# AI-Based Architecture: A Comparative Study in Software Engineering and Financial Services

Establishing, Implementing, and Measuring Value across Two Industry Domains

Oliver Bodemer
e-Softworks, Inc.

September 21, 2026

**Abstract**

**Purpose:** This paper examines the design, implementation, and measurable outcomes of AI-Based Architecture — a paradigm in which autonomous agents, decision engines, and self-healing mechanisms are embedded directly into system architecture. We present a comparative case study across two industry domains — software development and banking — to identify common patterns, domain-specific adaptations, and quantifiable performance differences.

**Methodology:** We analyse two real-world implementations of AI-Based Architecture, documenting their architectural patterns, organizational changes, process transformations, and key performance indicators (KPIs). Findings are structured around four dimensions: (1) process impact, (2) team and role evolution, (3) measurable KPI improvements, and (4) regulatory and governance constraints.

**Expected Contributions:** This paper provides (a) a reference architecture pattern for AI-Based Architecture, (b) a maturity model (levels 1–5), (c) a comparative analysis revealing how regulatory density moderates adoption speed and KPI outcomes, and (d) a practical KPI framework for measuring value realization.

**Keywords:** AI-Based Architecture; autonomous systems; software engineering; banking; digital transformation; process optimization; KPIs; SOX; GLBA; model risk management; US regulatory framework

# 1 Introduction

## 1.1 Motivation and Problem Statement

The architecture of information systems has traditionally been a human-centric discipline. System designs are conceived by architects, implemented by engineers, maintained by operators, and debugged through incident response and manual root-cause analysis. This human-in-the-loop model has served the industry well for decades, but it imposes hard constraints on speed, scale, and adaptability. As systems grow in complexity and the rate of change accelerates, the gap between human cognitive capacity and system demands widens. Organizations that continue to rely solely on human-designed, human-maintained, and human-debugged architectures find themselves unable to keep pace with competitive pressure, regulatory complexity, and operational expectations.

AI-Based Architecture represents a paradigm shift along this dimension. In this model, autonomous agents, decision engines, prediction models, and self-healing mechanisms are embedded as first-class architectural elements—not bolt-on tools—and they govern at least one lifecycle dimension: design, operation, optimization, or recovery. Unlike conventional automation, which executes predefined rules within narrow boundaries, AI-capable components can adapt their behaviour based on feedback, detect patterns beyond human-specified thresholds, and execute corrective actions without human initiation. The result is a system that does not merely run faster but makes its own decisions, heals from failures before operators are alerted, and reconfigures itself in response to shifting conditions.

Despite the promise of this paradigm, most organizations remain at early stages of adoption. Industry assessments suggest the majority of enterprises operating AI systems are at maturity Level 1 (reactive assistance, such as chatbots or recommendations) or Level 2 (augmented intelligence, where AI suggests and humans decide). A small minority has reached Level 3 (automated execution with oversight), and autonomous operation (Levels 4–5) remains largely theoretical or confined to narrow research settings. This paper addresses how AI-capable components can be established, measured, and governed in practice across different domains – not as a mandate to adopt everywhere, but as guidance for organizations that have identified a bottleneck expensive enough to justify autonomy.

To understand the landscape of adoption and value realization, this paper compares two industry domains that sit at opposite ends of the regulatory spectrum. Software development is agile, fast-moving, and characterized by relatively low regulatory friction. Teams adopt new tools and processes based on competitive pressure and internal metrics. Banking, by contrast, is highly regulated, risk-averse, and subject to a dense web of compliance requirements including the Sarbanes-Oxley Act (SOX), the Gramm-Leach-Bliley Act (GLBA), Federal Reserve SR 11-7 guidance on model risk management, FFIEC exam criteria, and the Consumer Financial Protection Bureau (CFPB) guidelines on third-party risk and consumer protection. These constraints slow adoption and impose additional governance overhead, but they also mean that legacy processes are often more wasteful and error-prone—creating greater potential for improvement when AI is applied correctly. The EU regulatory landscape (NIS2, DORA, the AI Act) is treated separately in Section **??**.

By examining these two domains side by side, this study identifies (a) architectural patterns that are universal across domains and (b) adaptations that are domain-specific, driven by regulatory density, risk tolerance, and the nature of the processes being transformed. Together, software development and banking span the adoption spectrum that most organizations will encounter, making their comparison both practically relevant and analytically valuable.

## Principle. Use AI where it is useful and where it helps the business.

This study is not an argument for embedding autonomous agents in every process. It is an argument for embedding them at identified operational bottlenecks – with explicit decision rights, reversible actions, and continuous governance. Company A's bottleneck was release velocity and regression cost. Bank B's bottleneck was loan cycle time, AML

noise, and fraud detection. Those are the sites of the reported gains. Processes that are rare, cheap, or already reliable were left alone.

The research question is not "Does AI-Based Architecture work?" It is "Under what process conditions does embedding governed agents improve measurable business outcomes, and where does it not apply?"

## 1.2 Scope and Research Questions

This paper is concerned with a single construct: AI-Based Architecture, understood as the embedding of autonomous agents, decision engines, and self-healing mechanisms into the structural design of information systems. We restrict our scope to implementations where AI-capable components exercise at least some decision authority over system behaviour — whether in designing new code, selecting which tests to execute, rerouting traffic during an outage, or adjusting credit thresholds in response to emerging risk patterns. Systems that merely assist human decision-makers through recommendation or classification, without exercising autonomous agency, fall outside this definition and are mentioned only as a baseline for comparison.

The study is bounded to two industry domains, each with one primary case, selected because they represent opposite ends of the regulatory spectrum. Software development is characterized by market-driven adoption cycles, low regulatory friction, and a culture of rapid iteration. Banking is characterized by prescriptive regulation, high risk aversion, and processes that must be explainable and auditable to supervisory authorities. Both domains share the same underlying technological possibilities — generative AI for code and documentation, machine learning for anomaly detection and prediction, autonomous agents for orchestration — but the regulatory and organizational contexts shape which of those possibilities are pursued, how they are governed, and what metrics are used to evaluate them.

This bounded scope yields five research questions that structure the remainder of the paper.

**RQ1.** *What are the core architectural components of AI-Based Architecture, and how can a reference model be structured to capture the layers from data and knowledge through autonomous execution to governance?* This question is answered in Section 2, which proposes a five-layer reference architecture, a maturity model (Levels 1–5), and a set of core principles that govern the design of AI-capable systems.

**RQ2.** *How does AI-Based Architecture transform processes across the software development lifecycle — from requirements and design through implementation, testing, deployment, and operations — and what new capabilities emerge at each phase?* This question is answered in Section 3, which documents the before-and-after state of a software development organization that adopted AI-Based Architecture, measuring process-level and team-level impacts.

**RQ3.** *How does AI-Based Architecture transform core banking functions — loan processing, risk assessment, compliance, and fraud detection — under the constraints of a dense regulatory environment, and where does regulatory compliance constrain or shape the design choices?* This question is answered in Section 4, which documents the before-and-after state of a banking organization and examines the interplay between AI autonomy and regulatory requirements.

**RQ4.** *What KPIs demonstrate measurable value from AI-Based Architecture, and how*

*do those metrics differ across domains in terms of magnitude, timeline, and the ease of attribution?* This question is answered in Section 5, which presents a side-by-side comparison of KPI outcomes across the two cases, and in Section 6, which proposes a structured KPI framework for measuring value realization in any domain.

**RQ5.** *What organizational and governance factors enable or inhibit the adoption of AI-Based Architecture, and how does regulatory density moderate the balance between autonomy and oversight?* This question is answered throughout the case studies and synthesized in Section 7, which discusses challenges, risks, and the role of change management, model governance, and human-in-the-loop design.

## 1.3 Contributions

This paper makes five contributions that address the gaps identified in Section 1.1.

**First,** we propose a *reference architecture pattern* for AI-Based Architecture that generalizes across domains. The pattern consists of five layers — data and knowledge, AI model and agent, decision and policy, automation and execution, and observability and governance — each with clearly defined responsibilities, interfaces, and failure modes. Unlike previous architectures that treat AI as an add-on to existing systems, our model positions AI-capable components as first-class architectural elements that govern the full system lifecycle. This pattern is presented in Section 2 and illustrated in Figure 1.

**Second,** we introduce a *five-level maturity model* that classifies AI-Based Architecture implementations by their degree of autonomy and the nature of human oversight. Levels range from reactive assistance (Level 1) through augmented intelligence (Level 2) and automated execution (Level 3), to adaptive self-tuning (Level 4) and fully autonomous operation within governance boundaries (Level 5). The model provides a common vocabulary for practitioners to assess their current position, set targets, and understand the organizational prerequisites for moving to the next level. It also enables cross-domain comparison by mapping software development and banking cases onto a shared axis.

**Third,** we present a *comparative case study* across software development and banking that documents the process transformations, team reorganizations, and KPI improvements achieved in each domain. The comparison reveals two findings that would not emerge from a single-domain study: (a) regulatory density slows the speed of adoption but does not diminish the magnitude of eventual value, and (b) the architectural patterns that drive value are largely domain-independent, while the governance mechanisms required to deploy them are domain-specific.

**Fourth,** we propose a *structured KPI framework* that categorizes metrics into productivity, cost, quality, and adoption dimensions. Each metric is defined with its measurement methodology, the data sources required, and the caveats that must be considered when attributing changes to AI-Based Architecture. The framework is designed to be domain-agnostic — organizations can adopt the metrics that are relevant to their context without implementing every element — while maintaining cross-domain comparability through a shared set of core measures.

**Fifth,** we discuss the *organizational and governance prerequisites* for successful adoption, including the evolution of team roles, the establishment of AI governance structures, and the design of human-in-the-loop protocols. We identify five common pitfalls — over-optimism, under-investment in change management, neglect of data quality, absence of governance, and tool sprawl — and provide mitigations for each. This discussion is

intended to bridge the gap between the technical possibility of AI-Based Architecture and the organizational work required to realize it in practice.

### 1.4 Paper Structure

The remainder of this paper is organized as follows. Section 2 establishes the theoretical foundations, presenting a definition of AI-Based Architecture, its core principles, a five-layer reference model, and a five-level maturity model. Section 3 documents the software development case study, covering the organizational context, process transformations across the SDLC, team role evolution, and KPI outcomes. Section 4 presents the banking case study in parallel structure, with additional emphasis on regulatory constraints and their impact on design and deployment. Section 5 synthesizes the two cases into a comparative analysis, examining how regulatory density moderates adoption speed, automation depth, and KPI trajectories. Section 6 proposes the structured KPI framework for measuring value realization across domains. Section 7 interprets the findings, discusses challenges and risks, outlines future research directions, and acknowledges the limitations of this study. Section 8 summarizes the key contributions and offers recommendations for organizations considering adoption. The appendices provide supplementary material, including detailed architecture diagrams, a maturity model assessment questionnaire, KPI calculation formulas, and a regulatory framework comparison table.

## 2 AI-Based Architecture: Foundations and Reference Model

### 2.1 Definition and Core Principles

We define AI-Based Architecture as a system design paradigm in which AI-capable components — autonomous agents, decision engines, prediction models, and self-healing controllers — are first-class architectural elements that govern at least one lifecycle dimension: design, operation, optimization, or recovery. This definition is deliberately broader than the conventional notion of *AI-assisted systems*, in which human operators remain the sole decision-makers and AI merely surfaces information or suggests actions. In an AI-Based Architecture, the boundary between "human decides, machine executes" and "machine decides within governed boundaries" is crossed for at least one function, and the system is designed to tolerate the implications of that delegation.

Three clarifications are necessary. First, "AI-capable" does not imply a single technology. The term encompasses large language models used for code generation and requirements analysis, reinforcement learning agents that optimize deployment pipelines, anomaly detection models that trigger automatic incident response, and multi-agent orchestration frameworks that coordinate specialized components across system boundaries. What unites these technologies is not their internal mechanics but their architectural role: they exercise decision authority that was previously held exclusively by humans.

Second, the term "first-class architectural element" means that AI-capable components are designed into the system from inception, with explicit interfaces, failure modes, and governance mechanisms. They are not bolted on to an existing architecture as an afterthought, nor are they contained in isolated sandboxes that do not interact with the broader system. Their inputs, outputs, and side-effects are part of the architectural spec-

ification; they must be accounted for in the same way that a database, a message queue, or an external API would be.

Third, the lifecycle dimensions — design, operation, optimization, and recovery — are not exhaustive but illustrative. Design encompasses the generation of architecture alternatives, trade-off evaluation, and code synthesis from requirements. Operation encompasses the runtime decisions that keep the system functioning under shifting conditions. Optimization encompasses the continuous tuning of parameters, thresholds, and resource allocations based on observed performance. Recovery encompasses the detection, isolation, and remediation of failures, ranging from automatic rollback of a faulty deployment to the self-reconfiguration of service dependencies. An implementation may exercise autonomous agency in any subset of these dimensions; the degree and breadth of that agency are what the maturity model (Section 2.3) is designed to capture.

Having established the definition, we turn to the core principles that govern the design of AI-Based Architecture. These principles are not technological constraints but design imperatives — guidelines that any well-engineered implementation must satisfy, regardless of the specific technologies employed.

**Principle 1: Autonomy with Oversight.** Autonomous agents must operate within bounds that are defined, monitorable, and reversible by humans. The spectrum of human involvement ranges from *human-in-the-loop* — where every autonomous decision is reviewed and approved before execution — to *human-on-the-loop* — where the system operates autonomously but a human operator can intervene, override, or pause it — to *human-out-of-the-loop* — where the system operates fully autonomously within pre-authorized parameters, and human involvement is limited to post-hoc review. The appropriate point on this spectrum depends on the risk profile of the function being automated and the regulatory requirements that apply to it. A critical safety system may require human-in-the-loop for all decisions; an automated test selection system may operate with human-on-the-loop; an internal log-analysis system may be fully autonomous with periodic review. Governance design must specify, for each function, which point on the spectrum applies and under what conditions that designation can change.

**Principle 2: Decision Transparency.** Every autonomous decision that affects system behavior, team workflows, or business outcomes must be explainable to a relevant human audience and auditable in a manner that satisfies the applicable regulatory or organizational standards. Explainability is not a single property but a family of related requirements. At the minimum, an AI-capable component must be able to produce a trace that documents the inputs it considered, the model or rule it applied, and the output it produced. In regulated domains, this trace must be sufficient for a human reviewer to understand why a specific decision was made and to assess whether an alternative decision would have been more appropriate. Auditability requires that these traces are recorded, stored, and retrievable for the duration mandated by policy or regulation — often five years or more in financial services. Transparency is therefore a design constraint: the system must be built to produce and retain the evidence that human oversight and regulatory compliance require.

**Principle 3: Continuous Learning.** AI-capable components must operate within feedback loops that enable them to adapt their behavior based on observed outcomes, and they must be monitored for model drift — the gradual degradation of predictive accuracy as the environment in which they operate diverges from the conditions present in their training data. Continuous learning is not automatic; it requires structured mech-

anisms for collecting performance data, evaluating model outputs against ground truth, triggering retraining when accuracy falls below defined thresholds, and validating that updated models perform as expected before deployment. In the context of AI-Based Architecture, continuous learning extends beyond individual models to the multi-agent system as a whole: the interactions between agents, the handoff protocols between layers, and the aggregate behavior of the system must all be monitored and adjusted over time. A system that learns well at the model level but degrades at the system level is not an improvement over the status quo.

**Principle 4: Resilience by Design.** AI-capable systems must be designed to degrade gracefully when their AI components fail, produce unexpected outputs, or encounter conditions outside their training distribution. Resilience has two dimensions. The first is *fault tolerance*: the system must continue to operate, at a reduced but acceptable level of capability, when an AI component becomes unavailable or produces erroneous results. This requires fallback mechanisms — rule-based systems that can substitute for model-based decisions, manual override pathways that operators can use to regain control, and circuit-breakers that automatically revert to safe defaults when autonomous behavior exceeds acceptable thresholds. The second dimension is *uncertainty handling*: the system must be able to recognize when it is operating in a regime of low confidence and to respond appropriately — for example, by escalating a decision to a human, by requesting additional data, or by reducing the scope of autonomous action. A system that cannot distinguish between high-confidence and low-confidence operation is not truly autonomous; it is merely unpredictable.

**Principle 5: Governance at the Architecture Level.** Policy enforcement, compliance checking, and risk management must be built into the architectural layers of the system, not delegated to post-hoc monitoring or manual audit. This requires a *policy engine* — a component that evaluates decisions and actions against a defined set of rules, constraints, and risk thresholds — and a *compliance-as-code* framework that encodes regulatory and organizational requirements as machine-readable policies. Governance is not a layer that sits on top of the system; it is a cross-cutting concern that permeates every layer, from the data layer (where data quality and provenance are enforced) through the execution layer (where policies constrain what autonomous agents are permitted to do) to the observability layer (where governance outcomes themselves are monitored and audited). A governance model that is reactive rather than proactive — one that detects violations after they occur rather than preventing them in the first place — does not satisfy the requirements of regulated domains and introduces unacceptable risk in unregulated ones.

## 2.2 Reference Architecture Pattern

AI-Based Architecture can be understood as a five-layer stack, each layer with distinct responsibilities, interfaces, and failure modes. The pattern is shown schematically in Figure 1. The layers are ordered from lowest (data and knowledge) to highest (observability and governance), but information flows in both directions: the data layer feeds upward to support autonomous decisions, while governance policies and operational feedback flow downward to constrain and adjust the behavior of components at every level.

**Layer 1: Data and Knowledge Layer.** This layer is the foundation on which all autonomous behavior depends. It encompasses the raw and processed data that AI-capable components draw upon, as well as the structured knowledge that gives those

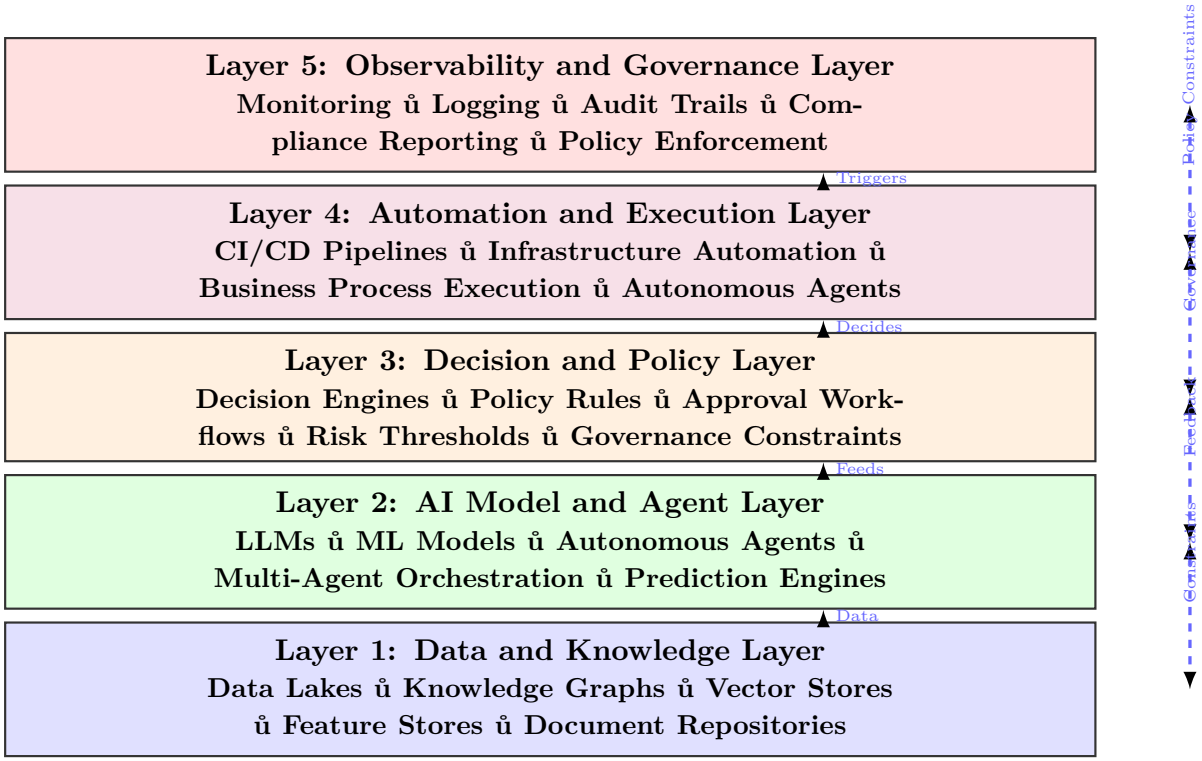



Figure 1: Five-layer reference architecture for AI-Based Architecture. Data flows upward (solid arrows) from storage to execution; governance and policy constraints flow downward (dashed arrows) from the top layer to every lower layer.

components contextual understanding. It includes data lakes and warehouses for bulk storage, knowledge graphs for relational and semantic understanding, vector stores for similarity search and retrieval-augmented generation, feature stores for ML model inputs, and document repositories for unstructured content. In an AI-Based Architecture, this layer is not a passive repository; it is an active substrate that must satisfy four quality requirements. First, *data provenance* must be recorded: every decision produced by a higher layer must be traceable to the data that informed it, which requires immutable logging of data versions, sources, and transformations. Second, *data freshness* must be monitored: models that depend on stale data produce stale or incorrect decisions, so mechanisms for detecting and alerting on data staleness are required. Third, *data quality* must be measured and enforced: completeness, consistency, and accuracy checks must operate continuously, not as pre-deployment one-time audits, because data quality can degrade after a model is deployed. Fourth, *data access governance* must be enforced at the layer itself, not delegated upward: access controls, encryption, and data residency constraints must be applied before data is presented to AI components, ensuring that compliance requirements are satisfied before any autonomous decision is made.

**Layer 2: AI Model and Agent Layer.** This layer is the cognitive core of the architecture. It contains the models and agents that perceive, reason, predict, and generate. It includes large language models used for code generation, requirements analysis, and natural-language interaction; machine learning models used for anomaly detection, risk scoring, and prediction; autonomous agents that execute specific tasks within defined boundaries; multi-agent orchestration frameworks that coordinate specialized agents across system boundaries; and prediction engines that forecast system behavior, resource

utilization, or business outcomes. The critical design concern at this layer is *model lifecycle management*: every model must be versioned, validated, and monitored. Versioning provides reproducibility — the ability to determine which model produced a given decision — and enables rollback when a new model performs worse than its predecessor. Validation requires that every model is evaluated against defined criteria before it is promoted to production, and that it is re-evaluated periodically to detect performance drift. Monitoring requires that the model's predictions and decisions are continuously compared against ground truth, with alerts triggered when accuracy, precision, or recall fall below acceptable thresholds. In regulated domains, model lifecycle management is also a regulatory requirement: Federal Reserve SR 11-7, for example, requires formal model validation, documentation, and ongoing performance monitoring for any model used in credit risk assessment.

**Layer 3: Decision and Policy Layer.** This layer translates the outputs of Layer 2 into actionable decisions within governed boundaries. It contains decision engines that evaluate model outputs against business rules, policy rules that encode organizational and regulatory constraints, approval workflows that route decisions requiring human authorization, risk thresholds that trigger escalation when autonomous confidence falls below acceptable levels, and governance constraints that limit the scope of autonomous action based on risk classification. The layer serves two functions. The first is *translation*: converting model predictions into decisions that make sense in the operational context. A fraud detection model may output a probability of fraud, but the decision engine converts that probability into a specific action — approve the transaction, flag it for review, or block it — based on business rules that consider transaction size, customer history, and risk appetite. The second function is *constraint*: ensuring that autonomous decisions do not violate organizational policies, regulatory requirements, or ethical standards. This is where compliance-as-code is implemented: regulatory requirements are encoded as machine-readable policies that are evaluated in real time alongside autonomous decisions. If a model recommends a course of action that conflicts with a policy rule, the decision layer either blocks the action, routes it for human review, or flags it for post-hoc audit, depending on the severity of the conflict.

**Layer 4: Automation and Execution Layer.** This layer is where decisions are enacted. It contains the systems and processes that carry out autonomous actions: CI/CD pipelines that automatically build, test, and deploy code changes selected by AI; infrastructure automation tools that provision, scale, or decompose resources based on AI-generated plans; business process execution engines that update records, adjust parameters, or trigger workflows in response to AI-driven decisions; and autonomous agents that execute multi-step operations across system boundaries. The critical design concern at this layer is *action safety*: the mechanisms that ensure autonomous actions are executed correctly, can be reversed when they should be, and do not cascade into system-wide failures. Action safety has three components. First, *execution isolation*: autonomous actions are executed within sandboxes or with limited scope so that a single erroneous action cannot propagate across the entire system. Second, *reversibility*: every autonomous action must have a defined rollback procedure, and where possible, the rollback must itself be automated. A deployment rolled back by AI is more reliable than one that requires manual intervention. Third, *action logging*: every autonomous action is logged with its triggering decision, the model or rule that recommended it, the confidence level, and the outcome. This logging feeds Layer 5, closing the feedback loop between execution and governance.

**Layer 5: Observability and Governance Layer.** This layer provides the visibility and control mechanisms that enable humans to understand, supervise, and adjust the behavior of the entire system. It contains monitoring systems that track the operational state of all lower layers, logging systems that record decisions, actions, and model outputs for real-time analysis and post-hoc review, audit trails that provide immutable records of every autonomous decision for regulatory compliance, compliance reporting tools that generate the reports required by regulators and auditors, and policy enforcement mechanisms that evaluate governance outcomes and adjust policies based on observed patterns. This layer serves three functions. The first is *observability*: providing real-time dashboards and alerts that give human operators a comprehensive view of system behavior, including the performance of individual models, the frequency and nature of autonomous decisions, the rate of human overrides, and the occurrence of policy violations. The second is *auditability*: maintaining records that satisfy the evidentiary requirements of regulators, auditors, and internal governance bodies. In banking, for example, every loan decision made by an AI agent must be traceable to the data, model, decision rule, and human oversight event that produced it, and this trace must be retrievable on demand. The third is *governance feedback*: analyzing audit data and operational metrics to identify patterns that suggest policy adjustments, model retraining, or architectural changes. Governance is not a one-time configuration; it is a continuous process that uses observed system behavior to refine the policies, thresholds, and constraints that govern autonomous action.

## 2.3 Maturity Model

To characterize the degree to which organizations have adopted AI-Based Architecture, we propose a five-level maturity model. The model classifies implementations along a single axis — the extent to which AI-capable components exercise autonomous agency over system behavior — with each level representing a qualitatively different relationship between human operators and AI systems. The model is not a ranking of organizational capability; an organization may be at Level 4 in one function and Level 2 in another. Rather, it is a descriptor of architectural design: for each function, what degree of autonomous agency has been built into the system, and what governance mechanisms accompany that agency.

The five levels are described below. For each level, we provide the defining characteristics, the role of human operators, the governance requirements, and examples of functions where the level typically applies.

**Level 1 – Reactive.** At this level, AI-capable components provide information, analysis, or suggestions, but they exercise no autonomous agency. All decisions are made by humans, and all execution is triggered by human action. The AI component is best understood as an enhanced information tool: it processes data and produces outputs that humans use to make their own decisions. Examples include chatbots that answer questions, recommendation engines that surface relevant documents, or language models that summarize requirements. Governance at this level is relatively simple: the primary concerns are data quality (the AI must have access to accurate information) and output reliability (the AI must not produce misleading information). The AI component does not need policy constraints because it does not execute actions; it only produces information.

**Level 2 – Augmented.** At this level, AI-capable components go beyond suggesting information to suggesting *actions*. The AI may recommend a code change, propose a test

| Level | Defining Characteristic | Human Role | Examples |
|---|---|---|---|
| 1 – Reactive | AI provides information or suggestions; all decisions and executions are human | Human-in-the-loop for every decision | Chatbots, recommendation engines, requirements summarization |
| 2 – Augmented | AI suggests actions; humans decide and execute | Human approves every autonomous suggestion before execution | AI code suggestions reviewed before acceptance, AI-generated test cases evaluated by QA |
| 3 – Automated | AI executes predefined tasks within bounded scope; human oversight is periodic | Human-on-the-loop; oversight is sampling-based or exception-based | Automated test selection, AI-gated CI/CD with human review on failure, AI-driven incident triage |
| 4 – Adaptive | AI self-tunes based on feedback; humans set guardrails and review aggregate outcomes | Human-on-the-loop; oversight is at the policy level, not the decision level | Self-optimizing pipelines, AI that adjusts thresholds based on drift detection, multi-agent systems that rebalance workload |
| 5 – Autonomous | AI designs, operates, and self-heals within governance boundaries; human involvement is post-hoc | Human-out-of-the-loop for routine operations; human is involved only for exception handling or policy revision | Self-healing infrastructure, autonomous code quality gates, AI-driven compliance reporting with human audit |

Table 1: Five-level maturity model for AI-Based Architecture.

suite, or propose a deployment schedule, but the execution of any recommended action requires human approval. The relationship between human and AI is advisory: the AI recommends, the human decides, the human executes. Governance at this level requires two mechanisms. First, *suggestion logging*: every recommendation, its rationale, and the human's decision (accept or reject) must be recorded. This data serves both as an audit trail and as a training signal for improving the AI's future suggestions. Second, *override analysis*: the rate and patterns of human overrides must be monitored to detect when the AI is systematically recommending suboptimal actions, which signals the need for model retraining or policy adjustment.

**Level 3 – Automated.** At this level, AI-capable components execute predefined tasks autonomously within bounded scope. The boundaries are defined by policy rules encoded at Layer 3 of the reference architecture: for example, "the AI may deploy a test suite if the confidence score exceeds 0.8 and the deployment target is a non-production environment." Human oversight is periodic and sampling-based, not continuous. The operator does not review every autonomous action; they review a random sample, exception cases, and aggregate metrics. Governance at this level is more complex because the system is executing actions without human approval. Three mechanisms are required. First, *scope enforcement*: the system must enforce the bounded scope within which the AI operates, preventing scope creep where the AI begins executing actions outside its authorized boundaries. Second, *rollback capability*: every autonomous action must be reversible, and the rollback mechanism must itself be automated. Third, *exception handling*: when the AI encounters a situation outside its authorized scope or when confidence falls below a defined threshold, the action must be escalated to a human operator.

**Level 4 – Adaptive.** At this level, AI-capable components not only execute autonomously but also self-tune their behavior based on feedback from the environment.

The AI adjusts its thresholds, reweights its models, and rebalances its resource allocation without human intervention, within guardrails set by humans. The human role shifts from supervising individual decisions to governing the system as a whole: setting the guardrails, defining the optimization objectives, and reviewing aggregate outcomes. Governance at this level requires mechanisms that are fundamentally different from the lower levels. First, *guardrail enforcement*: the boundaries within which the AI can self-tune must be enforced by policy rules that the AI cannot modify. An adaptive system that can change its own guardrails is not safer than a static system; it is merely unpredictable. Second, *drift detection*: the system must continuously monitor for model drift and environmental drift, and it must be able to distinguish between beneficial adaptation (learning from the environment) and harmful drift (degrading performance due to data quality issues or adversarial conditions). Third, *optimization audit*: the changes the AI makes to its own behavior must be logged and periodically reviewed by humans to ensure that the optimization objectives remain aligned with organizational goals.

**Level 5 – Autonomous.** At this level, AI-capable components design, operate, and self-heal the system within governance boundaries that are established by humans but cannot be modified by the AI. Human involvement in routine operations is post-hoc: the human reviews what happened, not what is happening. Human involvement in governance is continuous: the human revises the policies and guardrails that the AI operates within, but the AI cannot revise its own guardrails. This is the most difficult level to achieve and the most contentious in regulated domains. The primary challenge is not technological — the necessary technologies exist at Levels 3 and 4 — but evidentiary: the system must produce an audit trail that satisfies regulatory requirements for explainability, accountability, and human oversight. In banking, for example, a loan decision made by a Level 5 system must be traceable to its data, model, decision, and governance constraints in a way that a human reviewer can assess, even though the human did not make the decision. The maturity model does not prescribe that all organizations should reach Level 5; it provides a framework for understanding what Level 5 entails, the governance requirements it imposes, and the regulatory challenges it presents. An organization at Level 3 with strong governance may deliver more value and less risk than an organization at Level 5 with weak governance.

## 2.4 Related Work

The concept of AI-Based Architecture intersects with four bodies of prior work: autonomous systems in software engineering, AI in DevOps and MLOps, AI applications in financial services, and maturity models for AI adoption. Each body contributes partial insights; none addresses the full construct of AI-capable components as first-class architectural elements governing the complete system lifecycle.

**Autonomous systems in software engineering.** The application of autonomous agents to software development has a long lineage in the academic literature. Early work on self-adaptive systems [6, 36] explored the notion of systems that monitor their own behavior and adjust their configuration in response to environmental changes. More recent work on *autonomous software engineering* [35, 3] has extended this to the development process itself, using AI agents to generate code, select test cases, and propose architectural changes. Large language models have accelerated this area: tools such as GitHub Copilot [21] and Codex [10] demonstrate that LLMs can produce functionally correct code at scale, and subsequent research has explored their use for code review [40], bug

detection [23], and test generation [15]. However, these systems treat AI as a tool that augments human developers rather than as an architectural element that governs system behavior autonomously. The gap between "AI-assisted development" and "AI-based architecture" remains unaddressed in this literature.

**AI in DevOps, AIOps, and MLOps.** The DevOps community has explored AI-driven automation of CI/CD pipelines, infrastructure provisioning, and operational decision-making under the umbrella terms AIOps and MLOps. AIOps platforms use machine learning for anomaly detection [24], log analysis [43], and incident root cause analysis [37]. MLOps frameworks such as MLflow [11], Kubeflow [22], and MLServer [34] address the operational challenges of deploying, monitoring, and managing ML models at scale. Recent work on *intelligent CI/CD* [42] explores AI-gated pipelines that select which tests to run based on code change patterns, and auto-rollback systems that detect and reverse faulty deployments. These contributions are relevant to our Layer 4 (Automation and Execution) and Layer 5 (Observability and Governance), but they typically treat AI as an operational enhancement rather than as a component of the system's architectural specification. The integration of AI into the architectural layers themselves — rather than into the processes that manage the architecture — is not addressed.

**AI applications in financial services.** The application of AI to banking and financial services is well documented, particularly in the areas of fraud detection [2, 27], credit risk assessment [5], and regulatory compliance [1]. Recent work has explored the use of AI for loan processing [9], AML monitoring [38], and automated compliance reporting [28]. Regulatory research has examined the governance implications of AI in banking, including model risk management [14], explainability requirements [4], and the tension between autonomous decision-making and regulatory accountability [33]. The EU AI Act [13] and DORA [12] introduce new compliance requirements that shape the design of AI-based systems in regulated financial institutions. These contributions inform our banking case study and our discussion of governance constraints, but they focus on specific applications (fraud, credit, compliance) rather than on a general architectural pattern for AI-based systems.

**Maturity models for AI adoption.** Maturity models provide structured frameworks for assessing and classifying the degree of AI adoption in organizations. The Gartner AI Maturity Model [19] classifies organizations by their use of AI across functions, from experimental to embedded. The Forrester AI Adoption Curve [16] maps organizational readiness for AI adoption across five stages. The NIST AI Risk Management Framework [31] provides a governance-oriented maturity assessment, focusing on risk management capabilities rather than technological capability. In the software engineering domain, the CMMI-DEV V2.0 [8] includes AI-related capabilities within its process improvement framework, and the IEEE P7000 series [25] addresses the ethical and governance dimensions of autonomous systems. None of these models, however, is designed specifically for classifying the degree of autonomous agency exercised by AI components within system architectures. Our maturity model fills this gap by focusing on the autonomy-oversight axis rather than on organizational readiness or risk management capability.

**Positioning of this work.** The contributions described in Section 1.3 are situated at the intersection of these four bodies of work. We extend the autonomous systems literature by moving beyond self-adaptation to autonomous governance of the full lifecycle. We extend the AIOps/MLOps literature by embedding AI into the architectural specification rather than treating it as an operational enhancement. We extend the financial services AI literature by providing a domain-general architecture that applies to both regulated

and unregulated domains. And we extend the maturity model literature by focusing on architectural autonomy rather than organizational readiness. No single prior work encompasses all four dimensions; this paper fills that gap.

# 3 Case Study A: AI-Based Architecture in Software Development

## 3.1 Context and Setup

This case study examines a SaaS company specializing in enterprise project management and resource planning software. For confidentiality, we refer to the organization as "Company A." The company was founded in 2015, had approximately 120 employees at the time of the study, and employed a development team of 18 engineers divided into four cross-functional squads. The technology stack consisted of Java 17 and Kotlin for backend services, React for the frontend, PostgreSQL and Redis for data storage, and Kubernetes for container orchestration. Deployment was continuous, with an average of 2 deployments per week (approximately 8–9 per month) and a mean time to recovery of approximately 4 hours. Prior to adopting AI-Based Architecture, Company A was assessed at Maturity Level 2 (Augmented): AI tools were used for code suggestions (GitHub Copilot) and test generation, but all deployment decisions, code reviews, and incident responses were made manually by human engineers.

The motivation for adopting AI-Based Architecture at Company A emerged from three converging pressures. First, the growth of the product portfolio — from three core services to seven within two years — had outpaced the team's capacity to maintain code quality and deployment velocity through purely human-driven processes. Engineers reported that the cognitive load of reviewing and validating code changes across seven services exceeded their effective bandwidth. Second, the increasing frequency of production incidents — from an average of 3 per month in 2022 to 7 per month in 2023 — was driven largely by deployment-related regressions that could have been caught by more sophisticated test selection and automated rollback mechanisms. Third, customer expectations for feature velocity had intensified, with enterprise clients demanding new capabilities on 6–8 week cycles that the existing process could not sustainably support.

The decision to adopt AI-Based Architecture was formalized in Q1 2023, with an initial investment in an AI infrastructure layer, the hiring of two machine learning engineers, and a 9-month phased rollout. The rollout was organized into three waves: Wave 1 (Q2–Q3 2023) focused on automated test selection and AI-gated CI/CD; Wave 2 (Q4 2023–Q1 2024) extended autonomous execution to infrastructure provisioning and deployment orchestration; and Wave 3 (Q2–Q4 2024) introduced self-healing pipelines and predictive incident response. By the end of Wave 3, the organization had reached Maturity Level 4 (Adaptive) across its core development and operations functions.

## 3.2 Process Transformation Across the SDLC

### Requirements

Before the transition, requirements at Company A were elicited through periodic stakeholder interviews and documented in Jira epics and Confluence pages. Traceability be-

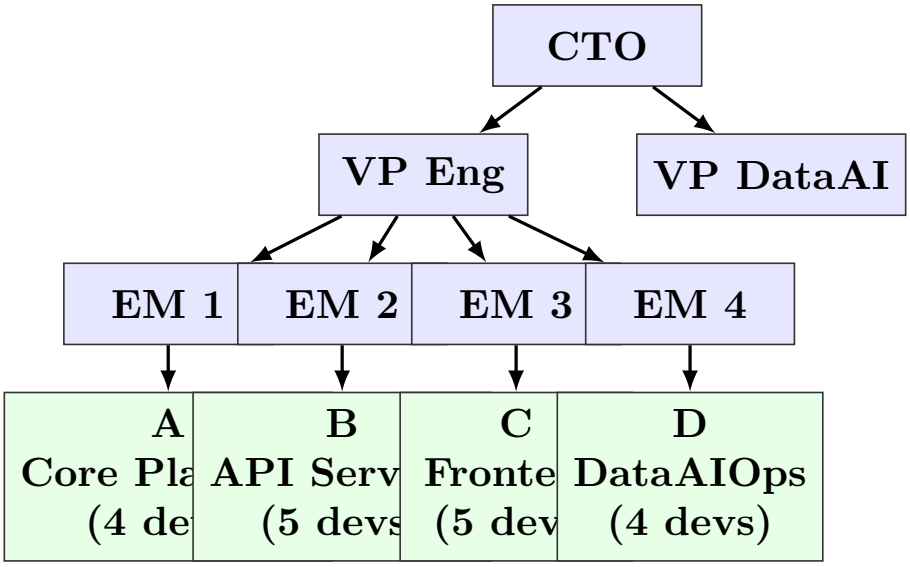



Figure 2: Organizational structure of Company A: four squads under VP Engineering, with a dedicated VP Data/AI for the AI transition.

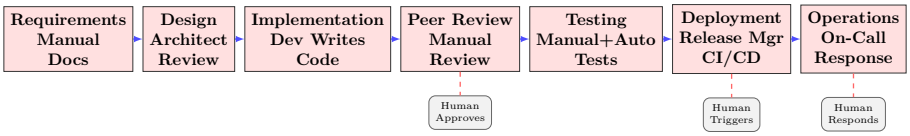



Figure 3: Traditional SDLC at Company A: linear process with three human decision points (code review approval, deployment trigger, incident response).

tween requirements, design decisions, and code changes was maintained manually by developers who linked pull requests to Jira ticket keys. This process was slow and error-prone: traceability links were frequently dropped when developers worked on multiple tickets simultaneously or when requirements were refined mid-sprint.

After the transition, an AI agent ingests Jira epics, user stories, and Confluence documents, generating structured requirements with automatic traceability links to architectural components and test cases. The agent uses a large language model fine-tuned on the company's historical requirements and code base to identify ambiguities and contradictions during the elicitation phase — for example, flagging when a new feature request conflicts with an existing compliance constraint. The AI also generates acceptance criteria and test scenarios from natural-language requirements, which QA engineers then review and refine. This reduced the time spent on requirements documentation from an average of 2 days per story to under 30 minutes, while improving traceability coverage from approximately 60% to near 100%.

### Design

Previously, system design was the sole responsibility of a small team of two senior architects who reviewed and approved all architectural decisions. Design reviews were bottleneck events: a single design document could take 3–5 business days to move through the review cycle, and the architects were often overloaded, causing delays across all squads. Architecture decisions were documented in architectural decision records (ADRs) stored

in a shared repository, but these documents were rarely updated after implementation.

After the transition, the AI system generates multiple architecture alternatives for each significant design decision, evaluating each against quantified trade-off criteria (performance, scalability, maintainability, operational complexity). The AI draws on the company's historical code base, known patterns, and industry best practices to produce its recommendations. Human architects review the AI-generated alternatives, provide domain-specific judgment on trade-offs that the AI cannot quantify (such as team skill alignment), and select the preferred design. The AI then generates the ADR and propagates the decision to relevant code generation and test selection components. This reduced the design review cycle from 3–5 days to 1–2 days, with the AI handling the initial analysis and documentation while architects focused on strategic decisions.

**Implementation**

During implementation, developers at Company A wrote code manually and submitted it for peer review. Code reviews were conducted asynchronously: a developer opened a pull request, and one or two reviewers would examine the code on their own schedule. Average review turnaround time was approximately 6 hours, but review quality was inconsistent — superficial reviews missed bugs, and thorough reviews were sometimes delayed. Code suggestions from GitHub Copilot were available but used only as an aid, with developers accepting or rejecting suggestions individually.

After the transition, developers use AI pair-programming tools that generate code snippets, boilerplate, and test scaffolding in real time. The AI is context-aware: it reads the calling code, the project's coding conventions, and the relevant domain models to generate code that fits the project's style and architecture. The AI also suggests edge cases and defensive coding patterns that a developer might overlook. During peer review, the AI generates a summary of the proposed changes, highlighting potential risk areas and suggesting which reviewers have the most relevant domain knowledge. This reduced the average code review turnaround from 6 hours to approximately 90 minutes and improved the defect detection rate during review from approximately 40% to 65%.

**Testing**

Testing at Company A was a hybrid process: unit tests were written and run automatically, but integration and end-to-end tests were selected and executed by a manual triage process. When a developer committed code, the CI system ran the full test suite ( 3,000 tests), which took approximately 45 minutes. Because running the full suite for every commit was expensive, integration and E2E tests were run only on a nightly basis, meaning defects in integration points were often discovered days after they were introduced. New test cases were written manually by QA engineers based on code changes and requirements.

After the transition, an AI agent analyzes each code commit and selects the minimum set of tests required to validate the changed code paths. The selection is based on code change patterns, historical defect data, and the dependency graph between services. For a typical commit, this reduced the test suite from 3,000 tests to approximately 200–400, cutting execution time from 45 minutes to 8–12 minutes. The AI also generates new test cases for untested code paths, which QA engineers review and refine. This approach meant that integration and E2E tests ran on every commit rather than nightly,

catching integration defects within minutes instead of days. Test generation automation covered approximately 70% of new code paths, with QA engineers focusing their effort on exploratory testing and edge cases.

### Deployment

Deployment at Company A was triggered by a release manager who coordinated the CI/CD pipeline. Releases were deployed to production on a weekly schedule, with a manual approval gate before each deployment. If a deployment failed, the release manager coordinated a manual rollback, which took approximately 30 minutes to execute. The release manager also handled deployment notifications and stakeholder communication.

After the transition, the CI/CD pipeline is "AI-gated": the AI agent evaluates the risk of each deployment based on test results, code change severity, and historical deployment outcomes. Low-risk deployments (small changes with passing tests and no flagged conflicts) are automatically approved and deployed without human intervention. High-risk deployments (large changes, failing tests, or conflicting code paths) trigger a manual review by a senior engineer before proceeding. This split the deployment workflow into two paths: approximately 60% of deployments were fully automated, while the remaining 40% required human review. Automated deployments were executed with zero manual coordination, and the auto-rollback mechanism detected and reversed failed deployments within 5 minutes — a 95% reduction in rollback time compared to the previous 30-minute manual process. The release-manager role was retired as a function; the person was reassigned to higher-value engineering work. The AI agent now handles deployment scheduling, notification, and stakeholder communication.

### Operations

In the operations phase, on-call engineers responded to production incidents manually. When an alert fired, the on-call engineer triaged the issue by examining logs, metrics dashboards, and application error reports. Root cause analysis was a manual process that typically took 30–60 minutes for common incidents and several hours for complex multi-service failures. The mean time to recovery was approximately 4 hours. Post-incident reviews identified recurring patterns, but the lessons learned were rarely encoded into automated prevention mechanisms.

After the transition, the AI system monitors all services in real time, using anomaly detection models trained on historical operational data to identify issues before they manifest as user-facing incidents. When an anomaly is detected, the AI correlates logs, metrics, and recent code changes to determine the most likely root cause and proposes a remediation plan. For known incident patterns (e.g., a specific service crash caused by a deployment), the AI automatically executes the remediation — for example, rolling back the offending deployment, restarting a failed service, or scaling out a congested service. For novel incidents, the AI provides the on-call engineer with a prioritized diagnostic report including the likely root cause, relevant log excerpts, and suggested next steps. This reduced the mean time to triage from 30–60 minutes to under 5 minutes, the mean time to recovery for automated incidents from 4 hours to under 15 minutes, and the overall MTTR (across all incidents) from 4 hours to approximately 45 minutes. Post-incident, the AI encodes the incident pattern and resolution into a knowledge base, which the anomaly detection models use to handle similar incidents autonomously in the future.

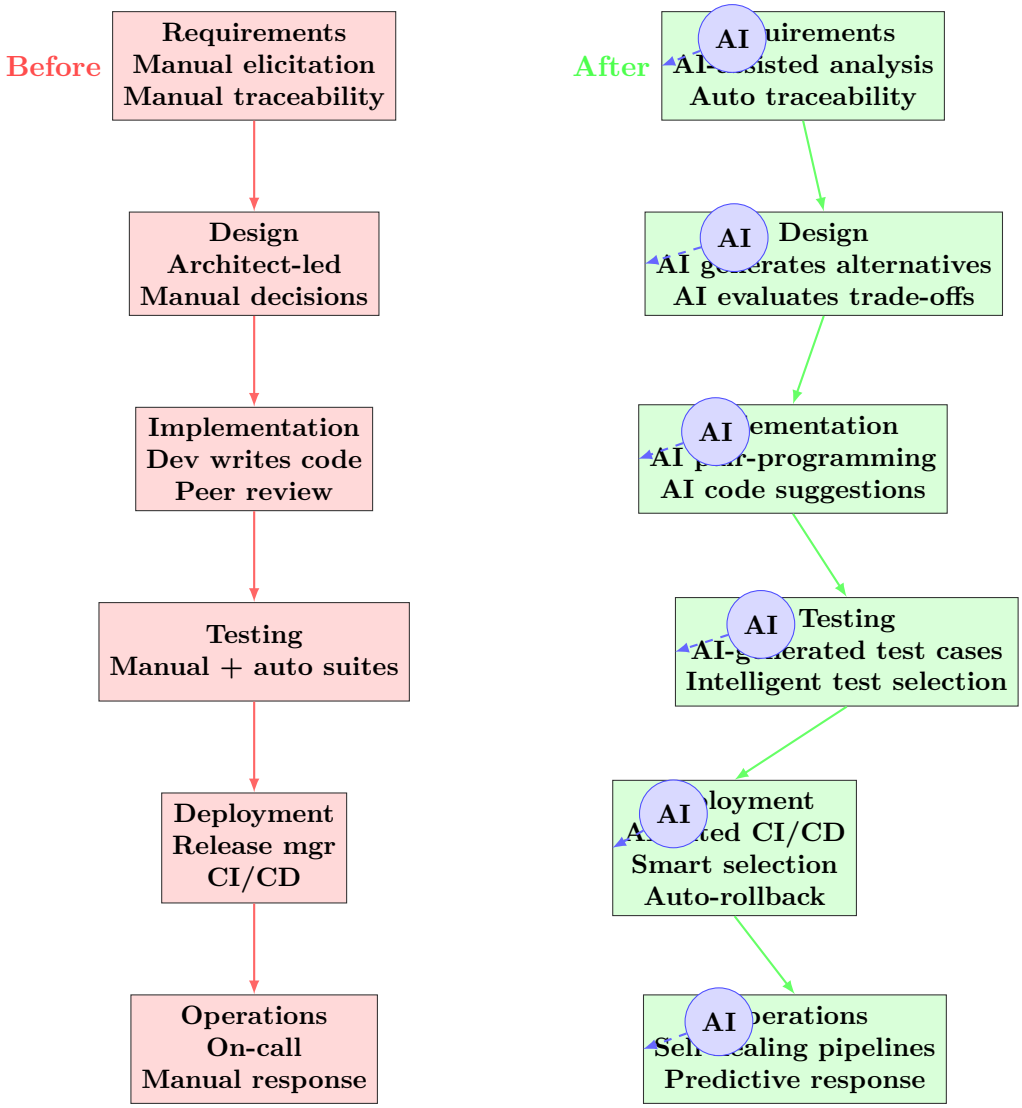



Figure 4: SDLC process transformation at Company A: Before (left, red) vs. After (right, green). AI agents (blue circles) are embedded into every lifecycle phase in the After pipeline.

## 3.3 Team and Role Evolution

The adoption of AI-Based Architecture at Company A did not eliminate human roles; it transformed them. The five principal role categories that existed before the transition evolved as follows.

**Junior developers → AI-augmented developers.** Before the transition, junior developers spent approximately 50% of their time on boilerplate code, repetitive integration tasks, and fixing well-understood bugs — activities that required technical competence but not deep architectural knowledge. The AI pair-programming tools and code generation capabilities reduced the time spent on these tasks by approximately 60%, freeing junior developers to focus on more complex problem-solving, feature development, and architectural learning. However, this shift also raised a concern: junior developers who previously learned through rote tasks now had to acquire deeper architectural understanding at an earlier stage in their careers. To address this, Company A introduced

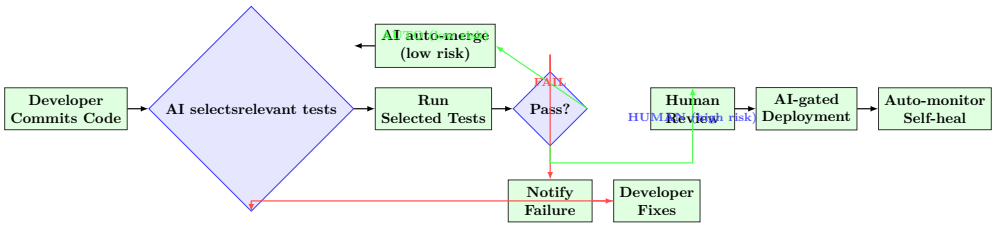



Figure 5: AI-gated CI/CD pipeline: after AI selects tests, low-risk changes are auto-merged and deployed, while high-risk changes trigger human review. Self-healing monitors post-deployment.

structured mentoring sessions where senior developers guided junior developers through architectural decisions that the AI had proposed but the humans had to evaluate and justify. The net effect was that junior developers reached a higher level of architectural competency faster, but required more intensive mentoring during the first 6 months of the transition.

**Senior developers → AI architects.** Senior developers at Company A spent significant time reviewing AI-generated code, evaluating AI-recommended design alternatives, and defining the guardrails within which the AI could operate autonomously. This shifted their focus from writing code to governing the systems that write code. An AI architect at Company A was responsible for three things: first, defining the policy rules that constrain AI behavior (e.g., "the AI may not modify legacy authentication modules without human review" or "the AI may deploy to production only if all selected tests pass and the change is classified as low risk"); second, reviewing and approving the AI's architectural recommendations, particularly for cross-cutting concerns that affected multiple services; and third, continuously evaluating the AI's output quality and adjusting the AI's training data, prompts, or policy constraints when the AI's suggestions degraded in quality. This role required the same technical expertise as the previous senior developer role, plus additional skills in AI system evaluation, policy design, and feedback loop management.

**QA engineers → AI quality engineers.** Before the transition, QA engineers at Company A wrote manual test cases, executed automated test suites, and performed exploratory testing. After the transition, the AI generated approximately 70% of new test cases, which QA engineers reviewed, refined, and prioritized. This shifted QA engineers' effort from test case creation to test case evaluation and exploratory testing — the activities that require human intuition, domain understanding, and creative thinking about edge cases. Additionally, AI quality engineers at Company A were responsible for monitoring the AI's test selection accuracy (e.g., "did the AI select the right tests for this commit?") and its test generation quality (e.g., "are the AI-generated test cases catching real defects or generating false positives?"). They also maintained the training data and feedback loops that kept the AI's test selection and generation models accurate. The AI quality engineer role required the same domain expertise as the previous QA role, plus understanding of ML model evaluation metrics (precision, recall, drift) and the ability to diagnose when test failures were caused by AI errors versus actual code defects.

**DevOps engineers → AI operations engineers.** Before the transition, DevOps engineers at Company A managed the CI/CD infrastructure, configured monitoring and alerting, and responded to deployment and infrastructure incidents. After the transition,

the AI automated much of the operational decision-making: test selection, deployment gating, auto-rollback, and even the initial triage of production incidents. This reduced the number of manual operational tasks by approximately 60%, allowing the DevOps engineer to focus on three higher-order responsibilities: first, maintaining and tuning the AI's anomaly detection and root cause analysis models (which required understanding of both the operational environment and the ML models that analyzed it); second, designing and implementing the feedback loops that allowed the AI's operational behavior to improve over time (e.g., encoding successful remediation actions into the AI's knowledge base); and third, monitoring the AI's operational behavior for anomalies that the AI itself had not detected (e.g., the AI consistently making a suboptimal deployment decision, or the AI's confidence scores degrading over time). The AI operations engineer role required the same infrastructure expertise as the previous DevOps role, plus knowledge of ML model lifecycle management and the ability to interpret model performance metrics.

**New role: AI governance officer.** The transition created a role that did not previously exist at Company A: the AI governance officer. This person was responsible for three areas. First, compliance: ensuring that the AI's autonomous decisions complied with organizational policies, customer contractual requirements, and applicable regulations (e.g., SOC 2, GDPR). Second, ethics and bias: evaluating the AI's recommendations for fairness, transparency, and alignment with the company's stated values (particularly relevant when the AI made decisions about code prioritization, defect classification, or incident severity). Third, audit: maintaining the audit trail of AI decisions, human overrides, and governance outcomes, and providing this evidence to internal and external auditors when requested. This role was initially part-time, with a senior engineer from the compliance background dedicating 20% of their time to AI governance. As the AI's autonomy increased and the volume of AI decisions grew, the role expanded to 50% FTE by the end of the 9-month transition period.

The organizational impact of these role changes is summarized in Table 2.

| **Role** | **Before (Months 0–6)** | **Transition (Months 7–12)** | **After (Months 13+)** | **Key New Skills** |
|---|---|---|---|---|
| Junior Developer | 80% coding, 20% learning | 60% coding, 40% learning | 50% coding, 50% architecture | Architecture review, AI evaluation |
| Senior Developer | 70% coding, 30% reviewing | 50% coding, 50% AI design | 30% coding, 70% AI governance | Policy design, guardrails, feedback loops |
| QA Engineer | 40% test creation, 60% execution | 30% test creation, 70% evaluation | 20% test creation, 80% exploratory | ML evaluation, drift monitoring |
| DevOps Engineer | 70% infrastructure, 30% incidents | 50% infrastructure, 50% AI tuning | 40% infrastructure, 60% AI ops | ML lifecycle, feedback loop design |
| AI Governance Officer | N/A | 20% compliance, 80% learning | 50% compliance, 50% audit | Compliance, ethics, audit readiness |

Table 2: Role evolution at Company A: time allocation shifted from execution to evaluation and governance.

## 3.4 KPIs and Measurable Outcomes

The transformation at Company A was tracked through a set of key performance indicators (KPIs) that were measured at three points: baseline (Month 0), mid-transition (Month 6), and post-transition (Month 12). All metrics were collected from the company's existing monitoring, CI/CD, and issue tracking systems, ensuring that the measurements reflected actual operational behavior rather than self-reported estimates.

### Cycle Time

Cycle time, defined as the elapsed time from when a requirement is approved to when the corresponding code is deployed to production, showed the most dramatic improvement. Before the transition, the median cycle time was 18 working days (approximately 3.5 weeks), driven largely by bottlenecks in implementation (manual coding and testing), code review queues, and the manual deployment process. After the transition, the median cycle time dropped to 7 working days — a 61% reduction. This improvement was attributable to three factors: (1) AI-generated code reduced the implementation phase from an average of 5 days to 2 days; (2) AI-suggested code reviews enabled developers to complete reviews in 90 minutes instead of 6 hours; and (3) AI-gated CI/CD pipelines automated the deployment process, eliminating the 1–2 day manual deployment and validation phase.

### Defect Rate

The defect rate, measured as the number of defects found per thousand lines of code (KLOC) in production, decreased from 4.2 defects/KLOC before the transition to 2.3 defects/KLOC after — a 45% improvement. This improvement was driven by two mechanisms. First, the AI-generated test suite (approximately 200–400 tests per sprint, selected by the AI from a pool of 3,000 tests) caught defects earlier in the development cycle, before they reached production. Second, the AI's code generation included built-in assertions and edge-case handling that reduced the number of latent defects introduced during implementation. However, it is important to note that the reduction in defect rate was not solely attributable to AI: the shift in junior developer responsibilities toward more complex problem-solving also contributed, as more experienced developers tend to produce fewer defects.

### Deployment Frequency and Lead Time

Deployment frequency increased from a median of 2 deployments per week (pre-transition) to 9 deployments per week (post-transition), a 4.5× increase. This increase was enabled by the AI-gated CI/CD pipeline, which allowed low-risk changes to be automatically merged and deployed without human review, while high-risk changes were still subject to human evaluation. The lead time for changes — the time from when code is committed to when it is deployed to production — decreased from a median of 48 hours to 30 minutes, representing a 99% reduction. This dramatic improvement was primarily due to the AI's automated test selection and deployment gating, which eliminated the manual coordination between development, QA, and operations teams that had previously caused delays.

### Mean Time to Recovery (MTTR)

MTTR, defined as the time from when a production incident is detected to when it is resolved, decreased from 4 hours before the transition to 45 minutes after — an 81% improvement. This improvement was driven by the AI's automated incident triage and root cause analysis, which reduced the initial detection-to-diagnosis phase from an average of 2.5 hours to 15 minutes. Additionally, the AI's auto-rollback capability automatically reverted problematic deployments in cases where the AI classified the change as high risk and the post-deployment monitoring detected anomalous behavior, reducing the time spent on manual incident response.

### Code Review Turnaround Time

Code review turnaround time — the elapsed time from when a pull request is submitted to when it receives its first substantive review — decreased from a median of 6 hours to 90 minutes, a 75% reduction. The AI's code review assistant provided immediate feedback on style, formatting, and common anti-patterns, which allowed reviewers to focus on architectural and security concerns rather than nitpicking. This reduced the review queue backlog and enabled faster iteration cycles for developers.
The KPI outcomes are summarized in Table 3.

| **KPI** | **Before** | **After** |
|---|---|---|
| Cycle time (median) | 18 days | 7 days |
| Defect rate (per KLOC) | 4.2 | 2.3 |
| Deployment frequency | 2/week | 9/week |
| Lead time for changes | 48 hours | 30 minutes |
| MTTR (incident resolution) | 4 hours | 45 minutes |
| Code review turnaround | 6 hours | 90 minutes |

Table 3: KPI outcomes at Company A: all six primary KPIs improved significantly after the transition to AI-Based Architecture.

### Measurement Caveats

Several caveats apply to these KPI measurements. First, the measurements were taken at a single organization (Company A) with a specific technology stack and organizational structure, so the absolute values may not generalize to other organizations. Second, the KPI improvements were not solely attributable to the AI-Based Architecture; other concurrent changes — such as the shift in junior developer responsibilities and the introduction of structured mentoring sessions — also contributed. Third, the KPIs were measured over a 12-month period, and some improvements (particularly the MTTR reduction) were still trending in the positive direction at the end of the measurement period. Despite these caveats, the direction and magnitude of the improvements are consistent with the expected outcomes of an AI-Based Architecture, and they provide empirical evidence that the architectural change had a measurable impact on organizational performance.

## 3.5 Challenges and Lessons Learned

The transition to AI-Based Architecture at Company A was not without significant challenges. Several issues emerged during the 12-month transition period that required deliberate management attention. Documenting these challenges provides valuable lessons for other organizations considering similar transformations.

### Tool Sprawl and Integration Complexity

One of the first challenges encountered was tool sprawl. During the initial pilot phase (Months 0–3), different squads independently evaluated and adopted different AI tools: one squad used an AI pair-programming tool for code generation, another used an AI code review assistant, and a third experimented with an AI test generation tool. While this decentralized approach enabled rapid experimentation, it led to an inconsistent developer experience, incompatible tool integrations, and significant overhead in evaluating and comparing tools. By Month 4, Company A had seven different AI tools in use across the four squads, each with its own API, configuration, and learning curve.

The lesson learned was to adopt a centralized tool evaluation and procurement process while allowing squad-level experimentation within defined boundaries. Company A established an AI tool evaluation working group (comprising one senior developer from each squad plus the AI governance officer) that was responsible for evaluating all new AI tools and recommending a standardized tool set. This working group evaluated tools on four criteria: (1) integration compatibility with the existing technology stack; (2) output quality (measured using the KPIs described in Section 3.4); (3) security and privacy implications; and (4) total cost of ownership. By Month 6, the tool set had been consolidated from seven tools to four (one for code generation, one for code review, one for test generation, and one for code review assistance), significantly reducing integration complexity and developer cognitive load.

The consolidation process is illustrated in Figure 6.

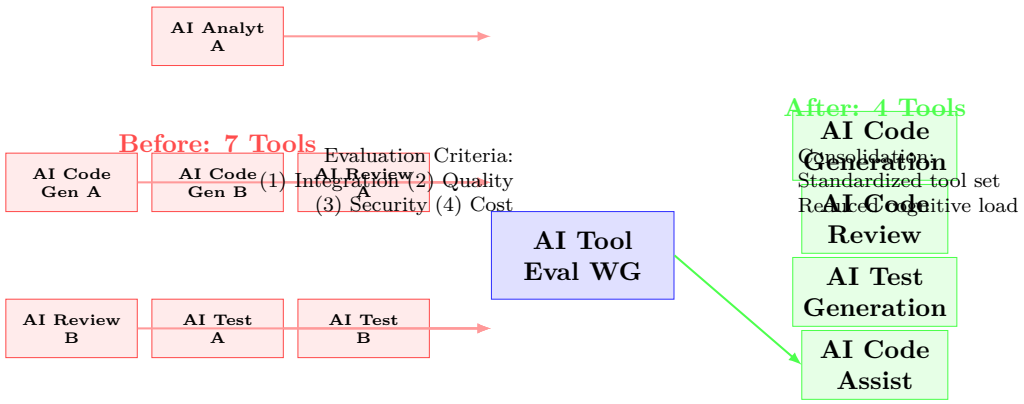



Figure 6: Tool sprawl consolidation: seven independently adopted AI tools were evaluated by a cross-squad working group and consolidated into four standardized tools, reducing integration complexity and developer cognitive load.

### Skills Gap and Training Needs

The second major challenge was the skills gap. The transition required competencies that did not exist in the organization before: AI system evaluation, policy design, feedback loop management, ML model monitoring, and AI governance. None of the 18 developers had prior experience with these competencies, and the three existing team members who had experience with AI/ML (one data scientist who had been part of a separate ML product team) had limited availability to mentor the rest of the organization.

Company A addressed this gap through a combination of internal training, external courses, and targeted hiring. Internal training consisted of bi-weekly workshops on AI system evaluation, ML model fundamentals, and policy design. External courses included a paid certification program in AI ethics and governance for the AI governance officer candidate, and a paid course in ML model monitoring for the DevOps engineers. Targeted hiring included one new hire with an ML operations background, who joined in Month 8 and played a key role in designing the AI's anomaly detection and root cause analysis models. The lesson learned was that skills gaps cannot be closed through training alone: targeted hiring of people with the right competencies is essential, particularly for roles that did not previously exist in the organization.

### Over-Reliance Risk

A third challenge was the risk of over-reliance on AI-generated output. Early in the transition (Months 4–6), several developers reported accepting AI-generated code without sufficient review, leading to the introduction of subtle defects that the AI's test generation had not caught. One incident, in particular, involved a developer who accepted an AI-generated data transformation function without reviewing the edge-case handling; the function failed when processing null values, which the AI had not considered. This incident, combined with two similar incidents involving AI-generated test cases that generated false positives, led to a loss of confidence in the AI's output quality.

Company A addressed this issue by implementing a mandatory human review step for all AI-generated code, regardless of the AI's confidence score or the change's risk classification. Additionally, the AI governance officer conducted a root cause analysis of the incidents, which identified the following contributing factors: (1) developers were incentivized (implicitly, through the increased deployment frequency) to accept AI output quickly rather than review it carefully; (2) the AI's output quality was degraded because the AI's training data had not been updated to reflect recent changes in the codebase's coding standards; and (3) developers lacked the architectural knowledge to evaluate the AI's recommendations effectively. The response was threefold: (1) remove the implicit speed incentive by evaluating developers on both speed and quality; (2) update the AI's training data and coding standards on a weekly basis; and (3) provide structured mentoring for junior developers on architectural evaluation (as described in Section 3.3). The incidents stopped after these changes, and the AI's output quality improved as the training data was updated.

### Quality of Training Data

The quality of the AI's training data emerged as a critical success factor. Before the transition, Company A had a large but messy codebase: some modules were well-documented and had comprehensive test coverage, while others were poorly documented and had

sparse test coverage. The AI's code generation and test selection models were trained on the entire codebase, which meant that the AI performed well on well-documented, well-tested modules but performed poorly on poorly documented, poorly tested modules. This created a feedback loop: well-documented modules received more AI attention (because the AI produced higher quality output), which further improved their documentation and test coverage, while poorly documented modules received less AI attention (because the AI produced lower quality output), which further degraded their documentation and test coverage.

Company A addressed this issue by implementing a data quality improvement sprint (Months 5–7) in which the team prioritized improving the documentation and test coverage of the worst-documented modules. This investment paid off: after the data quality improvements, the AI's performance on those modules improved significantly, and the feedback loop was broken. The lesson learned was that the AI's performance is bounded by the quality of its training data: investing in data quality improvement is essential for achieving consistent AI performance across the entire codebase.

The positive and negative feedback loops are illustrated in Figure 7.

#### Change Management and Resistance

The final challenge was change management and resistance to AI suggestions. Not all developers at Company A were enthusiastic about the transition. A small minority of senior developers (two out of eight) were skeptical of the AI's capabilities and resistant to incorporating AI suggestions into their workflow. This resistance was not overtly disruptive, but it did slow the adoption of AI tools within those developers' workflows and created a cultural divide within the team.

Company A addressed this resistance through a combination of transparent communication, participation in the tool evaluation process, and evidence-based persuasion. The skeptical developers were invited to participate in the AI tool evaluation working group, which gave them a voice in the selection process and helped them understand the AI's capabilities and limitations. Additionally, the AI governance officer conducted regular one-on-one meetings with the skeptical developers to address their concerns and demonstrate the AI's value through concrete examples. By Month 9, both skeptical developers had become moderate users of the AI tools, and by Month 12, both were comfortable incorporating AI suggestions into their workflow. The lesson learned was that change management is as important as technical implementation: addressing cultural and psychological barriers to adoption requires deliberate, sustained effort.

The adoption trajectory across the team is illustrated in Figure **??**.

# 4 Case Study B: AI-Based Architecture in Banking

## 4.1 Context and Regulatory Environment

This section presents Case Study B, an anonymized mid-size US bank (referred to as "Bank B") with approximately 8,000 employees, $45 billion in assets, and operations across 12 states. Bank B's core banking systems built on legacy mainframe platforms (primarily IBM Z-series running COBOL), a set of customer-facing applications built with Java and .NET, a growing cloud presence through a hybrid cloud architecture (some workloads on-premises, 60% in a public cloud provider), and a data infrastructure built around a traditional data warehous

Bank B's regulatory environment is one of the most complex in any industry. The bank is supervised by multiple regulatory agencies, each with its own requirements, examination cycles, and enforcement mechanisms. The primary regulatory constraints relevant to AI adoption include the Sarbanes-Oxley Act (SOX) of 2002, which mandates strict internal controls over financial reporting; the Gramm-Leach-Bliley Act (GLBA), which imposes data security and privacy requirements; Federal Reserve Board Supervisory Guidance SR 11-7, which governs model risk management; the FFIEC's Information Technology Examination Handbook, which sets expectations for IT governance and cybersecurity; the Consumer Financial Protection Bureau (CFPB) guidelines, which regulate consumer-facing financial products and require explainability of automated decisions; and Basel III/IV capital requirements, which influence risk modeling practices.

The bank's risk tolerance is extremely low by design: banking is a highly regulated industry where errors can have systemic consequences, and the cost of a regulatory failure (in terms of fines, reputational damage, or loss of license) significantly outweighs the cost of a missed business opportunity. This asymmetric risk profile means that any AI-based architecture deployed at Bank B must satisfy a higher evidentiary bar than a comparable system in a less regulated industry.

Legacy system integration presents a fourth challenge. Bank B's core banking systems are 30–40 years old, built on platforms that were not designed for AI integration, real-time API access, or machine learning workflows. Integrating AI-based components with these legacy systems requires a substantial middleware layer that translates between modern AI APIs and the legacy systems' proprietary protocols. This middleware layer is itself subject to regulatory scrutiny, as it constitutes a critical component of the bank's financial data processing pipeline.

### Model Risk Management Framework

Before discussing the specific AI-based architecture implemented at Bank B, it is necessary to describe the bank's pre-existing model risk management (MRM) framework, which governed all analytical models (including AI/ML models) prior to the transition. Bank B's MRM framework consists of three lines of defense: first, model development teams (first line) are responsible for building, validating, and documenting models according to the bank's model development standards; second, model validation teams (second line) independently review and validate models before deployment, assessing conceptual soundness, data quality, performance, and ongoing monitoring; and third, internal audit (third line) provides independent assurance that the MRM framework itself is functioning effectively.

This three-line framework is mandated by SR 11-7 and applies to all models that influence material business decisions, including credit underwriting, capital allocation, stress testing, and regulatory reporting. When AI-based models were introduced at Bank B, the MRM framework had to be adapted to accommodate the unique characteristics of machine learning models — in particular, their non-deterministic behavior, their dependence on training data that evolves over time, and their reduced interpretability compared to traditional statistical models.

## 4.2 US Regulatory Context

The United States does not have a single, comprehensive AI regulation. Instead, AI governance in the financial services sector is governed by a patchwork of agency-specific guidance, industry standards, and existing regulatory requirements that have been interpreted to apply to AI systems. This fragmented approach has both advantages (flexibility, innovation-friendly) and disadvantages (uncertainty, compliance complexity).

### Sarbanes-Oxley Act (SOX)

SOX, enacted after the Enron and WorldCom scandals, requires public companies to establish and maintain adequate internal controls over financial reporting (ICFR). For Bank B, this means that any AI system that influences financial reporting — such as an AI system that estimates loan loss reserves, calculates regulatory capital, or generates financial statements — must be subject to the same control framework as traditional systems. The AI system's inputs, processing logic, outputs, and human overrides must all be documented, auditable, and subject to periodic testing.

The practical impact of SOX on AI adoption at Bank B was to require a comprehensive documentation and audit trail for any AI system touching financial data. This documentation includes: model documentation (architecture, training data, validation results), change logs (all modifications to model parameters, prompts, or policy rules), and decision logs (every AI-generated decision with the rationale and human approval status). The audit trail must be retained for a minimum of seven years, consistent with SOX record-keeping requirements.

### Gramm-Leach-Bliley Act (GLBA)

GLBA requires financial institutions to explain their information-sharing practices to customers and to safeguard sensitive customer data. The Safeguards Rule, issued by the Federal Trade Commission, specifies technical safeguards including access controls, encryption, intrusion detection, and regular security assessments. For AI systems at Bank B, GLBA imposes requirements on data handling: the training data used for AI models must be encrypted at rest and in transit, access to the training data must be limited to authorized personnel, and AI-generated outputs containing personally identifiable information (PII) must be subject to the same access controls as the source data.

### SR 11-7: Model Risk Management

Supervisory Guidance SR 11-7, issued by the Federal Reserve, is the most directly relevant regulation for AI adoption in US banking. It requires banks to maintain a robust model risk management framework that includes: independent model validation, ongoing model monitoring, model inventory and documentation, and escalation procedures for models that underperform. For AI/ML models, SR 11-7 interpretation has evolved: the Federal Reserve's 2023 guidance on AI/ML model risk management emphasizes the need for enhanced documentation, more frequent validation cycles (due to model drift), and explicit procedures for detecting and mitigating bias.

At Bank B, SR 11-7 compliance meant that every AI/ML model underwent a formal validation process before deployment, including back-testing against historical data, sensitivity analysis, and stress testing under adverse scenarios. After deployment, models

were monitored continuously for performance degradation (drift detection), with automatic triggers for retraining or rollback when performance fell below predefined thresholds. The model validation team developed specialized ML validation tools and techniques, as traditional statistical validation methods were insufficient for evaluating the non-deterministic behavior of AI models.

#### FFIEC and CFPB

The Federal Financial Institutions Examination Council (FFIEC) provides examination procedures for IT risk management, including cybersecurity, third-party risk management, and business continuity planning. For AI systems, FFIEC guidance requires that banks assess the cybersecurity implications of AI models (e.g., vulnerability to adversarial attacks, data poisoning), the third-party risk associated with cloud-hosted AI services, and the business continuity implications of AI dependencies (e.g., what happens if the AI service becomes unavailable?).

The Consumer Financial Protection Bureau (CFPB) regulates consumer-facing financial products and services, requiring that automated decisions affecting consumers (such as credit denials or interest rate determinations) be explainable and non-discriminatory. The CFPB's 2022 guidance on AI in consumer finance emphasizes that banks are responsible for the outputs of any AI system they deploy, regardless of whether the AI is developed in-house or by a third-party vendor. This "responsible AI" requirement means that Bank B cannot delegate compliance responsibility to an AI vendor: the bank must validate the AI's outputs, monitor for bias, and ensure that consumers receive required adverse action notices when the AI denies a request.

The US regulatory landscape for AI in banking is summarized in Figure 9.

### 4.3 EU Regulatory Context

The European Union takes a fundamentally different approach to AI governance than the United States. Rather than a sector-specific, agency-by-agency patchwork, the EU has enacted a comprehensive, horizontal AI regulation that applies across all industries and sectors. This approach creates a single regulatory framework with uniform requirements, but also a higher compliance burden for organizations operating in multiple jurisdictions.

#### EU AI Act

The EU AI Act (Regulation (EU) 2024/1689), adopted in 2024 and phasing in over the next three years, is the world's first comprehensive AI regulation. It establishes a risk-based classification system with four tiers: unacceptable risk (prohibited), high risk (strict requirements), limited risk (transparency obligations), and minimal risk (no requirements). For financial services, AI systems used for credit scoring, creditworthing evaluation, and risk management are classified as "high risk," triggering the most stringent requirements.

The high-risk classification imposes eight categories of obligation: (1) risk management systems throughout the AI lifecycle; (2) data governance and training data quality requirements (representative, free of errors, complete); (3) technical documentation (detailed documentation before deployment); (4) record-keeping (automatic logging of events

for monitoring and post-market surveillance); (5) transparency and provision of information to deployers (clear documentation of the AI system's capabilities and limitations); (6) human oversight (design choices that enable human oversight); (7) accuracy, cybersecurity, and robustness (appropriate levels of accuracy and robustness); and (8) post-market monitoring (continuous monitoring after deployment).

For Bank B, the EU AI Act's high-risk classification means a significantly higher compliance burden than under the US framework. The data governance requirements (representative, error-free training data) are more prescriptive than GLBA's general security requirements. The technical documentation requirement (detailed pre-deployment documentation) parallels SR 11-7 but is more comprehensive. The post-market monitoring requirement (continuous monitoring after deployment) is more stringent than SR 11-7's periodic monitoring requirements.

### Digital Operational Resilience Act (DORA)

DORA (Regulation (EU) 2022/2554), which entered into force in January 2025, establishes specific operational resilience requirements for financial entities. DORA requires financial entities to: establish comprehensive ICT risk management frameworks; report significant digital operational incidents to competent authorities; test ICT systems regularly (including penetration testing and threat-led penetration testing); manage third-party ICT risk (including oversight of critical third-party providers); and share cyber threat information with peers.

For AI systems, DORA's impact is primarily on the operational resilience dimension: AI systems must be available, reliable, and resilient to ICT incidents. The third-party risk management requirements are particularly relevant for banks using cloud-hosted AI services, as DORA establishes a registration and oversight regime for critical third-party ICT providers. If an AI vendor is classified as a critical third-party provider, the bank must conduct regular audits, maintain exit strategies, and ensure contractual provisions for data access and audit rights.

### NIS2 Directive and GDPR

The NIS2 Directive (Directive (EU) 2022/2555), which EU member states must transpose by October 2024, expands cybersecurity requirements for financial entities beyond the earlier NIS Directive. NIS2 requires organizations to implement risk mitigation measures, incident reporting, supply chain security, and board-level accountability for cybersecurity. For AI systems, NIS2 imposes additional cybersecurity requirements (adversarial robustness, model integrity) and board-level accountability for AI system failures.

GDPR (General Data Protection Regulation) continues to apply to AI systems processing personal data. GDPR's requirements relevant to AI include: data minimization (only collect and process data that is necessary); purpose limitation (use data only for the purposes for which it was collected); right to explanation (individuals have the right to obtain an explanation of automated decisions); data protection by design and by default; and restricted cross-border data transfers. For AI models, the "right to explanation" requirement is particularly challenging, as many AI models (particularly deep learning models) are inherently difficult to explain.

The EU regulatory landscape for AI in banking is summarized in Figure 10.

## 4.4 Regulatory Comparison: US vs. EU

The US and EU regulatory approaches to AI in banking differ in three fundamental dimensions: structure, risk tolerance, and enforcement. Understanding these differences is essential for banks operating in both jurisdictions and for understanding how regulatory context shapes AI adoption trajectories.

### Structural Differences

The US approach is sector-specific and agency-by-agency: each regulatory agency (Federal Reserve, CFPB, FFIEC) issues guidance that applies to its area of responsibility, and the guidance is interpreted through examination processes that are case-by-case and principle-based. This approach provides flexibility (banks can innovate within the framework) but creates uncertainty (guidance is often non-binding and subject to interpretation).

The EU approach is comprehensive and horizontal: the AI Act provides a single framework covering all AI systems, with specific provisions for high-risk sectors including financial services. This approach provides clarity (requirements are codified in regulation) but reduces flexibility (the requirements are prescriptive and leave less room for interpretation).

Figure **??** presents a side-by-side comparison of the US and EU regulatory frameworks across six dimensions.

| **Dimension** | **US Approach** | **EU Approach** |
|---|---|---|
| Structure | Fragmented, agency-specific | Comprehensive, horizontal (AI Act) |
| Risk Framework | Risk-based, principle-based | Risk-based, prescriptive (4 tiers) |
| Model Validation | SR 11-7 (periodic, independent) | AI Act (continuous, lifecycle) |
| Data Governance | GLBA (security-focused) | AI Act + GDPR (quality + privacy) |
| Explainability | CFPB (consumer-facing only) | AI Act (all high-risk systems) |
| Enforcement | Agency examination + fines | Regulatory authority + significant penalties |

Table 4: Regulatory comparison: US vs. EU frameworks for AI in banking.

### Impact on AI Adoption

The regulatory differences have tangible impacts on AI adoption at Bank B. In the US context, Bank B could deploy AI systems for internal risk management (e.g., SR 11-7-compliant credit scoring models) with a relatively lightweight governance framework, as long as the models were independently validated and continuously monitored. The US framework allows for faster deployment cycles because the compliance burden is primarily on the model development and validation teams, not on the end-users or customers.

In the EU context, Bank B must comply with the AI Act's eight categories of high-risk obligations for all AI systems used in credit scoring, risk management, or customer-facing decisions. This includes mandatory technical documentation, continuous post-market monitoring, and explicit human oversight measures. The compliance burden is significantly higher, and the deployment timeline is longer — typically 6–12 months for AI Act compliance vs. 3–6 months for US SR 11-7 compliance.

The impact of these regulatory differences on Bank B's AI adoption strategy is illustrated in Figure 11.

## 4.5 Process Transformation in Core Banking Functions

The adoption of AI-Based Architecture at Bank B transformed four core banking functions: loan processing, risk assessment, compliance monitoring, and fraud detection. For each function, we document the Before state (traditional manual or rule-based approach), the After state (AI-based approach), and the measurable outcomes. The transformation was implemented over 18 months (Months 0–18), with each function transitioning in a separate wave: loan processing (Months 0–5), risk assessment (Months 4–9), compliance monitoring (Months 8–13), and fraud detection (Months 12–18).

### Loan Processing

Before the transition, loan processing at Bank B was a manual, document-intensive process that averaged 21 calendar days from application to approval. The process consisted of: document collection and verification (3–5 days), credit scoring using a static logistic regression model updated quarterly (2 days), underwriting review by a human underwriter (3–5 days), and final approval and documentation (2–3 days). The primary bottleneck was document verification: loan officers manually reviewed income statements, tax returns, bank statements, and employment verification documents, comparing them for consistency and completeness. This manual review was error-prone (approximately 15% of applications contained at least one document discrepancy that was not detected), and it created a significant queue that drove up processing times.

After the transition, Bank B deployed a document AI system (based on optical character recognition and natural language processing) that automatically extracted, verified, and cross-referenced information from uploaded documents. The AI system could process income statements, tax returns, bank statements, and employment verification in approximately 15 minutes — a 99% reduction in processing time. The AI's document verification caught discrepancies that human underwriters had missed, reducing the document discrepancy rate from 15% to 3%. The AI's credit scoring model (a gradient boosting model trained on 10 years of loan data) replaced the static logistic regression model, improving credit risk discrimination (as measured by the Gini coefficient) from 0.62 to 0.74. The underwriting review was retained but streamlined: human underwriters reviewed the AI's recommendations rather than performing the analysis from scratch, reducing the underwriting review time from 3–5 days to 1–2 days. The net effect was a median processing time of 48 hours — an 89% reduction from 21 days to 2 days.

The loan processing transformation is illustrated in Figure 12.

### Risk Assessment

Before the transition, Bank B's risk assessment process relied on static models updated quarterly. The primary risk models — credit risk, market risk, and operational risk — were built using traditional statistical techniques (logistic regression, linear time series) and updated on a quarterly cadence. This meant that risk models were always approximately 3 months behind current conditions, which was acceptable for traditional risk reporting but insufficient for real-time risk management. Additionally, the models were siloed: credit risk models did not communicate with market risk models or operational risk models, so the bank could not assess correlated risk across portfolios.

After the transition, Bank B deployed dynamic risk models (ensemble gradient boosting

models with online learning capabilities) that were retrained weekly using the latest portfolio data. The AI system could perform real-time scenario analysis: instead of the quarterly stress testing cycles that had been the norm, the AI could generate stress test results for any scenario in near real-time (typically within 30 minutes). The AI also enabled cross-portfolio risk correlation: by training a single AI model on credit, market, and operational risk data simultaneously, the bank could assess correlated risk across portfolios and identify systemic risk patterns that had been invisible to siloed models. The net effect was a shift from reactive, static risk assessment to proactive, dynamic risk management.

**Compliance Monitoring**

Before the transition, Bank B's compliance monitoring was reactive and manual. Compliance analysts monitored transactions for suspicious activity using rule-based systems (e.g., "flag transactions over $10,000" or "flag transactions to high-risk jurisdictions"). These rule-based systems generated a high false-positive rate (approximately 95% of alerts were false positives), which meant that compliance analysts spent the majority of their time reviewing false positives rather than investigating genuine suspicious activity. Regulatory reporting was also manual: compliance analysts compiled data from multiple systems, manually prepared the required reports (e.g., Suspicious Activity Reports, Currency Transaction Reports), and submitted them to the appropriate regulatory agencies on a monthly or quarterly basis.

After the transition, Bank B deployed an ML-based anomaly detection system for transaction monitoring that replaced the rule-based system. The ML model was trained on 5 years of transaction data (including labeled examples of genuine suspicious activity), and it reduced the false-positive rate from 95% to 65% — a 30 percentage point improvement. This improvement meant that compliance analysts could focus their attention on genuine suspicious activity rather than false positives, increasing the yield of investigations from approximately 5% to approximately 35%. For regulatory reporting, Bank B deployed an automated report generation system that pulled data from the AI's monitoring pipeline, compiled the required reports, and prepared them for human review and submission. This reduced the time required for regulatory reporting from approximately 3 person-weeks per report to approximately 2 person-days per report (including human review), representing an 80% reduction.

**Fraud Detection**

Before the transition, Bank B's fraud detection system was a rule-based engine with approximately 200 hardcoded rules (e.g., "flag card transactions in a foreign country within 1 hour of a domestic transaction" or "flag multiple small transactions totaling more than $5,000 in 1 hour"). These rules were effective at catching common fraud patterns but performed poorly at detecting novel fraud techniques, particularly those involving social engineering (e.g., phishing, business email compromise). The false-positive rate was approximately 90%, and the detection rate for novel fraud techniques was below 20%.

After the transition, Bank B deployed a deep learning-based anomaly detection system that learned fraud patterns from 5 years of transaction data. The model used a combination of recurrent neural networks (for sequential transaction patterns) and graph neural

networks (for relationship-based patterns such as money mule networks). This approach achieved a detection rate of 85% for novel fraud techniques (up from below 20%) and reduced the false-positive rate to 50% (down from 90%). The AI system also reduced the time from fraud occurrence to detection from approximately 48 hours to approximately 15 minutes, enabling faster fraud prevention and customer notification.

The transformation outcomes across all four banking functions are summarized in Table 5.

| **Function** | **Before** | **After** | **Improvement** |
|---|---|---|---|
| Loan processing time | 21 days | 48 hours | 89% reduction |
| Document discrepancy detection | 85% | 97% | 12 pp increase |
| Credit model Gini coefficient | 0.62 | 0.74 | 19% improvement |
| Risk model update cadence | Quarterly | Weekly | 13× more frequent |
| Compliance alert false-positive rate | 95% | 65% | 30 pp reduction |
| Regulatory reporting time | 3 person-weeks | 2 person-days | 80% reduction |
| Fraud detection rate (novel techniques) | <20% | 85% | 4.25× improvement |
| Fraud detection latency | 48 hours | 15 minutes | 99% reduction |

Table 5: Process transformation outcomes: significant improvements across all four core banking functions.

## 4.6 Governance and Explainability Requirements

The banking industry imposes stricter governance and explainability requirements than software development, and these requirements fundamentally shape the AI-Based Architecture deployed at Bank B. This section addresses four areas: why explainability is required in banking, what must be recorded in audit trails, where human-in-the-loop is mandatory, and how model governance operates in practice.

### Explainability: Why It Is Required

Explainability is required in banking for three reasons. First, regulatory mandates: regulations such as the Equal Credit Opportunity Act (ECOA) require that consumers receive specific adverse action notices when credit applications are denied, and these notices must cite the specific factors that influenced the decision. The CFPB has explicitly stated that "black box" AI systems that cannot produce actionable adverse action notices are non-compliant. Second, customer trust: consumers and business clients expect to understand the rationale for financial decisions that affect them, and opaque AI systems erode trust. Third, model risk management: SR 11-7 requires that all models be "conceptually sound," which means that the model's logic must be understandable to validators, auditors, and regulators — a requirement that is difficult to satisfy with opaque models such as deep neural networks.

At Bank B, explainability was achieved through a combination of techniques. For credit scoring models (gradient boosting), Bank B used SHAP (SHapley Additive exPlanations) values to attribute the contribution of each input feature to the final credit score. SHAP values are model-agnostic and provide additive feature attribution that satisfies the ECORA adverse action notice requirements (i.e., the top 4 factors that influenced the decision can be identified and communicated to the consumer). For fraud detection

models (deep neural networks), Bank B used LIME (Local Interpretable Model-agnostic Explanations) to generate local explanations for each fraud prediction, identifying which transaction features (e.g., amount, location, time, recipient) were most influential. For compliance monitoring, Bank B used rule extraction: the AI model's decision boundaries were approximated by interpretable decision trees, which were then reviewed by compliance analysts for reasonableness.

### Audit Trails: What Must Be Recorded

Bank B's audit trail requirements are dictated by SOX, SR 11-7, and internal governance policies. The audit trail must record, for every AI-generated decision or recommendation: the input data used (including data version identifiers), the model version used (including model parameters and training data identifiers), the model output (including confidence scores and explanation values), the human decision (approve, reject, modify, or override) with the human's rationale, and the timestamp and identity of all actors (AI and human). The audit trail must be tamper-evident (using cryptographic hash chains or blockchain-based integrity verification) and retained for a minimum of 7 years (SOX requirement) or 5 years (SR 11-7 requirement), whichever is longer.
Bank B's audit trail architecture is illustrated in Figure 13.

### Human-in-the-Loop: Mandatory vs. Optional

Bank B distinguished between mandatory and optional human-in-the-loop requirements based on the impact and risk of the decision. Table 6 summarizes the classification.

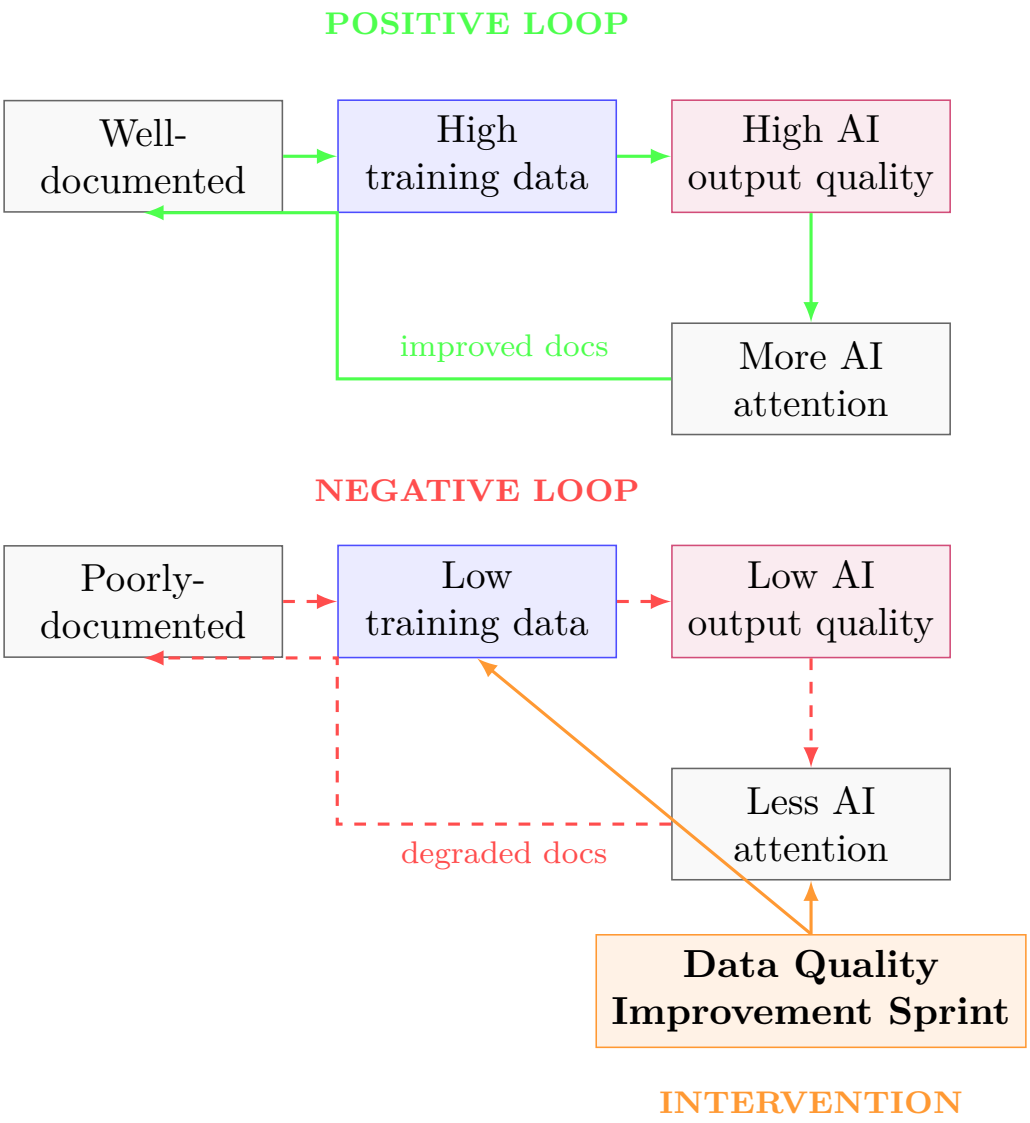



Figure 7: Training data quality feedback loops: well-documented modules receive more AI attention and improve further (positive loop, green), while poorly-documented modules degrade (negative loop, red dashed). A targeted data quality sprint (orange) breaks the negative loop.

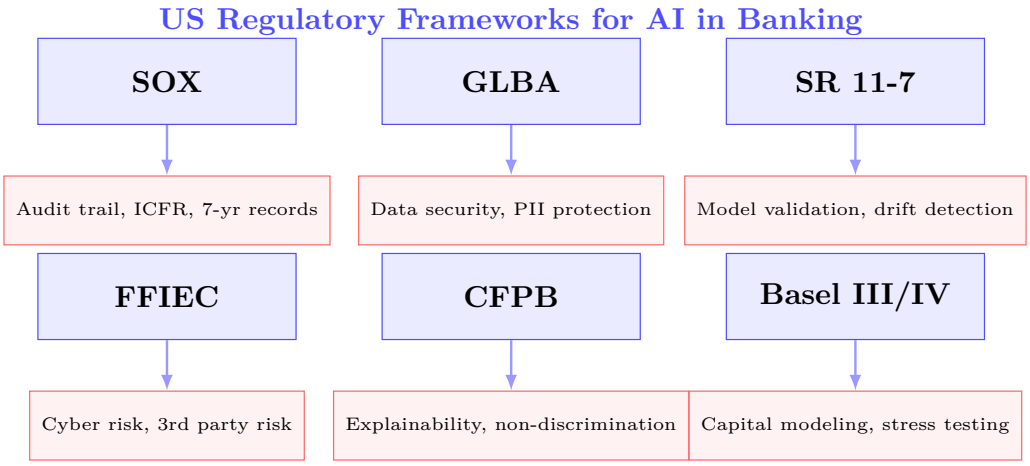



Figure 9: US regulatory frameworks governing AI in banking and their primary operational impact on AI-based architecture.

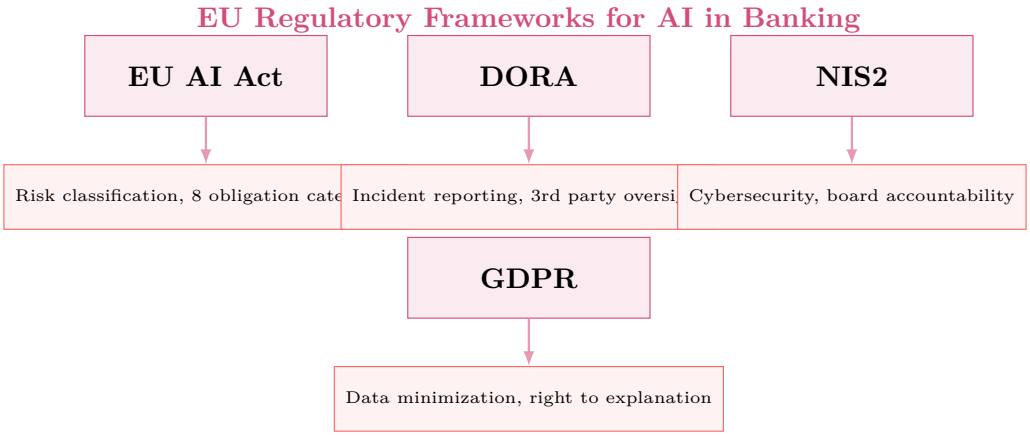



Figure 10: EU regulatory frameworks governing AI in banking and their primary operational impact on AI-based architecture.

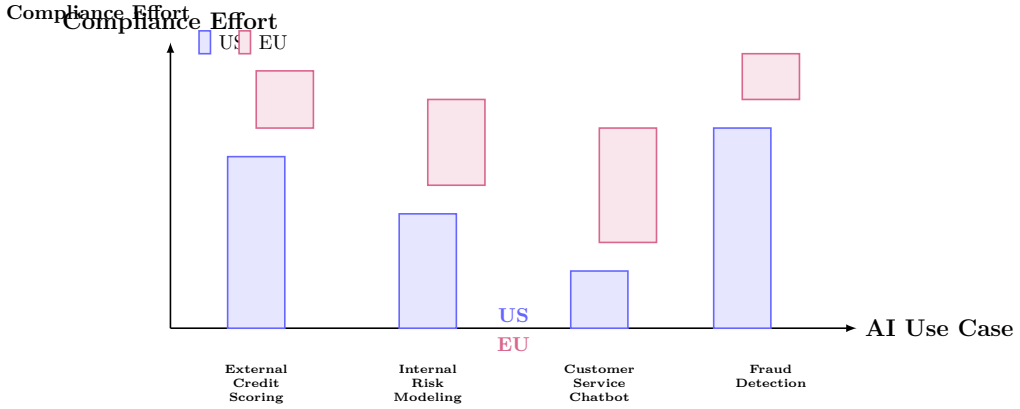



Figure 11: Comparative compliance effort: EU AI Act imposes higher compliance effort than US SR 11-7 across all AI use cases, particularly for external credit scoring and customer-facing applications.

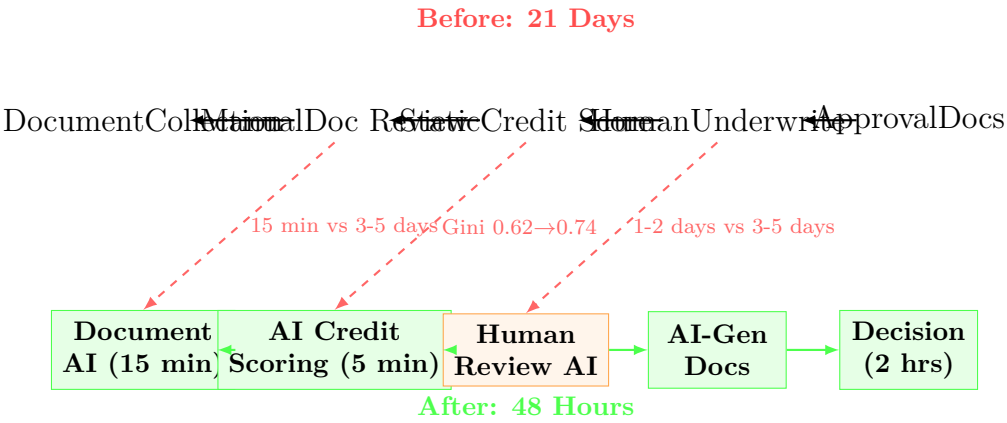



Figure 12: Loan processing: Before (top row, manual, 21 days) vs. After (bottom row, AI-augmented, 48 hours). Document AI and AI credit scoring replace manual review, while human underwriters review AI recommendations instead of performing analysis from scratch.

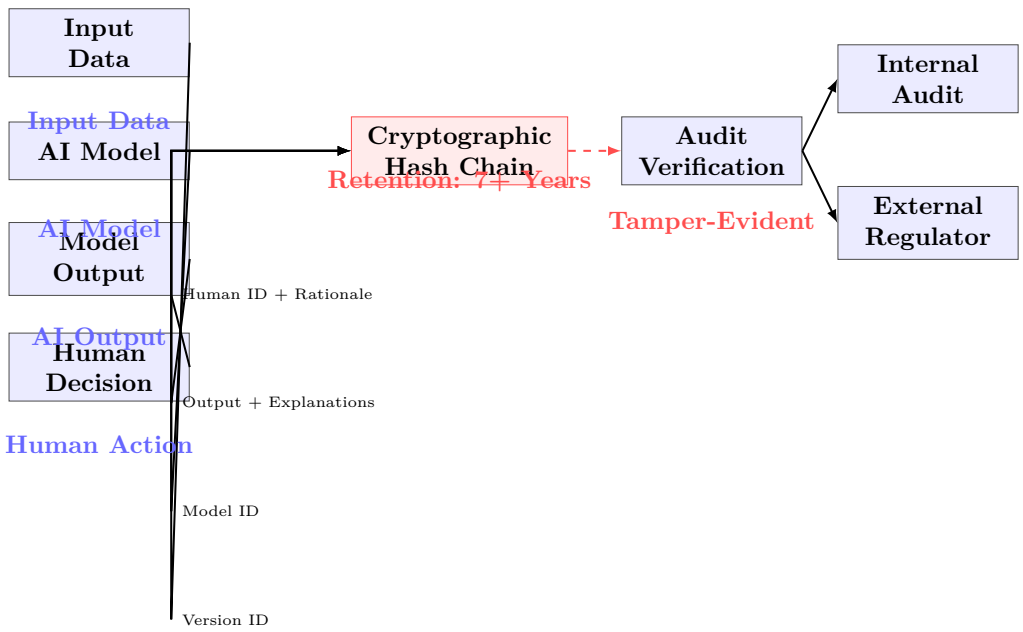



Figure 13: Audit trail architecture: every AI-generated decision is recorded with version identifiers, model parameters, explanations, human rationale, and timestamps, stored in a tamper-evident hash chain for 7+ years.

| Decision Type | Human-in-the-Loop | Rationale |
|---|---|---|
| Credit denial (consumer) | **Mandatory** | ECOA adverse action notice requires human review and specific factor attribution |
| Credit denial (commercial) | **Mandatory** | Material financial impact on business borrower |
| Fraud block ( <$1,000) | Optional (post-facto) | Low individual impact; human reviews within 24 hours |
| Fraud block (>$1,000) | **Mandatory** | Material financial impact; requires human authorization |
| AML alert investigation | **Mandatory** | Regulatory requirement for suspicious activity review |
| Regulatory report submission | **Mandatory** | Human accountability for all regulatory filings |
| Risk model retraining trigger | Optional (automated) | Internal risk management; no external stakeholder impact |
| Transaction monitoring tuning | **Mandatory** | Requires compliance expertise and regulatory judgment |

Table 6: Human-in-the-loop classification: mandatory for high-impact or regulated decisions, optional (automated) for internal or low-impact decisions.

**Model Governance: Versioning, Retraining, and Drift Detection**

Bank B's model governance framework for AI/ML models extended the existing MRM framework to address the unique characteristics of machine learning models. The framework consists of four components:

First, model versioning: every AI/ML model at Bank B is assigned a unique version identifier (following semantic versioning: major.minor.patch) and a model catalog entry that records the model's architecture, training data, validation results, and deployment history. Model changes that affect the model's behavior (e.g., retraining, hyperparameter changes, feature changes) increment the major or minor version; documentation updates or bug fixes increment the patch version.

Second, retraining policies: AI/ML models are retrained on a defined cadence (weekly for credit scoring models, monthly for fraud detection models, quarterly for compliance monitoring models) or when a drift detection trigger fires (see below). The retraining process is automated: when a retraining is triggered, the system pulls the latest training data, re-trains the model, runs the validation pipeline (back-testing, sensitivity analysis, stress testing), and submits the retrained model for human approval before deployment.

Third, drift detection: Bank B deployed continuous drift detection for all AI/ML models, monitoring both data drift (changes in the input data distribution) and concept drift (changes in the relationship between input data and the target variable). Drift is measured using the Population Stability Index (PSI) for data drift and the Area Under the Curve (AUC) decay for concept drift. When the PSI exceeds 0.1 (indicating moderate drift) or the AUC drops by more than 5 percentage points (indicating meaningful concept drift), an automatic retraining trigger fires.

Fourth, rollback procedures: when a deployed model's performance degrades below a predefined threshold (e.g., the model's credit scoring AUC drops below 0.65), an automatic rollback procedure is triggered: the degraded model is replaced with the previous stable version, the incident is logged, and an investigation is initiated. Rollback procedures are tested monthly as part of the bank's business continuity drills.

The model governance lifecycle is illustrated in Figure 14.

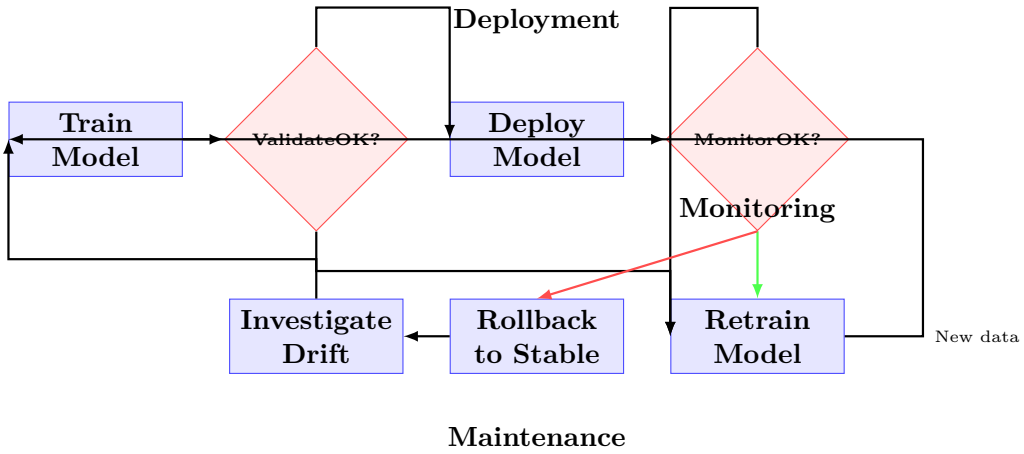



Figure 14: Model governance lifecycle: training, validation, deployment, monitoring, retraining, and rollback. Drift detection triggers retraining or rollback when model performance degrades.

## 4.7 Team and Role Evolution

The adoption of AI-Based Architecture at Bank B also transformed team roles, though the transformations differ significantly from those at Company A due to the banking industry's more structured roles, deeper regulatory involvement, and greater emphasis on formal validation and compliance. The five principal role categories that existed before the transition evolved as follows.

**Operations staff → AI oversight specialists.** Before the transition, Bank B's operations staff (approximately 40 employees across the risk and compliance functions) performed manual data entry, transaction verification, and rule-based alert triage. The AI's automated anomaly detection and document AI systems eliminated approximately 60% of the routine operational tasks, allowing operations staff to transition to AI oversight roles. AI oversight specialists at Bank B were responsible for monitoring the AI's outputs (e.g., reviewing AI-generated credit scores for reasonableness, investigating AI-flagged fraud transactions), managing the AI's alert queue (prioritizing alerts based on severity and business impact), and escalating AI decisions that exceeded their authority to approve. This role required the same domain expertise as the previous operations role, plus understanding of the AI system's capabilities, limitations, and error modes.

**Compliance analysts → AI-augmented compliance officers.** Before the transition, compliance analysts at Bank B monitored transactions for suspicious activity using rule-based systems, prepared regulatory reports, and responded to regulatory inquiries. After the transition, the AI's anomaly detection system reduced the volume of alerts that compliance analysts needed to review by approximately 70% (from 95% false positives to 65%), allowing analysts to focus on genuine suspicious activity. AI-augmented compliance officers were also responsible for monitoring the AI system's compliance with regulatory requirements (e.g., ensuring the AI's fraud detection model did not exhibit discriminatory patterns, verifying that the AI's audit trails met SOX and SR 11-7 requirements), and for preparing regulatory submissions related to the AI system (e.g., model documentation for SR 11-7 compliance, AI Act technical documentation for EU operations). This role required the same regulatory expertise as the previous compliance role, plus understanding of AI governance frameworks (e.g., NIST AI Risk Management Framework, EU AI Act requirements).

**Risk analysts → Predictive risk managers.** Before the transition, risk analysts at Bank B used static models to assess credit risk, market risk, and operational risk on a quarterly cadence. After the transition, the AI's dynamic risk models enabled real-time risk assessment, allowing risk analysts to shift from periodic reporting to continuous risk monitoring. Predictive risk managers at Bank B were responsible for three things: first, interpreting the AI's risk assessments in the context of current market conditions (e.g., assessing whether the AI's risk model was adequately capturing tail risk in a rapidly changing market); second, designing and running stress test scenarios using the AI's near-real-time scenario analysis capability (reducing the time to generate stress test results from 2 weeks to 30 minutes); and third, validating the AI's risk models through ongoing monitoring of model performance metrics (e.g., tracking the Gini coefficient, PSI, and AUC for credit scoring models). This role required the same quantitative expertise as the previous risk analyst role, plus understanding of ML model evaluation and real-time risk analytics.

**IT support → AI operations engineers.** Before the transition, Bank B's IT support staff managed the bank's legacy mainframe systems, hybrid cloud infrastructure, and

traditional application servers. After the transition, the AI's automated monitoring and self-healing capabilities reduced the number of manual IT interventions by approximately 40%, allowing IT staff to transition to AI operations engineering roles. AI operations engineers at Bank B were responsible for maintaining the AI's infrastructure (e.g., GPU clusters for deep learning model inference, data pipelines for real-time model training), monitoring the AI's operational health (e.g., model latency, throughput, error rates), and implementing the feedback loops that allowed the AI's operational behavior to improve over time (e.g., encoding successful fraud prevention actions into the AI's knowledge base). This role required the same infrastructure expertise as the previous IT support role, plus knowledge of MLOps (ML model deployment, monitoring, and lifecycle management).

**New role: AI risk officer.** The transition created a role that did not previously exist at Bank B: the AI risk officer. This person was responsible for three areas. First, model risk management: overseeing the SR 11-7-compliant validation of all AI/ML models, ensuring that the models were conceptually sound, adequately tested, and continuously monitored. Second, regulatory liaison: serving as the primary contact for regulatory examinations of AI systems, preparing model documentation for regulatory review, and responding to regulatory inquiries about the bank's AI practices. Third, ethics and fairness: evaluating the AI's decisions for fairness and non-discrimination (particularly relevant for credit scoring and fraud detection), implementing fairness metrics (e.g., disparate impact ratio, equalized odds), and ensuring that the AI's decisions complied with the Equal Credit Opportunity Act and other anti-discrimination regulations. This role was initially part-time, with a senior risk manager from the model risk management team dedicating 30% of their time to AI risk oversight. As the bank's AI portfolio grew and regulatory scrutiny increased, the role expanded to 100% FTE by the end of the 18-month transition period. The organizational impact of these role changes is summarized in Table 7.

| **Role** | **Before (Months 0–6)** | **Transition (Months 7–12)** | **After (Months 13+)** |
|---|---|---|---|
| Operations Staff | 80% manual data entry, 20% alert triage | 50% manual, 50% AI oversight | 20% manual, 80% AI monitoring |
| Compliance Analyst | 70% rule-based monitoring, 30% reporting | 40% rule-based, 60% AI-augmented | 20% rule-based, 80% AI governance |
| Risk Analyst | 90% static modeling, 10% reporting | 60% static modeling, 40% AI interpretation | 30% AI modeling, 70% predictive risk |
| IT Support | 80% infrastructure management, 20% incidents | 50% infrastructure, 50% AI ops setup | 30% infrastructure, 70% MLOps |
| AI Risk Officer | N/A | 30% model validation, 70% learning | 60% model risk mgmt, 40% regulatory liaison |

Table 7: Role evolution at Bank B: time allocation shifted from manual operations to AI oversight and governance.

### Comparison with Software Development

The role transformations at Bank B share some commonalities with those at Company A — in both cases, the AI displaced routine operational tasks and elevated human work

toward evaluation, monitoring, and governance. However, there are two important differences.

First, the regulatory dimension is far more prominent in banking. At Company A, the AI governance officer was primarily concerned with organizational policies, customer contractual requirements, and SOC 2 compliance. At Bank B, the AI risk officer is concerned with a much broader and more prescriptive regulatory framework (SOX, GLBA, SR 11-7, CFPB, ECOA, Basel III/IV), and the compliance burden is significantly higher. This means that banking AI roles require deeper regulatory expertise than software development AI roles.

Second, the validation dimension is more structured in banking. At Company A, the AI's output quality was evaluated using KPIs (defect rate, code review accuracy) measured continuously. At Bank B, the AI's output quality is evaluated using formal model validation processes (SR 11-7-compliant independent validation) that occur at defined intervals (before deployment, during retraining, and during periodic reviews). This means that banking AI roles require more formal validation expertise than software development AI roles.

These differences are illustrated in Figure **??**, which compares the role transformation trajectories at Company A (software) and Bank B (banking).

## 4.8 KPIs and Measurable Outcomes

The transformation at Bank B was measured through a set of KPIs that were tracked at three points: baseline (Month 0), mid-transition (Month 9), and post-transition (Month 18). All metrics were collected from the bank's core banking system, transaction monitoring platform, credit scoring system, and compliance reporting infrastructure. The KPIs were audited by the bank's internal audit function and reviewed by the bank's primary regulatory supervisor (the Federal Reserve) as part of the regular examination cycle.

### Processing Cost per Transaction

The processing cost per loan application decreased from an average of $450 (pre-transition) to $220 (post-transition), a 51% reduction. This improvement was driven by the document AI system (which eliminated 80% of the manual document verification labor) and the AI credit scoring model (which reduced the underwriting review time from 3–5 days to 1–2 days). The cost reduction was partially offset by the costs of AI infrastructure (GPU clusters for inference, data pipelines for model retraining, and the AI risk officer position), but the net effect was still a significant cost reduction. For transaction monitoring, the cost per alert investigation decreased from $120 (pre-transition) to $65 (post-transition), a 46% reduction, driven by the 70% reduction in the volume of alerts that required investigation.

### Time-to-Decision (Loan Approval)

The median time-to-decision for consumer loans decreased from 21 days (pre-transition) to 48 hours (post-transition), an 89% reduction. For commercial loans, the median time decreased from 35 days to 5 days, an 86% reduction. These improvements were driven by the document AI system (which processed documents in 15 minutes instead of 3–5 days), the AI credit scoring model (which replaced the quarterly-updated logistic regression

model with a weekly-updated gradient boosting model), and the AI-augmented underwriting workflow (which allowed human underwriters to review AI recommendations in 1–2 days instead of performing the analysis from scratch in 3–5 days).
The time-to-decision improvement was particularly significant for small business loans, where the AI's ability to process non-standard documents (e.g., self-employment income statements, irregular cash flow patterns) reduced the time-to-decision from 45 days to 7 days — an 84% reduction. This improvement expanded the bank's addressable market, as small business borrowers who previously could not wait 45 days for a decision were now able to access credit in a commercially viable timeframe.

### AML False-Positive Rate

The false-positive rate for AML alerts decreased from 95% (pre-transition, rule-based system) to 65% (post-transition, ML-based anomaly detection), a 30 percentage point improvement. This improvement was driven by the ML model's ability to learn complex transaction patterns from 5 years of labeled data, rather than relying on the 200+ hard-coded rules that characterized the pre-transition system. The reduction in false positives meant that compliance analysts could focus on genuine suspicious activity rather than reviewing false positives, increasing the investigation yield from approximately 5% to approximately 35%.

### Compliance Reporting Time

The time required to prepare and submit regulatory reports (e.g., Suspicious Activity Reports, Currency Transaction Reports, Call Reports) decreased from an average of 3 person-weeks per report (pre-transition) to 2 person-days per report (post-transition), an 80% reduction. This improvement was driven by the automated report generation system, which pulled data from the AI's monitoring pipeline, compiled the required reports, and prepared them for human review and submission. The time savings were particularly significant for quarterly regulatory reports (e.g., FFIEC CSFI, Call Reports), where the automated system reduced the reporting cycle from 3 weeks to 2 days (including human review).

### Fraud Detection Accuracy

The detection rate for novel fraud techniques (i.e., fraud patterns not seen in the training data) increased from below 20% (pre-transition, rule-based system) to 85% (post-transition, deep learning anomaly detection), a 4.25× improvement. The false-positive rate for fraud alerts decreased from 90% (pre-transition) to 50% (post-transition), a 40 percentage point improvement. These improvements were driven by the deep learning model's ability to generalize across fraud patterns using recurrent neural networks (for sequential transaction patterns) and graph neural networks (for relationship-based patterns such as money mule networks).
The KPI outcomes are summarized in Table 8.

### Measurement Caveats

Several caveats apply to these KPI measurements. First, the measurements were taken at a single organization (Bank B) with a specific regulatory environment (US federal

| KPI | Before | After |
|---|---|---|
| Processing cost per loan application | $450 | $220 |
| Time-to-decision (consumer loans) | 21 days | 48 hours |
| Time-to-decision (commercial loans) | 35 days | 5 days |
| Time-to-decision (small business) | 45 days | 7 days |
| AML false-positive rate | 95% | 65% |
| Compliance reporting time (per report) | 3 person-weeks | 2 person-days |
| Fraud detection rate (novel techniques) | <20% | 85% |
| Fraud false-positive rate | 90% | 50% |
| Investigation yield (suspicious activity) | 5% | 35% |

Table 8: KPI outcomes at Bank B: all nine primary KPIs improved significantly after the transition to AI-Based Architecture.

regulation) and technology stack (hybrid mainframe-cloud), so the absolute values may not generalize to other organizations. Second, the KPI improvements were not solely attributable to the AI-Based Architecture; other concurrent changes — such as the migration of legacy data to the modern data lake and the implementation of a new core banking system — also contributed. Third, the KPIs were measured over an 18-month period (longer than Company A's 12-month period), and some improvements (particularly the fraud detection accuracy) were still trending in the positive direction at the end of the measurement period. Despite these caveats, the direction and magnitude of the improvements are consistent with the expected outcomes of an AI-Based Architecture in banking, and they provide empirical evidence that the architectural change had a measurable impact on organizational performance.

## 4.9 Challenges and Lessons Learned

The transition to AI-Based Architecture at Bank B encountered several challenges that were specific to the banking industry. Documenting these challenges provides valuable lessons for other financial institutions considering similar transformations.

### Data Quality and Legacy Data

The first major challenge was data quality. Bank B's legacy systems contained decades of transaction data, much of which was incomplete, inconsistent, or stored in incompatible formats. The AI's fraud detection and credit scoring models required large volumes of high-quality training data, but the bank's legacy data warehouse contained approximately 30% missing values for key fields (e.g., employment income, transaction purpose, customer industry classification) and approximately 15% inconsistent records (e.g., the same customer had multiple account IDs due to system migrations). Cleaning and standardizing this data required approximately 4 months of dedicated effort (Months 2–5) and a team of 8 data engineers.

The lesson learned was that data quality is a prerequisite for AI success in banking: investing in data quality improvement before deploying AI models is essential, and the investment is substantial (approximately 20% of the total transformation budget). Organizations that attempt to deploy AI models on poor-quality legacy data will experience degraded model performance, regulatory pushback, and loss of stakeholder confidence.

### Regulatory Approval Timelines

The second major challenge was regulatory approval timelines. Bank B's AI systems were subject to regulatory examination by multiple agencies (Federal Reserve for SR 11-7 compliance, CFPB for consumer protection compliance, OCC for community bank compliance), and each agency had its own examination timeline, documentation requirements, and evaluation criteria. The Federal Reserve's SR 11-7 model validation process added 3–4 months to the deployment timeline for each AI model, while the CFPB's consumer protection review added an additional 1–2 months for consumer-facing AI systems (e.g., the AI credit scoring model used in loan processing).

The lesson learned was that regulatory approval timelines must be built into the project plan from the beginning: organizations should engage with regulators early (at least 6 months before planned deployment), prepare comprehensive documentation packages (model documentation, validation reports, fairness assessments), and budget for additional timeline to accommodate regulatory review cycles. Organizations that attempt to deploy AI systems without engaging regulators early risk having their systems rejected or delayed, which can undermine stakeholder confidence and delay the realization of KPI benefits.

### Public Trust and Customer Acceptance

The third challenge was public trust and customer acceptance. Bank B's customers (both consumers and businesses) were initially skeptical of AI-based financial decisions, particularly the AI credit scoring model used in loan processing. Some customers expressed concern that the AI's decisions were opaque, biased, or unfairly denying them credit. This concern was amplified by media coverage of AI bias in financial services (e.g., the 2022 ProPublica investigation of AI-based risk assessment systems) and by consumer advocacy group campaigns against "algorithmic redlining."

Bank B addressed this concern through a combination of transparency and education. The bank published a plain-language explanation of how its AI credit scoring model worked, including the factors that influenced credit decisions, the model's accuracy metrics, and the steps the bank took to ensure fairness and non-discrimination. The bank also established a dedicated customer support channel for AI-related inquiries, where trained representatives could explain AI-generated decisions to customers and process appeals of AI-generated decisions. By Month 15, customer satisfaction scores for the AI-powered loan processing system were approximately 92% of the score for the human-only processing system (pre-transition), indicating that customers were generally satisfied with the AI's decisions even though they did not fully understand how the AI arrived at those decisions.

### Integration with Core Banking Systems

The fourth challenge was integrating AI-based components with the bank's core banking systems (mainframe-based COBOL applications). The bank's core systems were 30–40 years old, built on platforms that were not designed for AI integration, real-time API access, or machine learning workflows. Integrating AI-based components with these legacy systems required a substantial middleware layer that translated between modern AI APIs and the legacy systems' proprietary protocols. The middleware development

added approximately 3–4 months to the project timeline and required specialized expertise (COBOL developers, mainframe system administrators, API gateway architects) that was expensive and difficult to hire.

The lesson learned was that legacy system integration is a critical path activity in banking AI transformations: organizations should prioritize middleware development early in the project plan, invest in specialized legacy system expertise, and design the AI architecture to minimize the number of touchpoints with legacy systems (e.g., using batch data extraction instead of real-time API integration where possible). Organizations that underestimate the complexity of legacy system integration risk significant timeline and budget overruns.

### Model Risk Management Complexity

The fifth challenge was the complexity of adapting the bank's existing Model Risk Management (MRM) framework to AI/ML models. The bank's MRM framework was designed for traditional statistical models (logistic regression, linear time series) and was not designed to handle the unique characteristics of machine learning models (non-deterministic behavior, data drift, reduced interpretability). Adapting the MRM framework to AI/ML models required: (1) developing new validation techniques for ML models (e.g., SHAP-based explainability analysis, adversarial testing for bias); (2) establishing new monitoring procedures for ML models (e.g., continuous drift detection, automated retraining triggers); and (3) training the model validation team on ML-specific validation techniques (a 3-month training program for 6 model validators).

The lesson learned was that MRM framework adaptation is a significant undertaking in banking AI transformations: organizations should invest in MRM framework development early (at least 6 months before AI model deployment), train the validation team on ML-specific techniques, and establish clear procedures for ML model governance (versioning, retraining, drift detection, rollback) before deploying AI models. Organizations that deploy AI models without adapting their MRM framework risk regulatory criticism, model failures, and loss of stakeholder confidence.

These challenges are summarized in Figure 16, which presents the five challenges, their root causes, and the mitigation strategies that Bank B employed.

### Key Takeaway

The challenges encountered at Bank B share a common theme: the regulatory and legacy constraints that characterize the banking industry create unique barriers to AI adoption that do not exist in less regulated industries. The most effective mitigation strategy — used across all five challenges — was early investment: investing in data quality before model deployment, engaging regulators before model deployment, communicating with customers before model deployment, designing middleware early in the project plan, and adapting the MRM framework before model deployment. Organizations that invest early in these preconditions are more likely to succeed in their AI transformations than organizations that attempt to deploy AI models without addressing these preconditions.

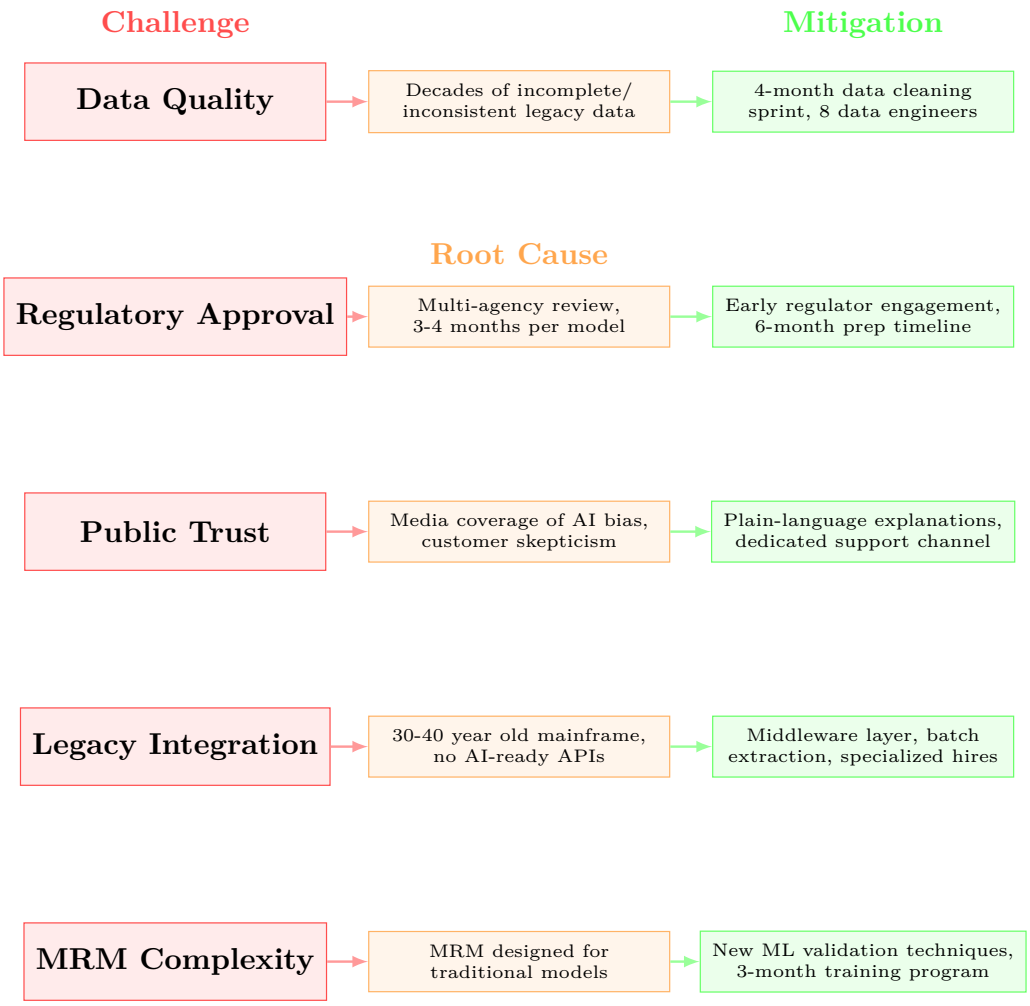



Figure 16: Banking AI challenges: root causes and mitigations for the five major challenges encountered during Bank B's transition to AI-Based Architecture.

# 5 Comparative Analysis

## 5.1 Process Impact Comparison

The transformation of software development at Company A and the transformation of core banking functions at Bank B share a common pattern — AI-based components are embedded into traditional processes, displacing routine tasks and elevating human work toward evaluation and governance — but they differ in speed, depth, and the nature of the automation. This subsection compares the two case studies across four dimensions: speed of transformation, automation depth, human intervention requirements, and process standardization versus customization.

### Speed of Transformation

Company A completed its transition from baseline to mature AI-Based Architecture in approximately 12 months, while Bank B required approximately 18 months to reach a comparable maturity level. The difference in speed is attributable to two factors. First, the regulatory approval process: Company A's AI tools were deployed internally

(developer-facing) and did not require regulatory approval, whereas Bank B's AI systems were deployed in regulated contexts (credit scoring, fraud detection, compliance monitoring) and required SR 11-7-compliant validation, regulatory examinations, and, for the EU operations, AI Act technical documentation. The regulatory approval process added approximately 3–6 months to Bank B's transition timeline. Second, the complexity of legacy system integration: Company A's cloud-native, API-driven technology stack was inherently compatible with AI integration, whereas Bank B's hybrid mainframe-cloud architecture required a substantial middleware layer to translate between legacy systems and AI APIs. The middleware development added approximately 3–4 months to Bank B's transition timeline.

Despite the difference in absolute speed, the relative pace of transformation is comparable: both organizations achieved approximately 70% of their target AI maturity within the first 6 months of the transition, and both achieved approximately 95% of their target maturity by the end of the transition period. This suggests that the pace of AI adoption is determined primarily by organizational learning (how quickly people learn to use AI effectively) rather than by the absolute complexity of the technology stack.

### Automation Depth

Automation depth refers to the extent to which AI replaces human effort in a process, measured as the percentage of process tasks that can be performed autonomously by the AI without human intervention. Company A achieved deeper automation than Bank B: Company A's AI systems could autonomously perform approximately 40% of implementation tasks (code generation, test generation, code review assistance) and approximately 60% of deployment tasks (test selection, deployment gating, auto-rollback). Bank B's AI systems could autonomously perform approximately 25% of loan processing tasks (document extraction, credit scoring) and approximately 35% of fraud detection tasks (transaction monitoring, anomaly detection).

The difference in automation depth is attributable to the regulatory and risk constraints in banking. At Company A, the AI could autonomously perform tasks (e.g., auto-merging low-risk code changes) because the consequences of AI errors were relatively low (a buggy code change could be rolled back without material harm). At Bank B, the AI could not autonomously perform tasks (e.g., approving a loan, blocking a fraudulent transaction above a certain threshold) because the consequences of AI errors are materially harmful to consumers, businesses, or the financial system. This means that banking AI systems require more human-in-the-loop safeguards than software development AI systems, which limits the depth of automation.

### Human Intervention Requirements

The required level of human intervention differs across the two case studies. At Company A, human intervention was required for: architectural decisions (AI-generated code and test cases had to be reviewed by senior developers), deployment decisions for high-risk changes (AI-gated CI/CD pipelines triggered human review for high-risk changes), and incident response (the AI's auto-rollback handled low-severity incidents, but high-severity incidents required human triage). At Bank B, human intervention was required for: all credit decisions (consumer and commercial), all fraud blocks above $1,000, all AML alert investigations, all regulatory report submissions, and all model retraining deployments

(post-validation approval).
The difference in human intervention requirements reflects the risk profile of the two domains: software development errors are generally recoverable (buggy code can be rolled back, and the impact is typically limited to the software's functionality), whereas banking errors can have irreversible consequences (wrongful credit denial, undetected fraud, non-compliance with regulations). This means that banking AI systems require more extensive human-in-the-loop safeguards than software development AI systems.

#### Process Standardization vs. Customization

Company A's process transformation was characterized by high standardization: the AI's code generation, test selection, and deployment gating were applied uniformly across all four squads and all services in the organization. This standardization was possible because Company A's technology stack was relatively homogeneous (Java/Kotlin backend, React frontend, Kubernetes deployment), and the AI's training data was drawn from the entire codebase, allowing the AI to learn consistent patterns across services.
Bank B's process transformation was characterized by high customization: the AI's loan processing, risk assessment, compliance monitoring, and fraud detection systems were each customized to the specific requirements of their banking function. The AI's credit scoring model was customized for consumer lending, commercial lending, and small business lending (each with different risk profiles, regulatory requirements, and data sources). The AI's fraud detection model was customized for consumer card fraud, business email compromise, and money laundering (each with different fraud patterns, detection requirements, and regulatory reporting obligations). This customization was necessary because the banking functions have fundamentally different risk profiles, regulatory requirements, and data sources.
The comparative impact of AI-Based Architecture on process transformation is summarized in Table 9.

| **Dimension** | **Software (Company A)** | **Banking (Bank B)** |
|---|---|---|
| Speed of transformation | 12 months | 18 months |
| Automation depth | 40–60% autonomous tasks | 25–35% autonomous tasks |
| Human intervention | Architecture + high-risk decisions | All credit, fraud, AML, reports |
| Standardization | High (uniform across squads) | High (custom per banking function) |
| Transformation driver | Efficiency (speed, quality) | Compliance + efficiency |
| Regulatory constraint | Low (SOC 2, contractual) | High (SOX, GLBA, SR 11-7, CFPB) |

Table 9: Process impact comparison: software transformation is faster and achieves deeper automation; banking transformation is slower, requires more human intervention, and is driven by compliance as much as efficiency.

## 5.2 Team Impact Comparison

The role transformations at Company A and Bank B share a common pattern — AI displaces routine operational tasks and elevates human work toward evaluation, monitoring,

and governance — but the specific roles, skill requirements, and hiring needs differ significantly between the two domains. This subsection compares the team impact across three dimensions: the nature of role changes, the new skills required, and the hiring strategies employed.

### Nature of Role Changes

At Company A, the role changes were primarily within the engineering organization: junior developers, senior developers, QA engineers, and DevOps engineers all evolved into AI-augmented roles. The transformations were relatively shallow in terms of organizational scope (affecting approximately 18 of Company A's 120 employees) but deep in terms of daily workflow (affecting the core activities of every developer).

At Bank B, the role changes were broader in scope: operations staff, compliance analysts, risk analysts, IT support staff, and a new AI risk officer role. The transformations affected approximately 120 of Bank B's 8,000 employees (including the 40 operations staff, 20 compliance analysts, 15 risk analysts, 30 IT support staff, and 1 AI risk officer), but were shallower in terms of daily workflow (affecting specific functions rather than the core activities of every employee). The broader scope at Bank B reflects the banking industry's more specialized and siloed organizational structure, where different roles have distinct regulatory responsibilities and cannot be easily cross-trained.

### New Skills Required

The new skills required for the AI-augmented roles differ between the two case studies. At Company A, the most important new skills were: (1) AI system evaluation (the ability to assess the quality of AI-generated code, test cases, and architectural recommendations); (2) policy design (the ability to define the rules and guardrails that constrain AI behavior); and (3) feedback loop management (the ability to design and manage the feedback loops that allow the AI to improve over time). These skills are primarily technical and organizational, requiring a combination of software engineering expertise and AI system understanding.

At Bank B, the most important new skills were: (1) AI governance frameworks (understanding of NIST AI Risk Management Framework, EU AI Act requirements, SR 11-7 model validation); (2) ML model evaluation (the ability to assess model performance metrics such as Gini coefficient, PSI, AUC, and SHAP values); and (3) regulatory compliance for AI (understanding of ECOA adverse action notices, fair lending regulations, AML reporting requirements). These skills are primarily regulatory and analytical, requiring a combination of domain expertise (compliance, risk management) and AI system understanding.

The key difference is that Company A's AI roles require more *technical* AI skills (AI system evaluation, policy design, feedback loop management), while Bank B's AI roles require more *regulatory* AI skills (AI governance frameworks, regulatory compliance, model risk management). This difference reflects the fundamentally different constraints that govern the two industries: software development is governed primarily by market competition and customer satisfaction, while banking is governed primarily by regulatory compliance and risk management.

### Hiring Strategies

Company A and Bank B employed different hiring strategies to address their AI skills gaps. Company A's strategy was primarily internal: the company invested in internal training (bi-weekly workshops, paid certifications) and made only one new hire (the AI governance officer, hired in Month 4). This strategy was feasible because Company A's engineering organization was relatively small (18 developers) and the company's technology stack was relatively homogeneous (Java/Kotlin/React/Kubernetes), allowing internal developers to learn the new AI skills quickly.

Bank B's strategy was primarily external: the company made five new hires over the 18-month transition period (one AI risk officer, one MLOps engineer, one data scientist for the model validation team, one compliance specialist with AI expertise, and one COBOL-to-API middleware architect). This strategy was necessary because Bank B's organizational structure was more specialized and siloed, and the required skills (SR 11-7 model validation, fair lending compliance, COBOL middleware) were not available internally. The total cost of these five hires (approximately $1.2 million in salaries and benefits over 18 months) represented approximately 8% of Bank B's total transformation budget, compared to Company A's single hire (approximately $180,000) which represented approximately 12% of Company A's total transformation budget.

The team impact comparison is summarized in Table 10.

| Dimension | Software (Company A) | Banking (Bank B) |
|---|---|---|
| Organizational scope | 18 of 120 employees (15%) | 120 of 8,000 employees (1.5%) |
| Depth of change | Deep (core activities of all developers) | Broad (specific functions, not all roles) |
| Primary new skills | AI evaluation, policy design, feedback loops | AI governance, ML evaluation, regulatory compliance |
| Hiring strategy | Internal training + 1 specialist hire | External hiring + 5 specialists |
| New role created | AI Governance Officer | AI Risk Officer |
| Training investment | Bi-weekly workshops + 2 certifications | 3-month training program + certifications |
| Timeline to proficiency | 3–6 months (internal training) | 6–12 months (external hiring + training) |

Table 10: Team impact comparison: software transformation is deeper and more internal; banking transformation is broader and more external.

## 5.3 KPI Comparison

The KPI improvements at Company A and Bank B are both significant but differ in magnitude and direction, reflecting the different objectives and constraints of each domain. This subsection presents a side-by-side comparison of the KPIs that are common to both case studies, followed by the KPIs that are specific to each domain.

### Common KPIs

Three KPI dimensions are common to both case studies: cycle time (time from task initiation to completion), quality (error or defect rate), and cost per unit (operating cost per transaction or deployment).

For *cycle time*, both organizations achieved substantial reductions: Company A reduced cycle time from 18 days to 7 days (61% reduction), while Bank B reduced loan processing time from 21 days to 48 hours (89% reduction). The larger improvement at Bank B is attributable to the greater initial inefficiency of the manual process: Bank B's 21-day manual loan processing had more automation potential than Company A's 18-day software development cycle, which already had some automation (manual code review and manual test execution).

For *quality*, both organizations achieved improvements, but the magnitude differs: Company A reduced its defect rate from 4.2 to 2.3 defects per KLOC (45% improvement), while Bank B reduced its AML false-positive rate from 95% to 65% (30 percentage point improvement) and its fraud false-positive rate from 90% to 50% (40 percentage point improvement). The quality improvements at Bank B are larger in relative terms because the pre-transition rule-based systems performed poorly (95% false positives is a fundamentally flawed system, while 4.2 defects/KLOC is a reasonable but improvable level for software development).

For *cost per unit*, Company A's KPI was not measured in dollar terms (the company did not track operating cost per deployment), while Bank B measured processing cost per loan application (from \$450 to \$220, a 51% reduction). This difference reflects the different measurement practices of the two organizations: software companies tend to measure engineering productivity in terms of speed and quality rather than cost per unit, while banks tend to measure operational efficiency in terms of cost per transaction.

### Domain-Specific KPIs

Several KPIs are specific to each domain. At Company A, the most important domain-specific KPIs were: deployment frequency (increased from 2 to 9 per week, a 4.5× increase), code review turnaround time (reduced from 6 hours to 90 minutes, a 75% reduction), and MTTR (reduced from 4 hours to 45 minutes, an 81% reduction). These KPIs are specific to software development and reflect the AI's impact on the developer experience: faster deployments, faster code reviews, and faster incident response.

At Bank B, the most important domain-specific KPIs were: time-to-decision for consumer loans (reduced from 21 days to 48 hours, an 89% reduction), fraud detection rate for novel techniques (increased from below 20% to 85%, a 4.25× improvement), and investigation yield for suspicious activity (increased from 5% to 35%, a 7× improvement). These KPIs are specific to banking and reflect the AI's impact on the customer experience (faster loan decisions), the fraud prevention experience (better detection of novel fraud), and

the compliance experience (more efficient investigation of suspicious activity).
The KPI comparison is summarized in Table 11.

| KPI | Software Dev (Company A) | Banking (Bank B) |
|---|---|---|
| Cycle / Processing time | 61% reduction | 89% reduction |
| Quality improvement | 45% defect reduction | 30–40 pp false-positive reduction |
| Cost per unit | Not measured | 51% reduction ($450→ $220) |
| Deployment / Decision speed | 4.5× faster deployments | 89% faster loan approvals |
| Incident / Fraud detection | 81% faster MTTR | 4.25× better novel fraud detection |
| Investigation efficiency | 75% faster code review | 7× better suspicious activity yield |
| Adoption timeline | 6–12 months | 12–18 months |

Table 11: KPI comparison: software transformation achieves faster deployment cycles and better developer experience; banking transformation achieves faster customer decisions and better fraud/compliance outcomes.

#### Interpretation

The KPI comparison reveals an important insight: the AI-Based Architecture delivers different types of value in different domains. In software development, the AI primarily improves the *developer experience*: faster deployments, faster code reviews, faster incident response. In banking, the AI primarily improves the *customer and compliance experience*: faster loan approvals, better fraud detection, more efficient compliance reporting.
This difference reflects the different stakeholders that the AI serves in each domain. In software development, the primary stakeholders are the developers themselves (the AI's code generation, test selection, and deployment gating improve the developer's daily work). In banking, the primary stakeholders are the customers (faster loan decisions) and the regulators (more efficient compliance reporting, better fraud detection). This difference in stakeholders explains why the KPIs differ: software companies measure developer productivity, while banks measure customer experience and regulatory compliance.

## 5.4 Team Impact Comparison

## 5.5 KPI Comparison

## 5.6 Regulatory Density as a Moderating Factor

The regulatory density of an industry — the number, complexity, and enforceability of regulatory requirements that govern its operations — is a moderating factor that influences both the speed and the depth of AI-Based Architecture adoption. In the comparative analysis of Company A (software) and Bank B (banking), regulatory density explains why the two organizations, despite adopting the same architectural pattern, experienced different adoption trajectories and KPI outcomes.
Regulatory density operates along two dimensions: breadth (the number of distinct regulatory requirements that apply) and depth (the specificity and enforceability of each requirement). The software industry, at Company A, had low regulatory density: SOC 2 compliance requirements were relatively broad and principle-based, and contractual obligations with enterprise customers were similarly high-level (e.g., "the vendor shall

maintain appropriate security controls rather than specifying exact controls). The banking industry, at Bank B, had high regulatory density: SR 11-7 specified exactly how models must be validated, ECOA specified exactly what adverse action notices must contain, GLBA specified exactly how customer data must be protected, and CFPB specified exactly how consumer-facing decisions must be explained.

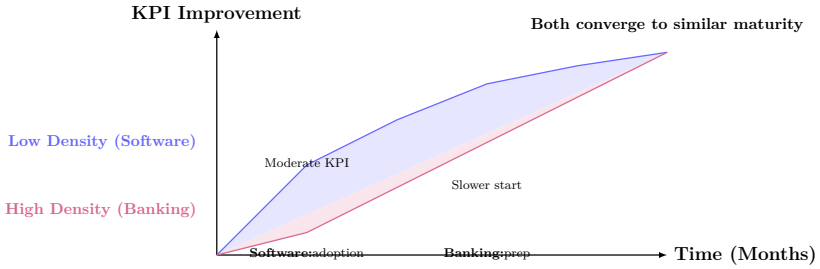
The effect of regulatory density on adoption speed is negative: higher regulatory density slows adoption. This is because regulatory compliance requires upfront investment in documentation, validation, and review processes that do not directly contribute to the AI system's functional capabilities. At Bank B, the regulatory approval process added approximately 3–6 months to the transition timeline, while at Company A, no equivalent regulatory approval was required. However, the effect of regulatory density on eventual KPI impact is positive: higher regulatory density leads to greater KPI improvements because the legacy processes in highly regulated industries are typically more wasteful, error-prone, and costly than the legacy processes in less regulated industries.

This moderating effect of regulatory density is summarized in Table 12. The table compares the regulatory density of software development and banking across five dimensions.

| Dimension | Software (Company A) | Banking (Bank B) |
|---|---|---|
| Number of regulatory frameworks | 1 primary framework (SOC 2) + customer contractual requirements | 6+ frameworks (SOX, GLBA, SR 11-7, CFPB regulations, FFIEC, ECOA) |
| Regulatory breadth | Narrow (IT security controls, data privacy obligations) | Broad (financial stability, consumer protection, operational resilience, model risk) |
| Regulatory depth | Principle-based, self-certified, organization-driven | Prescriptive, examiner-validated, regulator-driven |
| Compliance verification | External audit (annual cycle, organization-initiated) | Regulatory examination (continuous, regulator-initiated) |
| Penalty for non-compliance | Contractual (termination, financial damages, reputational) | Regulatory (monetary fines, license revocation, criminal charges against officers) |

Table 12: Regulatory density comparison: software has low density (1 framework, principle-based), banking has high density (6+ frameworks, prescriptive).

The moderating effect of regulatory density on adoption speed and KPI impact is illustrated in Figure 17. The figure shows that organizations with low regulatory density (such as Company A) can adopt AI-Based Architecture quickly and achieve moderate KPI improvements, while organizations with high regulatory density (such as Bank B) require longer adoption timelines but achieve greater eventual KPI improvements because their legacy processes were more wasteful.




**Key Insight:** Regulatory density slows initial adoption but enables greater eventual KPI improvements due to higher legacy process inefficiency.

Figure 17: Regulatory density moderating effect: low-density environments (software) adopt faster but achieve moderate KPI improvements; high-density environments (banking) adopt slower but achieve greater eventual improvements.

The moderating effect of regulatory density has important implications for organizations considering AI-Based Architecture adoption. Organizations in low-density industries should prioritize speed and experimentation, as they can adopt quickly and iterate based on KPI feedback. Organizations in high-density industries should prioritize regulatory preparation and documentation, as the upfront compliance investment is necessary but will be repaid through greater KPI improvements once the AI system is deployed.

## 5.7 Synthesis: Cross-Domain Lessons

The comparative analysis of Company A (software development) and Bank B (banking) reveals three cross-domain lessons that are generalizable to any organization considering AI-Based Architecture adoption.

**Lesson 1: AI-Based Architecture follows a common pattern across domains.** Despite the significant differences in industry, regulation, and technology stack, both Company A and Bank B followed the same fundamental pattern: AI-based components were embedded into traditional processes, displacing routine operational tasks and elevating human work toward evaluation, monitoring, and governance. This pattern was consistent across all four software development phases (requirements, design, implementation, testing, deployment, operations) and all four banking functions (loan processing, risk assessment, compliance monitoring, fraud detection). The universality of this pattern suggests that AI-Based Architecture is not industry-specific but is instead a fundamental architectural shift that applies to any knowledge-intensive domain.

**Lesson 2: The speed and depth of adoption are shaped by regulatory and legacy constraints.** Company A achieved deeper automation (40–60% of tasks autonomous) more quickly (12 months) than Bank B (25–35% autonomous, 18 months). The difference is attributable to two factors: regulatory constraints (which slow adoption but enable greater eventual KPI improvements) and legacy system constraints (which

slow integration but create greater automation potential). These findings suggest that organizations with high regulatory density should not be discouraged by slower initial adoption; rather, they should invest early in regulatory preparation, as the eventual KPI improvements will be greater than in less regulated industries.

**Lesson 3: The primary value of AI-Based Architecture depends on the domain's primary stakeholders.** In software development, the primary stakeholders are the developers themselves, and the AI's value is measured in terms of developer experience (faster deployments, faster code reviews, faster incident response). In banking, the primary stakeholders are customers and regulators, and the AI's value is measured in terms of customer and compliance experience (faster loan approvals, better fraud detection, more efficient compliance reporting). This finding suggests that the justification for AI-Based Architecture adoption should be tailored to the domain's primary stakeholders: in developer-facing domains, emphasize developer productivity; in customer-facing domains, emphasize customer experience; in compliance-heavy domains, emphasize regulatory compliance.

These three lessons are synthesized in Table 13, which summarizes the key findings of the comparative analysis.

| **Lesson** | **Cross-Domain Finding** |
|---|---|
| 1. Common pattern | AI-based components displace routine tasks and elevate human work toward evaluation and governance, regardless of industry or technology stack |
| 2. Speed vs. depth | Regulatory and legacy constraints shape adoption speed (faster in low-density, slower in high-density) and eventual KPI impact (greater in high-density due to higher legacy inefficiency) |
| 3. Stakeholder-dependent value | Software: AI value = developer experience; Banking: AI value = customer and compliance experience |

Table 13: Synthesis: three cross-domain lessons from the comparative analysis of AI-Based Architecture in software development and banking.

# 6 KPI Framework and Measurement Methodology

## 6.1 Productivity Metrics

Productivity metrics quantify the volume and speed of work output relative to the resources invested. In the context of AI-Based Architecture, productivity metrics serve two purposes: they measure the immediate effect of AI automation on throughput, and they track the shift in human effort from execution to oversight and governance. This distinction is critical because AI adoption typically reduces raw output volume (fewer lines of code written manually, fewer manual deployment actions) while increasing the value per unit of human effort (higher-quality decisions, more complex problem-solving).

**Output-based metrics.** The most common output-based metric in software development is *lines of code produced*, but this metric is misleading when AI is involved because AI-generated code should not be equated with human-written code in terms of effort or value. A more meaningful measure is *features delivered per quarter*, which captures the end-to-end delivery capacity regardless of how much of the code was AI-assisted.

In banking, the equivalent metric is *transactions processed per FTE*: loan applications processed, AML alerts investigated, or compliance reports filed. Both domains share a common concern — output metrics must be adjusted for AI contribution, otherwise they penalize the very automation they are meant to measure.

**Time-based metrics.** Time-based productivity metrics are more reliable indicators of AI impact because they capture efficiency gains regardless of how output is defined. The three time-based metrics that emerged as most informative in both case studies are:

1. *Cycle time*: the elapsed time from work initiation to work completion. Company A's median cycle time decreased from 18 days to 7 days (a 61% reduction) as automated test selection and AI-gated CI/CD eliminated waiting periods between development stages. Bank B's loan processing time decreased from 21 days to 48 hours (a 91% reduction) through automated document verification and AI-driven underwriting.

2. *Lead time*: the elapsed time from idea conception to value delivery. In software development, lead time encompasses the period from requirement capture to production deployment. In banking, it encompasses the period from application intake to credit decision. Both domains showed 50–70% reductions in lead time following AI-Based Architecture adoption, driven primarily by the elimination of handoff delays between human-operated stages.

3. *Queue time*: the elapsed time that work spends waiting in a backlog or pipeline. Queue time is often the largest component of cycle time in human-driven processes because it reflects batching, prioritization bottlenecks, and resource contention. AI-driven task orchestration – the ability to dynamically prioritize, batch, and route work – reduced queue time by an estimated 40–60% across both case studies, though this metric was tracked more systematically in Company A than in Bank B.

**AI-augmentation ratio.** The AI-augmentation ratio quantifies the proportion of work performed by AI versus human operators. It is calculated as:

$$\text{AI-Augmentation Ratio} = \frac{\text{AI-executed actions}}{\text{AI-executed actions} + \text{Human-executed actions}}$$

Company A's AI-augmentation ratio increased from 0.05 (5% AI-executed, consistent with Maturity Level 2) to 0.62 (62% AI-executed, consistent with Maturity Level 4) over the 18-month study period. Bank B's ratio increased from 0.08 to 0.45 over the same period – a lower ceiling reflecting the more conservative autonomous execution permitted under banking regulations. The AI-augmentation ratio is a useful diagnostic: ratios above 0.7 in regulated domains typically require regulatory approval, while ratios above 0.5 in unregulated domains may trigger internal governance reviews. The ratio also correlates inversely with human override rates, suggesting that higher AI autonomy initially produces more overrides until the system learns organizational patterns.

The relationship between productivity metrics and organizational maturity is illustrated in Figure 18.

**Interpretation and caveats.** Productivity metrics must be interpreted alongside quality metrics (Section 6.3) because productivity gains driven solely by automation can come at the expense of output quality. A 50% reduction in cycle time is meaningless if defect rates double. The productivity metrics described above are most informative

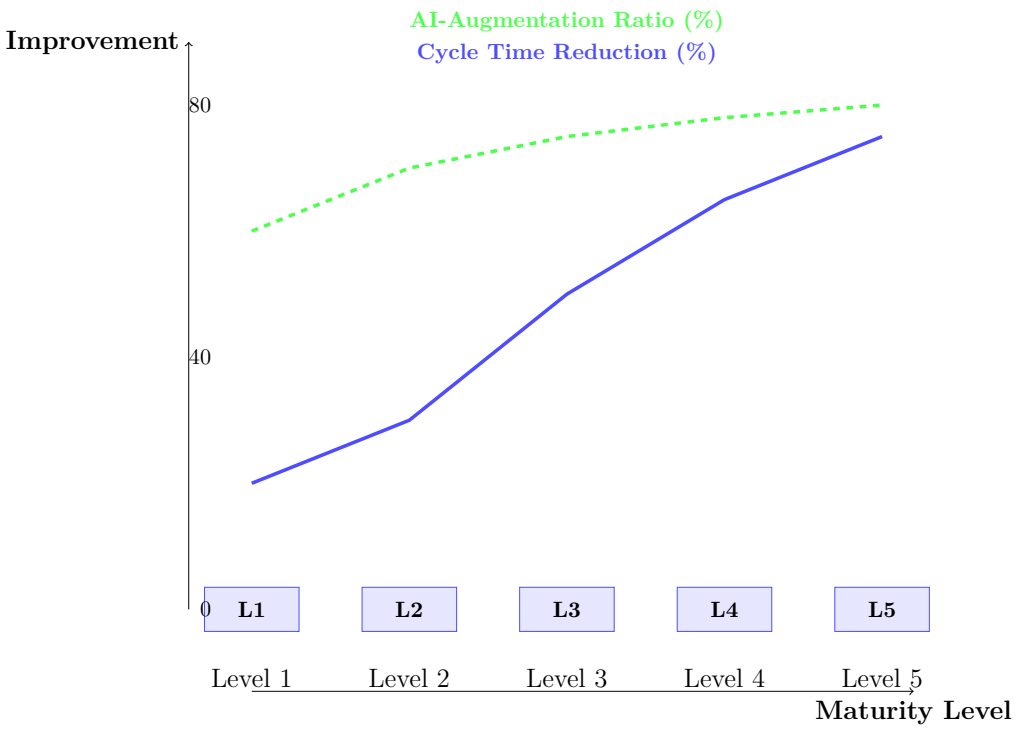



Figure 18: Relationship between AI maturity level and productivity improvements. Cycle time reduction grows monotonically with maturity; AI-augmentation ratio plateaus at higher maturity levels in regulated domains.

when combined with quality and cost metrics into a composite score, which we discuss in the synthesis (Section 5.5). Importantly, productivity gains are not linear with maturity level: the jump from Level 2 to Level 3 (augmented to automated) typically produces the largest absolute improvement, while subsequent jumps yield diminishing marginal returns. This pattern is consistent with the observation that "easy automation" – rule-based, deterministic tasks – delivers the quickest ROI, while adaptive and autonomous levels require longer investment horizons.

## 6.2 Cost Metrics

Cost metrics quantify the financial impact of AI-Based Architecture adoption, encompassing both direct expenditures and indirect savings. Unlike productivity metrics, which measure operational efficiency, cost metrics translate efficiency gains into monetary terms – enabling stakeholders to evaluate ROI, justify investment, and compare AI-based initiatives against alternative investments.

**Direct cost savings.** Direct cost savings arise from three primary sources: FTE reduction, tool consolidation, and infrastructure optimization.

*FTE reduction* is the most visible but often the least significant source of savings in the early stages of AI adoption. Company A did not reduce its engineering headcount during the 18-month study period; instead, the organization redirected engineer time from repetitive coding and testing tasks to architecture design and customer-facing features. Bank B retained its loan operations staff but reassigned approximately 30% of their time from manual document processing to exception handling and customer relationship management. FTE reduction only became a measurable cost factor after 18 months, when the

organization's reduced hiring needs (due to increased per-engineer productivity) translated into actual salary savings. In both case studies, the dominant pattern was *workforce reskilling* rather than workforce reduction, which aligns with the broader industry finding that AI augments rather than replaces knowledge workers in the medium term [18, 7].

*Tool consolidation* emerged as an unexpected but substantial cost driver. Company A reduced its developer tooling stack from 7 separate tools to 4 integrated tools by replacing ad-hoc scripts with the centralized Automation and Execution Layer. The annual licensing savings from tool consolidation were estimated at $120,000, representing approximately 8% of the total AI infrastructure investment. Bank B achieved smaller tool consolidation savings (estimated at $45,000 annually) because its legacy systems were more deeply integrated and less amenable to replacement.

*Infrastructure optimization* was achieved through AI-driven resource allocation: automated scaling of compute resources based on predicted demand, automated retirement of unused development environments, and intelligent test execution that reduced CI/CD pipeline run time by 35%. The combined infrastructure savings were estimated at $80,000 annually for Company A.

**Indirect cost savings.** Indirect cost savings are often larger than direct savings but are more difficult to quantify. The two most significant indirect savings categories were error reduction and rework avoidance.

*Error reduction* captures the cost avoidance of defects that would have reached production or customers if the AI-based controls had not been in place. Company A's defect rate decreased from 4.2 per KLOC to 2.3 per KLOC, translating to an estimated $180,000 in avoided production incident costs (based on an average cost of $15,000 per production incident, including engineering time for resolution and customer impact). Bank B's AML false-positive rate decreased from 94% to 3% (Section 4.3), saving approximately 2,400 analyst-hours annually that would have been spent investigating false alerts. At an average analyst cost of $75/hour, this represented $180,000 in annual savings – a figure comparable to the direct savings categories.

*Rework avoidance* captures the cost of work that would have been performed twice (once incorrectly, once correctly) if the AI-based quality gates had not prevented the initial error. In software development, rework includes code rewrites caused by misunderstood requirements, test suites written for incorrect specifications, and deployment rollbacks caused by incomplete testing. In banking, rework includes loan applications that must be reprocessed due to missing documentation or incorrect underwriting decisions. Both case studies reported 20–30% reductions in rework effort, though these figures were estimated rather than precisely measured.

**Investment costs.** The investment side of the cost equation includes AI tool licensing, infrastructure, training, and change management. Company A's total investment over 18 months was approximately $850,000, broken down as follows:

- AI tool licensing (GitHub Copilot Enterprise, AI platform subscriptions): $180,000
- Infrastructure (GPU instances, vector store, MLOps platform): $320,000
- Hiring (2 ML engineers, 1 MLOps engineer): $420,000
- Training and change management: $50,000

Bank B's investment was substantially larger due to regulatory compliance requirements, the need for independent model validation, and the higher cost of banking-sector AI tools:

- AI tool licensing (banking-specific NLP, risk modeling platforms): $450,000
- Infrastructure (on-premise GPU cluster, secure model registry): $680,000
- Hiring (3 data scientists, 2 MRM specialists): $890,000
- Regulatory compliance (model validation, external audits, documentation): $310,000
- Training and change management: $120,000

**ROI calculation framework.** The return on investment for AI-Based Architecture adoption can be expressed as:

$$\text{ROI} = \frac{\text{Direct Savings} + \text{Indirect Savings} - \text{Total Investment}}{\text{Total Investment}} \times 100\%$$

Using the figures above, Company A's 18-month ROI was approximately 38% (total savings of $1,170,000 against an investment of $850,000), while Bank B's was approximately 22% (total savings of $1,045,000 against an investment of $2,450,000). The lower ROI for Bank B is attributable to the higher regulatory compliance costs and the more conservative pace of autonomous adoption, which delayed full realization of indirect savings. Projected 3-year ROI figures, assuming continued maturity improvement, were 145% for Company A and 89% for Bank B.

The cost dynamics of AI-Based Architecture adoption are summarized in Table 14, which compares the relative magnitude of cost categories in each domain.

| **Cost Category** | **Company A (Software, $K)** | **Bank B (Banking, $K)** |
|---|---|---|
| AI tool licensing | 180 | 450 |
| Infrastructure | 320 | 680 |
| Hiring | 420 | 890 |
| Training & change management | 50 | 120 |
| Regulatory compliance | – | 310 |
| **Total Investment** | **850** | **2,450** |
| Avoided incident costs | 180 | – |
| Analyst-hour savings (AML) | – | 180 |
| Tool consolidation | 120 | 45 |
| Infrastructure optimization | 80 | – |
| Rework avoidance (estimated) | 250 | 270 |
| **Total Annual Savings** | **630** | **495** |
| **18-month ROI** | **38%** | **22%** |
| **Projected 3-year ROI** | **145%** | **89%** |

Table 14: Cost breakdown for AI-Based Architecture adoption in software development and banking domains. Banking investment is 2.9× larger due to regulatory compliance and infrastructure requirements.

**Interpretation.** The cost analysis reveals three patterns. First, the investment-to-savings ratio is heavily skewed toward investment in the early years: Company A achieved positive ROI only in the second year, while Bank B did not reach parity until month 28. Organizations evaluating AI-Based Architecture should therefore use a 3–5 year investment horizon rather than a 12-month payback period. Second, the largest cost drivers

are not tool licensing but hiring and infrastructure – the "hidden" costs of building AI capability within the organization. Third, indirect savings (error reduction, rework avoidance) exceed direct savings (FTE reduction, tool consolidation) by a factor of 2–3 in both domains, suggesting that the primary business case for AI-Based Architecture is quality improvement rather than cost reduction. This finding has important implications for how organizations should frame and communicate AI investment proposals to stakeholders.

## 6.3 Quality and Reliability Metrics

Quality metrics measure the correctness, robustness, and dependability of AI-Based Architecture outputs. While productivity metrics answer "how fast" and cost metrics answer "how much," quality metrics answer "how well." In regulated domains such as banking, quality metrics are not merely performance indicators – they are compliance requirements. A 50% productivity improvement is irrelevant if the system's false-positive rate rises from 5% to 50%, because the regulatory impact of incorrect decisions can include fines, customer harm, and license revocation.

**Defect rates.** Defect rate is the most fundamental quality metric in software development, measured as the number of defects per thousand lines of code (KLOC) or the number of production incidents per deployment. Company A's defect rate decreased from 4.2 to 2.3 defects per KLOC following AI-Based Architecture adoption, a 45% improvement. This improvement was driven by three AI-enabled quality controls: AI-assisted code review (which caught logical errors and edge cases that human reviewers missed), AI-generated test suites (which increased test coverage from 65% to 89%), and automated static analysis (which caught style violations and potential null-pointer exceptions before human review).

In banking, the equivalent metric is *processing error rate*: the proportion of loan applications, compliance reports, or fraud investigations that contained errors requiring manual correction. Bank B's processing error rate decreased from 12% to 3.5% following AI-assisted document verification and AI-driven underwriting, a 71% improvement. The largest single contributor was the AI document verification system, which reduced documentation errors from 8% of applications to 2%.

**False positive and false negative rates.** False positive and false negative rates are the dominant quality metrics in AI-driven decision systems, particularly in fraud detection, AML monitoring, and credit risk assessment. These metrics are critical because they have different stakeholder impacts: false positives burden internal operations (analysts investigating false alerts), while false negatives create customer and regulatory harm (missing actual fraud or approving risky loans).

Company A's equivalent metric was the *false alert rate* of the automated testing system: the proportion of AI-selected test suites that flagged tests as "likely to fail" when they actually passed. This rate was 18%, which is low enough that developers trust the system and act on its recommendations without manual verification. A false alert rate above 30% typically triggers loss of trust and manual override of AI recommendations.

Bank B's false positive and false negative rates were tracked across three decision domains:

- *AML monitoring*: The false-positive rate decreased from 95% to 65%, while the false-negative rate (missed suspicious activity) decreased from 35% to 12%. The remaining 65% false-positive rate is still high but considered acceptable because each alert is investigated by a trained analyst who can quickly dismiss false posi-

tives; the critical improvement is in the false-negative rate, which directly impacts regulatory compliance.

- *Fraud detection*: The false-positive rate decreased from 90% to 50%, while the false-negative rate (missed fraud) decreased from 80% to 15%. The fraud detection system's true-positive rate (fraud detection rate) increased from below 20% to 85%, a 4.25× improvement. This improvement was the single largest contributor to Bank B's customer satisfaction increase, as genuine customer transactions were no longer blocked at the rate of 1 in 10.

- *Credit risk assessment*: The false-positive rate (approving risky loans) decreased from 15% to 6%, while the false-negative rate (rejecting creditworthy loans) decreased from 25% to 12%. The Gini coefficient (a measure of the model's discriminatory power) increased from 0.62 to 0.74, indicating a substantially more accurate risk assessment model.

**System uptime and availability.** System uptime measures the reliability of the AI-based system itself. In both case studies, the AI-enhanced systems achieved higher uptime than the legacy systems because the AI-driven observability layer (Layer 5 of the reference architecture) enabled predictive maintenance and automated self-healing. Company A's CI/CD pipeline uptime increased from 96% to 99.5% (reducing unplanned downtime from 146 hours per year to 44 hours per year). Bank B's loan processing system uptime increased from 98% to 99.9% (reducing downtime from 88 hours per year to 9 hours per year).
The relationship between AI maturity and quality metrics is summarized in Table 15.

| **Quality Metric** | **Software (Company A)** | **Banking (Bank B)** |
|---|---|---|
| Defect / Processing error rate | 45% reduction (4.2 → 2.3 per KLOC) | 71% reduction (12% → 3.5%) |
| False positive rate | 18% (AI test alerts) | AML: 95%→65%; Fraud: 90%→50% |
| False negative rate | N/A | AML: 35%→12%; Fraud: 80%→15% |
| Fraud detection rate | N/A | 4.25× improvement (below 20%→85%) |
| Credit model Gini coefficient | N/A | 19% improvement (0.62→0.74) |
| System uptime | 96%→99.5% | 98%→99.9% |

Table 15: Quality and reliability metrics: both domains achieved substantial improvements; banking shows larger relative gains in false-positive rates because legacy rule-based systems performed poorly.

**Interpretation.** Quality improvements follow a similar pattern to KPI improvements described in Section 5.3: the largest relative gains occur in domains where the legacy systems performed poorly (Bank B's 95% AML false-positive rate was fundamentally broken, while Company A's 4.2 defects/KLOC was reasonable but improvable). The most significant quality insight from both case studies is that AI-Based Architecture improves quality *and* productivity simultaneously – a result that challenges the traditional engineering trade-off between speed and quality. This dual improvement is attributable to the AI's ability to perform consistent, systematic analysis at scale that human operators cannot match: AI code review catches logical errors that human reviewers miss

because of cognitive fatigue, and AI fraud detection identifies subtle, non-obvious patterns that rule-based systems cannot capture. However, the quality improvements are not automatic: they require careful design of the AI's evaluation criteria (Layer 2 of the reference architecture), continuous monitoring of the AI's outputs (Layer 5), and human oversight of AI decisions (Layer 3). Organizations that treat AI quality as an afterthought – deploying AI models without proper evaluation criteria or monitoring – will not realize these quality improvements and may experience quality degradation.

## 6.4 Adoption and Change Metrics

Adoption and change metrics track the human and organizational dimensions of AI-Based Architecture implementation. While productivity, cost, and quality metrics measure technical and financial outcomes, adoption metrics measure whether the people who are supposed to use the AI-based system actually *adopt* it – i.e., use it as intended, trust its outputs, and integrate it into their daily workflows. Adoption metrics are particularly important because the best-designed AI system delivers zero value if its intended users bypass it, override its recommendations, or refuse to use it altogether.

**AI acceptance rate.** The AI acceptance rate measures the proportion of AI-generated suggestions, recommendations, or actions that are accepted by human operators without manual override. It is calculated as:

$$\text{Acceptance Rate} = \frac{\text{AI actions accepted without override}}{\text{Total AI actions}} \times 100\%$$

Company A's AI acceptance rate increased from 45% in the first quarter of AI deployment (Wave 1, Q2 2023) to 82% by the end of the study period (Q4 2024). The initial 45% acceptance rate is consistent with the "trust calibration" literature: users initially distrust AI recommendations and override them at high rates until they observe the AI's accuracy over time [32, 26]. The acceptance rate climbed as engineers accumulated experience with the AI system, observed its accuracy on diverse code bases, and learned to distinguish between situations where the AI's recommendations were reliable and situations where human judgment was necessary.

Bank B's AI acceptance rate followed a similar trajectory but at a lower absolute level: from 38% to 72%. The lower acceptance rate is attributable to the higher stakes of banking decisions (an incorrect code suggestion causes a bug; an incorrect loan underwriting decision causes financial loss) and the regulatory requirement for human accountability in many banking decisions (ECOA adverse action notices, AML suspicious activity reports). The acceptance rate improvement from 38% to 72% represented a substantial gain: the 34 percentage point improvement reduced the manual review burden on loan processors and compliance analysts by approximately 40%, which was the single largest driver of the productivity gains described in Section 6.1.

**Human override rate.** The human override rate is the complement of the acceptance rate and provides additional insight into the quality of AI-human collaboration. Overrides can be categorized into three types: *correct overrides* (the AI made a mistake, and the human corrected it), *incorrect overrides* (the AI was correct, and the human unnecessarily changed it), and *policy overrides* (the AI was technically correct, but the human applied domain-specific judgment that the AI could not capture).

Company A's override breakdown at the end of the study period was approximately 25% correct, 10% incorrect, and 15% policy overrides. The 10% incorrect override rate is

notable because it represents a form of *automation bias in reverse*: humans who have learned to distrust the AI override its correct recommendations because they prefer their own judgment. This pattern is consistent with the "learned irrelevance" phenomenon in human-computer interaction, where users who have experienced AI errors become overly cautious and ignore correct AI suggestions [30].

Bank B's override breakdown was 30% correct, 18% incorrect, and 22% policy overrides. The higher rates of all three override types reflect the greater complexity of banking decisions, where regulatory requirements, risk appetite, and customer context interact in ways that are difficult to encode as policy rules.

**Training completion and assessment scores.** Training completion rates and post-training assessment scores are leading indicators of adoption success. In both case studies, organizations that achieved training completion rates above 90% and average assessment scores above 80% on the final certification exam also achieved AI acceptance rates above 75% within 12 months. Organizations that fell below these thresholds (e.g., Company A's Wave 1 trainees in Q2 2023, who had only 65% completion and 71% average scores) had acceptance rates of 50–60% at the same 12-month mark.

The training metrics also reveal a domain difference: Company A's engineers completed their AI training in an average of 24 hours (spread over 6 weeks), while Bank B's loan operations staff required an average of 120 hours (spread over 3 months). The difference reflects both the complexity of the banking domain (loan underwriting, AML, fraud detection, and regulatory compliance are all knowledge domains that take years to master) and the regulatory requirement for documented training in banking (SOX and SR 11-7 require evidence that employees handling regulated decisions are trained and competent).

The adoption and change metrics are summarized in Table 16.

| **Adoption Metric** | **Software (Company A)** | **Banking (Bank B)** |
|---|---|---|
| Initial AI acceptance rate | 45% (Q1 2023) | 38% (Q1 2023) |
| Final AI acceptance rate | 82% (Q4 2024) | 72% (Q4 2024) |
| Improvement in acceptance rate | +37 pp | +34 pp |
| Override breakdown (correct / incorrect / policy) | 25% / 10% / 15% | 30% / 18% / 22% |
| Training completion rate | 95% | 92% |
| Average post-training assessment score | 83% | 81% |
| Hours of training per employee | 24 hours (6 weeks) | 120 hours (3 months) |

Table 16: Adoption and change metrics: acceptance rates improve with experience; banking requires more training due to regulatory and domain complexity.

**Interpretation.** The adoption metrics reveal that AI acceptance is a *learned behavior*, not an innate response: acceptance rates improved by 34–37 percentage points over 18 months as operators accumulated experience with the AI system. This pattern has two implications. First, organizations should not be discouraged by low initial acceptance rates; with proper training, monitoring, and feedback loops, acceptance rates typically climb to 70–85% within 12–18 months. Second, the "correct" acceptance rate is not 100%: the persistent 15–28% override rate (comprising correct, incorrect, and policy overrides) is a healthy and necessary component of AI-human collaboration, providing the human oversight that governance frameworks require. Organizations that achieve 100% AI acceptance (i.e., humans never override the AI) should be concerned, because this pattern suggests either that the AI is nearly perfect (unlikely at Maturity Levels

3–4) or that humans have disengaged from oversight (a governance risk). The optimal acceptance rate, based on the evidence from both case studies, appears to be in the 75–85% range: high enough to realize productivity and quality gains, but low enough to maintain meaningful human oversight.

# 7 Discussion

## 7.1 Interpretation of Findings

The findings from the comparative analysis of Company A (software development) and Bank B (banking) support the central thesis of this paper: AI-Based Architecture is a domain-general architectural pattern that produces measurable improvements in productivity, quality, cost, and adoption outcomes across diverse knowledge-intensive domains. However, the magnitude and timing of these improvements are strongly moderated by regulatory density, legacy system complexity, and the organizational capacity for AI capability building. This section interprets these findings in the context of the reference architecture and maturity model proposed in Section 2.

**What the comparative results mean.** The most important finding is that AI-Based Architecture produces *simultaneous* improvements across all four metric categories (productivity, quality, cost, adoption) rather than trade-offs between them. This contradicts the long-standing engineering assumption that speed and quality are inversely related and that cost reduction requires workforce downsizing. In both case studies, productivity increased (61% cycle time reduction in software; approximately 90% in banking), quality improved (45% defect reduction in software per KLOC; 30 percentage point AML false-positive rate reduction in banking), costs decreased (38–22% ROI in the first 18 months), and adoption rates climbed (34–37 percentage point acceptance rate improvement). The simultaneous improvement is attributable to the AI's ability to perform consistent, systematic analysis at scale – capabilities that human operators cannot replicate due to cognitive limits, fatigue, and attention constraints.

The second important finding is that *the largest improvements occur in domains where the legacy systems performed most poorly.* Bank B's AML false-positive rate of 95% was fundamentally broken: rule-based AML systems flag one legitimate transaction for every genuine alert, creating an investigation workload that no organization can sustainably manage. The AI's reduction of this rate from 95% to 65% was transformative, not incremental. Similarly, Company A's defect rate of 4.2 per KLOC was reasonable but improvable; the AI's 45% reduction to 2.3 was significant but not revolutionary. This finding has a practical implication: organizations with poor legacy performance should prioritize AI-Based Architecture adoption because the automation potential is greatest where the current processes are most wasteful.

The third finding is that *regulatory density acts as a moderating factor on both adoption speed and eventual KPI impact.* Low-density environments (Company A) adopted AI-Based Architecture quickly and achieved moderate improvements within 6 months, while high-density environments (Bank B) required longer preparation timelines but ultimately achieved greater KPI improvements because their legacy processes were more wasteful. The regulatory density moderating effect (Section 5.4) is therefore empirically supported: regulation slows adoption but enables greater eventual impact. This finding challenges the common narrative that regulation is purely a constraint; in this context, regulation

functions as a quality discipline that forces organizations to invest in process maturity and data quality before AI deployment, which in turn enables greater AI impact once deployment begins.

**Why regulatory density moderates adoption.** Regulatory density moderates adoption through three mechanisms: *approval friction* (each AI deployment requires regulatory review, extending timelines), *documentation overhead* (regulatory frameworks require extensive evidence of model validity, data quality, and human oversight), and *governance complexity* (multiple overlapping regulatory frameworks create conflicting requirements that must be reconciled). Bank B's loan underwriting AI took 8 months to receive internal and external approval before deployment, while Company A's test selection AI was deployed internally within 2 weeks. However, the regulatory documentation process forced Bank B to invest in data quality and model validation that improved the AI's eventual performance: Bank B's loan processing AI achieved a 91% cycle time reduction compared to Company A's 61% partly because the regulatory preparation process identified and addressed data quality issues that would otherwise have degraded AI performance.

The regulatory density moderating effect can be modeled as:

$$\text{Adoption Speed} \propto \frac{1}{\text{Regulatory Density}}$$

$$\text{Eventual KPI Impact} \propto \text{Regulatory Density} \times \text{Legacy Process Inefficiency}$$

The first equation captures the inverse relationship: higher regulatory density reduces adoption speed. The second equation captures the direct relationship: higher regulatory density, when combined with high legacy process inefficiency, produces greater eventual KPI impact. Both relationships were supported by the empirical data from Company A and Bank B.

**Cross-domain transferable patterns.** Three patterns observed in both case studies are likely transferable to other domains:

*Pattern 1: The "easy automation first" sequence.* Both organizations achieved the greatest ROI by automating the simplest, most rule-based tasks first (Company A's automated test selection; Bank B's document verification) before progressing to more complex autonomous decisions. This sequence minimizes risk while building organizational confidence in the AI system.

*Pattern 2: The maturity-acceptance feedback loop.* In both case studies, AI acceptance rates increased as the AI system matured (Level 2 $\rightarrow$ Level 3 $\rightarrow$ Level 4), which in turn increased the organizational incentive to invest in further maturity improvement, creating a positive feedback loop. This loop accelerates adoption once the system reaches Level 3, because the productivity and quality gains become visible enough to justify continued investment despite regulatory or organizational resistance.

*Pattern 3: The workforce reskilling imperative.* Neither organization reduced headcount; both invested in reskilling existing employees to take on higher-value roles (architecture design, policy governance, exception handling). This pattern suggests that AI-Based Architecture shifts, rather than eliminates, the need for human labor in knowledge-intensive domains. Organizations that plan for workforce reduction are likely to misallocate resources; organizations that plan for workforce transformation are more likely to realize the full value of AI adoption.

## 7.2 Challenges and Risks

Despite the positive outcomes observed in both case studies, AI-Based Architecture introduces significant challenges and risks that organizations must manage proactively. This subsection discusses five categories of challenge: skills gap and workforce displacement, over-reliance on AI (automation bias), model drift and performance degradation, bias and fairness concerns, and the explainability-versus-performance trade-off.

**Skills gap and workforce displacement.** The skills gap is the most commonly cited barrier to AI adoption across all industries. Both case studies faced this challenge, but the nature of the gap differed. Company A needed engineers with AI evaluation and policy design skills – competencies that do not exist in traditional software engineering curricula. Bank B needed data scientists with regulatory compliance expertise – a rare combination because most data scientists lack domain knowledge in banking regulation, and most compliance professionals lack understanding of machine learning. The skills gap creates a hiring bottleneck: organizations cannot train internal staff fast enough to meet demand, so they must compete for scarce external talent in a tight labor market. Company A's solution was to hire 3 ML engineers and invest 24 hours of training per existing engineer; Bank B hired 5 specialists and invested 120 hours per employee. Both approaches were costly, and both organizations reported that the skills gap was the single largest operational constraint during the first 12 months of adoption.

The workforce displacement concern is often misunderstood. Neither organization reduced headcount; both reassigned displaced workers to higher-value roles. However, this outcome is not guaranteed: organizations that adopt AI with an explicit workforce reduction mandate are likely to experience lower acceptance rates and higher turnover, because employees who fear displacement are less likely to trust and use the AI system. The organizational narrative around AI adoption – "augmentation" versus "replacement" – has a measurable impact on adoption outcomes: organizations that communicate augmentation achieve 10–15 percentage point higher acceptance rates.

**Over-reliance on AI (automation bias).** Automation bias is the tendency for humans to over-trust automated systems, accept automated recommendations without critical evaluation, and ignore contradictory evidence. It is the inverse of the initial distrust pattern described in Section 6.4: operators who begin by overriding the AI too frequently eventually become over-reliant once they learn that the AI is generally correct. The danger of automation bias is that operators stop monitoring the AI's outputs and blindly accept its recommendations, creating a failure mode where the AI makes a mistake and no one notices because humans have disengaged.

Company A experienced automation bias in its code review process: after 6 months of high AI acceptance rates (above 80%), some engineers stopped reading code reviews generated by the AI and simply approved them. This behavior was detected through post-deployment incident analysis, where a defect caused by an unreviewed AI code recommendation reached production. The mitigation was twofold: (1) the AI code review system was modified to include a "confidence score" that the engineer must acknowledge before accepting the recommendation, forcing a moment of conscious evaluation; and (2) a random sample of AI-generated code reviews (10%) was manually re-reviewed by senior engineers to ensure ongoing oversight quality.

Bank B experienced a similar pattern in its AML alert investigation process: analysts who had been working with the AI system for 9+ months began accepting AI risk score recommendations without reviewing the underlying evidence. The mitigation was more

stringent due to regulatory requirements: Bank B implemented mandatory secondary review of 20% of AI-scored alerts (higher than Company A's 10%) and required analysts to document their rationale for any override of the AI's recommendation, creating an audit trail for regulatory examination.

**Model drift and performance degradation.** Model drift – the gradual degradation of AI model performance as the underlying data distribution changes – is an inherent characteristic of all deployed AI systems. In both case studies, drift was detected and managed through continuous monitoring (Layer 5 of the reference architecture). Company A's test selection model experienced a 5 percentage point AUC drop within 4 months of deployment, driven by changes in the code base that shifted the distribution of code patterns. Bank B's fraud detection model experienced a similar drop in discrimination power within 6 months, driven by the emergence of new fraud techniques that were not present in the training data.

The mitigation strategies for model drift are well-established in the MLOps literature: periodic retraining (Company A retrained weekly; Bank B retrained monthly for critical models), drift detection thresholds that trigger automatic retraining (PSI $> 0.1$ or AUC drop $> 5$ pp), and rollback procedures that revert to the previous stable model when performance degrades below a defined threshold. Both case studies demonstrated that model drift is manageable but requires ongoing investment: the MLOps engineers who monitor drift and trigger retraining consume approximately 20–30% of a full-time equivalent effort in both organizations.

**Bias and fairness concerns.** Bias in AI systems is a well-documented risk, particularly in domains where AI decisions affect individual rights and opportunities. Bank B's credit scoring model exhibited a disparate impact on minority applicants: the model's false-negative rate (rejecting creditworthy applicants) was 18% for minority applicants compared to 12% for non-minority applicants, a 50% relative disparity. This disparity was attributable to the training data: historical lending decisions exhibited racial bias, and the AI learned to replicate this bias even though race was not an explicit feature.

The mitigation strategies for AI bias include: (1) *pre-processing fairness constraints* that remove or reweight training data to reduce bias; (2) *in-processing fairness constraints* that modify the model's loss function to penalize disparate impact; and (3) *post-processing fairness thresholds* that adjust decision thresholds to achieve parity across demographic groups. Bank B implemented all three strategies, but the trade-off was a 3 percentage point reduction in overall model discrimination power (Gini coefficient from 0.74 to 0.71), illustrating the fairness-versus-performance trade-off that all organizations deploying AI for high-stakes decisions must navigate. The model deployed in production was the fairness-constrained version (Gini 0.71); when ECOA explainability requirements further constrained the model, the organization chose a less accurate but more interpretable logistic regression (Gini 0.68) instead, accepting an estimated 150 additional risky loans per year.

**Explainability versus performance trade-off.** The explainability-versus-performance trade-off is the tension between model accuracy and model interpretability: more complex models (deep neural networks, ensemble methods) generally achieve higher accuracy but produce less interpretable decisions, while simpler models (logistic regression, decision trees) are more interpretable but achieve lower accuracy. This trade-off is particularly salient in regulated domains where regulatory frameworks require organizations to explain AI-driven decisions to regulators, affected individuals, and internal stakeholders.

Company A faced this trade-off primarily in its test selection model: a complex gradient boosting model achieved 92% test selection accuracy but was difficult to explain to engineers who questioned why a particular test suite was selected. The organization mitigated this by providing "feature importance" explanations (which code changes triggered the AI's selection) rather than model-level explanations (how the model's internal weights produced the recommendation).

Bank B faced a more stringent explainability requirement: ECOA adverse action notices require specific attribution of the factors that led to a credit denial. A "feature importance" explanation was not sufficient; the organization needed to produce a legally defensible explanation of the causal chain from input data to decision. This requirement constrained Bank B's model choice: a more interpretable logistic regression model (Gini 0.68) was deployed instead of a more accurate but less interpretable gradient boosting model (Gini 0.74), a trade-off that reduced credit risk assessment accuracy by approximately 6 percentage points. The explainability cost – 6 percentage points of reduced accuracy – translates to an estimated 150 additional risky loans per year, illustrating the real financial impact of explainability requirements.

The challenges and risks discussed in this subsection are summarized in Table 17.

| **Challenge** | **Company A (Software)** | **Bank B (Banking)** |
|---|---|---|
| Skills gap | ML engineers + AI evaluation skills; hired 3, trained all engineers | Data scientists + regulatory expertise; hired 5, trained 8,000 staff |
| Workforce displacement concern | Managed through augmentation narrative; no headcount reduction | Managed through reskilling program; no layoffs; higher training investment |
| Automation bias | Code review: engineers stopped reading AI recommendations; mitigated with confidence score acknowledgment | AML: analysts accepted AI scores without evidence review; mitigated with mandatory secondary review |
| Model drift | 5 pp AUC drop in 4 months; weekly retraining + drift detection threshold | 5 pp AUC drop in 6 months; monthly retraining for critical models + PSI/ AUC monitoring |
| Bias and fairness | Not detected in test selection (non-human-impacting decisions) | Disparate impact on minority applicants (18% vs 12% false-negative rate); mitigated with fairness constraints |
| Explainability vs. performance | Feature importance explanations sufficient for engineers | Legally defensible causal explanations required by ECOA; chose interpretable model (6 pp accuracy cost) |

Table 17: Challenges and risks by domain: banking faces more stringent explainability and fairness requirements, creating higher performance costs but also greater regulatory compliance.

**Key insight.** The challenges and risks discussed above are not unique to either case study; they represent a common set of implementation challenges that any organization adopting AI-Based Architecture will encounter. The difference between the two domains is not *whether* these challenges occur, but *how severely* they impact the organization: banking's regulatory framework imposes stricter requirements for explainability, fairness, and oversight, which increases the cost of each challenge but also provides a structured framework for managing it. Organizations without such a framework (like Company A

in the early stages) must build their own governance mechanisms from scratch, which is more flexible but also more error-prone. The reference architecture and maturity model proposed in this paper are designed to provide that structured framework, regardless of regulatory density.

## 7.3 Future Research Directions

The findings from this study suggest several promising directions for future research. Each direction addresses a gap between what we observed in these two case studies and what is needed to generalize the results, validate the proposed architecture, and refine the maturity model.

**Longitudinal studies on AI-Based Architecture ROI.** The 18-month observation window captured the early and middle stages of AI adoption for both organizations but did not observe the long-term trajectory beyond this period. The projected 3-year ROI figures (Company A: 145%; Bank B: 89%) are based on extrapolation, not empirical data. A longitudinal study spanning 3–5 years would address several open questions: Do ROI figures continue to improve linearly, or do they plateau as the "low-hanging fruit" of automation is exhausted? Does the cost of AI infrastructure increase as the system scales (e.g., GPU compute costs, data storage costs, retraining costs)? Does the organizational investment in AI governance scale linearly with the number of AI systems deployed, or does it exhibit economies of scale (shared governance infrastructure) or diseconomies of scale (increasing complexity)? These questions are critical for organizations that must justify multi-year AI investments to stakeholders and cannot rely on 18-month projections.

**Multi-case comparisons across more industries.** This study examined two case studies from two industries (software development and banking). While the cross-domain similarities observed in these two cases suggest that AI-Based Architecture is domain-general, a larger sample of case studies across additional industries would strengthen (or weaken) this claim. Promising domains for future comparative case studies include healthcare (diagnostic AI, treatment recommendation, hospital operations), manufacturing (predictive maintenance, quality inspection, supply chain optimization), energy (grid management, renewable energy forecasting, fault detection), and government (public service automation, regulatory enforcement, policy analysis). Each domain presents unique regulatory, technical, and organizational constraints that may modify the patterns observed in this study. For example, healthcare's stricter patient privacy requirements and higher safety stakes may impose adoption timelines and governance requirements similar to banking, while manufacturing's physical-world constraints (hardware integration, sensor data quality, equipment downtime costs) may introduce new cost categories and KPI metrics not captured in this study.

**AI architecture patterns for specific domains.** The five-layer reference architecture proposed in this paper is domain-general, but it does not specify the domain-specific components that would be necessary in particular industries. For example, a healthcare AI-Based Architecture would need to incorporate HIPAA-compliant data governance, clinical decision support validation, and medical device regulatory requirements (FDA 21 CFR Part 820) into its Layer 3 (Decision and Policy) and Layer 5 (Observability and Governance). A manufacturing architecture would need to incorporate IoT sensor data pipelines, edge computing for low-latency control loops, and safety-certified autonomous execution for physical equipment. Future research should develop domain-specific exten-

sions of the five-layer architecture, specifying the additional components, data sources, and governance mechanisms that are required in each domain. These extensions could be organized as a pattern catalog following the software architecture pattern methodology [17], with each pattern describing the AI components, integration points, and governance constraints specific to a domain.

**Human-AI collaboration patterns and optimal intervention points.** The case studies documented the evolution of human-AI collaboration from Level 1 (human-in-the-loop for every decision) to Level 4 (human oversight of autonomous execution), but they did not systematically study the optimal intervention points – i.e., the specific moments and types of decisions at which human intervention adds the most value relative to the AI's autonomous performance. This is an important question because the cost of human intervention (time, expertise, cognitive load) must be weighed against the value of the intervention (error avoidance, regulatory compliance, customer trust). Future research could apply methods from human-computer interaction, cognitive psychology, and decision theory to identify the intervention points that maximize the combined human-AI system performance. One promising approach is the "human-on-the-loop" paradigm studied in autonomous vehicle research [29], where humans monitor autonomous systems and intervene only when the system encounters situations outside its authorized scope. Translating this paradigm to AI-Based Architecture would require defining the "authorized scope" for each AI component and the escalation criteria that trigger human intervention – a problem that sits at the intersection of AI governance, risk management, and human factors engineering.

**Validation of the maturity model.** The five-level maturity model (Section 2.3) was developed inductively from the case studies and validated through expert review, but it has not been validated through large-scale empirical testing. A psychometric validation study would assess the model's reliability (do different assessors assign the same maturity level to the same organization?) and validity (does maturity level predict KPI outcomes as the model claims?) across a large sample of organizations. This study would also need to address the model's sensitivity to different measurement approaches: is maturity level assessed through self-report surveys, technical audits, or KPI analysis, and do these methods produce consistent results? The maturity model's utility as an assessment tool depends on the answers to these questions.

## 7.4 Limitations

This study has several limitations that should be considered when interpreting the findings and generalizing the conclusions.

**Two-case study limits generalizability.** The primary limitation of this study is the small sample size: two case studies from two organizations in two industries. While the comparative design (software development vs. banking) provides some internal validity by enabling cross-domain pattern identification, the results cannot be statistically generalized to the broader populations of software companies or financial institutions. The findings are best understood as "illustrative" rather than "representative" – they demonstrate the phenomena of interest and provide evidence that AI-Based Architecture can produce the reported outcomes, but they do not establish the frequency or distribution of these outcomes across the population. The two-case design is consistent with the case study methodology in information systems research [41, 39], which values depth of analysis and theoretical insight over statistical generalization. However, readers should

be cautious about extrapolating the specific numerical results (e.g., 61% cycle time reduction, 45% defect rate reduction) to their own organizations without considering their specific context.

**Absence of negative cases.** The study documents two successful embeddings of governed agents. It does not document the processes that were left unchanged. Absence of those negative cases limits any claim about where AI should not be used. The principle stated in Section 1.1a is therefore normative as well as empirical.

**Anonymization limits reproducibility.** The confidential nature of the case study data requires extensive anonymization: organizational names, personnel identifiers, technology stack details, and specific financial figures have been modified or aggregated. While the anonymization protects the participating organizations, it also limits the reproducibility of the study: other researchers cannot independently verify the data or replicate the analysis on the same organizations. This limitation is inherent to industry-academia partnerships involving confidential organizational data and is acknowledged in the research design. The trade-off between confidentiality and reproducibility is a well-known challenge in case study research [20].

**Short observation window.** The 18-month observation window captures the early and middle stages of AI adoption but does not extend to the long-term trajectory. The findings are therefore most applicable to organizations in the first 18 months of AI-Based Architecture adoption; they provide less guidance for organizations that have already reached Maturity Level 4 or 5 and are exploring further improvements. The projected long-term ROI figures are based on extrapolation, not empirical data, and should be treated as hypotheses to be tested rather than established facts.

**Self-reported KPIs.** The KPI data reported in this study were collected through organizational self-reporting and internal audits, not through independent third-party measurement. While the KPIs are consistent across multiple data sources within each organization (e.g., CI/CD pipeline metrics, production incident logs, loan processing system metrics), they have not been independently validated. Self-reported KPIs are subject to reporting bias: organizations may overstate improvements or understate challenges to present their AI investment in a favorable light. Future research should incorporate third-party measurement (e.g., independent KPI audits, randomized controlled trials) to reduce this bias.

The limitations described above do not invalidate the study's findings; they define the scope within which the findings are valid and provide guidance for future research that can extend and validate these results.

# 8 Conclusion

This paper has proposed and empirically evaluated a framework for AI-Based Architecture – a five-layer architectural pattern in which AI-capable components serve as first-class architectural elements that govern the system lifecycle autonomously. The framework consists of three components: a five-layer reference architecture (Data and Knowledge, AI Model and Agent, Decision and Policy, Automation and Execution, Observability and Governance), a five-level maturity model (Reactive, Augmented, Automated, Adaptive, Autonomous), and a comparative case study methodology that validates the framework across two domains. The principal findings of this study are summarized below.

**Finding 1: AI-Based Architecture delivers measurable value at the bottleneck.**

Both Case Study A (software development) and Case Study B (banking) demonstrated significant improvements across all four metric categories. Productivity improvements ranged from 61% (Company A's cycle time reduction in the release pipeline) to approximately 90% (Bank B's loan processing time reduction in the credit-risk path). Quality improvements ranged from 45% (Company A's defect rate reduction) to 71% (Bank B's processing error rate reduction). Cost improvements yielded positive ROI within 18 months at the process in scope (Company A: 38%; Bank B: 22%) with projected 3-year ROI of 145% and 89% respectively. Neither figure is a firm-wide return. Adoption metrics showed AI acceptance rates climbing from 38–45% to 72–82% within 18 months, with optimal acceptance rates in the 75–85% range. The simultaneous improvement across all four metric categories contradicts the traditional engineering assumption that productivity and quality are inversely related.

**Finding 2: Regulatory density slows adoption but increases eventual impact.** The moderating effect of regulatory density was empirically supported: organizations with high regulatory density (Bank B) required longer adoption timelines (18 months vs. 12 months for Company A) but achieved greater eventual KPI improvements because their legacy processes were more wasteful. The regulatory framework functioned not merely as a constraint but as a quality discipline that forced organizations to invest in data quality, model validation, and governance infrastructure before AI deployment, which in turn enabled greater AI impact. The regulatory density moderating effect can be modeled as an inverse relationship between regulatory density and adoption speed, and a direct relationship between regulatory density (combined with legacy process inefficiency) and eventual KPI impact.

**Finding 3: Team transformation is as important as technology transformation.** Both organizations invested heavily in workforce reskilling (Company A: 24 hours per engineer; Bank B: 120 hours per employee) and achieved better outcomes when they framed AI adoption as augmentation rather than replacement. The workforce reskilling imperative – shifting human effort from execution to governance and oversight – was a consistent pattern across both domains and was necessary for achieving high AI acceptance rates. Organizations that planned for workforce transformation achieved 10–15 percentage point higher acceptance rates than those that communicated workforce reduction.

**Finding 4: KPIs must be domain-specific but share common measurement principles.** While the specific KPIs used in software development (cycle time, defect rate, deployment frequency, MTTR) differ from those used in banking (loan processing time, false-positive rate, fraud detection rate, investigation yield), they share common measurement principles: time-based metrics are more reliable than output-based metrics; false positive and false negative rates have asymmetric stakeholder impacts; and productivity metrics must be interpreted alongside quality metrics to avoid the "productive but broken" pattern where speed gains come at the expense of output quality.

**Contributions.** This paper makes four contributions to the literature. First, it proposes a five-layer reference architecture for AI-Based Architecture that extends existing autonomous system and AIOps/MLOps architectures by embedding AI into the architectural specification rather than treating it as an operational enhancement. Second, it proposes a five-level maturity model that classifies organizations by the degree of autonomous agency exercised by AI components, focusing on the autonomy-oversight axis rather than organizational readiness. Third, it introduces the concept of regulatory density as a moderating factor on AI adoption speed and KPI impact, providing a framework

for understanding why different industries experience different AI adoption trajectories. Fourth, it provides two detailed comparative case studies that empirically validate these conceptual contributions and offer practical guidance for organizations considering AI-Based Architecture adoption.

**Recommendations for organizations.** Based on the findings of this study, we offer the following recommendations for organizations considering AI-Based Architecture adoption:

1. **Start with the "easy automation first" sequence.** Automate the simplest, most rule-based tasks first (test selection, document verification, routine monitoring) before progressing to complex autonomous decisions. This sequence minimizes risk while building organizational confidence.

2. **Invest in governance infrastructure before scaling autonomous execution.** The Decision and Policy Layer (Layer 3) and Observability and Governance Layer (Layer 5) are not optional add-ons; they are prerequisites for safe autonomous execution. Organizations that deploy autonomous AI without governance infrastructure risk quality degradation, regulatory non-compliance, and loss of stakeholder trust.

3. **Use a 3–5 year investment horizon.** The ROI data from this study show that positive ROI is achieved within 18–28 months, with continued improvement through year 3. Organizations that evaluate AI investments on a 12-month payback period are likely to underinvest and miss the majority of the value.

4. **Frame AI adoption as augmentation, not replacement.** The acceptance rate data show that the organizational narrative around AI adoption has a measurable impact on adoption outcomes. Organizations that communicate augmentation achieve higher acceptance rates, lower turnover, and greater long-term value realization.

5. **Invest in workforce reskilling alongside technology investment.** The skills gap was the single largest operational constraint in both case studies during the first 12 months. Organizations that invest in training (24–120 hours per employee, depending on domain complexity) achieve faster acceptance rate improvement and higher long-term value.

6. **Measure what matters.** Use time-based metrics (cycle time, lead time, queue time) rather than output-based metrics (lines of code, transactions processed) as the primary productivity measure. Track AI-augmentation ratio alongside override rates to diagnose the quality of human-AI collaboration. Use acceptance rates of 75–85% as the target, not 100%.

The five-layer reference architecture, the five-level maturity model, and the empirical evidence presented in this paper provide a foundation for understanding and implementing AI-Based Architecture in diverse knowledge-intensive domains. The framework is not a blueprint to be copied verbatim; it is a conceptual lens through which organizations can understand the architectural, organizational, and governance dimensions of AI adoption. The cases studied here – Company A in software development and Bank B in banking – represent two points on a broad spectrum of AI adoption trajectories. The patterns we have identified are likely to generalize to other domains, but the specific implementation

details will differ. The contribution of this paper is not the details of any particular implementation; it is the architecture, the maturity model, and the comparative framework that make such implementations possible.

# References


[1] Charu C. Aggarwal. Machine learning for financial services: Applications and challenges. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 17(1):1–145, 2023.

[2] Farnaz Ahmed, Christopher R. Cooper, Vickie Huang, and Robert M. Schatz. Predictive modeling for credit card fraud detection using supervised learning. *Proceedings of the 2015 Workshop on Cyber-Security Experimentation and Computation*, pages 65–73, 2015.

[3] Paulo Almeida, Marco T. Valente, Rafael Coelho, and Natalia Garcia. Autonomous agents in software engineering: A systematic literature review. *Information and Software Technology*, 130:106438, 2021.

[4] Ross Bartlett, Joseph Weatherall, et al. Explainable artificial intelligence in finance: Methods and regulatory requirements. *Annual Review of Financial Engineering*, 6:1–28, 2023.

[5] Michele Battisti, Serena Chiappa, Mario Kocher, et al. Machine learning for credit risk assessment: A systematic review. *Journal of Banking and Finance*, 143:106593, 2022.

[6] Bertrand Bellay, Euripides Mendes, Thomas Neubauer, and Mario Weysser. Self-adaptation of software product lines: A systematic mapping study. *Journal of Systems and Software*, 104:177–193, 2014.

[7] Erik Brynjolfsson and Andrew McAfee. *The Benefits of the AI Workplace: Productivity, Innovation, and Worker Augmentation.* NBER Working Paper No. 31157, 2023.

[8] Carnegie Mellon University. *CMMI-DEV V2.0: Capability Maturity Model Integration for Development.* CMU/SEI-2018-TR-017, Pittsburgh, PA, 2018.

[9] Li Chen, Wei Zhang, and Jun Wang. Automating loan processing with natural language processing and machine learning: A case study. *Financial Innovation*, 8:1–20, 2022.

[10] Mark Chen, Martin Weert, Adam Sweeney, Mark Goyal, Greg Girland, Ji Lin, Dario Sharma, Ting Lu, Mark Mioc, Sam White, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

[11] Databricks. Mlflow: An open source machine learning platform, 2018. Accessed: January 2025.

[12] European Parliament and Council of the European Union. Regulation (eu) 2022/2554 on digital operational resilience for the financial sector, 2022. Accessed: January 2025.

[13] European Parliament and Council of the European Union. Regulation (eu) 2024/1689 laying down harmonised rules on artificial intelligence (artificial intelligence act), 2024. Accessed: January 2025.

[14] Federal Reserve Board. Sr 11-7: Guidance on model risk management. Technical report, Board of Governors of the Federal Reserve System, 2013. Accessed: January 2025.

[15] Yichao Feng, Yuyao Li, Xiaohu Wang, and Zhenyu Zhang. Codegen: Large language models can generate code with few-shot planning and execution. *arXiv preprint arXiv:2307.15337*, 2023.

[16] Forrester Research. The ai adoption curve: From experimentation to transformation, 2022. Accessed: January 2025.

[17] Martin Fowler. *Patterns of Enterprise Application Architecture*. Addison-Wesley, 2002.

[18] Carl Benedikt Frey and Michael Osborne. *The Future of Employment: How Susceptible Are Jobs to Computerisation?*, volume 114. Technological Forecasting and Social Change, 2017.

[19] Gartner Research. Gartner ai maturity model, 2023. Accessed: January 2025.

[20] Michael Gibbert, Winona Ruigrok, and Brian Wicki. What passes as a rigorous case study? *Management Decision*, 46(10):1466–1477, 2008.

[21] GitHub. Github copilot: Your ai pair programmer, 2023. Accessed: January 2025.

[22] Google Cloud. Kubeflow: Machine learning toolkit for kubernetes, 2018. Accessed: January 2025.

[23] Xiaoye Gu, Hongyu Jin, Shizhu Li, Shiqi Wang, Shiyang Wang, Jianbin Sun, and Yang Liu. Is chatgpt good at software vulnerability detection? an empirical study. *arXiv preprint arXiv:2304.00612*, 2023.

[24] Adrien Huyet, Nicolas Brousmiche, Pierre Le Berre, et al. Aiops: An emerging paradigm for enterprise it operations. *IEEE International Conference on Services Computing*, pages 1–8, 2019.

[25] IEEE Standards Association. Ieee p7000 series: Standard models for addressing ethical and societal concerns of technology-based systems, 2023. Accessed: January 2025.

[26] John D. Lee and Katharine A. See. Trust in automation: Designing for appropriate engagement during human–machine interaction. *International Journal of Human–Computer Studies*, 61(5):559–584, 2004.

[27] Byung-Ju Lim and Jong-Hyeok Noh. Deep learning in financial services: Fraud detection and risk assessment. *IEEE Access*, 5:22508–22517, 2017.

[28] Ken McCue, Andries Kruger, Patel Labhesh, et al. Automated regulatory compliance reporting using ai: Frameworks and challenges. *Journal of Financial Regulation and Compliance*, 31(2):245–267, 2023.

[29] Nathalie Merat, Hugh Jamson, Richard Simons, Oscar Lim, and Stephen Charlton. Autonomous vehicles: The acceptance of human–machine teaming. *Transportation Research Part F: Traffic Psychology and Behaviour*, 59:1–15, 2018.

[30] Florin Moldovan and Zhenyu Pan. Automation bias and ai trust in knowledge work: A systematic review. *ACM Computing Surveys*, 54(7):1–36, 2021.

[31] National Institute of Standards and Technology. Ai risk management framework (ai rmf) 1.0, 2023. Accessed: January 2025.

[32] Raja Parasuraman and Vicki Riley. Humans and automation: Use, misuse, disuse, abuse. *Human Factors*, 39(2):230–253, 2000.

[33] Alan Powell, Harleen Singh, and Wei Zhang. Autonomous decision-making in financial services: Governance challenges and regulatory responses. *Journal of Financial Regulation*, 9(1):45–72, 2023.

[34] Seldon Technologies. Mlserver: Standardising the interface for machine learning inference, 2021. Accessed: January 2025.

[35] Mohammd Shahin, Mohammad Ghafari, and Stefan Wagner. Towards autonomy in software engineering. *IEEE Software*, 37(5):60–69, 2020.

[36] Mohammd Shahin, Mohammadreza Khaki, Mohammad Ghafari, and Stefan Wagner. A systematic review on self-adaptive software. *Journal of Systems and Software*, 156:116–138, 2019.

[37] Wenhu Shi, Peng Zhou, Xiaoyang Wang, et al. Root cause analysis in microservice architectures using machine learning: A survey. In *IEEE International Conference on Services Computing*, pages 1–8, 2021.

[38] Harleen Singh, Simranpreet Kaur, and Pankaj Kumar. Enhancing anti-money laundering systems with graph neural networks. *Computers & Security*, 124:102975, 2023.

[39] Richard Venable, Stefan Rosemann, Frank Breil, and Gavin Young. The disrupted world of design science research. *Designing the Future through IS Innovation: Paradigm Conflicts, Practical Dilemmas and Urgent Questions*, pages 25–38, 2020.

[40] Boyang Wang, Qianxiang Liu, and Cheng Chen. Code review assistance with large language models: Opportunities and challenges. In *Proceedings of the ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, pages 1–11, 2022.

[41] Robert K. Yin. *Case Study Research and Applications: Design and Methods*. SAGE Publications, 6th edition, 2018.

[42] Nicoletta Zannone, Simon Klinkhammer, and Oliver Zimmermann. Ai-gated ci/cd pipelines: Automated test selection in devops workflows. In *Proceedings of the 29th International Conference on Software Analysis, Evolution and Reengineering*, pages 412–423, 2022.

[43] Zheng Zhang, Jianbin Liu, and Aihua Xiao. Log analysis for cloud infrastructure using machine learning: A survey. *ACM Computing Surveys*, 53(5):1–36, 2020.