Quality engineering & assurance

GitHub Copilot: Harnessing the power of AI for quality engineering

New era of efficiency with Gen AI

The advent of generative AI has created a significant stir in the technology landscape. This transformative shift has opened doors for innumerable opportunities for the quality engineering (QE) ecosystem and set to revolutionize the way QE is conducted. Gen AI has introduced a new era of efficiency and effectiveness, harnessing the power of AI algorithms trained on vast and varied datasets.

While we have been early adopters of code-assist tools such as GitHub Copilot and have skilled our Quality Engineers for its usage, we have kept pace with the Gen AI technology evolution. Cognizant has invested in platforms, tools and accelerators that help clients reap the benefits of LLM usage, Agentic AI deployment across the Quality Engineering lifecycle.

GitHub Copilot in QE lifecycle

Code-assist tools for testing have become an integral component of the QE process, driven by advancements in artificial intelligence (AI) and machine learning (ML). One such code-assist tool, Copilot augments traditional testing methodologies, streamlines processes, improves accuracy and creates bandwidth for testers to focus on more complex and strategic elements of QE.

Copilot has an intuitive interface that facilitates quick adaptation for teams. Its capability to learn from existing codebases ensures ongoing relevance and effectiveness. Regular updates and enhancements to Copilot keep it aligned with evolving industry standards, making it a vital tool for modern QE practices.

Cognizant has helped organizations leverage GitHub Copilot in their quality activities. This integration streamlines the development and testing phases, ensuring a more efficient workflow. Organizations are experiencing increased productivity and expedited delivery of high-quality software products.

Benefits driven by Copilot

→ Software quality:
- Increased test coverage
- Improved automation %
- More in sprint automation
- Better quality of test code

→ Time to market:
- Reduced onboarding time
- Testing cycle time reduction
- Faster release to market

→ Efficiency:
- Increase in ROI
- Decreased dependency on specific coding skills
- Better utilization of resources
- SMEs focus on more strategic activities

GitHub

Recognized as a Leader in the Gartner Magic Quadrant for AI code assistants, 2024

Highlighting its superior ability to execute and completeness of vision: "GitHub Copilot is recognized as a transformative tool in AI-driven software development and quality assurance (QA). With millions of developers and over 77,000 organizations using Copilot, its integration is helping teams accelerate their development processes by automating mundane tasks and boosting overall productivity. It not only enhances the speed of development but also maintains a high level of code quality, making it ideal for QA processes where precision and efficiency are crucial."

Exploring various use cases for quality engineering

Integrating GitHub Copilot throughout the software development lifecycle allows QE teams to reduce manual effort, boost speed and precision, and enhance collaboration, which ultimately improves software quality. Cognizant groups some of the widely implemented use cases under broader categories of generation, augmentation, and optimization:

→ Generation
Greenfield test suite generation

Enhanced scripting
Automating repetitive scripting tasks, allowing teams to focus on enhancing business coverage.

Test artifact generation
Copilot translates user stories directly into test artifacts, accelerating the testing cycle.

Full stack automation
Enables full stack automation with capabilities to enhance testing across UI, API, middleware and backend.

→ Augmentation
Suggestions to improve quality of existing artifacts

Real-time code suggestions
AI-powered Copilot provides real-time code suggestions, which can help you write efficient and error-free test scripts, improving the overall quality of your tests.

Code completion and snippet generation
GitHub Copilot provides code snippets for QE tasks.

Framework migration
Enhances productivity during test automation framework migration from one tech stack to the other.

→ Optimization
Recommendations to improve user efficiency

Code review and fix suggestions
Helps in identifying and suggesting fixes for bugs in code.

Code optimization & refactoring
Identifies inefficient code patterns and suggests improvements, ensuring performance optimizations without sacrificing quality.

Transforming nature of 'day in a life scenario' for QE

With its core feature of code generation, the applicability and wide adoption of AI code assistants can be observed in several areas:

- Test suite generation from requirement documents
- Test automation script generation from manual test cases
- Test data generation
- Code optimization and review
- Automation script maintenance
- Frame/script migration

One such variation for its applicability is when clients are migrating from one automation framework to another (e.g., Selenium to Playwright).

All existing test suites must be redesigned in the new framework being adopted. This migration involves identifying associates who have the necessary skills, accurate scripting in the new framework, design, and finally migration to the test suite.

With a lean team of quality engineers possessing framework scripting knowledge and assistance from AI code assistant tools, such framework migrations have been successfully completed. Cognizant has helped clients benefit from the quick turnaround and significantly reduced effort.

As industries move toward digital transformation,

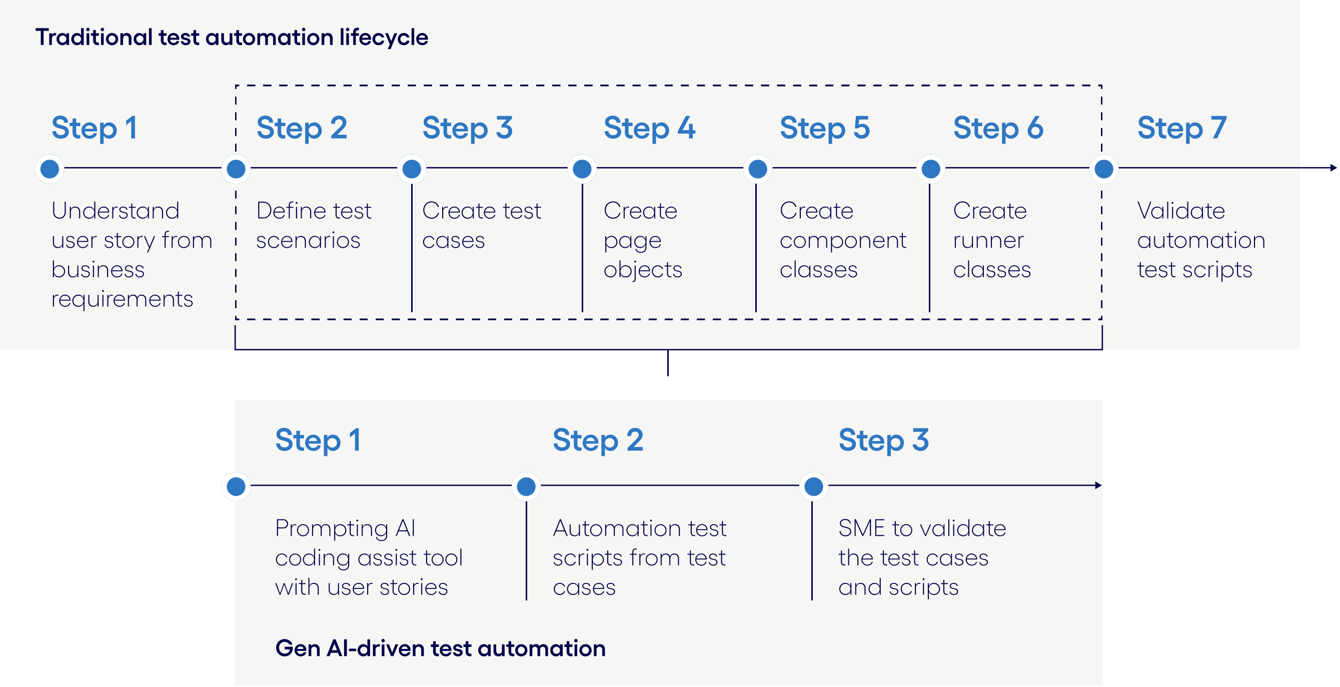
the adoption of **AI-driven tools** is proving indispensable in modernizing quality engineering practices.

Cognizant quality engineering use cases span across the SDLC

1	**Planning & requirements**	Epic / features to user story	Requirement to test cases	Test strategy generation		
2	**Design**	User story to BDD	User story to manual TC	Impact analysis	Test data requirements	
3	**Build**	BDD step definition	Manual to automation script generation	Synthetic test data generation	Web & API test script	NFT test script
		CI/CD pipeline YAML creation	Debug/fix	Code documentations	Generate release notes	Generate commit messages
4	**Testing**	Self healing auto script	Reg. test optimizer	Schema validation	RCA & failure analysis	
5	**Production release**	Metrics	Test summary reports			
6	**Maintenance**	Framework migration	Code refactoring	Real-time visibility		

Transformation of test automation lifecycle

Designing test automation scripts traditionally involves significant effort for creation of a modular, reusable and easily maintainable framework such as page-object models, which can be extended to an enterprise-grade test suites. The use of AI coding assistants has significantly shifted the way test automation scripts are designed and has reduced the framework creation efforts in generating standardized code snippets and real-time suggestions for code optimization.



Traditional test automation lifecycle

Step 1
Understand user story from business requirements

Step 2
Define test scenarios

Step 3
Create test cases

Step 4
Create page objects

Step 5
Create component classes

Step 6
Create runner classes

Step 7
Validate automation test scripts

Step 1
Prompting AI coding assist tool with user stories

Step 2
Automation test scripts from test cases

Step 3
SME to validate the test cases and scripts

Gen AI-driven test automation

Boosting quality engineering

A typical situation for service providers is the availability of test suites that are not adequately mapped back to the right business requirement. This leads to challenges in identifying a regression test suite, generation of test data, automation suite creation, etc. One of the ways GitHub Copilot has supported quality engineers is to reverse engineer test suites to generate requirement documents. This use case was significant when GitHub Copilot was leveraged to generate BDD feature files from the automation suite, easing the process of traceability and test suite maintenance.

Cognizant has also explored the GitHub Copilot feature for the non-functional testing events of the software development lifecycle (SDLC). For example, API script development assisted by GitHub Copilot for performance testing has shown significant design time reduction, decreasing the overall non-functional testing time.

The following are significant use cases in the performance testing and engineering space offered for clients:

- Performance test script development on APIs
- Migration of performance test scripts
- Creation of virtualized services
- Development of automated frameworks for the performance engineering lifecycle.

Redeploying quality engineer bandwidth

With support from code assisting tools, quality engineers can focus more on brand assurance and customer experience assurance.

- Refining test strategies
- Edge case analysis
- Risk based testing

- Optimizing automation frameworks
- Cross-platform testing capabilities
- Enhance CI/CD integration

- In-depth performance and security engineering
- Customer experience enhancements

- Improve industry-specific business rules coverage
- Regulatory compliance validations
- Exploratory testing

Maximizing gains with GitHub Copilot adoption

Cognizant technology experts have observed that businesses can maximize the potential of GitHub Copilot, focusing on few important elements:

Create thoughtful prompts

- Specific and clear short prompts
- Use role-based prompts to break down complex tasks, providing examples of input and output

Review the work of Copilot and maintain code quality

- Review the code suggestions to ensure they meet the project's coding standards, security and performance requirements
- The output from Copilot should be treated as starting points rather than final solutions

Provide context

- Open relevant files in the IDE to provide Copilot with more context
- Use comments to guide Copilot and improve the relevance of the suggestions

Stay updated

- Follow new features and updates to maximize the evolving capabilities of Copilot.

1

Strategic user stories with detailed acceptance criteria
Implement a user story analysis tool to validate and enhance the quality of user stories, ensuring they provide essential context for creating precise BDD scenarios and effective manual test cases

2

Robust automation framework
Usage of Modularized, data-driven and easily maintainable frameworks like Cognizant Reusable Automation Framework for Testing (CRAFT)

3

Integration with development tools and processes
Multi Agent architecture for achieving effective integration with existing development tools and processes to maintain workflow continuity

4

Comprehensive manual test cases
Use few-shot prompts to guide the creation of manual test cases, ensuring clarity in quality engineering objectives and facilitating accurate and effective automation scripts

5

Training and familiarity with the capabilities of Copilot
Conduct regular training sessions and updates to help the team adopt the tool effectively, boosting productivity and maximizing potential

Conclusion

The future of QE is undoubtedly intertwined with AI. At Cognizant, we have revolutionized every phase of Quality Engineering by integrating AI and Gen AI solutions that have enhanced the quality of releases, improved time to market and productivity. With our solutions, frameworks and accelerators that work seamlessly with LLMs, GenAI based code-assist tools, SLMs – our clients have seen early success in AI adoption.

Cognizant has leveraged GitHub Copilot to revamp quality engineering processes into highly efficient, automated workflows. The Quality Engineering community can benefit further with additional features and capabilities related to generation, augmentation and optimization in the tool.

As GitHub Copilot continues to evolve, it is positioned to become a key enabler in skill transformation. It is empowered to perform activities such as infrastructure as a code, pipeline as a code, defect fixes; in addition to automating quality assurance tasks, reducing human intervention in repetitive testing activities and improving overall software quality.

About Cognizant's quality engineering & assurance practice

We put AI-driven Quality Engineering at the forefront to drive technology and mindset change for IT and business. We achieve first time right quality for enterprises as they modernize their foundation, build new business models and reimagine customer experiences for success in digital and phygital ecosystems.

For more details about us click here.

"By 2027, 80% of enterprises will have integrated AI-augmented testing tools into their software engineering toolchain, which is a significant increase from approximately 15% in early 2023."

Gartner
Magic Quadrant for AI Code Assistants

cognizant®