# Testing the AI Agentic Way

**Praveen Bagare**

bagare@gmail.com

## Abstract

Conventional testing has its pros and cons. People, processes, and tools can be well-defined, documented, and implemented, but challenges around test data, environments, coverage, automation, and testing speed can arise unexpectedly. To address these challenges and shift the software testing life cycle (STLC) left, automating feasible steps using the novelty of generative AI agents is a great approach. These agents can be considered virtual peers that can help accelerate tasks not only across testing but the entire software development life cycle (SDLC). They generate the content for review, rather than a human creating it from scratch, thus giving the users a head start and significant time savings.

In this paper we will cover topics around introducing Gen AI agents, good practices around using them and what benefits they bring to the table. We will also touch upon some of the challenges faced with the AI agentic testing approach. A case study has also been shared to demonstrate the Return On Investment (ROI) and the benefits of testing the AI Agentic way.

## Biography

Praveen Bagare is a seasoned IT leader with over 21 years of experience in delivering innovative AI-powered solutions. His expertise encompasses Program Management, Artificial Intelligence solutions, and Quality Assurance, with a specialization in Test Data Management (TDM). Passionate about solving complex challenges, Praveen drives excellence and innovation.

Currently, he leads EPAM's North American TDM Competency Center, overseeing implementations that leverage generative AI and other TDM tools. He has architected, implemented, and managed TDM solutions for Fortune 500 companies globally.

Praveen is an eminent expert in the field, having authored a white paper on TDM in the Software Testing Life Cycle and another on AI and ML in TDM in PNSQC. An active senior IEEE member, TREWS fellow member, and presenter at AI Con USA 2025, he is committed to advancing industry standards. He is also ISTQB-certified and advanced certified in multiple TDM and AI tools. When not leading teams or driving innovation, Praveen enjoys spending time outdoors with his family in Wisconsin.

# 1. Introduction

In the current technological landscape, generative artificial intelligence (AI) has gained widespread adoption across various domains. This paper explores its impact on the software testing lifecycle (STLC) from an agentic perspective.

These AI agents are similar to virtual collaborators, facilitating the acceleration of tasks in a defined workflow to achieve desired outcomes. For example, agents can leverage user story inputs to build elaborate test cases using techniques like boundary conditions, positive and negative scenarios, equivalence partnering and more. These agents can bring in great efficiencies across the software development life cycle with individual agents or a series of orchestrated agents through workflows.

Market leaders have introduced agents at varying maturity levels, and their interactivity has significantly improved through the adoption of the Model Context Protocol (MCP). [1] and Agent to Agent (A2A) protocols.

# 2. What are Gen AI agents?

Generative AI agents can be explained as software programs or systems that can independently reason, make decisions, perform actions, learn, and orchestrate workflows to achieve specific objectives. They utilize a set of prompts, application connections, Retrieval Augmented Generation (RAG), and leverage generative AI from Large Language Models (LLMs) to produce desired outputs. [2] [3] [4]

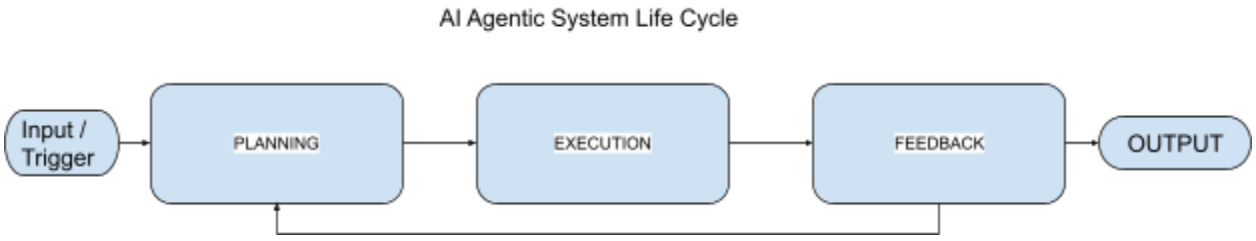
Following is a list of the typical Gen AI Agent attributes: [5]

- Agents are more complex than bots and assistants.
- They can automate simple tasks and provide limited decision-making capabilities.
- A single agent can achieve a desired objective, or a chain of agents can work together to reach a goal.
- Agents can interact with users or be triggered independently in the background.
- They can be categorized in various ways, including simple response-based, goal-based, learning agents, user-triggered, and event-driven agents. [6]

# 3. How do GenAI Agents work?

AI agents are typically built with an interfacing layer allowing integration with tools, use memory and leverage techniques like RAG to harness the power of Large Language Models (LLMs), thus delivering desired outputs. The recent evolution of the LLMs and effective prompt engineering has further made outputs more predictable, usable, and with minimal hallucinations.

Below is a simple depiction of an AL agentic system. [7]



# 4. Conventional Testing vs Testing the AI Agentic way

Let's explore the key activities in the testing life cycle, along with a brief overview of the conventional and agentic approaches. We'll outline the testing activities, describe them, and highlight their similarities and differences. While this overview doesn't cover everything, it provides a concise summary.

**Testing objective**

| Conventional | AI Agentic |
|---|---|
| Validate the product or application under test to ensure it meets the requirements and functions efficiently. Identify any defects and report them to the development teams for fixing. Keep stakeholders informed to ensure informed decision-making. | It achieves the same objective as conventional testing but additionally leverages agents and agent orchestrators to harness the generative capabilities of LLMs, facilitating shift-left testing strategies. Agents also enhance connectivity to various tools for test management, defect reporting, and code repositories. |

**Requirement Gathering** involves understanding the testing requirements and evaluating their completeness. This includes identifying the functionality that needs to be tested and determining the expected results. Additionally, it involves considering nonfunctional requirements such as the volume of users, accessibility, and security needs.

| Conventional | AI Agentic |
|---|---|
| Manually gather the requirements from epics, user stories and documents. Have meetings with business analysts to understand the functional and non-functional tasks. | Perform all the activities as in the conventional approach but get support from single or multiple AI agents to connect to systems like JIRA, Confluence, Git, meeting transcripts and other docs, and bring back content to one space, operate on it to enhance and standardise it and validate completeness. These agents help with better documentation of requirements, early unearthing of gaps in requirements. |

Planning and Estimation - Create a test strategy and test plan outlining the way testing would be done? Where would it be done? The tools that would be used and who would do it and when? Estimate the time and effort needed to complete the testing activities.

| Conventional | AI Agentic |
|---|---|
| Use templates, tools and experience to create a test strategy and a plan. | Leverage AI agents to create test strategies and test plans in standard templates with good |

| | |
|---|---|
| Leverage historic data and human experience to create the time and effort estimates. | coverage of all aspects of testing functional & non functional.<br>Leverage Gen AI estimation agents to get a reliable estimate based on historic data available (e.g. passing it through retrieval augmented generation) and the knowledge of the LLMs. |

Test Case and scenario authoring - The phase of testing where the requirements are converted into test scenarios and cases to validate the functionality with test steps and expected results or in Given, When, Then Gherkin format or other similar techniques.

| **Conventional** | **AI Agentic** |
|---|---|
| Human-authored test cases and scenarios based on an understanding of the expected functionality in the requirements. It can be ad hoc or very systematic, covering boundary conditions, positive-negative scenarios, and other proven approaches and depends on the expertise of the tester writing the test cases and scenarios. | Leverage AI agents that understand the requirements, create meticulous test cases across boundary conditions, positive, and negative test scenarios and cases with good coverage. The generated test cases can even be put into the test management tool from the agent directly. Predefined templates can be used to get the desired test case output formats. |

Test Environment Validation - The process of setting up the test environment along the path to production. *(Functional, Integration, User and finally Prod in parallel to performance and sometimes even Development)*

| **Conventional** | **AI Agentic** |
|---|---|
| The Test Environment team sets up the infrastructure either manually or through infrastructure as code. The Quality Assurance team verifies that the environment aligns with the path to production and meets usability requirements. | AI agents can be used to build the infra as code. Environment Scanning Agents can validate the environment's readiness for testing initially and later ensure it meets the required standards. |

Test Data Management - The process of procuring test data via TDM services like mining, creation, reservation, and cloning, contextual to the environment and system under test.

| **Conventional** | **AI Agentic** |
|---|---|

| | |
|---|---|
| TDM is typically done manually or in conjunction with utilities and or off the shelf expensive TDM tools. | Though TDM processes and tools can not easily be replaced, AI agents help automate a good portion of the manual steps in the TDM life cycle. (e.g., using Natural language to SQL agents, TDM tool config file generation agents and more) |

Test Execution - the process of executing test cases and scenarios against the system under test leveraging the test data created in the valid test environment to identify deviation from requirements and expected results by logging defects for identified anomalies.

| Conventional | AI Agentic |
|---|---|
| Manually testing the application, using test automation frameworks and tools are the non-AI way of doing testing. They are effective if well orchestrated and follow a streamlined process. Manually, CI/CD pipelines can be triggered to automate the execution as well. | AI agents can also seamlessly integrate with CI/CD pipelines, where the test automation code is directly merged into the test automation framework using connectors from the AI agents. Standardization, prioritization, and optimization of the automation scripts are key differentiators. [8] |

Test Execution Reporting - The process of documenting and communicating the outcomes of a testing cycle to stakeholders. It summarizes key findings like the number of tests executed, passed, or failed, providing a snapshot of the software's quality helping make Go no Go decisions.

| Conventional | AI Agentic |
|---|---|
| Test reporting can be done manually with simple Excel sheets for manual test execution or using test management tools like JIRA Xray. [9]<br>For test automation, tools like Allure Report [10] are used to generate a detailed report about the test execution. | AI agents can use all the methods of reporting in conventional testing and also leverage AI/ML based reporting tools like ReportPortal [11] to automate some of the manual activities like AI-based failure reason detection and so on. |

Defect Logging - the process of documenting discrepancies or bugs identified in the system under test compared to the specified functional and non-functional requirements.

| Conventional | AI Agentic |
|---|---|
| Manual methods can use Excel sheets for defect logging but are highly inefficient.<br>This is typically done in test management tools like JIRA or ADO | AI agents can identify test execution failures and directly log them as defects into the test management tool like JIRA or ADO. Furthermore orchestration of AI agents can be used with the power of the LLM to recommend the potential fixes to code as well right into the defect. |

# 5. Good Practices

Building AI agents that are reliable and effective requires a methodical approach, particularly during testing. Here are some good practices that are recommended.

Collaborative agent creation and domain expertise

- Include domain experts early - Partnering with a domain expert from the beginning ensures the agent's development is grounded in real-world knowledge, defining accurate metrics, identifying edge cases, and ensuring the agent's decisions are aligned with industry standards and human judgment.

- Establish a continuous feedback loop - Implement a robust feedback loop that enables the agent to learn and improve from its interactions and outcomes. This involves incorporating humans in the loop where humans review, correct, and provide feedback on the agent's actions.

Comprehensive testing and multiple inputs

- Test with diverse and varied sets of inputs. An agent's true robustness is proven when it can handle a wide range of data across projects. Instead of focusing on a single input type, test the agent across multiple sets of data sources to identify weaknesses and enhance its adaptability across programs and projects.

Avoid force fitting AI agents and Keep cost in mind

- Agents should solve problems that genuinely benefit from AI. Don't try to apply an AI solution to a use case that can be handled more efficiently or accurately with traditional software. Just because you have a hammer does not mean every problem is a nail.

- Please remember, the AI tokens costs can pile up soon. Use them effectively by pre-processing the inputs using native non-AI tools prior to shipping off the content to the LLMs.

Human-Centric design and accountability

- Integrating human oversight into the agent's workflow is critical for safety, trust and primarily ownership. This approach not only improves predictability and accuracy but also addresses concerns about job displacement by positioning AI as a collaborative tool rather than a replacement.

Ensure transparency and explainability

- For an AI agent to be truly trusted, its actions should be transparent. Focus on developing agents that explain the execution plan, show sources and references used.

Control the AI Hallucinations [12]

- Use Retrieval Augmented Generation (RAG) and effective prompt engineering to reduce the amount of Hallucinations and get desirable outputs consistently.

# 6. Benefits of AI Agents

The generative AI automation agents can help in multiple ways. The integration of AI agents into testing workflows can lead to more efficient, high-quality, and cost-effective software products.

- AI agents can enhance various aspects of testing processes across the STLC and even across steps in the SDLC. This allows people to focus on more complex and creative aspects of their work.

- The agents can improve testing coverage and quality by systematically exploring a wider range of test scenarios across positive / negative / boundary conditions / accessibility / user friendliness and more.

- The agents can help identify potential issues that might be overlooked by human testers, leading to more robust and reliable software products.

- AI agents can help reduce production defects by detecting errors early in the development cycle promoting Shift Left.

- Agents can lower the overall cost of testing, making it a more cost-effective solution for organizations with time / cost from 20 to 50% [13]. Let us note that it does need training and education to be effective AI agent users.

# 7. Challenges

Developing and deploying AI agents comes with several challenges too. Below are some of the key challenges faced while implementing the AI agentic approach for testing.

- The outputs are non-deterministic and can produce varying outputs for the same input due to factors like random initialization, stochastic processes, or model complexity. [14]. This means you can get different test cases and scripts every time you execute the agent. [15]

- There are hallucinations that are so good and feel true to a common eye and may feel like a really valid test case or script. Thus they may need a good factor of prompt engineering and feedback to improve them.

- AI agents rely on high-quality, diverse datasets and RAG inputs to generate test cases or identify defects. So, we need to be cautious of the fact garbage in, garbage out.

- Integrating AI agents into traditional software testing pipelines or legacy frameworks can be complex due to compatibility issues and may need improvements / enhancements on the framework.

- AI agents can not be a replacement for the flaws in the process. The process needs to be fixed prior to using the AI agents effectively and people need to be trained to use the agents.

# 8. Case Study

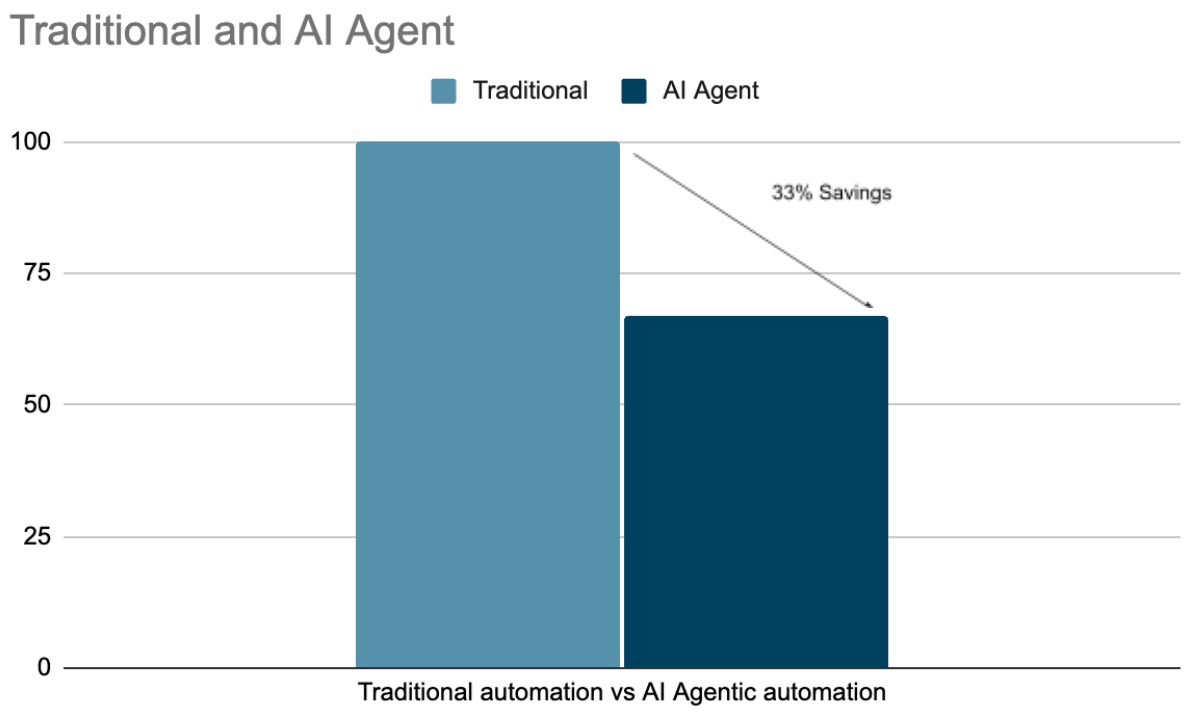
A financial company team facing a large automation backlog and seeking to reduce testing time deployed an AI Test Automation agent to generate test scripts in the gherkin format into an existing automation framework. The AI agent automated the creation of test scripts for previously manual tests, resulting in over **~33%** savings in time. This initiative successfully addressed the backlog and significantly accelerated the company's test automation efforts. The success led to early defect detection and overall a faster and higher quality release.

Tech Stack

- Test Cases Location - JIRA
- Test Automation Framework - Selenium WebDriver
- Behavior Driven Development (BDD) - Gherkin
- Programming Language - Java
- Version Control - Git

Savings and Benefits

- The average test automation speed was reduced from ~4.5 hours to under 3 hours, resulting in a 33% reduction in time spent on test script creation and test execution. *(factoring in some agent development and training time).*



# 9. References


1. “Model Context Protocol (MCP)”, https://modelcontextprotocol.io/introduction (accessed 01 Jul 2025)
2. “What are AI agents?”, https://www.ibm.com/think/topics/ai-agents (accessed 03 Jul 2025)
3. “What is an AI agent?”, https://cloud.google.com/discover/what-are-ai-agents (accessed 06 Jul 2025)
4. “Agentic AI”, https://youtu.be/kJLiOGle3Lw (accessed 03 Jul 2025)
5. “Building Agents”, https://platform.openai.com/docs/guides/agents (accessed Jul 2025)
6. “Types of AI Agents”, https://www.ibm.com/think/topics/ai-agent-types (accessed 05 Jul 2025)
7. “The Landscape of Emerging AI Agent Architectures for Reasoning, Planning, and Tool Calling: A Survey”, https://arxiv.org/abs/2404.11584 (accessed 05 Jul 2025)
8. “AI Agents in Software Testing”,https://testrigor.com/ai-agents-in-software-testing (accessed Sep 2025)
9. “How to create and manage test cases with Xray and Jira“, https://www.atlassian.com/devops/testing-tutorials/jira-xray-integration-manage-test-cases (accessed 06 Sep 2025)
10. “Allure Report”, https://allurereport.org/ (accessed Sep 2025)
11. “Report Portal”, https://reportportal.io/ (accessed Jul 2025)
12. “AI Agents Best Practices and Ethical Considerations: Implementing AI in Business”, https://writesonic.com/blog/ai-agents-best-practices, (accessed Sep 2025)
13. “Reshaping Software Engineering at Edward Jones”, https://partners.wsj.com/epam/ai-driven-software-engineering/reshaping-software-engineering-at-edward-jones/ , (accessed Sep 2025)
14. “Machine Learning Basics”, https://www.deeplearningbook.org/contents/regularization.html, author={Ian Goodfellow and Yoshua Bengio and Aaron Courville}, (accessed Sep 2025)
15. “Regularization for Deep Learning”, https://www.deeplearningbook.org/contents/ml.html, author={Ian Goodfellow and Yoshua Bengio and Aaron Courville}, (accessed Sep 2025)