical Evaluation of Using Large Language Models for Automated Unit Test Generation," arXiv, Sep. 2023, doi: https://doi.org/10.48550/arXiv.2302.06527.
6. A. Alshammari, C. Morris, M. Hilton, and J. Bell, "FlakeFlagger: Predicting Flakiness Without Rerunning Tests," 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE), May 2021, doi: https://doi.org/10.1109/icse43902.2021.00140.
7. F. Ricca, A. Marchetto, and A. Stocco, "A Multi-Year Grey Literature Review on AI-assisted Test Automation," arXiv (Cornell University), Aug. 2024, doi: https://doi.org/10.48550/arxiv.2408.06224.
8. "Gartner Market Guide for AI-Augmented Software-Testing Tools 2024," Tricentis, Mar. 04, 2024. https://www.tricentis.com/resources/gartner-ai-market-guide (accessed May 07, 2025).
9. D. Giudice, C. Gardner, A. Cornwall, A. Lozada, and K. Hartig, "The Forrester WaveTM: Continuous Automation Testing Platforms, Q4 2022," 2022. Accessed: May 10, 2025. [Online]. Available: https://www.tekmarkgroup.com/wp-content/uploads/2023/10/The-Forrester-Wave%E2%84%A2-Continuous-Automation-Testing-Platforms-Q4-2022.pdf
10. "Applitools Company Overview, Contact Details & Competitors," LeadIQ, 2024. https://leadiq.com/c/applitools/5a1d9d422300005b008d1c07 (accessed May 14, 2025).
11. "Mabl company profile," Tracxn, Aug. 17, 2021. https://tracxn.com/d/companies/mabl/__b9wljjy7_SAZSPVCj34bz9RedSOIojaJFYuWPFn4iBk (accessed May 16, 2025).
12. "Tricentis Recognized in the February 2024 Gartner® Market Guide for AI-Augmented Software-Testing Tools," Tricentis, Mar. 06, 2024. https://www.tricentis.com/news/tricentis-2024-gartner-market-guide-for-ai-augmented-software-testing-tools (accessed May 17, 2025).
13. "2023 Software Testing and Quality Report," Testrail, 2023. Accessed: May 16, 2025. [Online]. Available: https://content.testrail.com/hubfs/Downloadables/TestRail-2023-Software-Testing-and-Quality-Report.pdf
14. S. Azzabi, Z. Alfughi, and A. Ouda, "Data Lakes: A Survey of Concepts and Architectures," Computers, vol. 13, no. 7, p. 183, Jul. 2024, doi: https://doi.org/10.3390/computers13070183.
15. I. Prapas, B. Derakhshan, A. R. Mahdiraji, and V. Markl, "Continuous Training and Deployment of Deep Learning Models," Datenbank-Spektrum, vol. 21, no. 3, pp. 203–212, Nov. 2021, doi: https://doi.org/10.1007/s13222-021-00386-8.
16. M. Sulaiman, M. Farmanbar, S. Kagami, A. N. Belbachir, and C. Rong, "Online deep learning's role in conquering the challenges of streaming data: a survey," Knowledge and Information Systems, Feb. 2025, doi: https://doi.org/10.1007/s10115-025-02351-3.
17. B. Nushi, E. Kamar, and E. Horvitz, "Towards Accountable AI: Hybrid Human-Machine Analyses for Characterizing System Failure," arXiv (Cornell University), Sep. 2018, doi: https://doi.org/10.48550/arxiv.1809.07424.
18. Datadog Inc., "Datadog's State of Cloud Costs 2024 Report Finds Spending on GPU Instances Growing 40% as Organizations Experiment with AI," PR Newswire, Jun. 13, 2024. https://www.prnewswire.com/news-releases/datadogs-state-of-cloud-costs-2024-report-finds-spending-on-gpu-instances-growing-40-as-organizations-experiment-with-ai-302172281.html (accessed May 25, 2025).
19. "AI-enabled Testing Market Size, Share & COVID-19 Impact Analysis," Fortune Business Insights, May 12, 2025. https://www.fortunebusinessinsights.com/ai-enabled-testing-market-108825 (accessed May 26, 2025).