

## Next-Generation Software Testing: Integrating Artificial Intelligence with Automated Quality Engineering

Rohan Verma

Department of Artificial Intelligence,  
Institute of Intelligent Computing, Bengaluru, India

VOLUME 03 ISSUE 08 (2026)

Published Date: 17 August 2026 // Page no.: - 54-60

---

### ABSTRACT

The increasing complexity of software systems has created a need for testing approaches that are adaptive, automated, context-aware, and capable of operating continuously across the software development lifecycle. Traditional automation improves execution speed but often remains dependent on manually designed test cases, predefined rules, and static validation mechanisms. This research and review paper examines the conceptual integration of Artificial Intelligence (AI) with automated quality engineering, with particular attention to intelligent test generation, context-aware validation, automation frameworks, user-experience assessment, and technology-enabled quality assurance. The study employs a structured qualitative synthesis of the supplied literature, which includes research on augmented reality (AR), QR-code-based applications, mobile platforms, Unity3D, Vuforia, Blender, digital applications, and user-experience evaluation. Although the reviewed studies do not exclusively address AI-based software testing, they collectively demonstrate technological characteristics—automation, interaction awareness, application compatibility, platform integration, and user-centered validation—that are relevant to the design of next-generation testing environments. The analysis positions AI-driven testing as an evolution from script-centric automation toward adaptive quality engineering in which test selection, execution, validation, and feedback can be increasingly supported by intelligent mechanisms. The paper proposes a conceptual AI-enabled automated quality engineering framework comprising application understanding, test intelligence, execution orchestration, result interpretation, and continuous feedback layers. Findings indicate that the effectiveness of such frameworks depends not merely on AI adoption but on integration with application architecture, development workflows, platform constraints, and human quality expertise. The study concludes that AI can significantly strengthen automated testing when deployed as an augmentation mechanism rather than as an isolated replacement for established engineering practices.

**Keywords:** Artificial Intelligence, Automated Software Testing, Quality Engineering, Test Automation, Intelligent Test Generation, Continuous Testing, Software Quality Assurance, Adaptive Testing, User Experience

---

### INTRODUCTION

Software engineering has progressively shifted from periodic testing toward continuous quality assurance integrated with development and deployment activities. Modern applications frequently combine mobile interfaces, cloud services, interactive components, multimedia technologies, and heterogeneous platforms. Such environments increase the number of possible execution states and consequently make exhaustive manual validation impractical. Automation addresses part of this problem by reducing repetitive execution effort, but conventional automation frequently depends on predefined scripts and deterministic workflows. The next stage of software testing therefore requires mechanisms capable of interpreting application behavior, prioritizing relevant validation activities, learning from previous outcomes, and adapting testing activities to changing software conditions.

The importance of intelligent automation becomes clearer when considering applications that integrate multiple technologies. Research on mobile augmented reality applications demonstrates that application behavior can depend on interaction mechanisms, device capabilities, visual markers, and platform characteristics (Davidavičienė et al., 2020; Cheng et al., 2017). Similarly, QR-code-based augmented reality applications introduce dependencies between physical codes, digital content, application logic, and user interaction (Kan et al., 2009; Lin et al., 2021). Such dependencies create testing requirements that cannot always be adequately represented through simple fixed scripts.

The broader objective of next-generation software testing is therefore not simply to execute a larger number of tests. It is to improve the intelligence of the quality process itself. Ramamurthy (2023) characterizes AI-driven test automation as an important direction for modern software quality engineering because intelligent mechanisms can support automation beyond conventional rule-based execution. In this

context, AI can be positioned as a decision-support and optimization layer over automated testing rather than merely another testing tool.

The problem addressed in this paper is the fragmentation between application complexity and conventional automation. An automated test may execute rapidly while still failing to identify meaningful behavioral changes, prioritize critical scenarios, interpret unexpected outputs, or adapt to changes in application interfaces. This challenge becomes particularly significant for mobile and interactive applications, where compatibility and user experience are central quality dimensions. The reviewed literature provides evidence of these technological dependencies through studies involving Android platforms, Unity3D, Vuforia, augmented reality, QR codes, and user-experience assessment (Kaur & Sharma, 2014; Liu et al., 2018).

The primary objective of this research is to develop a theoretically grounded conceptual model for integrating AI with automated quality engineering. The paper specifically examines how application understanding, intelligent test selection, automated execution, result interpretation, and feedback can be connected into a continuous quality process. A secondary objective is to identify limitations and research gaps that must be addressed before AI-based testing can become a dependable component of production-grade quality engineering.

## LITERATURE REVIEW

The supplied literature presents several technological foundations relevant to intelligent automated testing, although its primary focus is augmented reality and interactive application development rather than AI-based software testing. This distinction is important because it prevents an unsupported assumption that AR research directly validates AI testing techniques. Instead, the literature is interpreted as evidence concerning the complexity of applications that next-generation testing systems must validate.

AlNajdi (2022) demonstrates the application of augmented reality and QR codes within educational textbooks, illustrating how software functionality may depend on interactions between physical learning materials and digital content. From a quality-engineering perspective, such systems require validation of recognition, content delivery, interface behavior, and compatibility. These characteristics illustrate why context-sensitive automated testing can be more useful than isolated functional scripts.

Kan et al. (2009) similarly examine QR codes within augmented reality applications. The integration of QR

recognition with AR functionality creates a chain of dependencies: the code must be recognized, the corresponding information must be interpreted, and the application must produce the intended augmented content. A failure at any stage can affect the complete user experience. This structure supports the argument that automated testing should evaluate workflows and dependencies rather than merely individual functions.

Lin et al. (2021) provide further evidence through a QR-code-based approach for differentiating the display of augmented reality content. The study illustrates the importance of conditional content presentation. In a testing context, conditional behavior increases the number of states that require validation because different inputs may trigger different visual or functional outputs. AI-assisted testing can theoretically address this challenge through intelligent input selection and adaptive prioritization.

Cheng et al. (2017) compare marker-based and markerless AR approaches and thereby highlight differences in application behavior associated with recognition mechanisms. Such variation has direct implications for compatibility and reliability testing. A testing framework operating across multiple recognition approaches must account for environmental and functional differences rather than assuming identical execution behavior.

Mobile application considerations are also apparent in Kaur and Sharma (2014), whose review of the Android platform demonstrates the importance of platform-specific characteristics. Liu et al. (2018) further investigate application development using Unity3D and Vuforia, illustrating the integration of development engines and AR libraries. From a quality perspective, technology stacks introduce multiple layers where defects may originate. An intelligent testing framework should therefore be capable of associating failures with relevant application layers.

Davidavičienė et al. (2020) focus on user experience in augmented reality mobile applications. This work is especially relevant because software quality cannot be reduced to technical correctness. An application may execute without crashing and still provide poor usability, confusing interactions, or unsatisfactory user experience. Automated quality engineering therefore requires a broader quality model in which functional, compatibility, performance, and experiential dimensions can be considered together.

Chaurasiya et al. (2019) and Gupta et al. (2020) demonstrate the use of AR and digital technologies in restaurant applications. These examples indicate that interactive software increasingly connects digital information with real-world decision-making. Such applications require validation of content accuracy, interface interaction, and application responsiveness. Koui (2017), through an interactive augmented-reality menu concept, similarly demonstrates the

role of interactive presentation in digital experiences.

The development tools themselves also contribute to the testing landscape. Hartmann (2020) explains Blender as a software environment for creating 3D content, while Li and Tang (2019) investigate Unity3D-based AR technology in education. These technologies reinforce the importance of testing applications composed of dynamically generated visual assets and interactive components. In such environments, conventional textual assertions may be insufficient for comprehensive validation.

Kiryakova (2021) discusses the immersive power of augmented reality, emphasizing the broader transformation of interaction produced by immersive technologies. This perspective strengthens the theoretical argument that future testing must consider the relationship between software behavior and user interaction. Arioputra and Lin (2015), through their mobile AR menu translation application, additionally illustrate how software can perform contextual interpretation between physical information and digital output.

Collectively, the literature reveals three major gaps. First, most studies focus on application functionality or user experience rather than intelligent testing itself. Second, the studies generally investigate individual technologies or application contexts rather than continuous quality-engineering architectures. Third, limited attention is given to learning-based test prioritization, automated failure interpretation, and feedback-driven test adaptation. These gaps establish the conceptual space for AI-driven quality engineering.

Ramamurthy (2023) provides the explicit conceptual bridge between AI and automated software quality engineering. The integration of this perspective with the supplied AR and mobile-application literature suggests that AI should be viewed as an intelligence layer capable of coordinating existing automation mechanisms rather than replacing the underlying testing infrastructure.

## METHODOLOGY

### 3.1 Research Design

This study adopts a qualitative research-and-review methodology based exclusively on the references supplied for the analysis. The methodological approach consists of thematic synthesis, conceptual comparison, and framework construction. Rather than conducting a quantitative meta-analysis, the study identifies recurring technological characteristics across the literature and maps them to the requirements of next-generation automated quality engineering.

The first analytical dimension is application complexity. Studies involving AR, mobile platforms, QR codes, Unity3D, and Vuforia were examined to identify dependencies that create testing challenges. The second dimension is quality scope, which considers functionality, compatibility, interaction, and user experience. The third dimension is automation intelligence, representing the movement from fixed scripts toward adaptive test selection, execution, and interpretation.

### 3.2 Conceptual Architecture of AI-Enabled Quality Engineering

A next-generation framework can be conceptualized as five interconnected layers.

The first layer is the Application Understanding Layer. It collects information about application structure, interfaces, workflows, dependencies, technologies, and historical execution behavior. For example, a mobile AR application may contain QR recognition, camera interaction, content rendering, and external library dependencies. The purpose of this layer is to construct an operational representation of the system that can support intelligent test decisions.

The second layer is the Test Intelligence Layer. This component determines what should be tested, when it should be tested, and which tests should receive higher priority. AI mechanisms can theoretically analyze historical failures, application changes, execution frequency, and risk indicators to prioritize tests. In a QR-based application, for example, modifications to QR recognition functionality should increase the priority of associated recognition and content-display tests.

The third layer is the Automated Execution Layer. Selected tests are executed across appropriate environments, devices, configurations, and application states. The literature on Android, Unity3D, and Vuforia demonstrates why execution environments matter (Kaur & Sharma, 2014; Liu et al., 2018). A test that passes in one environment may require additional verification in another because platform-specific behavior can influence application performance.

The fourth layer is the Intelligent Result Interpretation Layer. Traditional automation commonly categorizes outcomes as pass or fail. AI-enhanced systems can extend this process by clustering similar failures, identifying recurring patterns, and distinguishing potentially meaningful failures from environmental noise. This capability is particularly valuable where application behavior includes complex visual or interaction states.

The fifth layer is the Continuous Feedback Layer. Test outcomes are returned to development and quality processes. Failure patterns can influence subsequent test prioritization, regression selection, and defect investigation. This creates a feedback loop in which testing becomes progressively more context-aware.

### 3.3 Intelligent Test Generation and Selection

Test generation is a central component of the proposed model. Instead of treating all possible test cases equally, an intelligent framework should estimate their potential quality value. A conceptual priority score can be represented as:

$$\text{Test Priority} = \text{Risk} \times \text{Change Impact} \times \text{Failure History} \times \text{Usage Relevance}$$

This formulation is conceptual rather than an experimentally validated mathematical model. Its purpose is to demonstrate that test selection should incorporate multiple sources of evidence.

For example, if an AR application modifies its QR-processing module, tests associated with QR recognition, content retrieval, and visual rendering should receive greater priority than unrelated menu-navigation tests. The approach reduces regression cost while increasing the probability of detecting high-impact failures.

### 3.4 AI-Assisted Failure Analysis

Failure interpretation represents another major opportunity for intelligent quality engineering. Conventional automation can report that an expected element was not found, a response exceeded a threshold, or an interaction failed. However, the underlying reason may remain unclear.

An AI-assisted system could correlate the failure with application changes, previous executions, platform information, and related failures. If multiple tests fail after a change to a shared AR library, the framework could identify a common dependency rather than treating each failure independently. The relevance of dependency-aware analysis is reinforced by applications combining Unity3D, Vuforia, Android, QR recognition, and AR components (Kaur & Sharma, 2014; Liu et al., 2018).

### 3.5 User-Centered Automated Quality

Quality engineering should incorporate user experience because technical correctness does not guarantee effective interaction. Davidavičienė et al. (2020) demonstrate the importance of user-experience evaluation in mobile AR applications. Therefore, the proposed framework includes interaction quality as a complementary dimension.

For example, an AR menu may successfully recognize a marker and display digital content but still provide an inefficient or confusing user interaction. Automated quality engineering should consequently combine functional assertions with interaction-oriented indicators wherever technically feasible.

### 3.6 Continuous Testing in Agile Environments

The proposed framework is particularly compatible with continuous development because testing intelligence can be integrated into repeated development cycles. When a developer modifies application code, the system can identify impacted components, select relevant regression tests, execute them, interpret results, and return prioritized feedback.

Ramamurthy (2023) emphasizes the significance of AI-driven test automation for modern software quality engineering. Extending this perspective through the supplied application literature suggests that intelligent testing becomes particularly valuable when software includes multiple interacting technologies and rapidly changing user-facing functionality.

### 3.7 Methodological Limitations

The methodology is constrained by the nature of the supplied literature. The references do not provide a homogeneous empirical dataset concerning AI testing performance, defect-detection rates, execution time, or model accuracy. Consequently, the framework proposed here is conceptual and analytically derived rather than experimentally validated. Claims regarding the potential benefits of AI should therefore be interpreted as research propositions requiring empirical evaluation.

## RESULTS

The synthesis produced five principal findings concerning the integration of AI with automated quality engineering.

First, application complexity is a primary driver of intelligent testing requirements. The reviewed AR and mobile-application studies demonstrate that contemporary software frequently combines multiple technologies, including QR codes, mobile platforms, Unity3D, Vuforia, and interactive visual components (Kan et al., 2009; Liu et al., 2018). Such combinations create dependencies that increase the number of possible system states. Consequently, static regression scripts may become increasingly difficult to maintain as application complexity grows.

Second, testing must extend beyond functional correctness. The literature on user experience indicates that quality is multidimensional. Davidavičienė et al. (2020) demonstrate that evaluating user experience is important for mobile AR applications. This finding supports a broader quality-engineering model in which functionality, compatibility, interaction, and user experience are treated as related dimensions rather than independent testing activities.

Third, AI provides the greatest conceptual value when used for decision-making and adaptation. The analysis indicates that AI should not simply execute conventional scripts faster. Its more significant role is to determine which tests are most relevant, identify patterns in failures, and adapt testing activities

according to application changes. Ramamurthy (2023) similarly positions AI-driven automation as a significant direction in modern software quality engineering.

Fourth, technology-aware testing is essential for heterogeneous applications. Research involving Android, Unity3D, Vuforia, and AR demonstrates that the execution environment can influence application behavior (Kaur & Sharma, 2014; Liu et al., 2018). Therefore, an AI-enabled framework should incorporate environmental context when prioritizing and interpreting tests. A failure occurring only on a particular platform or configuration should not automatically be generalized to every environment.

Fifth, continuous feedback represents the principal architectural advantage of AI-enabled quality engineering. Traditional testing often produces isolated test reports. The proposed framework instead establishes a feedback cycle connecting application changes, test selection, execution results, failure interpretation, and subsequent testing decisions. This approach transforms testing from a verification stage into a continuous engineering capability.

However, the findings also indicate that AI does not eliminate the need for human expertise. Intelligent models can prioritize and classify information, but ambiguous failures, business-critical decisions, and architectural changes may require human judgment. The practical implication is therefore an augmented quality-engineering model in which AI handles large-scale pattern analysis and repetitive decisions while engineers retain responsibility for high-impact interpretation.

The evidence also reveals an important research gap: the supplied literature does not empirically establish how AI-based testing performs against conventional automation in terms of defect detection, false positives, maintenance effort, or computational cost. Future experimental studies are therefore necessary to quantify the conceptual benefits identified by this review.

## DISCUSSION

The findings suggest that next-generation software testing should be understood as a transition from automation of testing activities toward intelligence within the quality process. Conventional automation remains valuable because it provides reliable mechanisms for repeatable execution. However, its effectiveness can decline when applications become highly dynamic, technologically heterogeneous, or interaction-intensive. The reviewed AR literature provides several examples of such complexity, particularly where QR codes, mobile platforms, recognition mechanisms, and digital content interact (AlNajdi, 2022; Cheng et al., 2017; Lin et al., 2021).

Theoretically, the proposed framework extends the

concept of software quality engineering by treating testing as a closed-loop information system. Application changes generate information; the testing system interprets that information to select relevant validation activities; execution generates new evidence; and the evidence influences subsequent testing decisions. This feedback-oriented model provides a stronger foundation for continuous quality than a sequence of independent test executions.

The role of AI is therefore primarily transformational rather than substitutive. Ramamurthy (2023) supports the relevance of AI-driven automation to contemporary quality engineering, while the supplied AR studies reveal the kinds of complex application environments in which such intelligence may become useful. The combination suggests that AI can act as an orchestration layer connecting risk analysis, test prioritization, execution, and failure interpretation.

There are, however, important trade-offs. Greater intelligence introduces additional system complexity. An AI-based testing framework must itself be maintained, monitored, and validated. Poorly trained or inadequately configured models may prioritize inappropriate tests or incorrectly classify failures. This creates a paradox: the system intended to reduce testing complexity can introduce a new layer of engineering complexity.

Another limitation concerns explainability. Quality engineers must understand why a particular test was selected or why a failure was classified as significant. A system that produces recommendations without interpretable reasoning may reduce trust and make defect investigation more difficult. Consequently, AI-based testing should provide traceable evidence for important decisions, particularly in safety-critical or business-critical applications.

The literature also reveals that user experience should remain part of the quality equation. An application can be technically functional while remaining difficult or ineffective for users. The findings of Davidavičienė et al. (2020) reinforce this concern. For interactive AR systems, therefore, next-generation testing should increasingly combine functional validation with interaction and experiential assessment.

From a practical perspective, organizations should avoid replacing established automation systems immediately with AI-based solutions. A more sustainable strategy is incremental integration. Existing regression suites can serve as the execution foundation, while AI components are introduced for test prioritization, failure clustering, impact analysis, and maintenance assistance. This minimizes disruption while allowing organizations to evaluate measurable benefits.

The principal empirical limitation of this review is the lack of direct comparative experiments. The supplied references predominantly investigate AR, mobile applications, digital interfaces, and related technologies rather than AI testing

benchmarks. Therefore, the proposed framework should be regarded as a research model rather than evidence of guaranteed performance. Future studies should compare AI-enhanced and conventional automation using defect-detection effectiveness, regression execution time, test-maintenance effort, false-positive rates, and coverage as evaluation criteria.

Overall, the discussion indicates that AI-enabled quality engineering is most promising when it complements, rather than disregards, established software-testing principles. The central research opportunity lies in developing explainable, adaptive, technology-aware testing systems that can continuously learn from software changes without sacrificing engineering control.

## CONCLUSION

Next-generation software testing requires a shift from static test execution toward adaptive and intelligence-supported quality engineering. This paper developed a conceptual framework for integrating AI with automated testing through five interconnected capabilities: application understanding, test intelligence, automated execution, intelligent result interpretation, and continuous feedback.

The literature synthesis demonstrates that increasingly complex applications—particularly mobile and augmented-reality systems—contain multiple technological dependencies that create challenges for conventional automation. QR-code processing, AR recognition, mobile platforms, Unity3D, Vuforia, and interactive interfaces illustrate why testing must become more context-sensitive and comprehensive (Kan et al., 2009; Liu et al., 2018). The literature also demonstrates that quality must include user experience and compatibility rather than focusing exclusively on functional correctness (Davidavičienė et al., 2020).

The principal contribution of this study is the conceptual positioning of AI as an intelligence layer within automated quality engineering. Instead of merely increasing test execution speed, AI can potentially improve the selection, prioritization, interpretation, and continuous adaptation of testing activities. Ramamurthy (2023) reinforces the significance of AI-driven automation in modern quality engineering and provides a direct foundation for this conceptual integration.

Nevertheless, AI adoption introduces challenges involving explainability, model reliability, maintenance, data requirements, and human oversight. The supplied literature also does not provide sufficient empirical evidence to quantify the performance advantages of AI-based testing. Future research should therefore implement the proposed framework in real software

projects and compare it systematically with conventional automation.

The future of software testing is consequently unlikely to be defined by complete replacement of human testers or traditional automation. A more credible direction is collaborative quality engineering in which AI manages large-scale analytical and repetitive tasks while software engineers provide domain knowledge, architectural reasoning, and final quality judgment. Such integration can transform automated testing from a reactive verification mechanism into a continuous, adaptive, and strategically informed quality-engineering capability.

## REFERENCES

1. AlNajdi, S. M. (2022). The effectiveness of using augmented reality(AR) to enhance student performance: using quick response(QR) codes in student textbooks in the Saudi education system. *Educational Technology Research and Development*,70, 1105–1124.
2. Arioputra, D., & Lin, C. H. (2015). Mobile augmented reality as aChinese menu translator. In*IEEE International conferenceon consumer electronics Taiwan*.
3. Baruah, B. (2022, May 27). Augmented reality and QR code–What you need to know. Be aconstac.
4. Chaurasiya, A., Mhatre, S., Chaudhari, R., & Pawar, P. (2019).Smart restaurant menu card by using augmented reality.JETIR,6.
5. Cheng, J., Chen, K., & Chen, W. (2017). Comparison of marker-based AR and marker less AR: A case study on indoord ecoration system.
6. Davidavičienė, V., Raudeliūnienė, J.,&V iršilaitė, R. (2020).Evaluation of user experience in augmented reality mobile applications. *Journal of Business Economics and Management*,22, 467–481.
7. Gupta V., Gaddam N., Narang L., & Gite, Y. (2020). Digitalrestaurant . *International Research Journal of Engineeringand Technology (IRJET)*,07, 5340–5344.
8. Hartmann, T. (2020, May 21).What Is Blender (Software)?–SimplyExplained.All3DP.
9. Kan, T. W., Teng, C. H., & Chou, W. S. (2009). Applying QR code in augmented reality applications. In *Proceedings of the 8thinternational conference on virtual reality continuum and its applications in industry - VRCAI'09*.
10. Kaur, P., & Sharma, S. (2014). Google Android a mobile platform: Areview. *Recent Advances in Engineering*

and Computational Sciences (RAECS), 1–5,

11. Kiryakova, G. (2021). The immersive power of augmented reality. *Human 4.0–From Biology to Cybernetic*.
12. Kouï, E. (2017). *Avant l'appétit: An augmented reality interactive menu that elevates the gourmet food experience*.
13. Li, C., & Tang, B. (2019, February). Research on The Application of AR Technology Based on Unity3D in Education. *Journal of Physics: Conference Series*, 1168, 032045.
14. Lin, P. Y., Wu, W. C., & Yang, J. H. (2021). A QR code-based approach to differentiating the display of augmented reality content. *Applied Sciences*, 11, 11801.
15. Liu, X., Sohn, Y. H., & Park, D. W. (2018). Application development with augmented reality technique using Unity 3D and Vuforia. *International Journal of Applied Engineering Research*, 13, 15068–15071.
16. Ramamurthy, K. (2023). AI-Driven Test Automation Frameworks for the Modern Software Quality Engineering. *International Journal of Emerging Trends in Computer Science and Information Technology*, 4(4), 257-269.
17. Philip, P. G. (2025). Explainable Artificial Intelligence (XAI) for Project Governance: Improving Transparency and Stakeholder Trust in Automated Project Decision. *Journal of Project Management Studies*, 2(1), 37–55. <https://doi.org/10.58425/jpms.v2i1.570>