УДК 674.053:621.933.6 DOI 10.36910/775.24153966.2025.83.25

О.С Ковалишин
Львівський національний університет ветеринарної медицини та біотехнологій імені С.З. Ґжицького

ГЕНЕРАТИВНІ АГЕНТНІ РІШЕННЯ ШТУЧНОГО ІНТЕЛЕКТУ ДЛЯ ЗАБЕЗПЕЧЕННЯ ЯКОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

АІ-агенти демонструють автономність у плануванні, виконанні, аналізі та підтримці тестів, пропонуючи потенціал для подолання стійких викликів у забезпеченні якості, зокрема надмірних витрат на підтримку, тривалих циклів налагодження та обмежених можливостей прогнозування. Для структурування дослідження визначено сім основних агентних варіантів використання, що охоплюють повний життєвий цикл тестової автоматизації. Проведено порівняльну оцінку трьох SaaS-платформ — KaneAI, Zephyr Scale Automate та TestRigor — із застосуванням дуальної методики, яка поєднує кількісне оцінювання та якісний аналіз. Такий підхід забезпечує цілісне розуміння як зрілості підтримки агентних функцій, так і глибинних характеристик їх реалізації, роблячи внесок у дискурс щодо еволюції практик тестування програмного забезпечення.

Ключові слова: *генеративні АІ-агенти, якість програмного забезпечення, автоматизація тестування, агентні системи, штучний інтелект.*

O. Kovalyshyn

GENERATIVE ARTIFICIAL INTELLIGENCE AGENTIC SOLUTIONS FOR SOFTWARE QUALITY ASSURANCE

AI agents demonstrate autonomy in planning, execution, analysis, and maintenance, offering potential to address persistent challenges in quality assurance such as maintenance overhead, lengthy debugging cycles, and limited predictive capacity. To structure the investigation, seven core agentic use cases were defined, covering the full lifecycle of test automation. A comparative evaluation of three SaaS platforms—KaneAI, Zephyr Scale Automate, and TestRigor—was conducted using a dual framework that combines quantitative scoring with qualitative assessment. This approach enables an integrated understanding of both the maturity of support for agentic functionality and the qualitative depth of its implementation, contributing to ongoing discourse on the evolution of software testing practices.

Keywords: *Generative AI agents, software quality, test automation, agentic systems, artificial intelligence.*

Problem statement. The emergence of ChatGPT, GitHub Copilot, and similar generative AI (GenAI) systems has reshaped expectations for software engineering and testing. Early research signaled strong productivity boosts. GitHub's controlled study showed developers completing tasks 55.8% faster with Copilot [1]. AWS benchmarks reported up to 57% productivity gains from Amazon CodeWhisperer [2]. McKinsey estimated developers could be up to twice as fast in coding tasks, while GenAI overall could add \$2.6–4.4 trillion annually in value creation [3]. SoftServe pilots across the Software Development Lifecycle reported up to 45% productivity improvements: ~20% faster software development, 25% reductions in Quality Control effort, 20% acceleration in test automation, and 17% savings in documentation [4]. These figures are impressive but remain bounded to atomic activities such as code generation, boilerplate writing, or drafting tests, typically validated in pilot contexts with human orchestration. The gains plateau because current tools act as assistants rather than self-driving entities. As a result, organizations recognize that further progress requires moving beyond incremental efficiency toward systems that can operate with higher levels of autonomy and contextual awareness. To unlock transformative impact, the industry must shift toward agentic solutions—autonomous systems capable of planning, executing, and adapting across testing workflows.

Literature Review. In 2024–2025, the focus of AI in software testing has shifted from passive copilots toward agentic systems—autonomous, goal-directed entities. This transition reflects broader research in software engineering and multi-agent systems, which identifies the potential for agents not only to generate artifacts but also to orchestrate workflows in testing and quality management. [5]. Recent publications focus on the following primary applications of agentic AI in testing. First, dynamic test design, where agents derive test cases directly from requirements or UI exploration. Second, autonomous execution and environment control, reducing manual setup effort. Third, failure triage and root-cause analysis, with agents connecting logs and traces to likely causes. Fourth, test maintenance and refactoring, where agents adapt suites as applications evolve [6-7]. Beyond those, the following additional use cases are commonly discussed: self-healing / auto repair of broken tests, intelligent test selection / risk-based prioritization of tests, and predictive defect / error detection using historical test outcomes [8-10]. These use cases move well beyond static script generation and show iterative learning and adaptation.

Industry leaders are actively working to integrate agentic systems and GenAI agents into their testing solutions, moving beyond assistant-only approaches. Considering the main use cases of agentic AI in testing, several notable competitors are available on the market today. KaneAI offering execution across multiple frameworks and devices with self-healing capabilities [11]. TestRigor focuses on plain-English test authoring, autonomous execution, and strong self-healing to reduce maintenance overhead, while also leveraging user behavior for prioritization[12]. Zephyr Scale Automate extends test management with automation and execution features, making it easier to run and track large volumes of tests, though with less emphasis on full autonomy[13]. Together, these tools illustrate how vendors are already embedding agentic features into practical SaaS applications for software testing.

Goal of the research. Goal of the research is to analyse current market leading agentic solutions for software testing and potential for their application in ensuring quality of software applications.

Methodology and results. An AI agent is a computational system that can perceive its environment, set or interpret goals, and act autonomously to achieve them. Unlike assistants, which rely on continuous human prompting to complete narrowly defined tasks, agents operate with a higher degree of independence: they can plan sequences of actions, make context-aware decisions, and adapt their behavior based on feedback or changing conditions. This autonomy allows them not only to execute instructions but also to anticipate next steps, recover from errors, and optimize workflows over time. In the context of software testing, AI agents differ from assistant-style tools by orchestrating entire quality processes—such as designing, executing, analyzing, and maintaining tests—without requiring step-by-step human guidance [14].

While the concept of AI agents sets the theoretical foundation, the real shift in software testing comes from how these principles are embedded into practical tools and SaaS platforms. Early assistant-style applications proved useful for isolated tasks, but agentic systems are now moving into mainstream adoption through commercial products. These tools are designed not just to generate code or suggest test cases, but to self-heal, prioritize, orchestrate execution, and provide insights with minimal human intervention [15].

The selection of KaneAI, Zephyr Scale Automate, and TestRigor was driven by their explicit incorporation of agentic or GenAI-driven functionality, distinguishing them from traditional test automation tools. They also represent diverse market positions—emerging AI-first solutions, enterprise platforms extending into autonomy, and low-code environments with adaptive AI—ensuring a balanced and relevant comparison.

KaneAI (by LambdaTest) is a GenAI-native, end-to-end testing agent that lets teams plan, author, execute, debug, and evolve tests directly from natural-language objectives, rather than hand-coding steps. It's embedded in the LambdaTest cloud, so generated tests can run across browsers, OSs, mobile devices, and popular frameworks (e.g., Playwright, Selenium, Cypress, Appium) with built-in resilience features like self-healing and versioned "evolve" workflows. Public materials and launch coverage emphasize its aim to move beyond assistant-style code suggestions toward autonomous test orchestration and faster RCA/triage within CI/CD pipelines. LambdaTest has recently announced general availability, positioning KaneAI as a production-ready layer on top of its execution infrastructureKaneAI offers several key features. It enables natural language test authoring and evolution, allowing users to write objectives, scenarios, or steps in plain English and automatically generate tests. It supports multi-framework and multi-language export, including Selenium, Playwright, Cypress, WebdriverIO, and Appium, giving flexibility across environments. An Intelligent Test Planner can auto-generate detailed steps from high-level objectives. The platform includes self-healing and resilient test capabilities, such as handling UI or locator changes, versioning, and automatic healing. It provides broad device and browser coverage with cloud execution and scheduling through LambdaTest's infrastructure. Finally, KaneAI supports assisted debugging and observability with root-cause analysis, categorized errors, detailed reports, and advanced test analytics [16].

Zephyr Scale (by SmartBear) is an advanced test management solution that extends Jira with enterprise-grade capabilities for managing manual and automated testing. The Automate add-on expands this by enabling integration of automated test runs, cross-browser execution, and richer traceability between requirements, test cases, and results. Unlike agent-native tools, Zephyr Scale Automate is less focused on generative AI autonomy, but it plays a central role in connecting automated execution with structured test management at scale. It is widely adopted by QA teams already invested in Jira, CI/CD pipelines, and SmartBear's testing ecosystem. Zephyr Scale Automate provides a rich set of features tailored for enterprise test management. It natively integrates with Jira to organize, create, and manage large volumes of test cases. Full traceability links requirements to test cases and defects, ensuring coverage and compliance reporting. Automated test execution is supported through its Automate add-on, integrating with CI/CD tools such as

Jenkins, Bamboo, and GitHub Actions. It allows cross-browser and parallel runs, enabling faster validation across environments. Customizable reporting and dashboards deliver advanced test metrics, coverage views, and real-time project status inside Jira. Versioning and reuse features support test cycles, case control, and reusable components, while scalability ensures handling of thousands of test cases across multiple projects. Finally, Zephyr Scale Automate benefits from a broad integration ecosystem, connecting to SmartBear's own products like TestComplete and ReadyAPI, as well as external automation frameworks [17].

TestRigor is an AI-powered test automation platform designed to let teams create and maintain end-to-end tests entirely in plain English. Unlike traditional frameworks that depend on brittle locators or scripting, TestRigor interprets natural-language commands into executable actions, significantly reducing the barrier for non-technical testers. Its agentic features include self-healing tests, automatic adaptation to UI changes, and the ability to generate tests from user behavior patterns. Positioned as a SaaS solution, TestRigor integrates directly with CI/CD pipelines, enabling continuous, stable, and low-maintenance automated testing across web, mobile, and desktop applications. TestRigor focuses on simplifying automation through natural language and adaptive intelligence. It allows plain-English test authoring, enabling teams to create and execute tests without writing code. The platform supports self-healing automation, where tests automatically adjust to UI or locator changes, reducing maintenance effort. It offers cross-platform coverage across web, mobile, and desktop applications, with integrations into diverse environments. Test generation can also be driven from user behavior, analyzing real user flows and usage patterns to expand coverage. Seamless CI/CD integration is available with Jenkins, GitHub Actions, GitLab CI, CircleCI, and other pipelines, supporting parallel test execution. Regression and prioritization features help focus on the most critical functionality using metadata and analytics. Built for enterprise needs, TestRigor scales to thousands of tests while maintaining stability and speed. Finally, its low technical barrier makes it accessible to manual QA engineers, who can author and maintain automated suites without programming skills [18].

While these tools showcase different approaches to embedding AI into testing—from KaneAI's fully agentic orchestration to Zephyr Scale Automate's enterprise management layer and TestRigor's low-code intelligence—the real value emerges when we evaluate them against concrete challenges in modern QA. To move beyond high-level claims, it is essential to ground the discussion in specific agentic use cases that represent where autonomy can deliver the most impact: designing tests dynamically, executing and managing environments, triaging failures, maintaining suites, healing broken scripts, prioritizing runs, and predicting defects. Details and impact of each use case are represented in Table 1. This use cases cover the full lifecycle of test automation—from design and execution through failure analysis and maintenance, while also addressing efficiency, stability, and predictive quality assurance. These were chosen because they directly align with current challenges faced in modern testing: high maintenance overhead, slow feedback cycles, difficulty in diagnosing failures, and limited ability to anticipate risks[19,20].

To enable a structured evaluation of instruments, both quantitative and qualitative assessment was applied. Quantitative scoring was performed on a 1–5 scale to measure the maturity of support for each use case, while qualitative evaluation provided descriptive insights into how each capability is implemented in practice. This dual approach ensures that the analysis captures not only the presence of functionality but also its depth, reliability, and applicability in enterprise contexts.

With this evaluation framework in place, the Pic 1. translates the quantitative scoring into a structured comparison across the selected tools. By mapping KaneAI, Zephyr Scale Automate, and TestRigor against the common agentic use cases, the chart highlights not only where each tool demonstrates strong, production-ready capabilities but also where functionality remains emerging or limited. This provides a clear, evidence-based view of how current market offerings align with the practical needs of enterprise QA teams and where gaps in agentic adoption still exist.

Each tool was assessed on these dimensions using publicly available documentation, vendor claims, and independent reviews. Scoring system for quantitative use-case evaluation is following:

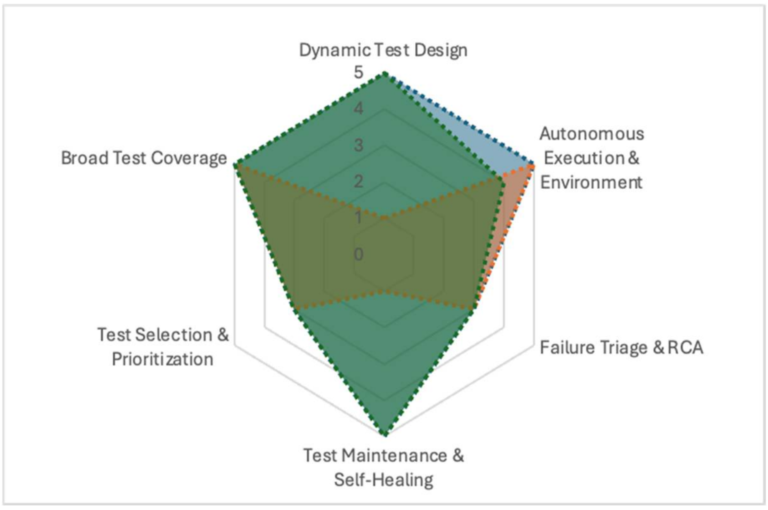
- 1 - Limited / No Support. The tool provides little to no native functionality for the use case. Any coverage would require extensive manual workarounds or integration with external tools.
- 2 - Emerging / Minimal Support. The tool shows early or experimental support for the use case. Features may be in beta, partially functional, or highly dependent on user customization. Reliability is limited, and the capability is not yet ready for consistent production use.

- 3 - Moderate / Partial Support. The tool offers meaningful functionality that addresses the use case, but coverage is incomplete or requires significant human input. Features may work reliably for simple scenarios but lack depth, scalability, or automation for complex environments.

Table 1.

Agentic AI use cases in software testing

Use Case	Description	Value/Impact
Dynamic Test Design	Agents derive test cases directly from requirements, user stories, or by exploring the UI.	Ensures broad and adaptive coverage while reducing manual test design effort.
Autonomous Execution & Environment Control	Agents handle environment setup, configuration, and execution of tests automatically.	Cuts down manual setup time, improves execution speed, and enhances reproducibility.
Failure Triage & Root-Cause Analysis	Agents correlate logs, traces, and telemetry to identify likely causes of failures.	Accelerates debugging, shortens defect resolution time, and provides actionable insights.
Test Maintenance & Refactoring	Agents update and refactor test suites as applications evolve (UI, APIs, workflows).	Reduces maintenance overhead, keeps automation relevant, and improves test reliability.
Self-Healing / Auto-Repair of Broken Tests	Agents detect failing steps (e.g., locator changes, timing issues) and repair them automatically.	Increases test stability, reduces flakiness, and lowers long-term upkeep costs.
Intelligent Test Selection / Risk-Based Prioritization	Agents prioritize tests based on recent changes, impacted areas, or historical failure patterns.	Optimizes execution cycles, provides faster feedback, and focuses on high-risk areas.
Predictive Defect / Error Detection	Agents analyze historical test outcomes and code changes to predict defect-prone areas.	Enables proactive defect prevention, increases release confidence, and improves overall quality.



Pic 1. **Quantitative evaluation of SaaS solutions coverage of Core agentic use cases.**

- 4 - Good / Near-Mature Support. The tool provides strong support and can address the majority of the use case requirements. Capabilities are practical for enterprise adoption, though some gaps remain in terms of full autonomy, integration, or advanced optimization.
- 5 - Strong / Mature Support. The tool demonstrates mature, robust, and widely validated support for the use case. It enables autonomous or near-autonomous execution with minimal human intervention, scales reliably in enterprise environments, and is recognized as best-in-class for this dimension.

While the quantitative scores provide a clear, numeric view of maturity across the agentic use cases, they do not capture the nuances of how each capability is actually delivered in practice. To complement the numbers, a qualitative assessment was conducted to highlight the nature, depth, and distinctiveness of each tool's implementation. This perspective reveals not only whether functionality exists, but also how usable, scalable, and enterprise-ready it is in real-world contexts. Table 1 summarizes these insights, contrasting KaneAI, Zephyr Scale Automate, and TestRigor across the core agentic use cases.

Table 2.

Qualitative evaluation of SaaS solutions coverage of core agentic use cases

Use Case	KaneAI	Zephyr Scale Automate	TestRigor
Dynamic Test Design	Natural-language test authoring; generates tests from specs, Jira tickets, and images.	Primarily manual test case creation; lacks AI-driven generation.	Plain-English test authoring; auto-generation from user behavior and existing test cases.
Autonomous Execution & Environment Control	Cloud-based execution across browsers/devices; scheduling and CI/CD integration.	Strong automated execution with parallel runs; integrates into CI/CD pipelines.	CI/CD and parallel execution supported; broad but slightly less seamless environment orchestration.
Failure Triage & Root-Cause Analysis (RCA)	Provides categorized error reporting and debugging tools; RCA suggestions are basic.	Offers reporting and traceability but triage is largely manual.	Detects failures from UI changes; analytics-based triage, RCA still limited.
Test Maintenance & Self-Healing	Auto-healing tests with versioning and evolution as applications change.	Maintenance is mostly manual; lacks self-healing features.	Robust self-healing; automatically adapts to locator and UI changes.
Test Selection & Prioritization	Planning and scheduling available; risk-based prioritization not fully automated.	Supports filtering and test cycles, but not AI-driven prioritization.	Prioritization based on user behavior and metadata; not fully autonomous.
Broad Test Coverage (Web/Mobile/API)	Supports web, mobile, API, and multiple frameworks (Selenium, Playwright, Appium, etc.).	Scales to enterprise projects; integrates with automation frameworks.	Cross-platform support for web, mobile, and desktop applications.

Conclusions. The transition from assistant-style test automation tools to agentic solutions marks a fundamental shift in software quality. Unlike assistants that require step-by-step prompting, AI agents can plan, execute, and adapt autonomously—capabilities now increasingly embedded into SaaS testing platforms. This study focused on seven core use cases—dynamic test design, autonomous execution, failure triage, test maintenance and self-healing, intelligent test selection, and predictive defect detection—chosen to reflect key challenges in QA, including maintenance overhead, slow debugging cycles, and limited risk anticipation.

Quantitative scoring revealed that KaneAI, as a GenAI-native solution, delivers the most advanced support for agentic workflows, particularly in dynamic test design, execution, and self-healing. TestRigor demonstrates strong applicability in plain-English automation and adaptive self-healing, bridging accessibility for non-technical testers with enterprise-level stability. Zephyr Scale Automate, while less agentic in its orientation, provides robust execution and enterprise-grade test management, ensuring integration with established workflows in Jira-driven organizations.

The qualitative evaluation further underscored these distinctions: KaneAI leads in autonomy and multi-framework support, TestRigor excels in simplifying authoring and maintenance through natural language, and Zephyr Scale Automate offers unmatched scalability in structured test management but lags in AI-driven adaptability.

Overall, the findings suggest that no single tool yet covers the entire spectrum of agentic use cases at a mature level. However, clear trends indicate that GenAI-first platforms are moving rapidly toward autonomous orchestration, while enterprise-oriented solutions are beginning to extend their ecosystems with agentic capabilities. For organizations, the choice of platform should be guided not only by current functionality but also by alignment with strategic goals—whether prioritizing autonomy, accessibility, or integration at scale.

Bibliography

1. arXiv. Feature-driven end-to-end test generation. URL: https://arxiv.org/abs/2302.06590 (дата звернення: 24.09.2025).

2. AWS Builder. 2023 CodeWhisperer year in review. URL: https://builder.aws.com/content/2ZrSTfMqY1P0FXrjafrDC85LIrS/2023-codewhisperer-year-in-review (дата звернення: 24.09.2025).
3. McKinsey & Company. Unleashing developer productivity with generative AI. URL: https://www.mckinsey.com/capabilities/mckinsey-digital/our-insights/unleashing-developer-productivity-with-generative-ai (дата звернення: 24.09.2025).
4. SoftServe. Redefining the economics of software development with GenAI. URL: https://info.softserveinc.com/hubfs/files/redefining-the-economics-of-software-development-gen-ai.pdf (дата звернення: 24.09.2025).
5. arXiv. URL: https://arxiv.org/abs/2407.17426 (дата звернення: 24.09.2025).
6. arXiv. URL: https://arxiv.org/html/2509.05197v1 (дата звернення: 24.09.2025).
7. Alian P., Nashid N., Shahbandeh M., Shabani T., Mesbah A. Feature-driven end-to-end test generation. IEEE/ACM 47th International Conference on Software Engineering (ICSE). IEEE Computer Society, 2025, pp. 678–678.
8. Le N.-K., Bui Q. M., Nguyen M. N., Nguyen H., Vo T., Luu S. T., Nomura S., Nguyen M. L. Automated web application testing: End-to-end test case generation with large language models and screen transition graphs. 2025. URL: https://arxiv.org/abs/2506.02529 (дата звернення: 24.09.2025).
9. Lee S., Choi J., Lee J., Wasi M. H., Choi H., Ko S., Oh S., Shin I. Mobilegpt: Augmenting LLM with human-like app memory for mobile task automation. Proceedings of the 30th Annual International Conference on Mobile Computing and Networking. 2024, pp. 1119–1133.
10. Liu C., Gu Z., Wu G., Zhang Y., Wei J., Xie T. Temac: Multi-agent collaboration for automated web GUI testing. arXiv preprint arXiv:2506.00520, 2025a.
11. LambdaTest. KaneAI. URL: https://www.lambdatest.com/kane-ai (дата звернення: 24.09.2025).
12. SmartBear. Zephyr Scale. URL: https://smartbear.com/test-management/zephyr (дата звернення: 24.09.2025).
13. TestRigor. URL: https://testrigor.com/ (дата звернення: 24.09.2025).
14. Sapkota R., Roumeliotis K. I., Karkee M. AI agents vs. agentic AI: A conceptual taxonomy, applications and challenges. arXiv preprint arXiv:2505.10468, 2025. URL: https://arxiv.org/abs/2505.10468 (дата звернення: 24.09.2025).
15. White J. Building living software systems with generative & agentic AI. arXiv preprint arXiv:2408.01768, 2024. URL: https://arxiv.org/abs/2408.01768 (дата звернення: 24.09.2025).
16. Shapiro D., Li W., Delaflor M., Toxtli C. Conceptual framework for autonomous cognitive entities. arXiv preprint arXiv:2310.06775, 2023. URL: https://arxiv.org/abs/2310.06775 (дата звернення: 24.09.2025).
17. Garousi V., Joy N., Keleş A. B. AI-powered test automation tools: A systematic review and empirical evaluation. arXiv preprint, 2024.
18. Sherifi B. et al. The potential of LLMs in automating software testing: From generation to reporting. arXiv preprint, 2024/2025.
19. Feldt R., Kang S., Yoon J., Yoo S. Towards autonomous testing agents via conversational large language models. arXiv preprint, 2023.
20. Lengyel F. AI agents in software testing and test automation. ResearchGate preprint, 2025.