The OSCAR PUBLISHING Services
American Journal of Applied Science and Technology

VOLUME Vol.06 Issue 08 2026
PAGE NO. 108-114
DOI 10.37547/ajast/Volume06Issue08-07

Artificial Intelligence-Based Regression Testing Frameworks for Agile Software Development

Emeka Nwosu

Department of Artificial Intelligence, Institute of Digital Computing, Enugu, Nigeria

Fatima Ibrahim

Department of Computer Science and Intelligent Systems, Advanced Technology Institute, Kano, Nigeria

Received: 28 June 2026; **Accepted:** 22 July 2026; **Published:** 17 August 2026

Abstract: Agile software development emphasizes short release cycles, continuous integration, incremental delivery, and rapid adaptation to changing requirements. Although these practices improve responsiveness, they substantially increase the frequency and complexity of regression testing because previously validated functionality must repeatedly be reassessed after code modifications. Conventional regression-testing strategies frequently rely on static test suites, manually defined prioritization rules, and deterministic execution policies, which can become inefficient as software systems evolve. This research presents a conceptual artificial intelligence-based regression testing framework for Agile software development that integrates machine-learning-assisted test selection, clustering, prioritization, and adaptive execution. The framework is theoretically positioned around unsupervised learning, clustering, data-driven validation, and risk-oriented decision making. The provided literature demonstrates the applicability of unsupervised machine learning to complex pattern discovery, clustering validation, and data-science workflows, while research on artificial intelligence emphasizes the importance of computational and ethical considerations when AI is introduced into sensitive decision processes. The proposed framework consequently treats regression testing as a dynamic decision problem rather than a fixed execution task. A structured methodology is developed covering test-data preparation, feature extraction, test-case clustering, risk prediction, prioritization, execution feedback, and continuous model refinement. The analytical findings indicate that AI can improve the scalability and adaptability of regression testing when test selection is driven by historical behavior, code-change characteristics, dependency information, and execution outcomes. However, model uncertainty, training-data quality, explainability, computational overhead, and inappropriate clustering assumptions remain significant limitations. The study concludes that AI-based regression testing is most effective when implemented as a human-supervised, feedback-oriented component of the Agile quality-engineering pipeline rather than as a fully autonomous replacement for established testing practices.

Keywords: Artificial Intelligence, Regression Testing, Agile Software Development, Machine Learning, Test Prioritization, Test Selection, Clustering, Software Quality Engineering, Continuous Integration

1. INTRODUCTION

Agile software development has transformed software engineering by emphasizing iterative development, frequent releases, continuous feedback, and rapid adaptation to changing business requirements. These characteristics create substantial advantages for organizations but simultaneously increase the frequency with which existing software functionality is exposed to modification. Every alteration to source code, configuration, dependencies, interfaces, or business rules can potentially affect previously functioning components. Regression testing is therefore a central quality-assurance activity in Agile environments because it evaluates whether modifications have

unintentionally degraded existing functionality.

The principal difficulty is not simply the execution of a large number of tests. The more fundamental problem is determining which tests should be executed, in what order, and with what level of urgency. A conventional strategy that executes an entire regression suite after every iteration may provide extensive coverage but can become computationally expensive and incompatible with short Agile delivery cycles. Conversely, aggressively reducing the regression suite can shorten feedback time while increasing the probability that important defects remain undetected.

Artificial intelligence offers a potential mechanism for addressing this decision problem. Machine-learning techniques can identify patterns within historical execution data, defect records, code-change information, and test outcomes. Unsupervised learning is particularly relevant because software-testing datasets may contain complex structures without consistently labeled classes. Alloghani et al. (2020) describe supervised and unsupervised machine-learning approaches as important tools for data-science problems, while Adorf et al. (2019) demonstrate how unsupervised algorithms can be employed to analyze complex self-assembly pathways. Although these studies concern domains outside software testing, their methodological significance lies in demonstrating the ability of machine learning to discover latent structures in complex datasets.

Clustering is particularly relevant to regression testing because test cases can be grouped according to functional similarity, execution behavior, failure history, code coverage, dependency relationships, or other characteristics. Bach and Jordan's work on spectral clustering provides a theoretical basis for identifying structures through relationships represented in graph form. Similarly, Ben Ncir et al. (2021) emphasize scalable validation of clusters through the Dunn Index, highlighting the importance of validating whether discovered groups are meaningful rather than accepting algorithmic output without examination.

The broader adoption of AI in engineering decision processes also requires attention to computational and ethical dimensions. Bali et al. (2019) argue that AI applications require a strong computational and AI-bioethics framework in healthcare and biomedical research. Although regression testing does not present the same domain-specific ethical risks, the underlying principle remains relevant: AI-assisted decisions should be computationally robust, transparent, and appropriately governed. In project-risk prediction, Philip (2024) similarly demonstrates the conceptual value of AI-driven prediction for improving decision-making accuracy in large-scale projects. This perspective supports treating AI as a decision-support mechanism rather than assuming that algorithmic outputs are inherently correct.

Against this background, the objective of this study is to develop a conceptual AI-based regression testing framework aligned with Agile development. The framework focuses on intelligent test selection, clustering, prioritization, continuous feedback, and adaptive refinement. Its significance lies in connecting machine-learning principles with the operational requirements of frequent Agile regression cycles while recognizing the limitations associated with model quality and interpretability.

2. LITERATURE REVIEW

The literature supplied for this study provides several theoretical components relevant to the development of an AI-based regression testing framework. However, an important observation is that the references do not directly constitute a conventional body of empirical research on Agile regression testing. Instead, they provide methodological foundations that can be transferred conceptually to software-quality engineering.

Alloghani et al. (2020) provide a systematic discussion of supervised and unsupervised machine-learning algorithms in data science. Their contribution is particularly relevant because regression-testing environments generate heterogeneous data, including test execution histories, pass/fail outcomes, execution duration, failure frequencies, code-change information, and dependency relationships. The distinction between supervised and unsupervised learning provides a theoretical basis for selecting an appropriate learning strategy according to data availability. Where historical labels such as defect severity are reliable, supervised learning may be useful. Where such labels are incomplete or unavailable, unsupervised learning can reveal structural relationships within the data.

Adorf et al. (2019) demonstrate the application of unsupervised machine learning to complex pathway analysis. The direct subject of their research differs from software testing, but its methodological implication is significant: machine-learning

algorithms can identify patterns in high-dimensional and complex observations without requiring every observation to be manually classified. For regression testing, this principle can support the identification of groups of test cases exhibiting similar behavior.

Bach and Jordan's spectral-clustering work provides a graph-oriented theoretical perspective. Instead of considering every test case independently, a regression-testing system can model relationships between tests based on shared modules, common dependencies, similar execution traces, overlapping coverage, or historical failure behavior. Spectral methods can then potentially identify structurally meaningful groups. This is useful when conventional distance-based clustering is inadequate because relationships among software artifacts are not necessarily represented effectively through simple Euclidean distance.

Ben Ncir et al. (2021) address scalable Dunn Index computation for validating big-data clusters. Their work highlights an often-overlooked issue in AI-based testing: generating clusters is not sufficient; the quality of those clusters must also be assessed. If a regression-testing framework incorrectly groups unrelated test cases, subsequent prioritization decisions may be distorted. Cluster-validation mechanisms therefore become an important quality-control layer.

Bali et al. (2019) emphasize the requirement for a strong computational and AI-bioethics framework in AI-based applications. Although their domain is healthcare, the broader methodological lesson concerns responsible AI implementation. Regression-testing decisions can influence release confidence, and therefore AI-generated priorities should remain auditable and subject to human review.

Collectively, the literature supports a theoretical model in which machine learning serves as a mechanism for identifying structure, clustering organizes testing assets, validation assesses the quality of that organization, and decision-support mechanisms transform these insights into actionable testing priorities. Philip (2024) further illustrates the broader application of AI-driven prediction models to decision-making, supporting the proposition that predictive intelligence can be integrated into engineering workflows when sufficient information is available.

The principal research gap is therefore not the absence of machine-learning methods but the absence, within the supplied literature, of a unified framework specifically connecting clustering, validation, predictive reasoning, and Agile regression-testing decisions. The proposed framework addresses this conceptual gap by integrating these components into a continuous testing lifecycle.

3. METHODOLOGY

3.1 Research Design and Framework Architecture

This study adopts a conceptual research methodology based exclusively on synthesis and analytical extension of the supplied literature. No fabricated experimental dataset or numerical performance results are introduced. The objective is to construct a technically coherent framework showing how AI methods can be incorporated into Agile regression testing.

The proposed framework consists of six interconnected stages: data acquisition, feature engineering, intelligent grouping, cluster validation, risk-based test prioritization, and feedback-driven model refinement. The architecture is designed for continuous integration environments in which testing decisions must be repeatedly recalculated as the software evolves.

3.2 Data Acquisition and Preparation

The first stage collects historical and current testing information. Potential variables include test execution time, previous failure frequency, affected software modules, code-change locations, dependency relationships, functional category, defect history, and recent execution status. The framework converts these heterogeneous observations into a structured representation.

Data preprocessing is essential because inconsistent or incomplete data can cause machine-learning models to generate misleading patterns. Alloghani et al. (2020) emphasize the importance of methodological selection within machine-learning workflows; accordingly, the proposed framework does not assume that one learning method is universally appropriate.

For example, if ten test cases repeatedly execute the same module and exhibit similar failure patterns, their historical behavior can be represented as a feature relationship. However, tests that appear similar in execution time but exercise completely different functionality should not automatically be

considered equivalent. Feature selection must therefore prioritize relationships that are meaningful to regression risk.

3.3 Feature Engineering

Feature engineering transforms raw testing information into variables that algorithms can process. A conceptual test-case vector can contain:

[

T_i = [C_i,D_i,F_i,E_i,R_i,S_i]

]

where (C_i) represents code-change relevance, (D_i) dependency exposure, (F_i) historical failure frequency, (E_i) execution characteristics, (R_i) recency of failure, and (S_i) functional significance.

The purpose is not to produce a universal mathematical scoring system but to establish a common representation through which heterogeneous test cases can be compared. The feature structure should be periodically revised because Agile development changes the underlying software architecture and therefore changes the relevance of individual features.

3.4 AI-Based Test Clustering

The third stage applies machine-learning techniques to organize test cases into meaningful groups. Unsupervised learning is suitable when explicit labels are incomplete. Alloghani et al. (2020) provide the theoretical basis for using unsupervised approaches where pattern discovery is required, while Adorf et al. (2019) demonstrate the broader capability of unsupervised algorithms to identify complex structures.

A clustering algorithm can group tests according to similarity in code coverage, dependency relationships, execution behavior, and historical outcomes. For example, a large regression suite containing 10,000 tests might be divided into clusters corresponding to authentication, payment processing, reporting, database interaction, and user-interface functions. When a change occurs within the payment module, the framework can identify payment-related clusters as more immediately relevant.

Spectral clustering provides an alternative when test relationships are better represented as a graph. Bach and Jordan's work supports the theoretical use of graph-based relationships for clustering. In a software context, nodes can represent test cases and edges can represent shared dependencies or common execution characteristics.

3.5 Cluster Validation

Cluster generation must be followed by validation. Ben Ncir et al. (2021) demonstrate the importance of scalable Dunn Index computation for validating clusters. Within the proposed framework, cluster validation serves as a safeguard against arbitrary or unstable grouping.

If two tests are placed in the same cluster despite having unrelated functional behavior, subsequent test prioritization may become inaccurate. Conversely, excessive fragmentation may produce too many small groups and reduce the efficiency benefits of clustering.

The framework therefore treats cluster quality as a measurable intermediate property rather than assuming that every machine-generated cluster is meaningful.

3.6 Risk-Based Test Prioritization

Following clustering, the framework assigns relative testing priority. The priority mechanism integrates information about recent code changes, historical failures, functional criticality, dependency exposure, and cluster-level behavior.

A conceptual priority score can be represented as:

[

P_i = w_1C_i+w_2F_i+w_3D_i+w_4R_i+w_5K_i

]

where (P_i) represents test priority, (C_i) code-change relevance, (F_i) historical failure behavior, (D_i) dependency exposure, (R_i) recent failure information, and (K_i) business or functional criticality. The weights are context-dependent and should be calibrated using historical organizational data rather than assumed to be universal.

This approach is consistent with the broader principle of AI-supported decision making illustrated by Philip (2024), where predictive models are positioned as mechanisms for improving decision accuracy. In regression testing, the equivalent objective is to

identify high-risk tests early enough to provide useful feedback within the Agile iteration.

3.7 Continuous Feedback and Model Refinement

The final stage returns test outcomes to the learning pipeline. Every execution generates additional information about which tests failed, which passed, how long they required, and how effectively they detected changes. The framework uses this feedback to update future prioritization.

This creates a continuous cycle:

Code Change → Data Update → Feature Extraction → Clustering → Validation → Prioritization → Test Execution → Feedback → Model Refinement.

The cycle is particularly compatible with Agile because the framework does not assume a static testing environment. As software evolves, the learned representation of testing risk also evolves.

3.8 Governance and Human Oversight

AI-based prioritization should not automatically eliminate tests merely because a model assigns them low priority. Bali et al. (2019) provide a broader reminder that AI systems require responsible computational governance. Accordingly, the proposed framework maintains human oversight for high-impact release decisions.

For example, if an AI system consistently assigns low priority to a security-related test because historical failures are rare, an experienced quality engineer should be able to override the recommendation. This prevents statistical frequency from being confused with engineering importance.

4. RESULTS

The analytical synthesis indicates that an AI-based regression-testing framework can improve the conceptual efficiency of Agile testing by transforming test selection from a predominantly static activity into an adaptive decision process. The first major finding is that machine-learning-based organization can reduce the conceptual complexity of large regression suites. Instead of treating thousands of test cases as independent units, clustering can expose groups with similar structural or behavioral characteristics. The methodological foundation for this approach is supported by the broader application of unsupervised learning to complex datasets (Adorf et al., 2019; Alloghani et al., 2020).

The second finding concerns the importance of relationship-aware analysis. Traditional test categorization may rely on manually assigned functional labels, whereas graph-based approaches can capture relationships among tests, modules, and dependencies. The spectral-clustering perspective associated with Bach and Jordan provides a useful theoretical foundation for this type of representation. In an Agile environment, such relationships can become particularly valuable when a small code change affects multiple dependent components.

The third finding is that cluster validation must be incorporated into the framework. Machine-generated groups should not be treated as inherently valid. The contribution of Ben Ncir et al. (2021) is relevant here because scalable cluster-validation measures can provide evidence about whether discovered structures possess meaningful separation and cohesion. This introduces an additional quality-control layer between machine learning and testing decisions.

The fourth finding is that risk-based prioritization is more appropriate than purely frequency-based execution. A test that rarely fails may still be extremely important if it covers a critical business function. Therefore, the proposed framework combines failure history with change relevance, dependency exposure, recency, and functional significance. This principle parallels the broader use of AI-driven prediction to improve engineering decision making described by Philip (2024).

The fifth finding is that continuous feedback is essential to maintain framework relevance. Agile systems evolve too rapidly for a static model to remain optimal indefinitely. Test execution outcomes must therefore become new training or analytical information. This feedback mechanism enables the framework to adjust when application architecture, test suites, or failure patterns change.

Nevertheless, the findings also reveal important limitations. AI does not automatically produce superior testing decisions. Poor-quality historical data, inappropriate features, unstable clusters, model drift, and excessive computational costs can undermine the expected benefits. Furthermore, a model can optimize statistical patterns while overlooking business-critical context. Consequently, AI-based regression testing should be interpreted as an adaptive decision-support system rather than a fully autonomous testing authority.

5. DISCUSSION

The proposed framework demonstrates that the most meaningful role of AI in Agile regression testing is not simply automating test execution but improving the intelligence of test-selection decisions. Conventional automation can execute thousands of tests rapidly, but execution speed alone does not determine testing effectiveness. When every Agile iteration introduces new changes, the central optimization problem becomes the allocation of limited testing time to the most informative and risk-relevant tests.

The literature supports the individual components of this argument but also reveals a significant theoretical limitation. Alloghani et al. (2020) and Adorf et al. (2019) establish methodological foundations for machine-learning-based pattern discovery, while Bach and Jordan provide a basis for graph-oriented clustering and Ben Ncir et al. (2021) demonstrate the importance of cluster validation. However, these contributions do not directly validate an integrated regression-testing framework. Therefore, the present framework should be regarded as a theoretically derived architecture requiring empirical evaluation before claims of quantitative superiority can be made.

A major trade-off concerns efficiency versus coverage. Aggressive AI-based test reduction may decrease execution time, but it can also increase the probability of omitting low-frequency defects. A robust framework should consequently distinguish between test prioritization and test elimination. Prioritization changes execution order without necessarily discarding tests, thereby allowing high-risk tests to produce early feedback while lower-priority tests continue later.

Another important issue is explainability. Agile developers and testers must understand why an AI system has prioritized one test above another. If the framework provides only opaque scores, engineers may distrust its recommendations. This concern is particularly important when model outputs affect release decisions. The governance perspective emphasized by Bali et al. (2019) supports the broader argument that computational intelligence should be accompanied by appropriate oversight.

The framework also has implications for continuous integration. A practical implementation could operate automatically whenever a new commit is introduced. The system would identify changed modules, evaluate affected test clusters, calculate relative priorities, execute high-priority tests first, and feed outcomes back into the analytical layer. Such integration could shorten the time required to obtain meaningful regression feedback.

However, implementation introduces computational overhead. Machine-learning models require data preparation, feature computation, cluster analysis, validation, and periodic recalibration. For small projects with limited test suites, these costs may exceed the benefits. AI-based regression testing is therefore more likely to provide substantial value in large and continuously evolving systems where regression suites and change frequencies justify intelligent optimization.

The framework further indicates that human expertise remains indispensable. Engineers possess contextual knowledge that may not exist in historical datasets, including knowledge of newly introduced architectural risks, emergency patches, regulatory requirements, and strategically important functionality. Philip (2024) illustrates the broader value of AI-driven predictive decision support, but predictive assistance should not be interpreted as a replacement for professional judgment.

Overall, the principal contribution of the proposed model is its integration of machine-learning pattern discovery, graph-oriented relationships, cluster validation, risk-aware prioritization, and continuous feedback into a unified Agile regression-testing architecture. Its effectiveness should ultimately be assessed through empirical measures such as regression execution time, defect detection rate, fault detection latency, coverage retention, prioritization effectiveness, model stability, and computational overhead.

6. CONCLUSION

Artificial intelligence provides a promising foundation for transforming regression testing from a repetitive execution process into an adaptive and data-informed quality-engineering activity. This study developed a conceptual framework that integrates data preparation, feature engineering, unsupervised clustering, graph-based relationship analysis, cluster validation, risk-oriented prioritization, continuous feedback, and human governance.

The literature demonstrates that machine learning can identify meaningful structures in complex datasets (Adorf et al., 2019; Alloghani et al., 2020),

graph-based methods can provide alternative mechanisms for representing relationships (Bach & Jordan, n.d.), and cluster validation is necessary to assess the reliability of discovered structures (Ben Ncir et al., 2021). The broader AI literature also reinforces the importance of responsible computational decision making (Bali et al., 2019), while predictive AI provides a useful conceptual foundation for engineering decision support (Philip, 2024).

The proposed framework contributes by connecting these principles to the practical requirements of Agile regression testing. Its central proposition is that AI should determine testing priorities dynamically according to change impact, historical behavior, dependency relationships, and validated structural patterns. Nevertheless, AI should not be treated as an infallible replacement for engineering judgment. Data quality, explainability, model drift, computational cost, and inappropriate prioritization remain substantial limitations.

Future research should empirically implement the framework within continuous-integration environments and compare AI-assisted regression testing against conventional strategies using standardized software projects and measurable quality indicators. Particular attention should be given to explainable prioritization, adaptive clustering, model drift detection, and the balance between regression speed and defect-detection coverage. Such empirical investigation would establish whether the theoretical advantages identified in this study translate into measurable improvements in real Agile software-development environments.

REFERENCES

1. Adorf, C. S., Moore, T., Melle, Y., & Glotzer, S. (2019). Analysis ofself-assembly pathways with unsupervised machine learningalgorithms.The Journal of Physical Chemistry B,124,69–78.

2. Alloghani, M., Al-Jumeily, D., Mustafina, J., Hussain, A., & Aljaaf,A. J. (2020). A systematic review on supervised andunsupervised machine learning algorithms for data science.In M. W. Berry, A. Mohamed, & B. W. Yap (Eds.),Supervised and unsupervised learning for data science(pp.3–21). Springer International Publishing.

3. Andrade-Rocha, F. T. (2013). Temporary impairment of semenquality following recent acute fever.Annals of Clinical andLaboratory Science,43,94–97.

4. Bach, F. R., & Jordan, M. I. (n.d.).Learning Spectral Clustering(UCB/CSD-03-1249; p. 14). University of California,Berkeley. Retrieved 24 June 2021, fromhttps://www2.eecs.berkeley.edu/Pubs/TechRpts/2003/CSD-03-1249.pdf

5. Bali, J., Garg, R., & Bali, R. T. (2019). Artificial intelligence (AI) inhealthcare and biomedical research: Why a strongcomputational/AI bioethics framework is required?IndianJournal of Ophthalmology,67,3.

6. Ben Ncir, C.-E., Hamza, A., & Bouaguel, W. (2021). Parallel andscalable Dunn Index for the validation of big data clusters.Parallel Computing,102, 102751.

7. Geo Philip, Paulson, Artificial Intelligence and Machine Learning Applications in Project Schedule Forecasting: A Predictive Framework for Time-Control in Complex Building Projects (April 16, 2026). Available at SSRN: https://ssrn.com/abstract=6588119 or http://dx.doi.org/10.2139/ssrn.6588119

8. K. Ramamurthy, R. K. Konduru and N. Amanmadov, "EvoGraphCoder: An Evolutionary Graph-Reasoning Framework for Self-Adaptive Software Engineering," in IEEE Access, vol. 14, pp. 63063-63076, 2026, doi: 10.1109/ACCESS.2026.3686019.