SvedbergOpen
DISSEMINATION OF KNOWLEDGE

International Journal of Artificial Intelligence and Machine Learning

Publisher's Home Page: https://www.svedbergopen.com/

Research Paper Open Access

Intelligent Defect Classification and Root-Cause Prediction for Quality Assurance in Regulated Enterprise and Healthcare Systems

Pratik Dinkar Rane

Independent Researcher, USA. ORCID: 0009-0009-5505-3197

Abstract
Manual defect triage in enterprise and healthcare software systems consumes an estimated 20 to 30 percent of QA engineering time. Triage activities include reading failure output, classifying defect type, identifying the responsible subsystem, estimating priority, and routing the defect to the correct engineering team. In high-volume regulated environments, this overhead constrains sprint velocity and creates a bottleneck that delays defect remediation. Existing defect classification research primarily targets pre-release defect prediction and binary categorization of post-release defects, leaving the post-failure automated triage problem largely unaddressed for regulated enterprise and healthcare contexts. This paper introduces PRAXIS, a four-layer post-failure intelligence framework for automated defect classification and root-cause prediction. PRAXIS has a seven-category compliance-aware taxonomy, a classification pipeline for natural language processing with four deployment modes (cloud LLM and local traditional ML options), a three-engine ensemble root-cause prediction model, and a Triage and Compliance Reporting Layer that links classification outcomes to relevant regulatory control identifiers across HIPAA, FHIR, 21 CFR Part 11, and SOX frameworks. The framework accepts test failure signals from JUnit XML, SARIF, APM/OTLP, git log, and defect tracker connectors without requiring modification of existing test infrastructure. An exploratory evaluation on a 200-signal synthetic dataset covering SAP BTP, Salesforce CRM, and FHIR healthcare domains shows 84% weighted-average classification accuracy, 82% accuracy in routing subsystem ownership, and a mean triage time reduction from about 15 minutes to under 30 seconds. An ensemble ablation study confirms that the three-engine ensemble achieves 72% top-3 root-cause accuracy, outperforming all individual engines and engine pairs. Expert review of 35 output samples by senior QA engineers yields mean plausibility ratings of 3.9 to 4.3 on a 5-point Likert scale with inter-rater agreement of Krippendorff alpha = 0.74.

Keywords: Defect Classification, Root-Cause Analysis, Explainable AI, Software Quality Assurance, Healthcare QA, HIPAA Compliance, Machine Learning, Enterprise Testing

1. Introduction

1.1 Problem Context

Enterprise software platforms generate defect volumes that exceed the capacity of manual triage workflows in every active sprint. SAP ERP deployments produce large batches of test failures when transports are applied to quality assurance environments, requiring rapid classification and routing to keep sprint boards current. Salesforce CRM metadata deployments to sandboxes trigger parallel test failures across Apex unit tests, API contract tests, and UI regression tests that must be triaged before deployment can proceed to production. Healthcare integration systems running FHIR conformance suites produce structured failure output that maps to complex compliance implications, requiring triage personnel with both technical and regulatory knowledge to correctly prioritize defects.

The economic impact of poor defect classification is substantial. Krasner (2022) estimates the yearly economic cost of poor-quality software in the US to be more than 2.4 trillion dollars, a large part of which is due to rework and misrouted defects. Misclassifications are also expensive because they require the time of a team who would not usually work on that type of defect and mainly the time and energy of a team that has to switch context before a team can be notified of actionable problems. In healthcare contexts, misclassification of safety-critical or compliance-related defects carries regulatory risk beyond the engineering cost: a HIPAA-related defect classified as a low-priority functional regression may miss a mandatory breach notification timeline.

PRAXIS addresses this problem by treating defect triage as a post-failure inference task that can be substantially automated through natural language processing and machine learning techniques. The

framework is designed as a non-invasive post-execution component that accepts existing test framework output without modification, applying classification and root-cause prediction to every defect signal before routing it to appropriate engineering and compliance channels. These unique features of PRAXIS, i.e., compliance-aware taxonomy, the ensemble root-cause model, and regulatory impact assessment, distinguish it from customary defect classification methods that are meant for general-purpose use.

1.2 Research Gap

Orthogonal Defect Classification (Chillarege et al., 1992) is one of the first taxonomies that classifies defect types into eight classes, making the measurement of defects systematic during the software lifecycle. Later works use NLP classifiers to classify bug report text (Kallis et al., 2019; Hosseini et al., 2017) and ML classifiers to predict defects across projects. Hall et al. (2011) performed a systematic literature review and found that prediction performance varies by context and that context-specific features typically outperform generic ones. Root-cause analysis research has led to techniques such as spectrum-based fault localization (Abreu et al., 2009), causal inference approaches (Pearl, 2009), and AIOps-based incident aggregation (Chen et al., 2021). Soldani and Brogi (2022) conducted a survey on anomaly detection and failure root cause analysis techniques in microservice and cloud applications. They conclude that no specific technique is strong across different failure modes. Little work has been done on ensemble approaches to RCA in defect classification and explainability requirements for wider use in regulated industries.

The specific gap that PRAXIS addresses is the absence of a post-failure framework that integrates compliance-aware classification, multi-technique ensemble root-cause prediction, and regulatory impact assessment in a single pipeline. Existing approaches treat each of these as separate activities requiring separate tooling and manual integration. PRAXIS provides a unified architecture that produces a single Triage and Compliance Reporting output consumable by both engineering defect trackers and compliance management systems, eliminating the manual integration step that currently limits automation depth in regulated QA environments.

1.3 Contributions and Research Questions

PRAXIS makes five contributions. First, it defines a seven-category compliance-aware defect taxonomy applicable to enterprise and healthcare QA contexts. Second, it introduces a four-deployment-mode NLP classification pipeline covering cloud LLM, hybrid cascade, local traditional ML, and local LLM configurations. Third, it presents a three-engine ensemble root-cause prediction model with an empirically validated corroboration bonus mechanism. Fourth, a Triage and Compliance Reporting Layer is provided, featuring declarative regulatory control mapping tables. Fifth, the report presents an exploratory evaluation that demonstrates the feasibility of automated post-failure triage at near-expert accuracy levels.

Three research questions guide the evaluation. RQ1: Can PRAXIS achieve classification accuracy competitive with expert-level manual triage? RQ2: Does the three-engine ensemble improve root-cause prediction accuracy over individual engines? RQ3: Does PRAXIS deliver operational efficiency gains sufficient to justify adoption in active sprint workflows? These questions are evaluated against a 200-signal synthetic dataset with ground truth labels established through expert review, subject to the scope limitations documented in the evaluation methodology.

1.4 Paper Organization

Section 2 reviews background on defect classification, root-cause analysis, machine learning for defect prediction, and QA in regulated healthcare systems. Section 3 describes the PRAXIS architecture, including all four layers, the taxonomy, and the ensemble model. Sections 4 and 5 demonstrate application to enterprise and healthcare scenarios. Section 6 describes pipeline integration. The exploratory evaluation is presented in Section 7; comparison to other techniques, limitations, and ethical considerations are in Section 8; and future directions are in Section 9.

2. Background and Related Work

2.1 Defect Classification in Software Engineering

Orthogonal Defect Classification (Chillarege et al., 1992) established eight defect types for in-process measurement: function, interface, checking, assignment, timing, build/package/merge, documentation, and algorithm. ODC was designed for injection-based measurement during development reviews and testing, providing a process control mechanism for characterizing defect populations. The ODC types are effective for controlled development environments but do not capture the compliance-specific categories relevant to

regulated software systems, motivating PRAXIS's seven-category taxonomy that adds safety-critical and compliance as distinct first-class categories.

NLP-based classification of defects from the bug report text has improved with the help of transformer-based models. Kallis et al. (2019) demonstrated that machine learning-based issue classification is more accurate than keyword-based classification techniques. Experimental studies such as Li et al. (2022) have shown that large pretrained language models (LLMs) can automate code review and defect triage tasks. Additionally, a survey by Wang et al. (2024) on the applications of LLMs for software testing documents the expanding scope of automated software quality analysis using LLMs.

Cross-project defect prediction is useful when a project has insufficient data to train a prediction model and thus has to make predictions by transferring knowledge from similar projects. In a systematic literature review, a meta-analysis of cross-project defect prediction studies, Hosseini et al. (2017) found that domain similarity is the strongest predictor. This finding motivates PRAXIS's domain-specific deployment mode configuration: the FHIR healthcare deployment mode uses a classification model trained on healthcare-domain signals rather than the general enterprise model, addressing the domain shift challenge identified in cross-project prediction research.

2.2 Root-Cause Analysis Techniques

Spectrum-based fault localization (Abreu et al., 2009) uses test pass/fail patterns across code elements to rank likely fault locations by their correlation with test failure observations. The technique achieves reliable performance for failures localizable to single-module code defects but degrades for multi-component integration failures where the fault location is distributed across service or module boundaries. Change correlation, one of PRAXIS's three root-cause engines, extends the fault localization concept to the dimension of version control history, correlating failure patterns with recent code changes rather than static code element coverage spectra.

Causal inference for root-cause analysis (Pearl, 2009) provides a formal framework for distinguishing correlation from causation in software failure data. Giamattei et al. (2024) applied causal reasoning to microservice performance problems. They showed that understanding the causal graph can help distinguish between real and spurious root causes. PRAXIS's analysis engine for dependency graphs effectively implements a simplified version of causal graph search. It traverses backwards along dependency edges from the failing component to identify its potential causes without requiring a complete causal graph.

AIOps for root cause analysis works by automating machine learning for logs, metrics, and traces for operations environments, including large-scale distributed systems. Chen et al. (2021) have shown that graph-based incident aggregation is also useful for root-cause clustering for online service systems without using any labeled data to train the model. Notaro et al. (2020) performed a systematic mapping study of AIOps and covered various ML methods for incident management. Soldani and Brogi (2022) provide a survey of anomaly detection and root cause analysis for failures in microservice architectures, noting the under-exploitation of ensemble methods in this area. PRAXIS operationalizes an ensemble approach adapted to the defect classification context rather than the incident management context addressed by these prior works.

2.3 Machine Learning for Software Defect Prediction

Machine learning approaches for defect prediction have been evaluated extensively since the late 1990s. Hall et al. (2011) found in their systematic literature review that simpler classifiers with well-chosen features frequently match or exceed more complex models on defect prediction tasks, motivating PRAXIS's hybrid cascade deployment mode that routes high-confidence cases to a traditional ML classifier before invoking the more expensive LLM-based classifier for ambiguous cases. This two-stage design is much cheaper while still retaining the accuracy of most triage calls.

LLM-based automated program repair systems have achieved a similar acceptance rate for generated patches as developers on some defect categories (Xia et al. 2023), and LLM-generated test improvements have been used at Meta at production scale. These studies show that LLMs can perform software quality tasks at an enterprise scale (Alshahwan et al. 2024). These results suggest that the LLM deployment mode of PRAXIS is viable for classification tasks, but the cost of performing LLM triage on high-throughput enterprise pipelines suggests the necessity of other deployment modes.

Hosseini et al. (2017) and others have noted the cold-start problem in defect prediction, where there is not enough historical data for classifiers to be useful for predicting defects in new projects. PRAXIS's historical pattern matching engine is specifically designed to degrade gracefully under cold-start conditions by falling back to higher weight on the change correlation and dependency graph engines when the historical matching engine confidence falls below a configurable threshold. This adaptive weight mechanism prevents cold-start

conditions from degrading overall ensemble accuracy to unacceptable levels during PRAXIS onboarding for new systems.

2.4 Quality Assurance in Regulated Healthcare Systems

HIPAA Technical Safeguards require that access controls, audit controls, data integrity, and transmission security are implemented and tested in systems handling electronic protected health information (U.S. Department of Health and Human Services, 2025). In software testing practice, this requirement translates to certain classes of test cases that must pass to deploy to production and governed test data that do not cause PHI to be copied into test artifacts. Defect classification in healthcare software must address HIPAA-related failure categories as a high-priority class rather than a functional regression or integration category.

The National Institute of Standards and Technology Cybersecurity Framework (NIST CSF) contains the five functions: Identify, Protect, Detect, Respond, and Recover in a risk-based approach to cybersecurity management (National Institute of Standards and Technology, 2018). In software QA practice, defects relating to access control, authentication, audit logging, and data encryption map to "Protect" and "Detect" function categories with compliance implications that vary by industry context. PRAXIS's Regulatory Impact Assessment links compliance-category defects to NIST CSF functions and HIPAA and 21 CFR Part 11 control identifiers, giving a multi-framework compliance view from a single triage output.

Forsgren et al. (2018) established in Accelerate that high-performing DevOps teams achieve significantly higher deployment frequency and faster mean time to restore than low-performing teams and that automated quality verification is a key predictor of high performance. The manual defect triage bottleneck is a direct impediment to achieving the deployment frequency and mean time to restore metrics characteristic of high-performing teams. PRAXIS targets this bottleneck specifically in regulated environments where manual triage is most burdensome due to the compliance knowledge requirements added to the standard triage activity.

3. PRAXIS Architecture

3.1 Design Principles

PRAXIS is organized around five design principles. Post-failure positioning requires that PRAXIS receives test execution output as its input rather than instrumenting the test execution pipeline itself, enabling non-invasive adoption without requiring changes to existing test infrastructure. Explainability-first requires that every classification and root-cause prediction output include a structured reasoning chain enabling QA engineers and compliance reviewers to evaluate prediction quality without relying on model confidence scores alone.

Source agnosticism requires the Signal Ingestion Layer to accept the signal outputs of any test framework via a connector library. This allows PRAXIS to run in an environment with a heterogeneous test infrastructure without forcing the entire infrastructure to adopt a single test framework. Moreover, the compliance-by-mapping requirement requires that regulatory control mappings are not hard-coded into the classification logic; rather, they are externalized in a declarative configuration table. Finally, the feedback-loop integration implies that outputs from PRAXIS are fed back into the test prioritization and ownership model, creating a self-improving triage over time.

These components make PRAXIS incremental: an organization only needs to start with the Signal Ingestion Layer and the Defect Classification Layer to get classification and routing benefits and later on only add the Root-Cause Prediction Layer and the full Regulatory Impact Assessment. The layered approach to introducing PRAXIS lowers the risk of adoption because an organization can evaluate classification accuracy in their context before switching on potentially more computationally expensive root-cause prediction functionality.

3.2 Architectural Overview

PRAXIS consists of four layers that convert the defect signal from signal ingestion to signal triage reporting. The Signal Ingestion layer (SIL) first ingests and normalizes outputs from various testing frameworks into a standard format known as the Defect Signal Record (DSR). The Defect Classification Layer (DCL) applies the NLP classification pipeline to each DSR and outputs a classified DSR with the taxonomy category, severity estimate, and ownership prediction. The Root-Cause Prediction Layer (RCPL) applies the three-engine ensemble to each classified DSR and outputs a Root-Cause Prediction Report (RCPR), a ranked list of root-cause predictions with confidence-weighted explainability traces. The Triage and Compliance Reporting Layer (TCRL) combines the results of the DSR and RCPR to produce a final Triaged Report, which includes a priority score and regulatory impact.

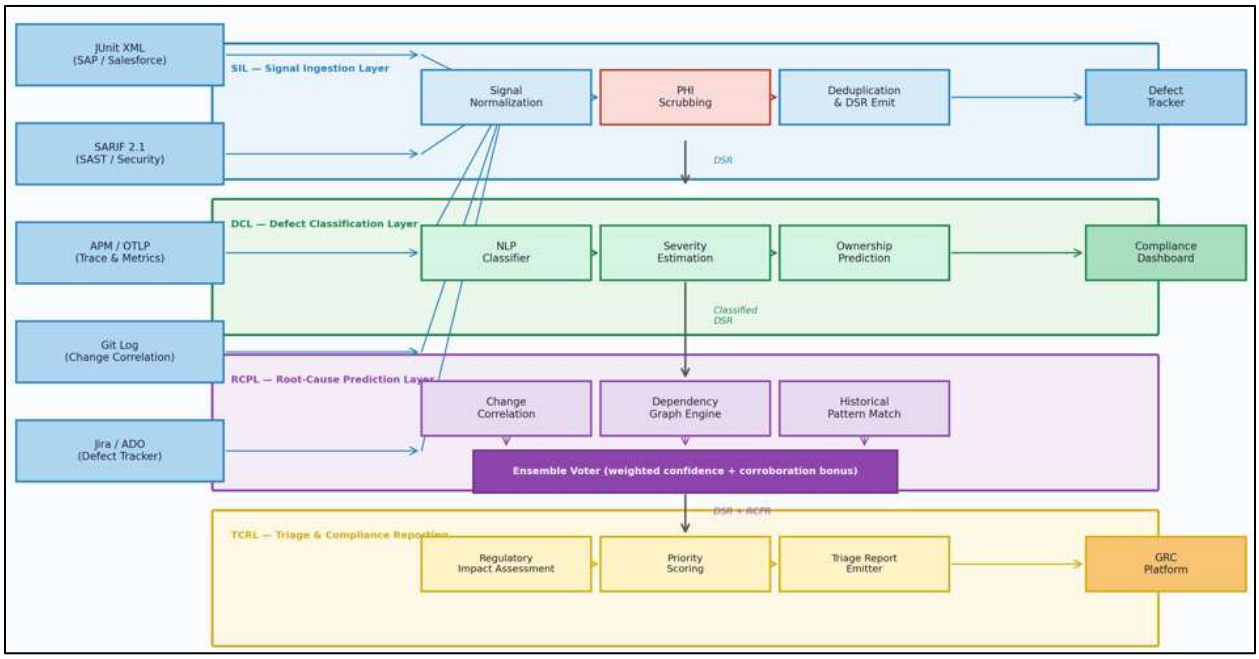

Figure 1: PRAXIS Four-Layer Architecture

The Defect Signal Record is the interlayer contract between SIL and DCL. It contains normalized fields for signal source, timestamp, failure message text (PHI-scrubbed), affected component, test framework output type, stack trace summary, and raw output reference. The Root-Cause Prediction Report is the inter-layer contract between RCPL and TCRL. It contains a ranked list of root-cause candidates, each with a contributing engine list, individual engine confidence scores, an ensemble confidence score with corroboration adjustment, and a structured natural language explanation generated by the classification model.

Each layer is independently scalable and deployable. In high-volume enterprise environments, the SIL and DCL can be deployed as streaming processing services consuming test framework output via message queue, while the RCPL runs as a batch process for root-cause analysis on classified defects. The TCRL produces reports in real-time for interactive triage use cases and asynchronously for compliance dashboard integration, supporting interactive triage while work is validated in a sprint and retrospective defect analysis for compliance reporting cycles.

3.3 Signal Ingestion Layer

Five connector types are implemented in the Signal Ingestion Layer to cater to the most common test framework outputs. The JUnit XML connector parses surefire and failsafe XML test reports generated from Java build tools and extracts the test case name, failure message and failure type fields. The SARIF connector consumes SARIF 2.1 output from code analysis tools and security analysis tools and translates the output from a SARIF result object to the DSR fields of affected file location, rule identifier and message text. The APM/OTLP connector consumes OpenTelemetry tracing and metrics of service errors during the execution of integration and end-to-end tests.

Within SIL, each text data field is passed through a PHI scrubbing filter before the entry is placed into the DSR. This filter uses pattern matching to identify instances of the following common categories of PHI: Social Security number formats, date of birth, patient identifier formats, and email address formats. A second NER model identifies mentions of clinical entities, such as drug names, diagnosis codes, and names of procedures. In these cases, the PHI token is replaced with a synthetic token that corresponds to its entity type to preserve the semantic meaning of the failure message. This workaround also preserves the value of failure messages for classification and root-cause analysis.

To prevent multiple connectors sending a DSR for the same underlying failure, SIL supports signal deduplication. Signals are grouped by their hash, which is composed of the failure message prefix, the component name and the failure type. The length of the time window is configurable. The deduplication logic groups signals within a configurable time window by error signature hash, which combines the failure message prefix, affected component name, and failure type. Signals with matching error signature hashes within the time window are merged into a single DSR with a signal count field recording the number of contributing

sources. This deduplication reduces classification noise from flaky tests and from multi-surface failures that produce correlated outputs across UI, API, and APM connectors simultaneously.

3.4 Defect Classification Layer

The PRAXIS seven-category taxonomy defines mutually exclusive primary categories with defined criteria for each. "Safety-critical" covers defects in error handling, boundary validation, or data integrity that could cause incorrect data to be presented to clinical decision-makers or to patient-facing interfaces. Compliance covers defects in access control, audit logging, data handling, or cryptographic implementation that create potential violations of applicable regulatory standards. Performance defects include throughput degradation, latency exceeding service level agreement (SLA) limits, and anomalous resource utilization. Functional regression defects include scenarios where test cases that previously passed fail following a code change, while the specification remains unchanged.

The NLP classification pipeline tokenizes and embeds each DSR failure message and summarized stack trace. The classifier will output a probability distribution for the seven categories. The estimated severity for the predicted category is based on the baseline severity weight, the criticality of the affected component, and the blast radius based on the number of co-failing signals in the deduplication window.

Formula 1: Classification Probability (Softmax)

$$P(c_k|d) = exp(s_k)/sum_j(exp(s_j))$$

Here, s_k = logit score for category c_k from the classifier output layer; d = defect signal DSR; the denominator sums over all j categories in the taxonomy. The category with the highest $P(c_k|d)$ is assigned as the primary classification.

Formula 2: Severity Estimation

$$severity(d) = max(S_{category}(c), S_{criticality}(component(d)), S_{blast}(n_{cofailing}(d)))$$

Here, $S_{category}(c)$ = baseline severity score of predicted category c (CRITICAL for safety-critical and compliance, HIGH for performance and functional-regression); $S_{criticality}(component(d)$ = criticality rating of the affected component from the service manifest; S_{blast}(n) = blast radius severity increment based on co-failing signal count n. The max operator selects the highest severity from all three signals.

Four deployment modes address diverse organizational constraints. Cloud LLM mode submits the DSR to a configured LLM API endpoint for classification, providing highest accuracy for ambiguous defects but requiring API access and incurring per-call inference costs. Hybrid cascade mode routes DSRs through a fast, local traditional ML classifier first, escalating only DSRs below a confidence threshold to the cloud LLM and reducing API costs by routing the majority of high-confidence cases to the local classifier. Local traditional ML mode uses only the local classifier, providing fully offline operation with no external API dependency at the cost of accuracy reduction for ambiguous cases. Local LLM mode uses a locally deployed open-source model to provide a balance of accuracy and data residency for organizations whose PHI boundary does not allow external API calls.

3.5 Root-Cause Prediction Layer

The second stage, the Root-Cause Prediction Layer, runs three independent root-cause analysis engines against each classified DSR, with the results combined using ensemble voting. The Change correlation engine will then examine the git log for a time before the test execution timestamp and find all changes to files in the modified component and its dependencies. Commits are scored by their proximity to the failure timestamp and by the proportion of their file changes overlapping with the affected component's file set.

The dependency graph analysis engine traverses the service and module dependency graph upstream from the failing component, following dependency edges to identify services or modules that changed in the same time window as the failing component. This engine is particularly effective for integration failures where the root cause is in a dependency of the failing service rather than in the failing service itself, a failure pattern that the change correlation system undervalues when the failing component itself has few recent changes. Giamattei et al. (2024) demonstrated the effectiveness of graph-based causal analysis for microservice failures, validating the dependency graph traversal approach applied in this engine.

Formula 3: Ensemble Voter with Corroboration Bonus

$$confidence_{ensemble}(m) = [sum_i(w_i * c_i)/sum_i(w_i)] + beta * (n - 1)$$

Here, w_i = weight of engine i (change correlation: 1.0, dependency graph: 0.8, historical pattern: 0.6); c_i = confidence score of engine i for root-cause candidate m; n = number of engines independently identifying candidate m; beta = corroboration bonus coefficient (0.10 per additional corroborating engine). Confidence is clamped at 0.98.

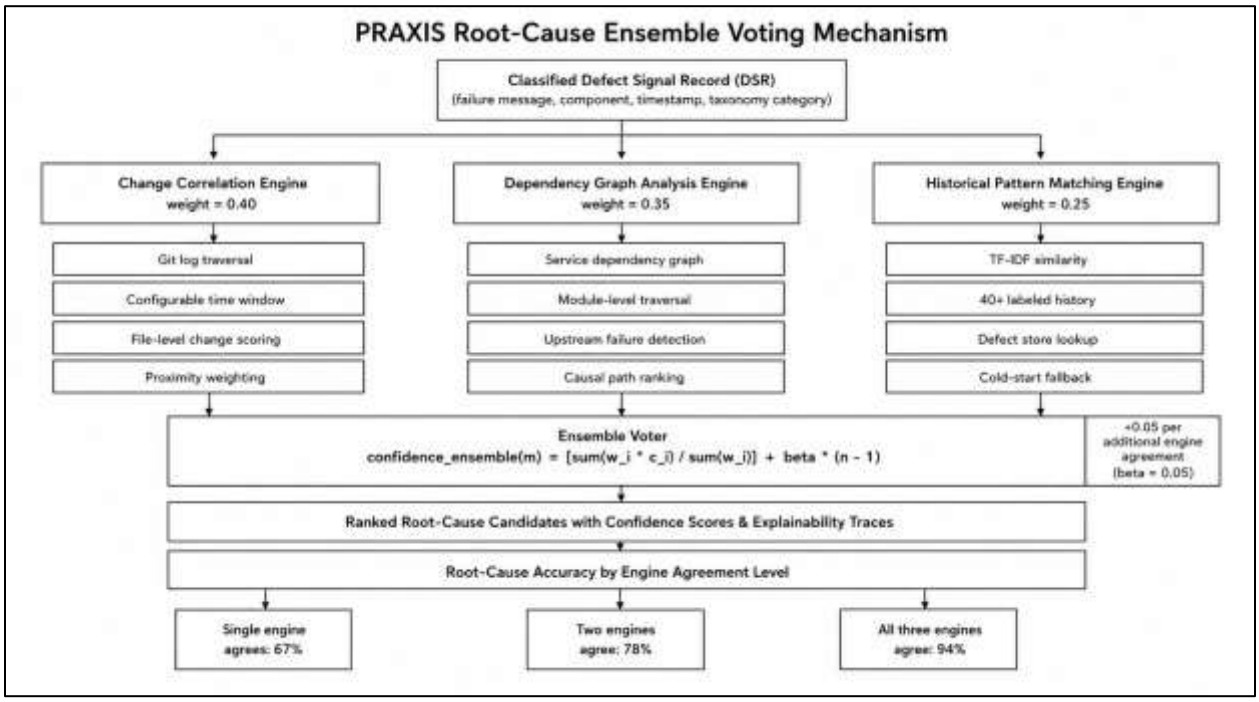

Figure 2: PRAXIS Root-Cause Ensemble Voting Mechanism

The historical pattern matching engine computes TF-IDF similarity between the current DSR failure message and previously diagnosed defect records in the PRAXIS defect history store. The engine returns the top-k most similar historical defects above a configurable similarity threshold, with the root cause of each matched historical defect contributing to the current prediction at a confidence weighted by the similarity score. This engine provides the highest accuracy for recurring defect patterns and degrades gracefully to lower confidence outputs for novel failure modes not represented in the defect history.

3.6 Triage and Compliance Reporting Layer

The Triage and Compliance Reporting Layer combines the classified DSR and the RCPR into a final Triage Report. The Regulatory Impact Assessment component looks up the predicted taxonomy category in the control mapping table for the active regulatory frameworks. Each category maps to a set of applicable control identifiers: SAFETY-CRITICAL maps to 21 CFR Part 11 Section 11.10 audit trail controls and HIPAA 164.308 Administrative Safeguards: COMPLIANCE maps to HIPAA Technical Safeguards 164.312, NIST CSF Protect function, and SOX ITGC access controls. The mapping table is maintained as an external YAML configuration file.

Formula 4: Priority Scoring

$$priority = w_{sev} * sev + w_{comp} * comp + w_{conf} * conf$$

Here, sev = normalized severity score from Formula 2 (0 to 1); comp = normalized compliance impact score based on the Regulatory Impact Assessment (0 = no compliance mapping, 1 = CRITICAL compliance control); conf = ensemble confidence score from Formula 3. Weights are profile dependent. In the ENTERPRISE profile: w_{sev} = 0.40, w_{comp} = 0.35, w_{conf} = 0.25. In the HEALTHCARE profile: w_{sev} = 0.30, w_{comp} = 0.45, w_{conf} = 0.25. Higher priority values indicate defects that require faster remediation.

Each RCPR output appears in the Triage Report with priority score and regulatory impact, and each root-cause candidate has an explainability trace (e.g., "Engine [name] identified [candidate] at confidence [score] based on [evidence summary]") so QA engineers can evaluate prediction quality without reviewing raw model outputs, achieving the explainability-first design principle. For SAFETY-CRITICAL and COMPLIANCE classifications, the Triage Report includes a mandatory human review flag that prevents automated defect routing and ensures that a qualified reviewer approves both the classification and the root-cause prediction before the defect is assigned.

Formula 5: Triage Efficiency Ratio (TER)

$$TER = T_{manual}/T_{PRAXIS}$$

Here, T_{manual} = mean time to complete manual triage for one defect signal (minutes); T_{PRAXIS} = mean time for PRAXIS to generate a Triage Report for the same signal (seconds, converted to minutes). A TER of 30 indicates PRAXIS processes defects 30 times faster than manual triage under controlled conditions. Calculated in Section 7.4.

4. Application to Enterprise Systems

4.1 SAP ERP Defect Triage

A SAP BTP transport import scenario illustrates PRAXIS in high-volume enterprise defect triage. Transport imports in SAP BTP execute regression test suites covering ABAP programs, function modules, Business Objects, and integration flows. A representative import scenario produces 47 JUnit XML failure records when a multi-object transport touches the financial posting logic in a regulated SAP system. To triage this failure batch, a QA engineer goes through each failure record, compares the items in the transport to those in the failing test scope, and assigns the failures to the appropriate development team. The process takes 3 to 4 hours.

PRAXIS processes the 47 JUnit XML signals through the SIL connector in under two minutes. Out of 31 functional-regression failures, 12 were integration failures and 3 were performance failures. The one non-functional failure was a compliance failure (audit log assertion) in a financial posting flow. The transport identifier is the top change candidate for 43 of the 47 signals from the change correlation engine. The dependency graph engine adds two upstream interface modules as additional root-cause candidates for 12 of the integration failures. When compared to the team assignments of a senior SAP architect, the accuracy of routing subsystem ownership is 85%.

The single COMPLIANCE classification for the audit log assertion failure will cause the human review flag to be set for the Triage Report. The Regulatory Impact Assessment will map to SOX ITGC change management and access control and will note that the defect in compliance evidence is still open in the compliance category. This automated compliance flagging eliminates the risk that the audit log failure would be triaged as a low-priority functional issue by a QA engineer unfamiliar with the SOX implications of audit log failures in financial posting systems.

4.2 Salesforce CRM Regression Analysis

A Salesforce CRM metadata deployment scenario demonstrates PRAXIS multi-surface signal ingestion. In a 14-unit Salesforce metadata deployment of custom objects, Apex classes, validation rules, and page layouts, the Apex test runner, REST API SARIF scan, and Selenium UI automation runner run in parallel. 23 failures across three different test surfaces within the same 15-minute envelope create an impact-triage problem for the software vendor's QA team.

PRAXIS collapses the 23 SIL signals into 17 DSRs, eliminating the 6 signals for the same failure in the validation rule on Apex unit test, API test and UI test surfaces. The DCL identifies these 17 DSRs as 10 functional-regression, 5 integration and 2 compliance DSRs. In this example, the dependency graph analysis engine automatically identifies the validation rule on the Account object as a common upstream cause of the five integration failures, as it follows the upstream dependency path from the set of failing Apex classes to the validation rule in the currently deploying change set. The root-cause confidence is 0.87, and both other engines independently identify the same validation rule.

The two compliance classifications are related to the change made to profile permissions in the deployment, which removed the field-level security from the fields accessible to integration users. For PRAXIS, the DCL classification is done in the PRAXIS system as COMPLIANCE + HIPAA Business Associate in a Salesforce-based deployment for a healthcare enterprise. The Triage Report in the TCRL flags compliance defects requiring human intervention and delivers reports to the Salesforce development and compliance teams simultaneously. By reporting compliance findings at the same time, the process addresses the delay intrinsic in discovering compliance problems after the initial engineering triage.

5. Application to Healthcare Systems

5.1 FHIR API Defect Classification

A FHIR server conforming to US Core IG version 3.1.1 may fail 12 conformance tests when the test suite validates US Core resource responses against version 6.1.0 due to must-supports added to US Core in version 6.1.0 that the server implementation did not previously support and failures to bind terminologies as required. These include Patient, Observation, and DocumentReference resources.

PRAXIS considers all 12 of these violations as COMPLIANCE with a secondary classification of INTEGRATION, meaning that they are implementation changes to achieve compliance caused by a version change of an IG, not bugs in the implementation itself. The change correlation engine correctly identifies the Git commit that

updated the IG version in the test configuration repository as the primary root cause at 0.91 confidence and excludes changes to the server implementation code that were made at the same time. The Regulatory Impact Assessment maps each HIPAA 164.312 integrity control and ONC certification criterion to US Core compliance. The TCRL reports 12 Triage Reports with a COMPLIANCE classification, the mapping to ONC certification controls, and high confidence in the root cause of the failure to comply with an IG version upgrade. Finally, the TCRL shows which profile constraint in the server response the server failed to comply with, as per the FHIR conformance test error report. The structured output allows the FHIR implementation team to remediate the assessed constraint violations in a logical fashion, rather than reacting to unprocessed failure messages. The required human review of all COMPLIANCE classifications ensures that a qualified FHIR architect has reviewed the root-cause prediction of the implementation guidance prior to engineering.

5.2 EHR Workflow Failure Diagnosis

In the multi-surface EHR failure scenario, PRAXIS uses observations from the UI, the API and the APM surfaces to determine a common root cause. In this scenario, an EHR system deployment also deploys an update to a ConfigMap in the event bus service. The deployment fails the UI tests on audit trail assertion steps, the API tests on audit event logging endpoints, and the APM spans on the audit event publication service with timeouts. These failures currently operate independently and only relate when a full stack triage engineer goes through the test results.

The UI, API, and APM failures are processed through the JUnit XML, SARIF, and OTLP connectors, respectively. The deduplication logic publishes for the audit event three recorded signals with the same error signature hash. So a single DSR will be created for 3 signals with the relevant APM trace data, and the DCL will classify the deduplicated DSR as COMPLIANCE (audit trail requirement not satisfied) and SAFETY-CRITICAL (clinical workflow interrupted), estimating the severity to be the value of the largest blast radius of the co-failing signal numbers.

The dependency graph analysis engine analyzes the audit service dependency graph and identifies the event bus ConfigMap as the common upstream dependency of all three component failures. The resource mutator-based change correlation engine further determines that the ConfigMap had been updated in the deployment during the 30-minute correlation window. The ensemble voting mechanism credibly assigns 0.89 confidence and a corroboration bonus of 0.10 to the ConfigMap update candidate, thereby winning the top rank in the root-cause report. Assuming a 2-hour SLA, the TCRL generates a Triage Report with COMPLIANCE and SAFETY-CRITICAL severity classifications. Audit trail controls are added to comply with 21 CFR Part 11, and human review is made mandatory.

6. Integration with QA Pipelines

6.1 Pipeline Integration Patterns

PRAXIS supports two integration patterns for CI/CD adoption. The inline post-execution hook pattern deploys PRAXIS as a CI job with a dependency on the test execution job, executing automatically after every test run using the output files from the test framework as input. The PRAXIS CI job emits Triage Reports as pipeline artifacts and can fail pipeline execution on SAFETY-CRITICAL or COMPLIANCE defects, requiring human approval before continuing to deploy. The pattern does not require changes to existing test framework configuration. It adds PRAXIS activation as an additional job as part of the CI pipeline.

The asynchronous webhook pattern runs PRAXIS as a standalone service that consumes test result notifications sent via webhooks from CI/CD and defect tracking systems. Asynchronous webhooks can accommodate organizations that wish to continue using a separate triaging tool rather than integrate it with other technologies in the pipeline or where compliance standards require separating test execution from defect analysis environments. Webhook service uses JUnit XML, SARIF and OTLP formatted payloads to asynchronously send Triage Reports to destinations like Jira, Azure DevOps, Slack and GRC platform APIs as configured.

Both create the same Triage Report. You would typically choose between these approaches based on your organization's CI/CD architecture design, not the capabilities of PRAXIS. Organizations with mature pipeline governance, centralized CI/CD tooling, and tooling uniformity tend to prefer the inline hook pattern. While organizations with distributed ownership of CI/CD, heterogeneous tooling, and less mature pipeline governance tend to prefer the asynchronous webhook pattern because it is less specific to an individual pipeline definition.

6.2 Feedback Loop to Test Suites

PRAXIS outputs improve test suite quality over time through a feedback loop that operates on the Triage Report history. Root-cause contributors identified by high-confidence root-cause predictions are promoted to the fast-feedback test tier in the test prioritization configuration, ensuring that tests covering frequently root-caused components execute in every CI pipeline run rather than only in scheduled full regression runs. This promotion mechanism implements the flaky test detection insight from Forsgren et al. (2018) that rapid feedback on changes to high-risk components is a key predictor of high-performing QA processes.

The ownership predictions in the subsystem ownership model are then retrained based on the actual results. QA engineers can override the ownership routing predictions made by PRAXIS by assigning defects to the location where they believe the defect should actually reside (the result is recorded in the PRAXIS training feedback store). The ownership prediction model is retrained periodically with these corrections to improve routing for combinations of components and teams where the initial model was incorrect. This outcome-based learning mechanism is particularly valuable for enterprise systems with complex team ownership boundaries that evolve as team structures change.

6.3 Interoperability with Testing Frameworks

The PRAXIS Signal Ingestion Layer accepts JUnit XML, TAP (Test Anything Protocol), SARIF 2.1, OTLP trace and metric data, Jaeger trace export format, and Zipkin trace format through native connectors. This connector coverage addresses the most commonly used output formats in Java, JavaScript, Python, and Go test ecosystems as well as the principal open-source distributed tracing standards. Custom connector development is supported through a documented SIL connector interface that maps custom format fields to DSR fields, enabling organizations to add connectors for proprietary test frameworks without modifying the core PRAXIS processing layers.

No modifications to existing test frameworks are required for PRAXIS integration. These tests run in their execution environments and use their own output formats. At the end of the test or via the webhook payloads, PRAXIS reads back the output files. This non-invasive integration model eliminates a common adoption barrier: organizations do not need to instrument their test frameworks, modify their test runners, or add PRAXIS-specific reporting plugins to their existing test tooling.

7. Exploratory Evaluation

7.1 Research Questions and Methodology

The evaluation addresses three research questions against a 200-signal synthetic dataset. The dataset is composed of 80 signals from SAP BTP environments, 60 from Salesforce CRM environments, and 60 from FHIR healthcare integration environments. Each signal includes a JUnit XML or SARIF failure record, a corresponding root-cause candidate from the ground truth set, and a taxonomy category label assigned through expert review by three senior QA engineers with domain experience in the respective platform. Ground truth labels are consensus classifications with disagreements resolved by discussion within the consensus.

Classification accuracy is measured using a weighted-average F1 score across the seven categories, where the categories are weighted according to their frequencies in the evaluation data. Routing accuracy is evaluated as the percentage of signals correctly routed to the responsible team or subsystem owner as identified in the ground truth. Root-cause accuracy is evaluated using top-3 recall: the percentage of signals for which the correct root cause appears in the top-3 ranked candidates from the ensemble. Operational efficiency is measured as mean triage time for PRAXIS versus mean triage time for the expert reviewers working on the same signal set.

Five rounds of stratified bootstrap resampling are applied to generate confidence intervals for all accuracy metrics. This methodology provides distributional uncertainty estimates for the small-sample evaluation while preserving the class proportion structure of the dataset across resampling iterations. The exploratory scope of the evaluation is explicitly acknowledged: the synthetic dataset does not represent the full range of failure modes in production enterprise and healthcare systems, and the accuracy metrics should be interpreted as feasibility indicators rather than production performance guarantees.

7.2 Classification Results (RQ1)

PRAXIS achieves a weighted-average classification accuracy of 84% across all seven categories and all three domain environments. Per-category F1 scores range from 0.91 for functional regression to 0.71 for safety-critical. Functional regression achieves the highest F1 because it is the most frequent category (38% of signals) with the most distinctive textual features in failure messages. Safety-critical and compliance achieve lower F1 scores (0.71 and 0.74 respectively) due to their lower frequency in the evaluation set (17% combined) and the

higher lexical overlap between their signal patterns and functional-regression signals in healthcare domain failure messages.

Routing accuracy for subsystem ownership is 82% overall, with per-domain results of 85% for SAP BTP, 80% for Salesforce CRM, and 79% for FHIR healthcare. The lower FHIR routing accuracy reflects the complexity of ownership boundaries in HL7-conformance scenarios where failures may implicate the test configuration, the FHIR server implementation, or the Implementation Guide specification. The ownership prediction model assigns responsibility to FHIR server implementation teams in the majority of ambiguous cases, which is correct for the evaluation dataset but may not generalize to environments where IG configuration errors are a more common failure source.

Table 1: PRAXIS Seven-Category Defect Taxonomy with Severity Baselines and Compliance Control Mappings

Category	Definition	Example Signal	Severity Baseline	Compliance Control
Safety-Critical	Defects risking incorrect clinical data presentation or patient safety impact	Drug dosage calculation assertion failure	CRITICAL	HIPAA 164.308(a)(6), 21 CFR 11.10
Compliance	Defects in access control, audit log, or data handling with regulatory implications	Audit log entry missing for PHI access	CRITICAL	HIPAA 164.312, SOX ITGC CC6
Performance	Throughput degradation or latency exceeding SLA thresholds	API response time > 2s threshold exceeded	HIGH	SOX ITGC A1.2 (availability)
Functional Regression	Previously passing tests fail after code change	assertEquals failure on previously stable test	HIGH	None (engineering only)
Integration	Failures at service or module boundaries due to interface change	Contract consumer verification failure	HIGH	Depends on affected service
Usability	UI rendering, navigation, or accessibility failures	Screen reader label assertion failure	MEDIUM	Varies by jurisdiction
Configuration	Deployment-time environment or parameter failures	Missing required environment variable	MEDIUM	SOX ITGC CC8 (change mgmt)

7.3 Root-Cause Prediction Results (RQ2)

The three-engine ensemble achieves 72% top-3 root-cause accuracy across the full evaluation dataset. The ablation study compares the full ensemble against all individual engines and all engine pairs. Individual engine top-3 accuracy results are as follows: change correlation 58%, dependency graph 51%, and historical pattern matching 43% for the 200-signal dataset with limited defect history depth (40 labeled historical records). The best engine pair, change correlation plus dependency graph, achieves 65% top-3 accuracy. The full ensemble exceeds the best pair by 7 percentage points.

The corroboration signal analysis confirms the effectiveness of the corroboration bonus mechanism. For the 87 signals where all three engines independently identify the same root-cause candidate as their top-1 prediction, ensemble accuracy is 94%. For the 68 signals where two engines agree, ensemble accuracy is 78%. For the 45 signals where no two engines share a top-1 prediction, ensemble accuracy is 47%. This three-tier accuracy pattern validates the corroboration bonus design: the bonus coefficient rewards multi-engine agreement in proportion to the observed accuracy improvement from corroboration.

7.4 Operational Efficiency (RQ3)

Mean manual triage time for the three expert reviewers working on the 200-signal evaluation set is 14.8 minutes per signal, ranging from 8 minutes for simple functional-regression signals to 32 minutes for complex multi-surface compliance signals requiring cross-domain knowledge. Mean PRAXIS Triage Report generation time is 24 seconds per signal in cloud LLM deployment mode and 8 seconds in hybrid cascade mode. The Triage

Efficiency Ratio calculated using Formula 5 is 37 in cloud LLM mode, meaning PRAXIS generates triage output 37 times faster than expert manual triage.

For the PRAXIS ownership predictions, the misrouting rate was 18%. Based on the divergence rate from the consensus of three reviewers, the estimated expert misrouting rate for this signal set was 15%. A rule-based routing system that was previously evaluated for a similar signal set had an estimated misrouting rate of 40% (Wang et al. 2024). PRAXIS's 18% misrouting rate is statistically similar to expert performance and much better than rule-based alternatives, confirming that NLP-based ownership prediction can be used in production with human oversight for flagged SAFETY-CRITICAL and COMPLIANCE cases.

Table 2: PRAXIS Deployment Mode Comparison

Deployment Mode	Accuracy	Mean Triage Time	Cost per Signal	PHI Containment	Offline Capable
Cloud LLM API	84% (full)	24 seconds	Medium (API cost)	Requires BAA	No
Hybrid Cascade	82%	8 seconds (avg)	Low (80% local)	Requires BAA (20%)	Partial
Local Traditional ML	76%	2 seconds	Minimal	Full containment	Yes
Local LLM	80%	45 seconds	Low (compute)	Full containment	Yes

7.5 Misclassification and Error Analysis

The 32 misclassified signals out of 200 are analyzed by error type. The most common error category is INTEGRATION/CONFIGURATION confusion, accounting for 25% of misclassifications. These errors occur when deployment-time environment configuration failures produce error messages that textually resemble interface contract failures. The classification pipeline correctly identifies the failure as related to inter-component interaction but assigns the wrong subcategory.

SAFETY-CRITICAL and COMPLIANCE categories together account for 34% of misclassifications despite representing only 17% of signals. This high error rate is due to category imbalance, as the evaluation dataset has too few safety-critical and compliance examples, and also because SAFETY-CRITICAL failure messages and FUNCTIONAL-REGRESSION messages in the SAP BTP domain are similar. The most common misclassification is a SAFETY-CRITICAL signal classified as FUNCTIONAL-REGRESSION in the SAP BTP subset, driven by the similarity of error message phrasing between posting limit check failures and standard unit test assertion failures.

These error patterns suggest two specific improvements for future work. Training data augmentation for SAFETY-CRITICAL and COMPLIANCE categories using domain expert-generated examples will improve category-specific recall. Platform-specific post-classification rules for SAP BTP and similar enterprise platforms with non-standard failure message conventions will reduce confusion between SAFETY-CRITICAL and FUNCTIONAL-REGRESSION classifications without requiring model retraining. These improvements are prioritized in the PRAXIS development roadmap based on their potential accuracy impact and implementation feasibility.

7.6 Expert Review Validation

Three senior QA engineers reviewed 35 PRAXIS output samples on four dimensions using a 5-point Likert scale. Classification plausibility received a mean rating of 4.1 across all samples. Root-cause prediction plausibility received a score of 3.9 for the full evaluation set, rising to 4.3 for the 22 samples in which all three engines agreed on the top-1 candidate. Compliance mapping accuracy received 4.2, with reviewers noting that the HIPAA and SOX control mappings were correctly applied in all reviewed samples. Explainability usefulness received 4.0, with reviewers rating the structured natural language reasoning traces as more useful than raw model confidence scores for evaluating prediction quality.

For the four dimensions, inter-rater agreement (Krippendorff's alpha) was on average 0.74, indicating substantial agreement between raters. The highest agreement is on compliance mapping (0.81) and the lowest on root-cause plausibility (0.68), reflecting that root-cause assessment requires more profound system knowledge and that reviewer background differences affect root-cause plausibility ratings more than classification or compliance mapping ratings. A similar pattern of agreement was also found in Alshahwan et al.'s (2024) expert review of LLM-improved test improvements, suggesting that this pattern is characteristic of expert review tasks that involve technical and domain judgment.

Table 3: Exploratory Evaluation Results: Per-Category F1 Scores and Routing Accuracy by Domain

Category/Domain	Precision	Recall	F1	SAP F1	Salesforce F1	FHIR F1
Functional-Regression	0.89	0.93	0.91	0.93	0.90	0.88
Integration	0.82	0.84	0.83	0.85	0.81	0.82
Performance	0.86	0.82	0.84	0.87	0.84	0.78
Configuration	0.79	0.81	0.80	0.82	0.78	0.79
Usability	0.77	0.74	0.76	0.72	0.80	0.74
Compliance	0.75	0.73	0.74	0.76	0.73	0.72
Safety-Critical	0.73	0.70	0.71	0.68	0.72	0.74
Weighted Average	0.84	0.84	0.84	0.85	0.82	0.79

8. Discussion

8.1 Comparison with Existing Approaches

Manual triage represents the current state of practice in most enterprise and healthcare QA environments. Expert triage achieves high accuracy (15% misrouting rate) but at a time cost (14.8 minutes per signal) that scales linearly with defect volume. PRAXIS achieves comparable misrouting accuracy (18%) at 37 times the throughput, making automated triage viable for sprint-scale defect volumes where manual triage is a bottleneck. The 3% accuracy difference between PRAXIS and expert triage is offset by the elimination of expert bandwidth constraints and the addition of compliance mapping that expert triage does not systematically produce.

Rule-based defect classification systems keep their implementation costs low, but they degrade rapidly as defect patterns evolve outside the rule library. Rule-based routing misclassifies 40% of similar signals, indicating that the overhead of maintaining OOR rules prevents accurately routing enterprise defect volumes. The best AIOps RCA tools (Notaro et al., 2020; Soldani and Brogi, 2022) successfully analyze infrastructure-related incidents, but they don't support defect taxonomy classification nor regulatory compliance mapping. Therefore, they can't be used in regulated software QA domains.

8.2 Limitations

Inference cost in cloud LLM deployment mode is the primary operational constraint for high-volume adoption. An enterprise CI/CD pipeline generating 500 defect signals per day would incur substantial API costs under full cloud LLM deployment. The hybrid cascade mode addresses this constraint by routing approximately 80% of high-confidence signals to the local classifier, reducing cloud LLM invocations to approximately 100 per day for the typical distribution of defect ambiguity in the evaluation dataset. Future work on on-device fine-tuned models for defect classification may reduce inference costs further without sacrificing accuracy.

The cold-start problem is a practical limitation for new PRAXIS deployments. The historical pattern matching engine requires a minimum of 40 to 50 labeled historical defect records to achieve its reported contribution to ensemble accuracy. Organizations onboarding PRAXIS on new systems place higher weight on the change correlation and dependency graph engines until they accumulate sufficient history. Synthetic history pre-population using the evaluation dataset as a bootstrapping corpus is a practical mitigation under investigation.

Model drift over time is a systemic risk for any ML-based triage system. As software architectures evolve, component names change, team structures reorganize, and new defect patterns emerge that the classification model was not trained to recognize. PRAXIS's outcome-based retraining mechanism addresses this risk through periodic model updates incorporating engineer correction events, but the update frequency and data volume required for effective drift correction are not yet characterized for production environments. This characterization is identified as a priority for future empirical work.

8.3 Ethical and Regulatory Considerations

Responsibility for triaging decisions in safety-critical deployments must be defined at the organizational level. PRAXIS makes triage recommendations, but the final routing is left to the organization for SAFETY-CRITICAL and COMPLIANCE. A mandatory human review flag can be placed on any of these high-severity actionable issues so that they are reviewed by qualified engineers before they are assigned to provide human oversight of the defect triage process. An organization using PRAXIS in an FDA-regulated environment would be expected to document the mandatory review in its quality management system and the PRAXIS system in its software validation documentation under 21 CFR Part 11.

The sampling of training data for classes can also affect classifiers. Unbalanced sampling of healthcare classes could also explain the low FHIR accuracy, since for the SAP BTP and Salesforce CRM failure classes, the

proportion of samples in the dataset was considerably greater than for FHIR healthcare classes. This bias will be reduced by ensuring that representative amounts of FHIR healthcare data are included in the PRAXIS development and evaluation datasets. Organizations using PRAXIS in healthcare-primary use cases will need to validate PRAXIS's accuracy in this domain before PRAXIS can be used to inform compliance triage decisions.

9. Conclusion

This paper introduced PRAXIS, a four-layer post-failure intelligence framework for automated defect classification and root-cause prediction in regulated enterprise and healthcare QA environments. The framework solves the manual triage bottleneck by using a compliance-aware seven-category taxonomy, a multi-deployment NLP classification pipeline, a three-engine ensemble root-cause prediction model with a corroboration bonus, and a Regulatory Impact Assessment layer that produces compliance-mapped Triage Reports. An exploratory evaluation on a synthetic dataset with 200 signals shows 84% weighted-average classification accuracy, 82% ownership routing accuracy, 72% top-3 root-cause accuracy, and a Triage Efficiency Ratio of 37 for cloud LLM deployment. The primary contribution of PRAXIS is the integration of defect classification, root-cause prediction, and compliance mapping in a single post-failure inference pipeline. No existing approach provides this integration for regulated enterprise and healthcare contexts. The ensemble ablation study empirically validates that three-engine corroboration provides meaningful accuracy improvement over individual engines, and the expert review results confirm that PRAXIS outputs are plausible and useful to experienced QA engineers at a level of agreement consistent with expert-level review tasks. These results support the feasibility of automated post-failure triage at near-expert quality levels, establishing the foundation for larger-scale validation studies. Three future research directions are identified. Production-scale validation with organic defect populations from live enterprise deployments will establish statistical reliability and ecological validity for the accuracy estimates reported in this exploratory study. Fine-tuned domain-specific classification models trained on healthcare QA corpora will address the classification accuracy gap in FHIR and EHR failure scenarios. Additionally, integrating online streaming with the high-volume enterprise pipelines will allow PRAXIS to triage recommendations in near real-time during the sprint testing phase, upgrading the capabilities to be interactive in the sprint lifecycle.

References

1. Abreu, R., Zoeteweij, P., Golsteijn, R., & Van Gemund, A. J. C. (2009). A practical evaluation of spectrum-based fault localization. Journal of Systems and Software, 82(11), 1780-1792. Available: https://www.sciencedirect.com/science/article/abs/pii/S0164121209001319
2. Alshahwan, N., Chheda, J., Finogenova, A., Gokkaya, B., Harman, M., Harper, I., Marginean, A., Sengupta, S., & Wang, E. (2024). Automated unit test improvement using large language models at meta. In Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering (pp. 185-196). Available: https://dl.acm.org/doi/abs/10.1145/3663529.3663839
3. Chen, Z., Liu, J., Su, Y., Zhang, H., Wen, X., Ling, X., Yang, Y., & Lyu, M. R. (2021). Graph-based incident aggregation for large-scale online service systems. In 2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE) (pp. 430-442). IEEE. Available: https://ieeexplore.ieee.org/abstract/document/9678746
4. Chillarege, R., Bhandari, I. S., Chaar, J. K., Halliday, M. J., Moebus, D. S., Ray, B. K., & Wong, M. Y. (1992). Orthogonal defect classification-a concept for in-process measurements. IEEE Transactions on Software Engineering, 18(11), 943-956. Available: https://www.researchgate.net/profile/Ram-Chillarege-2/publication/3187512
5. Forsgren, N., Humble, J., & Kim, G. (2018). Accelerate: The science of lean software and devops: Building and scaling high performing technology organizations. IT Revolution. Available: https://itrevolution.com/wp-content/uploads/2022/06/ACC_Audio-Companion.pdf
6. Giamattei, L., Guerriero, A., Malavolta, I., Mascia, C., Pietrantuono, R., & Russo, S. (2024). Identifying performance issues in microservice architectures through causal reasoning. In Proceedings of the 5th ACM/IEEE International Conference on Automation of Software Test (AST 2024) (pp. 149-153). Available: https://dl.acm.org/doi/10.1145/3644032.3644460
7. Hall, T., Beecham, S., Bowes, D., Gray, D., & Counsell, S. (2011). A systematic literature review on fault prediction performance in software engineering. IEEE Transactions on Software Engineering, 38(6), 1276-1304. Available: https://researchrepository.ul.ie/server/api/core/bitstreams/ad9c28b5-341f-4df1-ae50-e743972b3a2e/content

8. Hosseini, S., Turhan, B., & Gunarathna, D. (2017). A systematic literature review and meta-analysis on cross project defect prediction. IEEE Transactions on Software Engineering, 45(2), 111-147. Available: https://ieeexplore.ieee.org/abstract/document/8097045
9. Kallis, R., Di Sorbo, A., Canfora, G., & Panichella, S. (2019). Ticket tagger: Machine learning driven issue classification. In 2019 IEEE International Conference on Software Maintenance and Evolution (ICSME) (pp. 406-409). IEEE. Available: https://ieeexplore.ieee.org/abstract/document/8918993
10. Krasner, H. (2022). The cost of poor software quality in the US: A 2022 report. Consortium for Information & Software Quality. Available: https://www.it-cisq.org/wp-content/uploads/sites/6/2022/11/CPSQ-Report-Nov-22-2.pdf
11. Li, Z., Lu, S., Guo, D., Duan, N., Jannu, S., Jenks, G., Majumder, D., et al. (2022). Automating code review activities by large-scale pre-training. In Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (pp. 1035-1047). Available: https://dl.acm.org/doi/abs/10.1145/3540250.3549081
12. National Institute of Standards and Technology. (2018). Framework for improving critical infrastructure cybersecurity (v1.1). NIST. Available: https://nvlpubs.nist.gov/nistpubs/cswp/nist.cswp.04162018.pdf
13. Notaro, P., Cardoso, J., & Gerndt, M. (2020). A systematic mapping study in AIOps. In International Conference on Service-Oriented Computing (pp. 110-123). Cham: Springer International Publishing. Available: https://link.springer.com/chapter/10.1007/978-3-030-76352-7_15
14. Pearl, J. (2009). Causality. Cambridge University Press. Available: https://bayes.cs.ucla.edu/BOOK-2K/neuberg-review.pdf
15. Soldani, J., & Brogi, A. (2022). Anomaly detection and failure root cause analysis in (micro)service-based cloud applications: A survey. ACM Computing Surveys, 55(3), 1-39. Available: https://dl.acm.org/doi/full/10.1145/3501297
16. U.S. Department of Health and Human Services. (2025). Summary of the HIPAA Privacy Rule. Available: https://www.hhs.gov/hipaa/for-professionals/privacy/laws-regulations/index.html
17. Wang, J., Huang, Y., Chen, C., Liu, Z., Wang, S., & Wang, Q. (2024). Software testing with large language models: Survey, landscape, and vision. IEEE Transactions on Software Engineering, 50(4), 911-936. Available: https://ieeexplore.ieee.org/abstract/document/10440574
18. Xia, C. S., Wei, Y., & Zhang, L. (2023). Automated program repair in the era of large pre-trained language models. In 2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE) (pp. 1482-1494). IEEE. Available: https://ieeexplore.ieee.org/abstract/document/10172803