



COPYRIGHT ISTOCKPHOTO, CREDITLUCANDY

Software Testing in the Generative AI Era: A Practitioner's Playbook

Douglas C. Schmidt , William & Mary

This article presents a practitioner-focused playbook on how generative artificial intelligence is transforming software testing by automating tasks like test generation, simulation, and anomaly detection, while also introducing new challenges like hallucinations, bias, and nondeterminism.

Software testing is undergoing a paradigm shift due to generative artificial intelligence (AI). Imagine test cases that write themselves by analyzing your documentation and code, user behaviors being simulated before any real user ever touches the software, and continuous

feedback guiding development in real time. Generative AI—especially large language models (LLMs) trained on vast data sets—is turning this vision into reality. These tools can analyze your requirements and code to produce test cases automatically, synthesize realistic test scenarios on the fly, and even monitor running systems for anomalies. The promise is compelling: you and your team can test faster, smarter, and more thoroughly.

However, these AI-augmented advances also introduce new risks. The same AI tools that boost testing can produce unpredictable behavior—from biased outputs or “hallucinated” results to inconsistent, nondeterministic responses that are hard to reproduce or verify. In high-stakes domains like health care, finance, and defense, where lives and missions are on the line, ensuring the reliability and trustworthiness

Digital Object Identifier 10.1109/MC.2025.3562940
Date of current version: 27 June 2025

of AI-augmented testing is paramount. Traditional quality assurance (QA) practices must adapt to issues like AI bias, unexpected outputs, and regulatory compliance.

This article explores how generative AI is reshaping software testing—the benefits it offers, the challenges it raises, and how you can adapt. With AI coding assistants like OpenAI’s Codex, GitHub Copilot, and Meta’s Code Llama increasingly integrated development environments, continuous integration/continuous delivery (CI/CD) pipelines, and cloud platforms, teams need to adjust quickly. Understanding how these tools work—and recognizing when they don’t—is becoming a critical skill. The guidance here draws on my recent stint as the Director of Operational Test and Evaluation and related collaborations with colleagues.^{1,2,3}

HOW GENERATIVE AI IS AUTOMATING AND SCALING TESTING TODAY

Generative AI is extending the reach of test automation. Instead of manually writing every test or trying to guess all possible user behaviors, you can offload many tedious and error-prone tasks to AI. The AI-augmented testing

capabilities shown in **Figure 1** serve as accelerators for software QA, and are described as follows.

Automatic test case generation

Tools like Diffblue Cover and CodiumAI can analyze your code and generate unit tests across a variety of languages. They can even integrate with your continuous integration pipelines (for example, Jenkins or GitLab) for automatic execution.⁴ For instance, an LLM can read a specification and produce a suite of tests covering the expected behaviors. This automation helps validate intended functionality early and targets known weak spots. It also expands coverage while reducing the manual effort of writing tests.

Continuous testing and feedback

Generative AI supports continuous testing by monitoring systems and providing rapid feedback. Machine learning models can watch test executions or even live production logs to spot anomalies and failures in real time.⁶ If a test fails, an AI assistant might immediately analyze the recent code changes and suggest likely causes. AI can also detect patterns in past defects to recommend new test cases or fixes. In modern

Development, Operations, and Security pipelines, every code commit can trigger AI-generated tests and anomaly detection, providing instant feedback and catching issues earlier.

Early prototyping and “shift-left” testing

Generative AI blurs the line between development and testing by enabling earlier prototyping of features. Rather than waiting for a fully integrated system to begin testing, an AI tool can create a mock interface or simulate a new feature from just a design description. This “shift-left” approach allows you to exercise and evaluate components before writing any real code, thus shortening the QA feedback loop and catching design flaws sooner in the development process.

Simulating diverse usage

Generative AI enables you to simulate a wide range of user behaviors and environmental conditions. For instance, you might prompt an LLM to act as various personas⁵ (for example, a novice user, an expert, even an adversarial attacker) to see how your software handles different inputs and sequences of actions. AI-augmented simulations can also mimic conditions that are

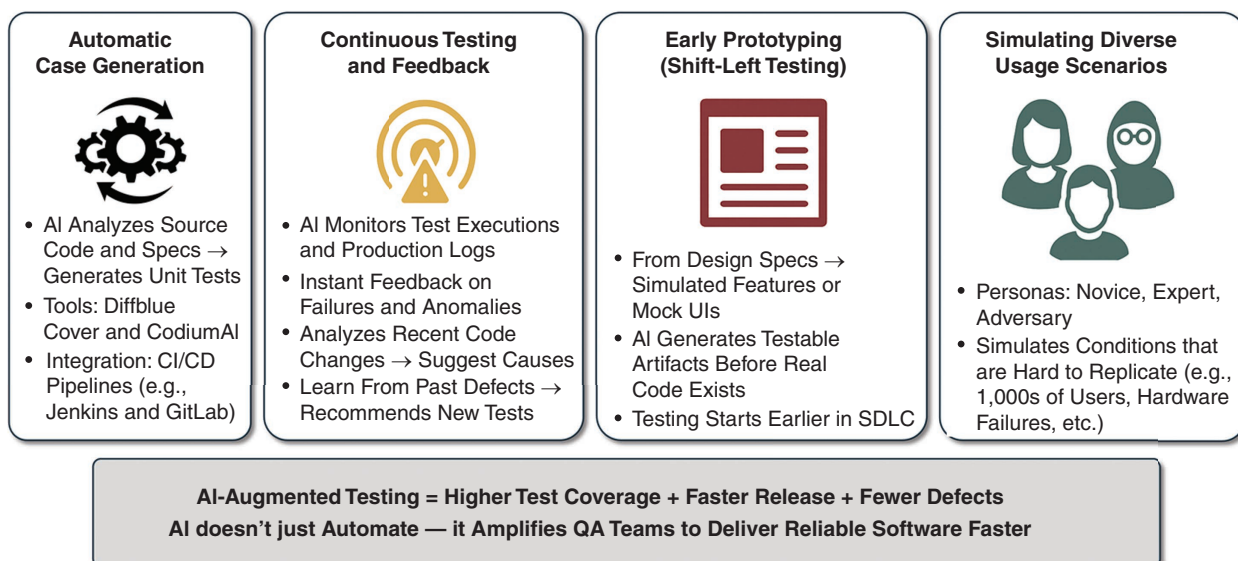


FIGURE 1. Accelerating software testing with generative AI. UI: user Interface; SDLC: software development life cycle.

impractical to reproduce manually (imagine thousands of virtual users or sudden sensor failures). Testing under these diverse scenarios gives you greater confidence in the robustness of your system.

These AI-augmented techniques can widen your test coverage (including edge cases), catch bugs earlier, and speed up the feedback cycle. By offloading tedious work to AI and injecting intelligence into testing, generative AI becomes a force multiplier for QA, enabling you to deliver more reliable software faster.

CHALLENGES OF AI-AUGMENTED TESTING

Along with the benefits of AI-augmented testing come new challenges, there are several key areas you need to address, which are shown in [Figure 2](#) and described as follows.

AI “hallucinations”

AI systems sometimes produce answers or code that look correct and confident but are completely wrong.⁷ For example, an AI-generated test might assert an output value that appears plausible (perhaps it even aligns with the documentation) but is incorrect. If you rely on that test, it could pass or fail for the wrong reasons. In short, you can’t blindly trust AI-generated outputs—especially for critical logic—without verification. Keep a human in the loop to review important AI-generated tests and results.⁸ Likewise, perform adversarial testing; for example, feed malformed or malicious inputs to the AI and see if the system goes off track. Your goal is to ensure the system handles uncertainty properly, for example, by indicating when it doesn’t know an answer instead of confidently fabricating one.

Nondeterministic behavior

Unlike traditional software, AI components might not behave the same way every time. A conventional function will produce identical outputs for identical inputs, but an AI model might respond slightly differently on each run,

which makes it hard to consistently reproduce results. For example, if a test passes once but fails on the next run, is that due to randomness or a real bug? You may need to rerun certain tests numerous times to understand the range of possible outcomes. Even small input tweaks or data updates

shouldn’t, such as make an unauthorized decision or reveal sensitive information. Emerging regulations like the European Union’s AI Act will likely mandate evidence of an AI system’s fairness, transparency, and risk controls. AI tools can assist by scanning for compliance issues,⁹ but ultimately,

Generative AI blurs the line between development and testing by enabling earlier prototyping of features.

can shift an AI’s responses, so watch out for model drift—when an AI’s behavior changes over time as its data or parameters evolve. Ensuring quality for nondeterministic systems means gathering far more test data and often defining new metrics. It’s a shift away from the traditional “pass/fail” mindset toward measuring whether the AI’s performance stays within an acceptable range of reliability.

Compliance and ethics

In regulated domains, you must ensure that AI systems comply with all relevant laws and ethical standards, for example, a healthcare AI should never leak patient data. Your testing should include scenario-based checks that exercise policy constraints, for example, check that AI never does something it

you and your team are accountable for ensuring that AI-augmented software stays within legal and ethical bounds.

Bias and fairness

Generative models learn from their training data, so any biases in that data can creep into AI-generated tests or outputs. If the training data underrepresents certain user groups or conditions, the AI might overlook those scenarios, leading to blind spots in quality. So don’t just ask “Does it work?”—also ask “Does it work for everyone using our software?” In practice, you’ll need to use diverse inputs, cover underrepresented scenarios, and watch for skewed or unfair results. If you detect bias, you may need to adjust your test data or retrain the model to improve its fairness, even if

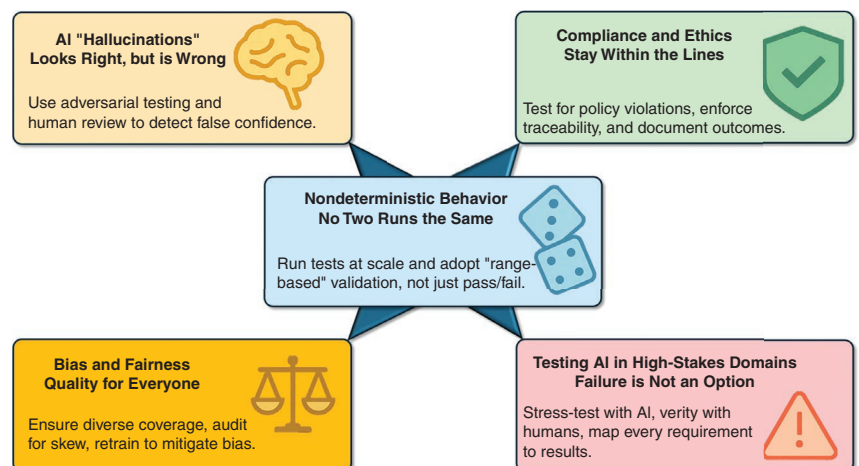


FIGURE 2. Challenges of AI-augmented software testing.

that takes extra effort. These considerations broaden the notion of software QA to include factors like fairness, explainability, robustness under uncertainty, and ethical integrity. They aren't insurmountable but addressing them requires evolving your testing strategies—especially in safety-critical environments where even a small AI misstep can be disastrous.

Testing AI in high-stakes domains

The stakes are elevated in high-stakes domains like defense, aerospace, and health care, which can't tolerate failures in AI-augmented systems. A flaw in an autonomous emergency response drone's software or a medical decision-support AI could be deadly. In these cases, you must demonstrate that the system works correctly even under extreme or unusual conditions. Generative AI can help by simulating numerous variations of scenarios (for example, different operating conditions, emergency situations, rare edge cases) to stress-test the system and expose weaknesses.

Even so, rigorous human oversight is essential, so "trust but verify" at every step. If AI-augmented testing tools report everything passed, you should still spot-check the results to identify edge cases the AI might have missed. In high-stakes

domains, even rare oversights can be unacceptable. Traceability is also crucial, so document how each requirement maps to specific tests/outcomes to justify your test coverage and explain test results to stakeholders and regulators. Whenever AI is involved in a regulated, safety-critical system, be prepared to show exactly what was tested and verify nothing was left to chance.

The lesson from high-stakes domains is clear: use AI to amplify your testing, but maintain strict human oversight and accountability. Reliability, transparency, and safety must remain paramount whenever AI is part of a mission-critical system.

ADAPTING TESTING PRACTICES FOR THE AI ERA

To harness generative AI in testing while mitigating its risks, you'll need to evolve your QA practices. Several ways to update your approach for the AI era are shown in Figure 3 and described as follows.

Foster AI literacy in QA teams

Ensure your testing team learns how AI works—and where it can fail. Testers should learn the basics of model training and evaluation, familiarize themselves with common AI failure modes, and adopt prompt engineering

techniques and patterns⁵ to apply LLMs effectively. This knowledge helps them use AI-based tools robustly and recognize when an AI's output might be misleading. Encourage your team to practice crafting prompts and critically analyze AI-generated results. With greater AI literacy, your team can treat AI as a powerful assistant under their supervision, rather than a mysterious black box.

Encourage cross-functional collaboration

Testing AI-infused systems isn't just the QA team's job. Developers, testers, data scientists, and domain experts should all collaborate on AI-augmented projects. Involve QA early in the design of AI features so they can advise on testability and anticipate problems. Likewise, developers should build transparency and test hooks into AI components, for example, logging an AI model's key decision factors for later review. Involve domain experts to ensure test scenarios reflect real-world conditions, for example, have medical doctors contribute cases to test a healthcare AI or have financial experts suggest scenarios to test fintech AI. Breaking down silos and sharing knowledge across roles yields more robust testing strategies for AI-augmented systems.

Upgrade your tool kit and techniques

Keep your proven QA methods but augment them with AI-augmented approaches. For example, introduce adversarial testing (to try "breaking" the AI with malicious or unexpected inputs) and apply statistical analysis to interpret test results since AI outputs are probabilistic rather than deterministic.³ You may need new metrics to quantify qualities like bias or uncertainty in the AI's output. Likewise, experiment with emerging AI-augmented test generation and analysis tools to see how they can boost your productivity. By expanding your toolbox, you can better cover AI-specific

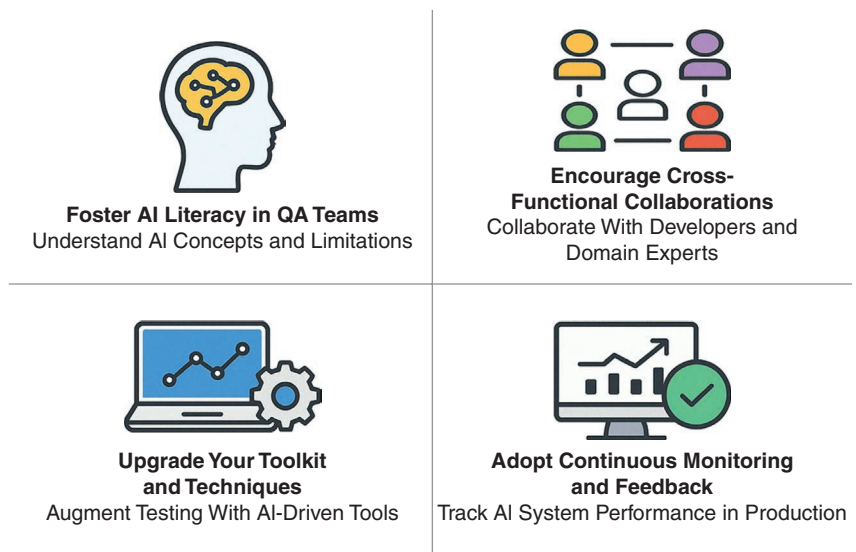


FIGURE 3. Adapting testing practices for the AI era.

failure modes and make your testing regime more effective.

Adopt continuous monitoring and feedback

Don't treat testing as a one-and-done phase. After deployment, continue to monitor your AI components in production and feed those insights back into your tests. AI-augmented monitoring tools (such as Amazon DevOps Guru or Sentry) can watch production systems for anomalies, performance drift, or new types of errors, then alert your team. If a model's performance degrades over time, be ready to update test cases or retrain the model as needed. Schedule regular audits of AI outputs to catch issues that only appear with production deployments. AI can automate parts of this monitoring and analysis, but human experts should review the results and guide improvements. Maintaining this feedback loop will keep your AI systems reliable long after deployment.¹⁰

By putting these strategies into practice, you can turn generative AI into a real asset in your QA process. Testing will become more adaptive and comprehensive, keeping pace with AI-augmented development while upholding high standards of quality and trust.

YOUR AI TESTING PLAYBOOK: WHAT TO DO NEXT

Generative AI is rapidly reshaping software testing and early adopters are already seeing significant benefits. Autogenerated tests and large-scale simulations are catching bugs that previously slipped through manual efforts and faster feedback loops are helping teams iterate quickly. When applied properly, these techniques can yield more reliable software and shorter development cycles by enabling smarter testing.

At the same time, you need a clear view of AI's limitations. An AI-augmented system is only as good as the data and algorithms behind it, so it can still exhibit bias, produce incorrect results, or behave unpredictably.

Human judgment, disciplined testing practices, and strong processes remain essential to validate and rein in AI contributions. Remember to "test the tester" by building in safeguards to catch mistakes or quirks in the AI tools themselves. By updating your practices and skill sets to account for these issues, you can harness the benefits of AI while mitigating its risks.

So what should you do now? Figure 4 shows a few concrete next steps for different roles on your team, which are explained as follows.

- **Software engineers and developers:** Integrate generative AI tools into your development and testing workflow to boost coverage and catch issues early. Use AI to generate unit tests and to simulate complex user flows. Don't assume the AI is always right—review and refine every AI-generated test case or code suggestion. Design your software with testability in mind, for example, add logging and hooks around AI-augmented features so you can verify their outputs. Let AI handle the tedious parts of testing but apply your judgment to discern what's correct.
- **QA and testing professionals:** Leverage AI as a partner to extend your testing. Use AI tools to generate diverse test scenarios and synthetic data, uncovering bugs that traditional methods might miss. Focus on validating the AI's behavior, for example, look for biased outputs or hallucinations and apply adversarial testing to stress-test AI-augmented features. Champion new types of checks in your test plans (for example, include explicit fairness and bias tests for AI outputs). Orchestrate the best of both worlds—use AI to work more efficiently while you focus on ensuring accuracy and integrity.
- **Technical leaders:** Create an environment that integrates AI into quality assurance thoughtfully. Invest in training your teams on AI tools and concepts and give them room to experiment with AI-augmented testing solutions. Update your QA processes to add checkpoints for AI-related risks, for example, require bias testing for machine learning components, and mandate human review of critical AI-augmented decisions or recommendations. Promote cross-team collaboration so that development, QA,

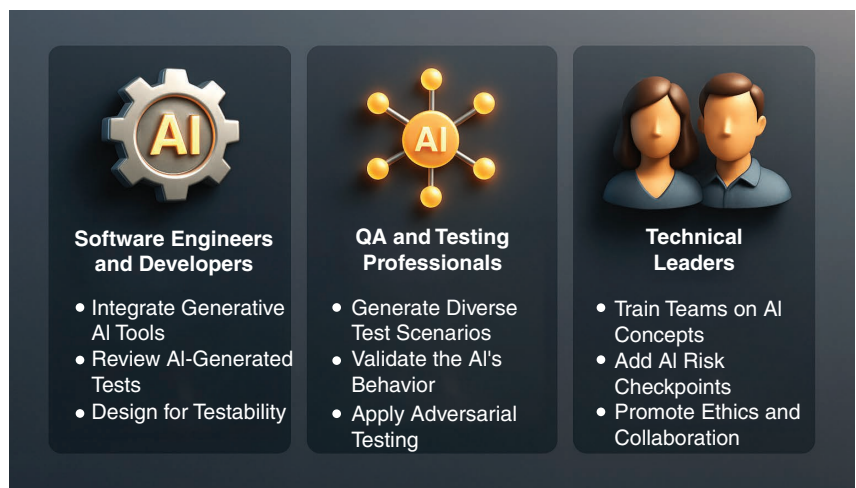


FIGURE 4. Testing roles for AI-augmented software systems.

and data science teams are aligned on quality goals. Likewise, insist on high ethical standards (such as transparency, fairness, and accountability) for any AI usage in your products and processes. With strong leadership and the right guardrails, your organization can harness generative AI responsibly to deliver better software faster! **C**

REFERENCES

1. L. J. Freeman, D. C. Schmidt, A. Bonnell, J. Robert, H. Wojton, and Acquisition Innovation Research Council, AIRC Panel on Generative AI in the Acquisition Lifecycle. (Jan. 7, 2025). [Online Video]. Available: <https://www.youtube.com/watch?v=ZlCc94w-2bY>
2. L. J. Freeman, J. Robert, and H. Wojton, "The impact of generative AI on test & evaluation: Challenges and opportunities," in *Proc. Int. Workshop Envisioning AI-Augmented Softw. Develop. Lifecycle*, Trondheim, Norway, 2025, pp. 1-5.
3. P. Roy, J. Chandrasekaran, E. Lanus, L. J. Freeman, and J. Werner, "A survey of data security: Practices from cybersecurity and challenges of machine learning," 2023, *arXiv:2310.04513*.
4. F. Ricca, A. Marchetto, and A. Stocco, "AI-based test automation: A grey literature analysis," in *Proc. IEEE Int. Conf. Softw. Testing, Verification Validation Workshops (ICSTW)*, 2021, pp. 263-270, doi: [10.1109/ICSTW52544.2021.00051](https://doi.org/10.1109/ICSTW52544.2021.00051).
5. J. White et al., "A prompt pattern catalog to enhance prompt engineering with ChatGPT," in *Proc. 30th Pattern Lang. Program. (PLoP) Conf.*, Allerton Park, IL, USA, pp. 1-31, 2023.
6. P. Dogga, K. Narasimhan, A. Sivaraman, S. Saini, G. Varghese, and R. Netravali, "Revelio: ML-generated debugging queries for finding root causes in distributed systems," in *Proc. Mach. Learn. Syst. 4 (MLSys)*, pp. 601-622, 2022.
7. F. Liu et al., "Exploring and evaluating hallucinations in LLM-powered code generation," 2024, *arXiv:2404.00971*.
8. N. Maleki, B. Padmanabhan, and K. Dutta, "AI hallucinations: A misnomer worth clarifying," in *Proc. IEEE Conf. Artif. Intell. (CAI)*, 2024, pp. 133-138, doi: [10.1109/CAI59869.2024.00033](https://doi.org/10.1109/CAI59869.2024.00033).
9. A. S. Chandrasekaran, "Harnessing the power of generative artificial intelligence (GenAI) in governance, risk management, and compliance (GRC)," *Int. Res. J. Eng. Sci. Technol. Innov.*, vol. 11, no. 5, p. 11, 2024.
10. E. Lanus, B. J. Lee, L. Pol, D. Sobien, J. Kauffman, and L. J. Freeman, "Coverage for identifying critical metadata in machine learning operating envelopes," in *Proc. IEEE Int. Conf. Softw. Testing, Verification Validation Workshops (ICSTW)*, 2024, pp. 217-226.

DOUGLAS C. SCHMIDT is dean of the School of Computing, Data Sciences & Physics, William & Mary, Williamsburg, VA 23185 USA. Contact him at douglas.c.schmidt@wm.edu.

IEEE Annals of the History of Computing



IEEE Annals of the History of Computing publishes work covering the broad history of computer technology, including technical, economic, political, social, cultural, institutional, and material aspects of computing. Featuring scholarly articles by historians, computer scientists, and interdisciplinary scholars in fields such as media studies and science and technology studies, as well as firsthand accounts, *Annals* is the primary scholarly publication for recording, analyzing, and debating the history of computing.



www.computer.org/annals

Digital Object Identifier 10.1109/MC.2025.3573140