International Journal of Emerging Research in Engineering and Technology
Pearl Blue Research Group| Volume 5, Issue 4, 160-164, 2024
ISSN: 3050-922X | https://doi.org/10.63282/3050-922X.IJERET-V5I4P116

Original Article

Decision Intelligence for AI-Driven Agile Lifecycle Governance: Linking Architecture-Centered Management to Defect Risk Forecasting

Aarav Mehta[1], Riya Sharma[2], Kunal Verma[3], Sneha Iyer[4]
[1,3]Computer Science Department, Vellore Institute of Technology, Vellore, Tamil Nadu, India.
[2,4]Artificial Intelligence Department, Vellore Institute of Technology, Vellore, Tamil Nadu, India.

Abstract *- Agile delivery improves responsiveness but can amplify architectural drift, defect leakage, and governance ambiguity when quality controls are not explicitly linked to measurable risk signals. In practice, many teams implement defect prediction or operational analytics as standalone dashboards rather than as decision drivers for sprint planning, release gating, and architecture investment. This paper presents DI-AIGLG, a decision intelligence-based governance framework that links architecture-centered management with defect risk forecasting and observability evidence to produce repeatable, auditable, and low-overhead governance actions across the agile lifecycle. DI-AIGLG formalizes governance as a closed-loop system consisting of evidence capture, predictive risk modeling, decision policies, and outcome learning. We define an architecture exposure score, propose a calibrated risk score that combines predicted defect likelihood with architectural blast radius, and map risk to actionable controls spanning backlog selection, CI/CD gating, and release readiness. A worked example demonstrates risk budgeted sprint planning under capacity constraints. The framework is designed to be implementable using CI/CD telemetry, automated testing signals, and AI-enhanced observability pipelines.*

Keywords *- Decision Intelligence, Agile Governance, Architecture Centered Management, Defect Prediction, Risk Forecasting, SRE, Observability, CI/CD, Explainable AI.*

1. Introduction

Agile frameworks emphasize iterative delivery, customer collaboration, and responsiveness to change [1]. Scrum operationalizes these principles through recurring decision points such as sprint planning, sprint review, and retrospectives [2]. As systems scale into distributed cloud native architectures, governance challenges increase: architectural dependencies deepen, failure modes become non-local, and regulatory requirements demand traceability. DevOps research shows that performance depends on tight feedback loops and disciplined engineering practices rather than heavyweight approvals [3]. This paper argues that the governance problem is fundamentally a decision quality problem: teams need a repeatable way to connect architecture evidence and delivery telemetry to concrete lifecycle decisions.

2. Background and Related Work

2.1. Architecture Evidence and Architecture-Centered Management

Architecture centered management treats architecture as a management instrument for controlling complexity, integration risk, and long-term evolution. A practical prerequisite is a consistent architecture description, including viewpoints, concerns, and key architectural decisions [4]. Widely adopted guidance on applying architecture in practice emphasizes component boundaries, interfaces, and dependency management as core levers for quality outcomes [5].

2.2. Reliability Governance and SRE Principles

Reliability governance requires proactive controls that are measurable and sustainable at delivery speed. Integrating Site Reliability Engineering (SRE) principles into enterprise architecture supports predictive resilience by making reliability targets explicit and by treating risk as a first-class planning constraint [6].

2.3. Observability as Decision Evidence

In cloud systems, observability and automated root cause analysis can convert operational telemetry into structured evidence that informs delivery decisions. AI-enhanced observability architectures enable faster diagnosis and, importantly for governance, can surface recurring failure patterns and high exposure areas for preventive investment [7].

2.4. Automated Testing and Quality Signals

Automated testing remains a primary mechanism for reducing defect leakage. Comparative analysis of automated testing frameworks for Java enterprise systems highlights how test strategy choices influence reliability, cycle time, and defect escape risk [8].

2.5. Explainability and Auditability in Regulated Decision Systems

Governance in regulated settings requires explainable and auditable decision pathways. Regulatory-grade fraud detection studies show that explainable AI and auditable pathways are essential to justify automation, support compliance, and improve stakeholder trust [9].

2.6. Privacy Preserving Learning and Operational Governance

When learning across organizational boundaries, data sharing constraints can block centralized modeling. Federated learning architectures provide privacy-preserving training while relying on governance protocols to manage participation, auditing, and operational controls [10].

2.7. Decision Intelligence for Agile Lifecycle Governance

Decision Intelligence (DI) treats governance as an engineered system of decisions: each decision has explicit inputs, options, constraints, and measurable outcomes. A DI methodology for AI-driven agile software lifecycle governance links architecture-centered project management to decision modeling and continuous outcome learning [11].

2.8. Compliance Driven Architecture Controls

Secure microservices architectures in healthcare illustrate how governance constraints can be embedded at design time and deployment time. A HIPAA-compliant microservices architecture for prescription processing demonstrates boundary controls, security design, and platform-aware deployment considerations [12].

2.9. ML-Enabled Software Engineering and Decision Loops

ML increasingly supports software engineering tasks such as risk prediction, automation, and decision support. Prior work on the expanding role of ML models motivates integrating predictive outputs into lifecycle decisions rather than isolating them as dashboards [13].

2.10. ML Driven Risk Detection in CI/CD

To remain low overhead, governance actions should be enforced through automated pipelines. Intelligent CI/CD for banking systems shows how ML-driven risk detection can be integrated into continuous delivery with deployment evaluation in real environments [14].

2.11. Cloud Native Automation and Operationalization

Cloud native automation examples demonstrate the feasibility of embedding ML outputs into operational workflows. An AI-driven fax-to-digital prescription automation framework illustrates OCR and ML integrated microservices orchestration patterns [15].

2.12. Fault and Defect Prediction Studies

Comparative studies on fault prediction indicate that classical models such as random forests, logistic regression, and nearest neighbors can be effective depending on feature quality and context [16]. Predictive analytics also demonstrates operational benefit when integrated with systems for inventory or fulfillment, reinforcing the need to operationalize predictions rather than report them [17]. Further comparative analysis of machine learning models for software defect prediction motivates selecting algorithms based on empirical performance and deployment constraints [18].

3. Problem Statement and Design Goals

Many agile programs lack a repeatable, architecture-aware method to translate risk signals into concrete governance actions. As a result, teams either under-govern and accumulate architectural drift and defect leakage, or over-govern and slow delivery. The objective of DI-AIGLG is to enable governance that is fast, explainable, and evidence-driven. Design goals are: (G1) low overhead integration with sprint and pipeline workflows, (G2) architecture awareness through dependency and blast radius signals, (G3) risk-informed planning and gating based on calibrated forecasts, (G4) explainability and auditability for regulated decision contexts, and (G5) continuous learning through outcome feedback.

4. DI-AIGLG Framework

DI-AIGLG implements governance as a closed-loop decision system with five stages: Sense, Model, Decide, Act, and Learn. Sense collects architecture evidence, delivers telemetry, and operational signals. Model produces calibrated risk estimates and architecture exposure scores. Decide applies policies that map evidence to actions under constraints. Act enforces actions across sprint planning, code review, testing, and release. Learn measures outcomes and updates both models and decision thresholds.

Table 1: Decision Points and Primary Evidence Signals

Decision Point	Primary Evidence	Typical Action
Sprint portfolio selection	Backlog value and effort, risk forecasts, and architecture exposure	Risk-budgeted sprint scope
Change review intensity	Change size, hotspot, prior defects, test health	Add reviewer, require additional tests
CI/CD gate strictness	Pipeline volatility, risk tier, security checks	Block, stage, or allow with guardrails
Release readiness	Recent incidents, error rates, RCA signals, and dependency exposure	Canary rollout, hold release
Architecture investment	Hotspots, coupling growth, recurring RCA themes	Refactor, decouple, add SLOs

5. Linking Architecture Exposure to Defect Risk

To connect architecture-centered management to defect risk forecasting, DI-AIGLG defines an Architecture Exposure Score (AES) for a component or change. AES captures hotspot intensity, coupling and dependency blast radius, business criticality, and volatility. Defect risk forecasting produces a calibrated probability estimate $P(defect|x)$ from change and component features. DI-AIGLG combines these into a decision-ready risk score: $Risk(i) = P(defect|i) \times Impact(i) \times AES(i)$. Impact reflects business and compliance sensitivity, particularly for regulated workflows.

6. Decision Policies And Governance Actions

DI-AIGLG maps risk scores to actionable controls. For sprint planning, teams can set a risk budget per sprint and select backlog items under both capacity and risk constraints. For CI/CD, automated gates can intensify checks for higher risk tiers, require staged rollout, or trigger architecture review. For release readiness, observability, and RCA signals can act as stop or proceed evidence when recent operational instability overlaps with high exposure components.

7. Implementation Blueprint

A practical implementation integrates (1) data ingestion from version control, code review, CI/CD pipelines, test systems, and issue trackers, (2) telemetry ingestion from observability stacks, (3) feature extraction and storage, (4) model training, calibration, and drift monitoring, and (5) a governance policy engine that computes AES, risk tiers, and recommended actions. The system should emit an audit trail: evidence summaries, risk drivers, thresholds applied, and the resulting action.

8. Worked Example: Risk Budgeted Sprint Selection

Assume a sprint capacity of 20 story points and a backlog of candidate items with estimated value, effort, and risk. A value-only plan may select multiple high-exposure items touching the same hotspot component, increasing expected defect leakage. A risk budgeted plan replaces one high-risk item with a lower-risk alternative while preserving most value. This makes tradeoffs explicit and reduces surprise defects without adding heavy governance overhead.

Algorithm 1
Risk-Budgeted Sprint Selection (Greedy Heuristic)
Input: backlog items i with Value(i), Effort(i), Risk(i); Capacity C; RiskBudget Rmax
Output: selected set S
1: $S \leftarrow \emptyset$; usedEffort $\leftarrow$ 0; usedRisk $\leftarrow$ 0
2: Sort items by Score(i) = Value(i) / (Effort(i) + ε) descending
3: For each item i in sorted order:
4: if usedEffort + Effort(i) $\leq$ C and usedRisk + Risk(i) $\leq$ Rmax then
5: $S \leftarrow S \cup \{i\}$
6: usedEffort $\leftarrow$ usedEffort + Effort(i)
7: usedRisk $\leftarrow$ usedRisk + Risk(i)
8: return S

9. Evaluation Plan

Evaluation should cover both model quality and governance outcomes. For forecasting, measure discrimination and calibration, and track drift over time. For governance, measure escaped defects, change failure rate, incident recurrence in hotspots, and delivery throughput stability. In regulated contexts, evaluate explanation fidelity and audit completeness.

10. Threats to Validity

Threats include label noise in defect datasets, concept drift as systems evolve, intervention effects where governance changes developer behavior, and limited generalizability across domains. Risk thresholds and AES weights typically require tuning per organization.

11. Conclusion

DI-AIGLG links architecture-centered management to defect risk forecasting to enable low overhead, explainable, and evidence-driven agile lifecycle governance. The framework converts telemetry and predictions into concrete actions: risk budgeted planning, pipeline gating, and observability-informed release readiness. Future work includes large-scale empirical validation across multi-team programs and deeper automation of mitigation recommendations.

References

[1] "Manifesto for Agile Software Development," 2001. https://www.geeksforgeeks.org/software-engineering/agile-manifesto-for-software-development/

[2] K. Schwaber and J. Sutherland, "The Scrum Guide," 2020. https://www.scrum.org/resources/scrum-guide

[3] Nicole Forsgren, Jez Humble, and Gene Kim, "Accelerate: The Science of Lean Software and DevOps," *IT Revolution*, 2018. https://dl.acm.org/doi/book/10.5555/3235404

[4] ISO/IEC/IEEE 42010:2011, "Systems and Software Engineering—Architecture Description," 2011. https://ieeexplore.ieee.org/document/6129467

[5] Len Bass, Paul Clements, and Rick Kazman, "Software Architecture in Practice," 4th ed., Addison-Wesley, 2021. https://www.oreilly.com/library/view/software-architecture-in/9780136885979/

[6] Siva Kantha Rao Vanama, "Integrating Site Reliability Engineering (SRE) Principles into Enterprise Architecture for Predictive Resilience," *IJETCSIT*, vol. 4, no. 3, pp. 164–170, 2023. https://www.ijetcsit.org/index.php/ijetcsit/article/view/514

[7] Idrasena Manga, Sai Dheeraj Sivva, and Vamshi Krishna Manga, "The Adaptive Intelligence in Cloud Systems: A Unified Architecture for AI Enhanced Observability and Automated Root Cause Analysis", *IJAIDSML*, vol. 5, no. 1, pp. 160–166, 2024, https://ijaidsml.org/index.php/ijaidsml/article/view/366

[8] Srikanth Reddy Gudi, "Enhancing Reliability in Java Enterprise Systems through Comparative Analysis of Automated Testing Frameworks," *International Journal of Emerging Trends in Computer Science and Information Technology*, vol. 4, no. 2, pp. 151–160, 2023. https://www.ijetcsit.org/index.php/ijetcsit/article/view/476

[9] Sai Santhosh Goud Bandari, Sai Dheeraj Sivva, and Rakesh Reddy Thalakanti, "Regulatory Grade Fraud Detection using Explainable Artificial Intelligence with Auditable Decision Pathways and Empirical Validation on Banking Data," *International Journal of Artificial Intelligence, Data Science, and Machine Learning*, vol. 5, no. 3, pp. 139–147, 2024. https://ijaidsml.org/index.php/ijaidsml/article/view/367

[10] Rakesh Reddy Thalakanti, Sai Santhosh Goud Bandari, and Sai Dheeraj Sivva, "Federated Learning for Privacy Preserving Fraud Detection across Financial Institutions: Architecture Protocols and Operational Governance," *International Journal of Emerging Research in Engineering and Technology*, vol. 5, no. 2, pp. 108–114, 2024. https://ijeret.org/index.php/ijeret/article/view/394

[11] Sai Krishna Gunda et al., "Decision Intelligence Methodology for AI Driven Agile Software Lifecycle Governance and Architecture Centered Project Management," 2023. https://ijaidsml.org/index.php/ijaidsml/article/view/301

[12] Srikanth Reddy Gudi, "Design and Evaluation of Secure Microservices Architecture for HIPAA Compliant Prescription Processing on AWS and OpenShift," *International Journal of Artificial Intelligence, Data Science, and Machine Learning*, vol. 5, no. 2, pp. 144–149, 2024. https://scholar.google.com/scholar?hl=en&as_sdt=0%2C5&q=Design+and+Evaluation+of+Secure+Microservices+Architecture+for+HIPAA+Compliant+Prescription+Processing+on+AWS+and+OpenShift&btnG=

[13] Sai Krishna Gunda, "The Future of Software Development and the Expanding Role of ML Models," *International Journal of Emerging Research in Engineering and Technology*, vol. 4, no. 2, pp. 126–129, 2023. https://ijeret.org/index.php/ijeret/article/view/347

[14] Rakesh Reddy Thalakanti and Sai Santhosh Goud Bandari, "Intelligent Continuous Integration and Delivery for Banking Systems using Machine Learning Driven Risk Detection with Real World Deployment Evaluation," *International Journal of AI, BigData, Computational and Management Studies*, vol. 5, no. 4, pp. 168–175, 2024. https://ijaibdcms.org/index.php/ijaibdcms/article/view/335

[15] Srikanth Reddy Gudi, "AI Driven Fax to Digital Prescription Automation: A Cloud Native Framework Using OCR, Machine Learning, and Microservices for Pharmacy Operations," *International Journal of Emerging Research in Engineering and Technology*, vol. 5, no. 1, pp. 111–116, 2024. https://ijeret.org/index.php/ijeret/article/view/358

[16] Sai Krishna Gunda, "Fault Prediction Unveiled: Analyzing the Effectiveness of Random Forest, Logistic Regression, and K Neighbors," *in Proc. 2nd Int. Conf. on Self Sustainable Artificial Intelligence Systems (ICSSAS)*, pp. 107–113, 2024. https://ieeexplore.ieee.org/abstract/document/10760620

[17] Srikanth Reddy Gudi, "Leveraging Predictive Analytics and Redis Backed Caching to Optimize Specialty Medication Fulfillment and Pharmacy Inventory Management," *International Journal of AI, BigData, Computational and*

Management Studies, vol. 5, no. 3, pp. 155–160, 2024. https://ijaibdcms.org/index.php/ijaibdcms/article/view/327https://ijaibdcms.org/index.php/ijaibdcms/article/view/327

[18] Sai Krishna Gunda, "Comparative Analysis of Machine Learning Models for Software Defect Prediction," *in Proc. Int. Conf. on Power, Energy, Control and Transmission Systems (ICPECTS)*, pp. 1–6, 2024. https://ieeexplore.ieee.org/abstract/document/10780167

[19] Sai Dheeraj Sivva et al., "AI-Driven Decision Intelligence for Agile Software Lifecycle Governance: An Architecture-Centered Framework Integrating Machine Learning Defect Prediction and Automated Testing," *IJETCSIT*, vol. 4, no. 4, pp. 167-72, 2023s. https://www.ijetcsit.org/index.php/ijetcsit/article/view/554