AI-Based Automated Defect Detection and Test Prioritization in Software Quality Engineering

Azizbek Karimov
Department of Artificial Intelligence, Institute of Digital Technologies, Tashkent, Uzbekistan

Abstract.
The increasing complexity of software systems has created a need for quality engineering approaches that can identify defects efficiently and allocate testing resources according to risk. AI-based automation offers a mechanism for transforming conventional test execution from a largely rule-driven process into an adaptive decision-making activity. This paper develops a conceptual framework for AI-based automated defect detection and test prioritization by synthesizing principles from the provided literature on graphical authentication, password usability, user behavior, and security-oriented evaluation. Although the cited studies primarily address authentication rather than software testing, they collectively demonstrate the importance of usability, behavioral variability, attack resistance, user choice, and systematic evaluation in automated security-sensitive systems. These principles are mapped to defect detection and test prioritization through a proposed pipeline consisting of test-data acquisition, defect-feature extraction, risk scoring, intelligent classification, prioritization, execution, and feedback-based model refinement. The analysis argues that AI-driven test automation can improve the allocation of testing effort by prioritizing cases associated with higher predicted defect probability or security impact. The framework also incorporates the broader perspective of AI-driven test automation frameworks for modern software quality engineering (Ramamurthy, 2023). The paper concludes that AI-based prioritization should not replace conventional testing controls; rather, it should function as an adaptive decision-support layer whose effectiveness depends on representative data, explainability, continuous validation, and careful management of false-positive and false-negative outcomes.

Keywords: Artificial Intelligence, Automated Defect Detection, Test Prioritization, Software Quality Engineering, Machine Learning, Risk-Based Testing, Test Automation, Security Testing, Defect Prediction

INTRODUCTION

Software quality engineering has progressively moved beyond the traditional objective of detecting failures after implementation. Contemporary quality practices emphasize continuous identification of defects, risk assessment, automated validation, and early intervention throughout the software development lifecycle. As software applications become more interconnected and security-sensitive, exhaustive execution of every available test case becomes increasingly expensive. The central problem is therefore not simply whether a test

can be automated, but whether available testing resources can be directed toward the tests most likely to reveal important defects.

Artificial intelligence provides a potential solution by allowing testing systems to learn relationships between historical execution data, software characteristics, test outcomes, and defect patterns. An AI-enabled testing system can assign a predicted priority to individual test cases and dynamically determine which tests should be executed first. This changes testing from a static sequence into a risk-sensitive process. AI-driven test automation frameworks similarly seek to integrate intelligent decision-making into modern quality engineering rather than treating automation merely as repetitive execution (Ramamurthy, 2023).

The relevance of this approach can also be understood through the provided authentication literature. Research on graphical passwords demonstrates that security systems must be evaluated not only according to whether they function, but also according to usability, user behavior, resistance to attacks, and variations in user choice. For example, graphical authentication research has examined usability among children, alternative password structures, shoulder-surfing resistance, and user-selection behavior (Assal et al., 2018; Davis et al., 2004; Gao et al., 2010). These concerns are directly relevant to software quality engineering because a technically functional system can nevertheless contain defects that emerge only under particular behavioral, usability, or security conditions.

The objective of this paper is to propose a conceptual AI-based framework for automated defect detection and intelligent test prioritization. The framework focuses on how test information can be transformed into predictive defect-risk indicators and subsequently used to establish an execution order. The paper also examines the limitations of such an approach, particularly where training data are incomplete, system behavior is highly variable, or AI recommendations are insufficiently explainable.

LITERATURE REVIEW

The provided literature establishes several theoretical principles that can support an AI-oriented software quality framework. Biddle et al. (2012) provide a broad examination of graphical passwords and emphasize the historical development of graphical authentication mechanisms. Their work is relevant to quality engineering because it illustrates how security mechanisms evolve through repeated evaluation, comparison, and identification of weaknesses. For AI-based defect detection, this suggests that historical testing outcomes can represent an important learning resource rather than merely an archive of previous failures.

Usability is another important dimension. Assal et al. (2018) explored graphical password authentication for children, highlighting the importance of considering user characteristics when evaluating authentication systems. Similarly, De-Angeli et al. (2005) investigated whether graphical representations provide meaningful advantages for authentication. These studies imply that defects cannot always be interpreted exclusively through system-level functional outcomes. A system may technically execute correctly while producing usability-related failures for particular user groups. Consequently, an intelligent defect-detection

framework should incorporate contextual characteristics rather than relying only on binary pass/fail results.

The literature also identifies the importance of user behavior. Davis et al. (2004) examined user choice in graphical password schemes, while Gao et al. (2013) presented a survey of graphical password use in security. Such research demonstrates that user interaction introduces variability into system behavior. From a testing perspective, this variability can be transformed into test-prioritization information. Test cases involving historically unstable user interaction patterns, unusual input sequences, or security-sensitive operations may deserve higher priority than consistently successful and low-risk cases.

Security resilience is another recurring theme. Gao et al. (2010) examined graphical password schemes resistant to shoulder-surfing, while Azad et al. (2017) discussed a secure graphical password approach for smart devices. These studies reinforce the distinction between ordinary functional correctness and resistance to adversarial or abnormal conditions. An AI-based testing framework should therefore treat security-sensitive test cases as potentially high-impact even when their historical failure frequency is low.

Danish et al. (2016) investigated an alignment-based graphical password authentication system, demonstrating the continuing development of alternative authentication mechanisms. Brosto and Sasse (2000) evaluated the usability of Passfaces through a field trial, indicating the value of empirical assessment under realistic conditions. Elftmann (2006) considered alternatives to password-based authentication, while Gayathiri (2012) examined the transition of text-password approaches across generations. Collectively, these works demonstrate that evaluation criteria change as technologies and attack surfaces evolve.

A significant gap becomes apparent when these studies are considered in relation to the present research problem. The provided literature focuses primarily on authentication rather than AI-driven software defect prediction. It does not directly establish a validated machine-learning model for test prioritization. Therefore, the theoretical contribution of this paper is not to claim that these studies empirically prove AI-based testing effectiveness. Instead, their findings are used to derive quality dimensions—security, usability, behavioral variability, historical failure, and contextual risk—that can inform an AI-based testing architecture.

METHODOLOGY

3.1 Research Design

This study adopts a conceptual research and review methodology. Because the provided references do not contain a common experimental dataset for software defect prediction, the paper does not fabricate empirical performance measurements. Instead, it synthesizes the quality and security principles present in the supplied literature and maps them to an AI-based test automation architecture.

The proposed architecture consists of six logical stages: test-data acquisition, feature extraction, defect-risk prediction, test prioritization, automated execution, and feedback-based

learning. The architecture is designed to operate as a decision-support mechanism within software quality engineering.

3.2 Test-Data Acquisition

The first stage collects information generated during software testing. Relevant information can include historical test outcomes, execution duration, test-case frequency, changed software components, previous defect associations, failure categories, input characteristics, and security sensitivity.

For example, assume an authentication module contains 500 automated tests. A conventional regression strategy might execute all tests according to a predetermined sequence. An AI-based system instead records historical evidence and identifies which tests have demonstrated a greater probability of exposing defects following changes to authentication logic.

The underlying principle is consistent with the literature's emphasis on empirical evaluation. Brosto and Sasse (2000), for instance, demonstrate the value of field-based assessment rather than relying exclusively on theoretical assumptions. In software testing, execution history similarly provides evidence from which prioritization decisions can be derived.

3.3 Feature Extraction

Raw test information must be converted into features that an AI model can process. Potential features include recent failure frequency, code-change proximity, execution history, defect severity, test age, input complexity, security relevance, and interaction variability.

Security-sensitive features are particularly important for applications involving authentication. Research examining shoulder-surfing resistance demonstrates that some system conditions have consequences beyond ordinary functional failure (Gao et al., 2010). Accordingly, a test-prioritization model should incorporate impact as well as probability.

A conceptual risk score can be represented as:

$$\text{Risk Score} = \alpha P(D) + \beta S + \gamma C + \delta H$$

where $P(D)$ represents predicted defect probability, S represents potential severity, C represents change-related risk, and H represents historical failure evidence. The coefficients α, β, γ, and δ determine the relative importance of each factor.

This formulation avoids the assumption that the test with the highest predicted failure probability is automatically the most important. A low-probability security test may deserve greater priority than a high-probability test associated with a minor defect.

3.4 AI-Based Defect Detection

The defect-detection component can employ supervised classification when labeled historical test outcomes are available. Test executions can be classified into categories such as likely successful, potentially defective, security-sensitive failure, or uncertain outcome.

A model could learn patterns linking software changes and test characteristics to previous defects. When a new software build is generated, the model estimates the likelihood that each test will reveal an issue.

However, classification confidence should not be treated as absolute truth. False positives may increase unnecessary testing, while false negatives may cause important tests to be delayed. Therefore, the framework should preserve deterministic testing controls alongside AI recommendations.

The conceptual relationship is especially important in security applications. The authentication literature demonstrates that user choices and attack conditions can substantially alter system behavior (Davis et al., 2004; Gao et al., 2010). AI models must consequently be trained on sufficiently diverse conditions if they are expected to identify defects associated with unusual behavior.

3.5 Intelligent Test Prioritization

After generating defect-risk scores, the system orders test cases according to expected value. High-risk tests are executed first, followed by medium- and lower-risk cases.

Consider three hypothetical tests:

Test	Predicted Defect Probability	Severity	Priority
Authentication input validation	0.72	High	Very High
Password recovery interface	0.54	Medium	High
Visual theme validation	0.80	Low	Medium

A probability-only approach would execute the visual theme test first. A risk-aware model instead prioritizes authentication input validation because its potential impact is substantially greater. This distinction is essential for practical quality engineering.

The principle also corresponds to the broader direction of AI-driven test automation, in which intelligent mechanisms can assist in determining what should be tested and when rather than simply automating predetermined test scripts (Ramamurthy, 2023).

3.6 Continuous Feedback and Model Updating

Following execution, actual test outcomes are returned to the AI layer. Correct predictions reinforce existing patterns, while incorrect predictions become new training information.

This creates a feedback cycle:

Software Change → Feature Extraction → Defect Prediction → Risk Scoring → Test Prioritization → Execution → Result Analysis → Model Updating

The feedback mechanism is particularly important because software systems change continuously. A model trained on historical behavior can become unreliable when application architecture, user behavior, or security mechanisms change substantially.

3.7 Theoretical Model

The proposed theoretical model is based on four interacting dimensions: probability, impact, behavioral variability, and historical evidence.

Probability estimates how likely a test is to reveal a defect. Impact measures the consequences if the associated component fails. Behavioral variability accounts for differences in user interaction and abnormal input. Historical evidence captures previous failure patterns.

This model is conceptually supported by the provided literature. User choice has been identified as a significant consideration in graphical password systems (Davis et al., 2004), while usability research demonstrates that authentication experiences can differ according to user context (Assal et al., 2018; De-Angeli et al., 2005). Security studies additionally demonstrate that specific attack conditions require specialized evaluation (Gao et al., 2010).

RESULTS

The conceptual analysis produces four principal findings. First, AI-based defect detection is most valuable when it incorporates multiple dimensions of testing evidence rather than treating previous pass/fail results as the sole predictive signal. Historical failures are useful, but they cannot fully represent usability, behavioral, security, and contextual risks. The provided authentication literature repeatedly illustrates this multidimensional nature of system evaluation (Assal et al., 2018; Biddle et al., 2012; De-Angeli et al., 2005).

Second, test prioritization should be risk-sensitive rather than probability-sensitive alone. A test associated with a security-critical component may deserve priority even when its predicted failure probability is relatively low. Research on shoulder-surfing resistance demonstrates why security conditions may require explicit attention independent of ordinary functional behavior (Gao et al., 2010). Similarly, secure graphical password mechanisms demonstrate the need to evaluate protection properties alongside basic functionality (Azad et al., 2017).

Third, behavioral variability represents a valuable feature for intelligent testing. Studies concerning user choice and graphical authentication show that users do not necessarily interact with security mechanisms in identical ways (Davis et al., 2004; Gao et al., 2013). Consequently, a test-prioritization model that relies only on deterministic execution history may underrepresent defects associated with unusual input or user behavior. Incorporating interaction patterns can improve the conceptual coverage of AI-based testing.

Fourth, AI should operate as an adaptive layer rather than an independent replacement for established quality controls. AI predictions can improve execution ordering, but incorrect predictions remain possible. A high-confidence prediction does not guarantee the absence of defects. This supports a hybrid model in which deterministic regression testing, security validation, and expert review remain available while AI dynamically allocates testing effort.

Overall, the proposed framework indicates that the greatest potential benefit of AI-based prioritization is resource allocation. Instead of reducing testing to fewer tests, the approach attempts to execute the most informative and consequential tests earlier. This aligns with the broader objective of AI-driven test automation frameworks in modern software quality engineering (Ramamurthy, 2023).

DISCUSSION

The theoretical significance of the proposed framework lies in connecting adaptive test prioritization with multidimensional quality evaluation. Traditional automation can execute thousands of cases quickly, but speed alone does not establish whether the most consequential defects will be detected early. AI introduces the possibility of assigning different levels of importance to tests according to historical evidence, predicted failure probability, software changes, and potential impact.

The authentication literature provides an important conceptual foundation for this argument. Graphical password research demonstrates that quality cannot be reduced to a single technical property. Usability, security, user selection, and resistance to specific attacks all contribute to system quality (Biddle et al., 2012; De-Angeli et al., 2005; Gao et al., 2010). Translating this principle to software testing means that AI models should avoid optimizing solely for defect frequency. They should consider the nature and consequences of the defect.

A major practical implication is improved regression-test scheduling. In a large software project, executing every test after every modification may be computationally expensive. A prioritization engine can place tests related to recently modified or historically unstable components at the beginning of the execution queue. If a critical defect is detected early, developers can potentially receive feedback before lower-value tests consume additional resources.

However, several trade-offs must be recognized. An AI model trained on historical defects can reproduce historical testing biases. If a particular component has rarely been tested, the model may interpret its low failure history as evidence of low risk even when the actual uncertainty is high. This creates an important distinction between low observed failure and low expected risk.

False negatives are another serious limitation. If the model incorrectly assigns low priority to a security-sensitive test, the resulting delay may be more consequential than an ordinary prioritization error. Security research on authentication systems reinforces the need to evaluate specialized threat conditions rather than relying exclusively on common execution paths (Gao et al., 2010; Azad et al., 2017).

Explainability is therefore essential. Test engineers should be able to determine why a particular test received a high or low priority. Features such as recent failures, code changes, security classification, and historical instability can provide interpretable reasons for prioritization. An opaque model may achieve predictive performance but still create operational resistance because quality engineers cannot confidently validate its decisions.

The framework also has limitations arising from the evidence base. The supplied references primarily concern graphical authentication and password security, not empirical AI defect-detection datasets. Therefore, the proposed model should be regarded as a theoretically grounded framework rather than an experimentally validated performance model. Claims concerning accuracy, execution-time reduction, or defect-detection improvement require future empirical evaluation using real software repositories and labeled testing datasets.

Future research should compare several prioritization strategies, including traditional regression ordering, historical failure-based ordering, risk-based ordering, and machine-learning-based prioritization. Evaluation should include defect-detection rate, time to first critical defect, testing cost, false-negative rate, and explainability. Such experiments would establish whether the conceptual advantages identified here translate into measurable quality improvements.

CONCLUSION

AI-based automated defect detection and test prioritization represents a significant direction for software quality engineering because it changes automation from simple test execution toward adaptive decision-making. The proposed framework integrates test history, software-change information, defect probability, security relevance, behavioral variability, and potential impact into a unified prioritization process.

The literature supplied for this study, although primarily focused on graphical authentication, provides useful theoretical principles for this framework. Research on usability, user choice, security resistance, authentication alternatives, and empirical evaluation demonstrates that software quality is multidimensional and context-dependent (Assal et al., 2018; Biddle et al., 2012; Davis et al., 2004; Gao et al., 2010). These principles support the argument that intelligent test prioritization should consider more than historical failure frequency.

The primary contribution of the proposed approach is therefore conceptual: AI should be positioned as an adaptive risk-assessment layer that helps determine which tests deserve earlier execution. It should complement rather than eliminate deterministic regression testing and expert quality assurance. The effectiveness of such a system depends on representative data, continuous feedback, appropriate risk modeling, explainability, and explicit protection against false-negative predictions.

Future implementation should validate the framework experimentally using real software projects and sufficiently diverse defect datasets. Comparative evaluation of predictive accuracy, critical-defect detection, execution efficiency, and model explainability would provide the empirical evidence required to transform the proposed conceptual framework into a validated software quality engineering methodology.

REFERENCES

1. Assal, H., Imran, A., & Chiasson, S. (2018). An exploration ofgraphical password authentication for children.InternationalJournal of Child-Computer Interaction,18,37–46.

2. Azad, S., Rahman, M., Ranak, M. N., Ruhee, B. K., Nisa, N. N.,Kabir, N.,:::, & Zain, J. M. (2017). VAP code: A securegraphical password for smart devices. Computers &Electrical Engineering, 59, 99–109.

3. Biddle, R., Chiasson, S., & Van Oorschot, P. (2012). Graphicalpasswords: Learning from the first twelve years.Journal ofACM Computing Surveys (CSUR),44,19–41.

4. Brosto, S., & Sasse, M. A. (2000). Are Passfaces more usable thanpasswords? InA field trial investigation people and computersXIV–Usability or else!(pp. 405–424). Springer.

5. Danish, A., Sharma, L., Varshney, H., & Khan, A. M. (2016, March).Alignment-based graphical password authentication system. In2016 3rd International conference on computing for sustainableglobal development (INDIACom)(pp. 2950–2954). IEEE.

6. Davis, D., Monrose, F., & Reiter, M. (2004).On user choice ingraphical password schemes. InProceedings of the 13thUSENIX security symposium(pp. 151–164).

7. De-Angeli, A., Coventry, L., Johnson, G., & Renaud, K. (2005). Is apicture really worth a thousand words? Exploring thefeasibility of graphical authentication systems.InternationalJournal of Human-Computer Studies,63, 128–152.

8. Elftmann, P. (2006). Secure alternatives to password-basedauthentication mechanisms (Diploma Thesis, RWTH AachenUniversity, Aachen, Germany),https://www1.cs.fau.de/lepool/thesis/diplomarbeit-2006-elftmann.pdf. Accessed 20April 2017.

9. Gao, H., Ren, Z., Chang, X., Liu, X., & Aickelin, U. (2010). A newgraphical password scheme resistant to shoulder-surfing. InInternational conference on Cyberworlds (CW)(p. 194).

10. Gao, H. C., Wei, J., Ye, F., & Ma, L. C. (2013). A survey on the useof graphical passwords in security. Journal of Software, 8,1678–1698.http://www.jsoftware.us/vol8/jsw0807-16.pdf

11. Gayathiri, C. (2012). Text password survey: Transition fromfirst generation to second generation.http://blogs.ubc.ca/computersecurity/les/2012/04/Text-Password-SurveyGAYA.pdf. Accessed 20 April 2022.

12. 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

13. 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.