Journal of Mainframe Computing and Legacy System Modernization

ORACLE APEX APPLICATION FOR MANAGING INTENSIVE CARE UNIT RECORDS, ALERTS, AND CLINICAL OBSERVATIONS

Dylan Baker
United States

ABSTRACT

Oracle APEX applications support intensive care unit record management, clinical alerts, and continuous patient observation through a centralized web-based platform. The system records vital signs, medications, laboratory results, ventilator settings, fluid balance, nursing notes, diagnoses, and treatment updates. Real-time dashboards allow doctors and nurses to monitor patient condition and review changes across ICU beds. Automated alerts can identify abnormal heart rate, blood pressure, oxygen saturation, temperature, laboratory values, or delayed clinical tasks requiring immediate attention. Shift-reporting tools improve communication between critical care teams and reduce information loss during handovers. Integration with Oracle databases enables secure storage, rapid retrieval, reliable reporting, and accurate record management. Overall, the application can improve ICU documentation, clinical coordination, early risk recognition, patient safety, and critical care decision making.

keywords: Oracle APEX; Intensive Care Unit; Clinical Alerts; Patient Monitoring; ICU Record Management.

1. Introduction

Enterprise applications depend on reliable diagnostics to detect runtime errors, failed workflows, API issues, slow transactions, and integration faults [1-2]. Logs are the main evidence source for understanding what happened before, during, and after an incident [3]. Effective application diagnostics require structured logging, traceability, contextual metadata, error classification, and operational visibility [4-5]. However, fixed logging strategies often create two problems. Low-detail logs may miss important causes [6], while high-detail logs may generate excessive storage cost and alert noise [7].

The main challenge is that diagnostic needs change during runtime [8]. A normal transaction may require only basic logs, but a failing workflow, repeated exception, or security-sensitive event may need deeper traces [9-10]. Adaptive logging improves this process by changing log depth according to system condition, transaction risk, and error behavior [11-13]. AI-assisted monitoring can identify when additional diagnostic detail is needed and when logging can return to normal levels [14-15]. This article proposes an adaptive logging framework for enterprise application diagnostics.

2. Methodology

The proposed framework captures log metadata from application servers, API gateways, databases, background jobs, authentication modules, and integration services [16]. It records event type, severity, timestamp, user role, transaction ID, request path, error frequency, response time, affected component, and workflow state [17-18]. These features are converted into diagnostic context profiles [19]. Adaptive logging requires context because the same event may be low-risk in one workflow but critical in another [20].

The logging controller classifies runtime states as normal, warning, error-prone, performance-risk, security-sensitive, workflow-critical, or incident-active [21]. Based on this classification, it adjusts log level, trace depth, parameter capture, correlation ID tracking, and diagnostic enrichment. Lightweight scoring and policy rules are used so logging decisions remain fast and explainable [22]. Sensitive fields are masked before storage, and transaction-critical events are linked with workflow and database identifiers for easier diagnosis.

The framework is evaluated using simulated enterprise applications involving APIs, database workflows, authentication events, scheduled jobs, and service integrations. Static logging is compared with adaptive logging under repeated exceptions, slow requests, failed jobs, API timeout, suspicious login, and workflow interruption scenarios [23]. Evaluation metrics include diagnosis time, log volume reduction, root-cause evidence quality, missed-event rate, storage overhead, alert usefulness, and administrator acceptance rate. Feedback from resolved incidents, ignored logs, and administrator notes is used to refine logging policies over time.

3. Results and Discussion

The results show that adaptive logging improves diagnostics by capturing richer evidence only when runtime risk increases. In the baseline condition, static logs either lack enough detail or produce large volumes of low-value records. With the proposed framework, detailed traces are activated for high-risk events and reduced during normal execution. The strongest improvement appears in intermittent failures, where deeper logs are needed only around the failure window.

The discussion shows that adaptive logging must balance diagnostic depth with privacy and storage cost. Excessive logging can expose sensitive data or overload monitoring systems, while weak logging may hide failure causes. The main limitation is that log-level decisions depend on accurate runtime classification. Regular policy review and masking controls are necessary for safe long-term use.

4. Conclusion

This article presented an adaptive logging framework for enterprise application diagnostics. The framework adjusts log detail using runtime context, risk scoring, workflow state, and error behavior. It improves troubleshooting by increasing diagnostic evidence during abnormal conditions while reducing unnecessary log noise during normal operation. The study shows that adaptive logging can strengthen enterprise diagnostics when combined with structured metadata, privacy controls, and continuous incident feedback.

References

1. Kavuluri, V. V. R., Bandla, S., Gorumutchu, M. R., Mandapatti, J. K., & Jagadhabi, N. (2022). Execution Trace Forensics in Low-Code Platforms for Critical Workflows. Journal of Grid, Machines and Drives, 9, 1-8.
2. Bandla, S., Kavuluri, V. V. R., Gorumutchu, M. R., Jagadhabi, N., & Mandapatti, J. K. (2021). Memory Pressure Amplification in Declarative Runtimes. Transactions on Smart Electrical and Computational Systems, 8, 39-45.
3. Jagadhabi, N., Gorumutchu, M. R., Bandla, S., Mandapatti, J. K., & Kavuluri, V. V. R. (2023). Control Plane Saturation in Metadata-Driven Platforms. Journal of Privacy-Aware Computing Systems, 10, 36-43.
4. Kavuluri, H. V. R. (2019). Bug Severity Classification in Issue Tracking Systems with Support Vector Machines. High-Performance Distributed Computing Review, 6, 7-12.
5. Kavuluri, H. V. R. (2023). Change Data Capture for High-Volume Transaction Systems with Consistency Validation. Emerging Digital Systems, 10, 1-8.

6. Bandla, S., Kavuluri, V. V. R., Jagadhabi, N., Gorumutchu, M. R., & Mandapatti, J. K. (2023). Operational Risk Surfaces of AIIntegrated Database Frameworks. Cross-Disciplinary Knowledge Systems, 10, 1-8.
7. Kande, R., Kotla, C., & kanth Thottempudi, K. (2024). Trustworthy Data Intelligence: Neuro-Symbolic Verification for Provenance Reasoning, Confidence Propagation, and Trust Scoring Across Enterprise Analytics Ecosystems. Perception, Navigation and Intelligent Devices, 11, 1-8.
8. Kavuluri, H. V. R. (2019). Artificial Neural Network-Based Software Effort Estimation with Project Metrics. Journal of Grid, Machines and Drives, 6, 1-6.
9. Kavuluri, H. V. R. (2021). Test Case Prioritization with Genetic Algorithms for Regression Testing. Transactions on Smart Electrical and Computational Systems, 8, 1-6.
10. Kavuluri, H. V. R. (2020). Feature Selection for Machine Learning Fault Prediction in Software Modules. Journal of Privacy-Aware Computing Systems, 7, 7-12.
11. Kavuluri, H. V. R. (2022). Adaptive Storage Tiering for Mixed Analytical and Operational Workloads in Data Engineering. High-Performance Distributed Computing Review, 9, 9-16.
12. Kavuluri, H. V. R. (2020). Software Reliability Prediction with Weibull Failure Models. Emerging Digital Systems, 7, 7-12.
13. Jagadhabi, N., Mandapatti, J. K., Bandla, S., Kavuluri, V. V. R., & Gorumutchu, M. R. (2021). Cross-Layer Dependency Inflation in Model-Augmented Engineering Stacks. Cross-Disciplinary Knowledge Systems, 8, 32-38.
14. Kavuluri, H. V. R. (2021). Source Code Complexity Assessment with Static Metrics and Maintainability Index. Perception, Navigation and Intelligent Devices, 8, 1-6.
15. Kavuluri, H. V. R. (2021). Evaluation of Supervised Machine Learning for Software Defect Detection. Journal of Grid, Machines and Drives, 8, 7-11.
16. Kavuluri, V. V. R., Gorumutchu, M. R., Bandla, S., Jagadhabi, N., & Mandapatti, J. K. (2024). Query Plan Instability Patterns in Self-Adaptive Optimizers. Transactions on Smart Electrical and Computational Systems, 11, 31-36.
17. Mandapatti, J. K., Gorumutchu, M. R., Kavuluri, V. V. R., Jagadhabi, N., & Bandla, S. (2022). Causal Inference Failures in Machine-Learning-Driven Database Tuning. Journal of Privacy-Aware Computing Systems, 9, 1-7.
18. Kavuluri, V. V. R., Gorumutchu, M. R., Jagadhabi, N., Mandapatti, J. K., & Bandla, S. (2021). Schema Volatility Propagation in AI-Driven Data Architecture Pipelines. High-Performance Distributed Computing Review, 8, 1-7.
19. Kande, R., Kotla, C., kanth Thottempudi, K., Anjuru, P., & Nithyanandam, S. (2021). Adaptive SOAR Using Deep Reinforcement Learning for Autonomous Threat Detection in Cloud-Native Architectures. Emerging Digital Systems, 8, 31-38.

20. Kande, R., kanth Thottempudi, K., & Kotla, C. (2021). Autonomous DevSecOps: A Quantitative Maturity Model and ML-Driven Security Orchestration for Continuous Compliance. Cross-Disciplinary Knowledge Systems, 8, 1-8.
21. Kotla, C., kanth Thottempudi, K., & Kande, R. (2022). Software Supply Chain Security for Infrastructure-as-Code Pipelines with Vulnerability Detection and Remediation. Perception, Navigation and Intelligent Devices, 9, 29-36.
22. Gorumutchu, M. R., Mandapatti, J. K., Kavuluri, V. V. R., Jagadhabi, N., & Bandla, S. (2023). Deterministic Execution Models for Platforms Under Enterprise Transactional Load. Journal of Grid, Machines and Drives, 10, 32-38.
23. Kavuluri, H. V. R. (2019). Defect Prediction in Object-Oriented Software with Decision Tree Classifiers. Transactions on Smart Electrical and Computational Systems, 6, 6-11.