THE COMING ENTERPRISE SOFTWARE QUALITY CRISIS: WHY AI-GENERATED CODE DEMANDS BETTER TESTING

As AI accelerates software development and cybersecurity threats, enterprises need a new approach to software quality—powered by design-driven synthetic data.

Nine Major Outages. One Warning for Enterprise Software.

Between November 12 and 22, 2026, nine widely reported technology outages disrupted enterprise services, financial transactions, air travel, and government operations. The incidents affected organizations whose systems support everyday business at enormous scale:

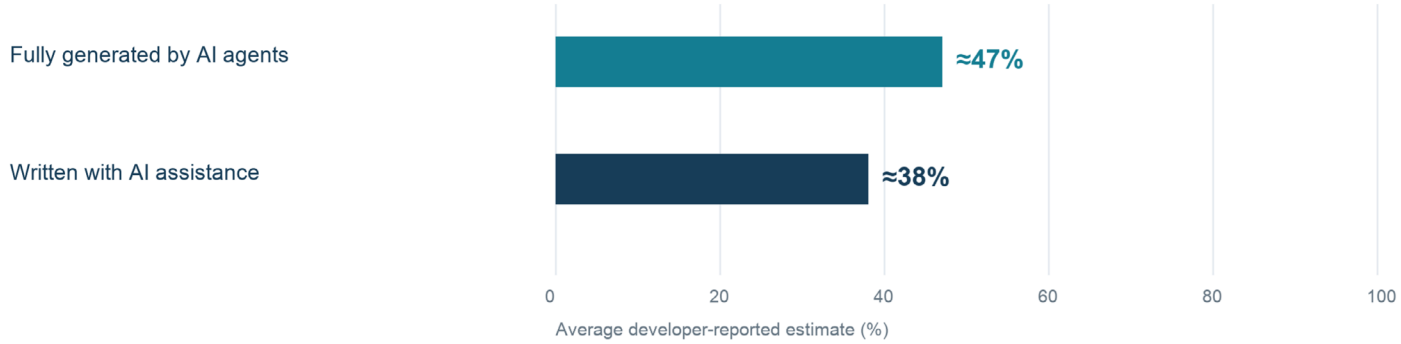
- Microsoft, November 12–13. An outage disrupted Exchange Online and dependent Microsoft 365 features.
- Workday, November 13–14. An issue involving its disaster-recovery environment forced affected production tenants into maintenance.
- Federal Aviation Administration, November 13. A network problem prompted a nationwide ground stop, disrupting air travel.
- Amazon Web Services, November 15. Authentication and connectivity problems affected services in its US-East-1 region.
- JPMorgan Chase, November 16. A software problem delayed transaction processing, with effects extending to other financial services.
- United Airlines, November 16. A technology outage delayed departures while the airline worked to restore systems.
- Apple, November 17. An internal update disrupted services including the App Store, Apple Music, and iCloud sign-in.
- National Security Agency, reported November 17. An internal network disruption affected agency communications during maintenance and security patching.
- Delta Air Lines, November 22. A weight-and-balance automation failure delayed more than 1,000 flights and required manual workarounds.

AI Is Changing the Economics—and Risks—of Software Development

Generative AI is transforming software engineering. Coding assistants and autonomous development agents can generate functions, modify applications, refactor systems, and produce working code in a fraction of the time required by traditional methods.

According to the 2026 JetBrains Developer Ecosystem research, based on responses from more than 15,000 professional developers worldwide, respondents estimated that an average of 47% of their work code was fully generated by AI agents and 38% was written with AI assistance.

AI shapes a substantial share of coding work



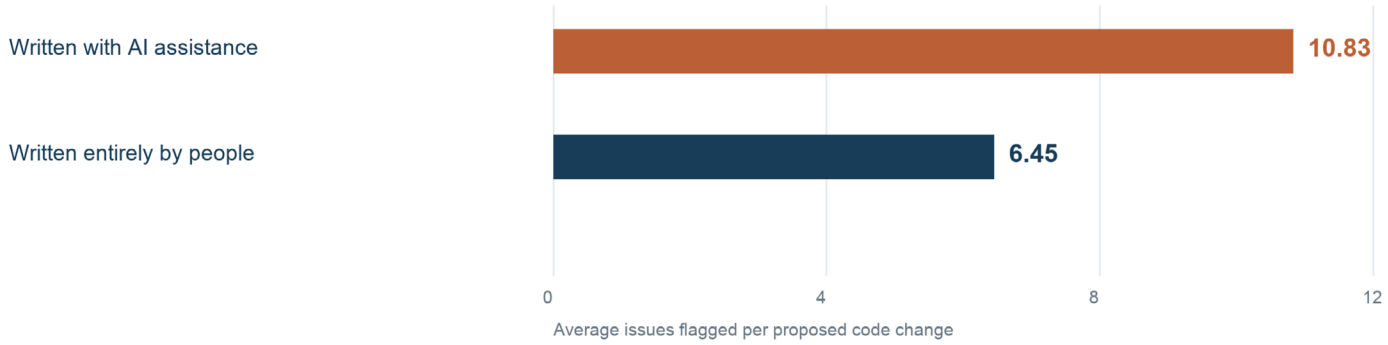
Source: JetBrains Developer Ecosystem Survey 2026 • More than 15,000 professional developers
Separately reported estimates of work code; percentages are not additive. [1]

But there is a critical distinction between generating code that works and generating code that behaves correctly under the full range of conditions it may encounter.

CodeRabbit examined 470 proposed software changes submitted for review before being added to a codebase—process developers call a "pull request." Its automated reviews flagged an average of 10.83 issues in changes written with AI assistance, compared with 6.45 in changes written entirely by people. That is approximately 1.7 times as many issues. Problems involving logic and correct behavior were 75% more common in the AI-assisted group.

These results show why generated code needs careful review and testing before release. The study measured issues flagged during code review.

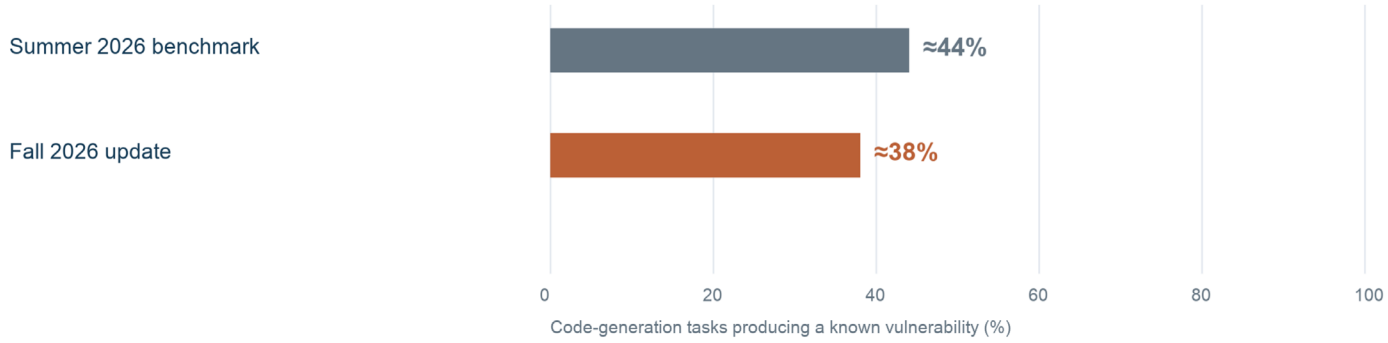
Reviews flagged more issues in AI-assisted code changes



CodeRabbit • 470 proposed code changes reviewed • Approximately 1.7× as many issues overall
Logic and correctness issues were 75% more common in the AI-assisted group. [2]

Veracode's summer 2026 security benchmark found that approximately 44% of tested AI code-generation tasks produced code with a known vulnerability. Its fall update reported improvement to approximately 38%, or a 62% security pass rate. Even with that improvement, more than one in three tested tasks still produced vulnerable code—reinforcing the need for security validation as part of software delivery.

Security improved, but the benchmark still reveals risk



Veracode GenAI Code Security research • Summer and fall 2026 results
Approximate share of tested code-generation tasks producing a known vulnerability. [3]

AI can dramatically increase software development velocity. But increased velocity without a corresponding improvement in validation creates a growing gap between the speed of code production and the ability to establish its quality.

The Hidden Weakness of Production-Based Test Data

For decades, enterprise software testing has depended heavily on production data. Organizations extract subsets of production databases, mask sensitive information, and use those datasets to validate applications before release.

This provides realistic business entities, relationships, transaction patterns, and operational histories. But production data represents what has already happened. It does not necessarily represent everything that could happen.

Delta's November 22 disruption shows the operational consequences of losing a critical automated system. ABC News reported that an outage in one of Delta's weight-and-balance systems delayed more than 1,600 flights nationwide. Delta said 49% of its flights that day were delayed by more than 15 minutes. The disruption was concentrated at its Atlanta hub and also affected flights heading to Atlanta from other airports.

Delta employees used manual workarounds to prepare flights for departure while teams restored the automation. The airline said the system was operating normally again by Sunday afternoon, but the disruption required passenger rebooking, meals, hotel rooms, and refunds for customers who chose not to travel after a cancellation. A failure in one operational dependency had become a network-wide service problem during the Thanksgiving travel period.

For enterprise testing, the operational lesson is clear: validation must address the behavior of interconnected systems when a critical service becomes unavailable, including fallback procedures and recovery. Designing test data around these requirements gives teams a way to exercise failure paths alongside normal business transactions.

Testing must evolve beyond validating applications against representative historical data. It must systematically challenge applications with conditions most likely to expose defects.

Design-Driven Synthetic Data: Engineering Tests to Find Defects

The solution begins by changing the question. Instead of asking, “What production data can we use to test this application?” organizations should ask, “What data do we need to generate to demonstrate that this application behaves correctly under the conditions that matter?”

This is the foundation of design-driven synthetic test data generation. Rather than relying exclusively on existing production records, synthetic data is engineered from explicit test requirements, business rules, data relationships, and application behaviors.

Production Data versus Design-Driven Synthetic Data

PRODUCTION DATA

Recorded business transactions

Recorded transaction	Amount
Payment 001	$125.00
Payment 002	$480.00
Payment 003	$75.00
Payment 004	$920.00
Payment 005	$250.00

Real transactions provide business context.

Rare or unobserved conditions may be absent.

DESIGN-DRIVEN SYNTHETIC DATA

Engineered test conditions

Test condition	Engineered input
Zero and negative values	$0.00 and −$1.00
Boundary values	At and above a defined limit
Invalid input	Missing account number
Selected permutations	Currency × account status
Business-rule exception	Payment from a frozen account

Test requirements determine the data.

Build the conditions needed to challenge behavior.

Illustrative payment examples

Production data represents the business. Engineered data challenges the software.

GenRocket enables quality engineering teams to design and generate datasets for positive and negative testing; boundary and extreme-value conditions; selected combinations of business conditions; complex relational dependencies; exception and failure paths; and deterministic regression testing.

Positive and negative scenarios verify both successful transactions and appropriate rejection or recovery behavior. Boundary testing explores minimums, maximums, empty fields, numerical limits, and unusual data lengths. Combinatorial testing targets interactions among attributes and business conditions that might reveal defects.

Relational testing constructs connected entities and transactions with the referential integrity applications require. Exception testing introduces missing records, conflicting values, duplicates, and invalid inputs. Deterministic generation makes the same dataset reproducible, so a defect can be investigated and a correction verified.

Code coverage measures which portions or branches of an application execute. Test data coverage asks whether the inputs and scenarios adequately exercise business rules, boundaries, dependencies, and failure conditions. High code coverage alone does not establish correct behavior.

The goal is not simply to execute more code during testing. It is to expose more of the conditions under which that code could fail.

Closing the Gap Between AI Development and Software Quality

Traditional testing practices evolved around development cycles in which people wrote code, assembled environments, prepared datasets, and ran validation at predictable intervals. AI-assisted development is changing that cadence.

Testing must operate at a comparable pace. This requires moving beyond manually prepared datasets toward automated, requirements-driven test data engineering integrated directly into software delivery workflows.

With GenRocket, synthetic test data can be generated programmatically and incorporated into automated testing and CI/CD pipelines. Test environments can be populated with precisely defined datasets that exercise requirements, business rules, boundary conditions, and complex relational scenarios.

When a defect is discovered, its triggering dataset can be reproduced deterministically. As applications evolve, test scenarios can be expanded for new functionality, changing business rules, and newly identified failure conditions.

The approach also creates a natural role for AI agents in testing. Development and testing agents can request or orchestrate synthetic datasets to validate specific application behaviors, making purposeful data generation part of the automated development process.

As AI accelerates code generation, design-driven synthetic data can help ensure that software validation accelerates with it.

The Future of Software Quality Is Designed, Not Sampled

The recent wave of enterprise technology outages reminds us how much modern business depends on software reliability. Rapid adoption of AI-generated code raises the stakes, while increasingly sophisticated AI-powered tools can search applications for weaknesses at scale.

The response cannot be to slow innovation. It must be to transform software testing.

Production data will continue to provide valuable insight into real-world business conditions. But historical data alone cannot deliver the breadth of scenarios needed to validate increasingly complex and rapidly evolving applications. Enterprises need the ability to design, generate, and reproduce the precise data conditions necessary to challenge their software before those conditions emerge unexpectedly in production.

AI is transforming how software is created. Design-driven synthetic data can transform how enterprises establish confidence in the software they release.

The outages that opened this article offer a warning about the road ahead. As AI accelerates software development, enterprises can gain efficiency while also accumulating technical debt: defects, fragile dependencies, and untested assumptions that become harder to resolve as systems evolve. If validation fails to keep pace, outages risk becoming more frequent—and more costly.

A new standard for software testing must accompany this new pace of development. Production data provides valuable business context, but effective validation also requires deliberately engineered edge cases, boundary values, invalid inputs, and combinations of conditions that challenge application behavior.

Design-driven synthetic data makes those conditions available for systematic, repeatable testing. Integrated into software delivery, it helps teams expose defects, reproduce failures, and verify corrections before software reaches production.

The coming enterprise software quality crisis is a risk enterprises can address now. By applying the same ambition to validation that they bring to AI-assisted development, organizations can realize AI's efficiencies while finding and correcting defects before they become costly outages.