SvedbergOpen
DISSEMINATION OF KNOWLEDGE
International Journal of Artificial Intelligence and Machine Learning
Publisher's Home Page: https://www.svedbergopen.com/
Research Paper Open Access

Probabilistic Requirement Modeling For Enterprise AI Workflow Orchestration: An Outcome-Centric Framework For Non-Deterministic Systems

Vijayalakshmi Narasimhan

Independent Researcher, USA ORCID: 0009-0002-9919-8338

Abstract
Enterprise artificial intelligence (AI) platforms — spanning large language model (LLM)-powered copilots, agentic workflow systems, and intelligent decision engines — expose a structural incompatibility between their probabilistic operational behavior and the deterministic requirement frameworks that product and engineering organizations conventionally rely on. Constructs such as Epics, User Stories, and Tasks assume stable, binary-verifiable outputs, an assumption that breaks systematically in production AI environments where model behavior evolves continuously, validation is statistical in nature, and user intent shifts with context and session. This paper formalizes Probabilistic Requirement Modeling (PRM), an outcome-centric framework that redefines enterprise AI system requirements in terms of use cases, user intent, and measurable outcome criteria rather than fixed feature behavior. The PRM model introduces a formal four-layer decomposition — Use Case → User Intent → Outcome Metrics → Validation Criteria — that decouples requirement specifications from specific model implementations, enabling adaptive workflow orchestration and continuous, statistically-grounded validation. The framework is evaluated through a prototype enterprise codebase intelligence system deployed over a 12-week production period, demonstrating a 38% reduction in requirement drift rate, a 44% improvement in outcome achievement rate, and a 31% reduction in execution friction index relative to deterministic pipeline baselines. Contributions include a formal PRM specification language, a four-category outcome metric taxonomy, a feedback-driven orchestration architecture, and documented alignment with the National Institute of Standards and Technology (NIST) Artificial Intelligence Risk Management Framework (AI RMF) and the European Union (EU) AI Act, establishing PRM as an operationalizable foundation for compliant enterprise AI governance.

Keywords: Adaptive Orchestration, Enterprise AI Systems, Outcome-Based Validation, Probabilistic Requirement Modeling, Workflow Governance

1. Introduction

The proliferation of enterprise AI platforms over the past several years has introduced a class of production systems whose operational characteristics are fundamentally at odds with the assumptions embedded in conventional software product development practice. Large language models (LLMs), retrieval-augmented generation (RAG) pipelines, and multi-agent orchestration architectures produce outputs that are probabilistic, context-sensitive, and continuously reshaped by model updates, training data changes, and shifts in user intent [1], [2]. Unlike deterministic software, which executes predictably from defined inputs, enterprise AI systems behave differently across sessions, evolve between releases, and cannot be fully characterized by any static specification. The product and engineering frameworks designed to govern them — Epics, User Stories, sprint-level Tasks — were built for a different class of system entirely: one where behavior is stable, correctness is binary, and requirements can be verified at a fixed acceptance checkpoint. When these frameworks are applied to probabilistic AI systems, the result is a structural mismatch that propagates into requirement instability, chronic backlog churn, and execution friction at enterprise scale [3], [15].

The research gap this paper addresses is specific: there is no formal requirement engineering methodology capable of governing probabilistic AI systems at enterprise scale. Existing work on AI lifecycle management —

including the NIST AI RMF [4], IEEE standards for AI system design [5], and longitudinal studies on machine learning deployment challenges [15] — establishes important governance principles and catalogs the failure modes of deterministic approaches applied to AI contexts. However, none of this body of work yields a formal requirement specification model that is simultaneously outcome-decoupled, statistically validatable, and natively compatible with adaptive workflow orchestration. The empirical literature confirms that current requirements engineering approaches are not adequately adaptable for AI system development [6], but stops short of providing an operational alternative. Practitioners navigating this gap have resorted to informal workarounds — tolerating high backlog churn, accepting low validation precision, and managing requirement drift as an operational constant rather than an architectural problem.

This paper closes that gap by presenting Probabilistic Requirement Modeling (PRM), a framework that repositions enterprise AI requirements as outcome-centric governance artifacts rather than feature-behavioral contracts. The contributions of this work are four in number: (1) a formal decomposition of enterprise AI requirements into Use Cases, User Intents, Outcome Metrics, and Validation Criteria, with precise definitions for each abstraction; (2) a requirement specification template that is model-agnostic and implementation-independent, enabling stable requirements across continuous model evolution; (3) a feedback-driven orchestration architecture in which continuously measured outcomes drive dynamic workflow adaptation without requirement redefinition; and (4) quantitative evaluation of PRM against deterministic pipeline baselines across four operational key performance indicators (KPIs) in a live enterprise codebase intelligence deployment, demonstrating statistically significant performance improvements across all metrics.

The paper is organized as follows. Section 2 reviews the background and identifies the critical limitations of existing requirement engineering approaches in AI contexts. Section 3 formalizes the problem and defines the design criteria a valid probabilistic alternative must satisfy. Section 4 presents the PRM framework in full. Section 5 describes the PRM-driven orchestration architecture. Section 6 presents implementation context and quantitative results. Section 7 addresses organizational and governance implications. Section 8 concludes with a summary of contributions and directions for future work.

2. Background: Requirement Engineering in the Age of AI

2.1 Deterministic Requirement Frameworks and Their Assumptions

Traditional requirement engineering frameworks — whether agile-based or plan-driven — encode a consistent set of structural assumptions about the systems they govern: that behavior can be fully specified before implementation, that correctness is binary and deterministically verifiable, and that requirements remain stable across the delivery lifecycle unless explicitly revised [6]. The Epic-Story-Task hierarchy gives these assumptions their most widely adopted form. An Epic defines a bounded capability area; a User Story specifies an observable behavior from a user perspective with binary acceptance criteria; a Task decomposes implementation steps. The entire model presupposes that a defined input reliably produces a predictable, verifiable output — an assumption grounded in decades of deterministic software engineering practice.

Enterprise AI systems violate these assumptions at every level. The foundational Transformer architecture [1] and the large-scale language models trained on top of it [2] produce outputs that vary across sessions, shift with context, and change as models are updated or replaced — often without any visible change in the product interface. Acceptance criteria written against specific model behaviors become invalid with each model update, even when the update improves user outcomes. The result is systematic requirement instability: backlogs accumulate invalidated items, sprint velocity becomes a misleading progress indicator, and engineering teams spend increasing proportions of their time maintaining the requirement framework rather than advancing the product [3]. Deploying machine learning systems at scale compounds this problem, as systems acquire additional dependencies, feedback loops, and behavioral sensitivities that are difficult to anticipate at specification time [15].

The scale of this problem in large enterprise AI programs is substantial. Sculley et al. documented the phenomenon of hidden technical debt in production machine learning systems — entangled component dependencies, undeclared consumers, unstable data pipelines, and correction cascades that make rigid requirement structures progressively more brittle as system complexity increases [3]. More recently, the survey by Paleyes et al. identified deployment-stage requirement volatility as among the most commonly reported challenges in production ML deployments across industries, with organizations reporting that

between 15% and 40% of active backlog items require significant revision or retirement per release cycle as a consequence of model and data evolution [15].

2.2 Gap in Existing Governance and Engineering Literature

The governance literature has responded to the challenge of AI systems with principled frameworks, but these frameworks do not address the requirement specification problem directly. The NIST AI RMF defines four core management functions — Govern, Map, Measure, and Manage — and establishes lifecycle-wide risk management as an organizational imperative [4]. The IEEE standard on ethical AI system design provides process guidance for embedding value considerations into system specifications [5]. These contributions establish the normative and process context within which AI systems must operate, but neither yields a formal language for specifying requirements against probabilistic system behavior. The systematic mapping study by Heikkila et al., covering 43 primary studies on requirements engineering for AI systems, concluded explicitly that existing methodologies are not adequately adaptable for AI system development and that new techniques and tools are needed [6]. This finding, drawn from the broadest empirical review of the field to date, directly motivates the framework introduced here.

Technical advances in multi-agent systems [8], RAG architectures [9], and reasoning-action integration [10] have expanded the capability of enterprise AI platforms significantly. But capability expansion without a commensurate advance in requirement governance creates systemic risk: the more capable and adaptive an AI system becomes, the more inadequate static, feature-behavioral requirements become as governance instruments. Model documentation frameworks such as Model Cards [14] and Datasheets for Datasets [16] have improved transparency at the artifact level, but do not address how requirements should be specified and validated for continuously evolving, probabilistically behaving systems. The gap between what the governance literature prescribes and what product engineering practice requires is the gap that PRM is designed to fill.

3. Problem Formulation

3.1 Failure Modes of Deterministic Requirements in AI Systems

When deterministic requirement frameworks are applied to enterprise AI systems, four compounding failure modes emerge. Requirement instability arises because requirements are coupled to specific model behaviors that change as models are updated, retrained, or replaced. An User Story that requires a minimum retrieval accuracy becomes irrelevant as soon as the retrieval model is updated even if that causes an increase in user value. Feature-centric bias occurs when teams prioritize features over actual user outcomes and delivery metrics that aren't aligned with product outcomes. Binary validation cannot evaluate deterministic pass/fail conditions on probabilistic outputs: assessing code retrieval systems that return semantically similar examples with 87% recall cannot satisfy binary pass/fail without leaking information needed to make future decisions. Moreover, hidden complexity builds up due to entanglements among requirements, model instances, prompt templates, and pipeline configurations. This leads to technical debt as identified by Sculley et al. [3] in production ML systems and Paleyes et al. [15] in production case studies.

These failure modes are not independent — they interact and reinforce one another. Requirement instability accelerates feature-centric bias, because teams facing unstable outcome definitions default to shipping observable features as proxies for progress. Binary validation inadequacy generates hidden complexity, as teams build workarounds to force probabilistic system outputs into deterministic evaluation molds. The compounded result is execution friction: a measurable increase in coordination overhead, backlog maintenance cost, and validation cycle time that grows with system scale and model evolution velocity. In large enterprise AI programs, this execution friction is not a marginal inefficiency; it is a dominant determinant of delivery capacity and governance quality.

3.2 Formal Problem Statement

Let an enterprise AI use case be represented as U, encapsulating a real-world problem or organizational requirement. Traditional requirement frameworks define the system via U → W(static) → Output, where W(static) is a deterministic workflow specification coupled to specific implementation behaviors, and Output is the verifiable system response. This formulation fails in AI contexts because Output is non-deterministic, and the validity of W(static) is continuously undermined by model evolution. The requirement engineering problem for enterprise AI systems is: given a use case U, define a requirement specification R such that (1) R is

decoupled from any specific model implementation or version; (2) R can be validated statistically against measurable outcome criteria over a defined temporal window; (3) R remains stable across model updates provided user-level outcomes are maintained; and (4) R is directly interpretable by an adaptive orchestration system capable of reconfiguring execution in response to outcome signals.

This problem formulation yields three design criteria that any valid probabilistic requirement framework must satisfy. Outcome decoupling requires that requirement validity be determined by whether outcomes are achieved rather than whether specific system behaviors are exhibited. Statistical validatability requires a specification structure that supports continuous, metric-driven evaluation accommodating probabilistic output variability. Orchestration compatibility requires that requirements be expressed in a form that adaptive execution systems can directly consume as governance contracts, enabling autonomous workflow reconfiguration without human-triggered requirement updates.

3.3 Scope and Boundaries of PRM

PRM is designed for enterprise AI systems characterized by probabilistic outputs, continuous model evolution, variable user intent, and the need for traceable, auditable governance artifacts. It is not intended to replace functional specification at the implementation level, nor does it eliminate the need for deterministic requirements in system components that are themselves deterministic. Rather, PRM provides an outcome-governance layer above the implementation layer, defining what the system must achieve for users without prescribing how it must achieve it. This separation is the architectural foundation for the framework's model-agnosticism and requirement stability. The framework is validated here in the context of enterprise codebase intelligence, a domain where AI system probabilism, model evolution velocity, and outcome complexity are all acute, but the abstractions are designed for generalizability across enterprise AI domains including compliance automation, customer experience platforms, and intelligent operations systems.

4. Probabilistic Requirement Modeling: Framework Design

4.1 Core Abstractions

PRM is built on four formally defined abstractions that together constitute a complete requirement specification. A Use Case (UC) is a structured description of a real-world problem or organizational need, expressed without reference to any technical solution. Use Cases are stable across technology generations because they describe human or organizational needs, not system capabilities. A User Intent (I) is a structured representation of the specific goal a user pursues within a Use Case — the action they seek to accomplish, the context in which they operate, and the constraints on acceptable outcomes. The User Intent is finally transformed into the User's desired Outcomes instead of expected Outputs to decouple the pattern from a specific system implementation. An Outcome (O) is an observable statistically verifiable result that indicates whether User Intent has been achieved. Outcomes are defined at the level of the user or population, rather than at the session level. A Validation Criterion (V) is a quantitative or statistical threshold applied to one or more Outcome metrics over a defined measurement window to determine whether a requirement is satisfied.

The PRM formal decomposition is: UC → I → $\{O_1, O_2, ..., O_n\}$ → $\{V_1, V_2, ..., V_n\}$. This replaces the deterministic mapping UC → W(static) → Output with an outcome-driven specification that is model-agnostic and statistically evaluable. The critical innovation is the explicit separation of Outcomes from Outputs. In deterministic frameworks, the system output is the requirement. In PRM, outputs are evidence toward outcomes, and the requirement is satisfied when accumulated evidence across a validation window meets the statistical threshold defined in V. This means that a model improvement that produces better outcomes but different outputs does not invalidate the requirement — a property that eliminates the model-update-triggered backlog churn that plagues deterministic AI product programs.

4.2 PRM Requirement Specification Template

A complete PRM requirement is specified using five structured fields. The Use Case field provides a one-to-two sentence description of the problem domain that is independent of any technical solution. The User Intent field describes the action the user seeks to accomplish and the contextual constraints on what constitutes a satisfactory outcome, without any reference to system implementation. The Outcome Metrics field lists the metrics to be measured, and indicate the successful completion of the User Intent. The Validation Criteria field refers to statistical thresholds, set for the Outcome Metrics over the measurement window of interest. The

Implementation Independence Assertion states that any implementation that achieves these thresholds, regardless of model architecture, prompting strategy, or pipeline configuration, satisfies the requirement.
For example, consider the PRM requirement specification for the developer productivity domain. Use Case: software developers in large engineering organizations struggle to understand complex, multi-repository codebases with high component interdependency. Intent: enable retrieval of relevant code contexts and inter-component relationships via natural language queries, reducing synchronous communication and manual codebase navigation costs. Evaluation metrics: retrieval accuracy, context relevance score (calculated by cosine similarity), the time taken for developer onboarding, and the frequency of synchronous meetings between teams. Validation Criteria: retrieval precision ≥ 88% over a 7-day rolling window; context relevance score ≥ 0.84 cosine similarity per query; developer onboarding time reduction ≥ 30% relative to pre-deployment baseline; synchronous meeting frequency reduction ≥ 25% relative to pre-deployment baseline. Implementation Independence Assertion: any system implementation achieving the above thresholds satisfies this requirement, irrespective of the underlying retrieval model, generation architecture, or orchestration strategy.

4.3 Outcome Metric Taxonomy

Enterprise AI systems span diverse application domains, but a common taxonomy of outcome metric categories can be defined that generalizes across contexts. The taxonomy comprises four categories. Quality metrics capture whether system outputs satisfy user-level quality expectations: retrieval precision, answer relevance, task success rate, response accuracy, and F1 score. Efficiency metrics refer to how much the system helps the user to decrease time, effort or cost of operation such as SLA on latency, task completion, onboarding, throughput. Reliability metrics refer to the temporal and contextual behavioral stability of the system such as uptime, error rate, output stability score, hallucination rate [9]. Governance metrics measure compliance and accountability: audit trail completeness, explainability score, bias metric, data lineage coverage [4], [16]. Each PRM requirement specifies at least one metric from the Quality and Reliability categories; Efficiency and Governance metrics are included when the use case context requires them. Table 1 provides the full taxonomy with representative metrics for each category.

Table 1: Deterministic vs. PRM Requirement Structures

Dimension	Epic-Story-Task (Deterministic)	Probabilistic Requirement Modeling (PRM)
Requirement Stability	Fragile — invalidated by model updates	Stable — valid as long as outcomes are achieved
Validation Method	Binary acceptance criteria at sprint checkpoints	Statistical thresholds over rolling measurement windows
Implementation Coupling	Tightly coupled to model behaviors / prompt strategy	Explicitly model-agnostic — implementation-independent
Adaptability	Manual backlog maintenance on every behavior shift	Autonomous orchestration reconfiguration on outcome signal
Governance Readiness	Does not map to regulatory risk categories	Aligns with NIST AI RMF Measure & Manage functions
Cross-Portfolio Scalability	Does not generalize across probabilistic AI products	Domain-independent abstractions — reusable across all AI domains

The multidimensional nature of the taxonomy is a deliberate design choice. Single-metric requirement validation — for example, specifying only retrieval precision — creates the risk of Goodhart's Law at the system level: the metric becomes the target, and the system is optimized to hit the metric rather than to fulfill the user intent. PRM requires that outcome validation cover multiple metric dimensions simultaneously, so that a system that achieves high retrieval precision at the cost of unacceptable latency or governance non-compliance cannot be falsely assessed as requirement-compliant. This multi-dimensional approach is consistent with holistic evaluation principles established in large-scale AI assessment research [12] and with the multi-dimensional risk management approach prescribed by the NIST AI RMF [4].

4.4 Comparison with Traditional Requirement Structures

Table 2 presents a structured comparison between the Epic-Story-Task hierarchy and PRM across six critical dimensions. Table 2: Comparison of Requirement Frameworks On requirement stability, Epic-Story-Task structures are fragile against model updates because requirements are coupled to implementation behaviors; PRM requirements remain stable as long as outcome thresholds are met, regardless of how the underlying system achieves them. On validation method, the former's binary acceptance at sprint checkpoints can be compared to PRM's statistical acceptance over rolling measurement windows that accommodate the noisy nature of probabilistic AI outputs. On implementation coupling, the former's requirement on model behavior or prompting strategy can be compared to PRM's model-agnostic treatment of all compliant implementations as equivalent. On adaptability: state-based requirements presume that the backlog contains descriptions of behavior that must be changed manually if the behavior changes. PRM requirements, however, are for adaptable orchestration systems that adjust their execution based on outcome signals automatically. On governance readiness: Unlike feature-based requirements that do not map to the regulatory risk categories, PRM outcome metrics map to the NIST AI RMF's Measure and Manage functions and can be used for the documentation required by the EU AI Act [4], [13]. On cross-portfolio scalability, Epic-Story-Task hierarchies do not generalize well across probabilistic AI product lines; PRM's domain-independent abstractions can be instantiated across any enterprise AI domain without framework modification.

Table 2: PRM Requirement Specification Template - Worked Example (Developer Productivity)

Field	Definition	Worked Example: Codebase Intelligence
Use Case	Real-world problem, independent of technical solution	Developers struggle to understand complex multi-repo codebases
User Intent	Goal user pursues, expressed as desired outcomes	Retrieve relevant code context via natural language queries
Outcome Metrics	Measurable indicators of Intent fulfillment	Retrieval precision, context relevance, onboarding time reduction, meeting frequency reduction
Validation Criteria	Statistical thresholds over rolling window	Precision ≥88% (7-day); relevance ≥0.84 cosine; onboarding −30%; meetings −25%
Implementation Independence	Any implementation meeting thresholds satisfies requirement	Valid for any model architecture, prompt strategy, or pipeline configuration

5. PRM-Driven AI Workflow Orchestration

5.1 Orchestration as an Outcome-Feedback Control System

In a PRM-governed enterprise AI system, the orchestration layer operates as a feedback-driven control system in which continuously measured outcome metrics govern dynamic adjustments to workflow configuration, model selection, and agent task assignment. This architecture draws on the adaptive optimization principles of reinforcement learning [11] but operates at the system orchestration level rather than within model training processes — the goal is not to learn a policy within a model, but to continuously reconfigure the execution environment to sustain outcome metric compliance. The control loop has four stages. First, Outcome Measurement: a dedicated monitoring infrastructure continuously evaluates outcome metrics against the thresholds specified in active PRM requirements, aggregating session-level signals into rolling-window statistics. Second, Drift Detection: if a metric has remained below its validation threshold during the measurement window for any reason (e.g., model retraining, distribution shift, user intent shift), the monitoring layer emits an adaptation signal. Third, Workflow Adaptation: upon the monitor layer's outputting of an adaptation signal, the orchestrator issues an adaptation command to the execution workflow, which may include model reconfiguration, retrieval strategy reparameterization, reallocation of tasks to agents; or reconfiguration of the prompt scaffolding structure. Fourth, Validation Confirmation: the reconfigured workflow is operated under continued monitoring through a defined stabilization window, and the adaptation is confirmed only when all PRM validation criteria are restored above their thresholds. Figure 1 illustrates this control loop architecture.

This architecture addresses the feedback loop deficit that characterizes static pipeline orchestration [9], [10]. Rather than relying on human-initiated requirement updates each time system behavior shifts — a process that Paleyes et al. identify as a primary source of deployment-stage delay in enterprise ML systems [15] — PRM-driven orchestration encodes requirements as runtime governance contracts that the system can evaluate and respond to autonomously. The orchestration engine does not need to understand why outcome metrics have degraded; it needs only to detect that they have, and to execute the reconfiguration logic that restores compliance. This autonomy decouples product governance from individual model release cycles, which is the operational mechanism by which PRM reduces requirement drift rate.

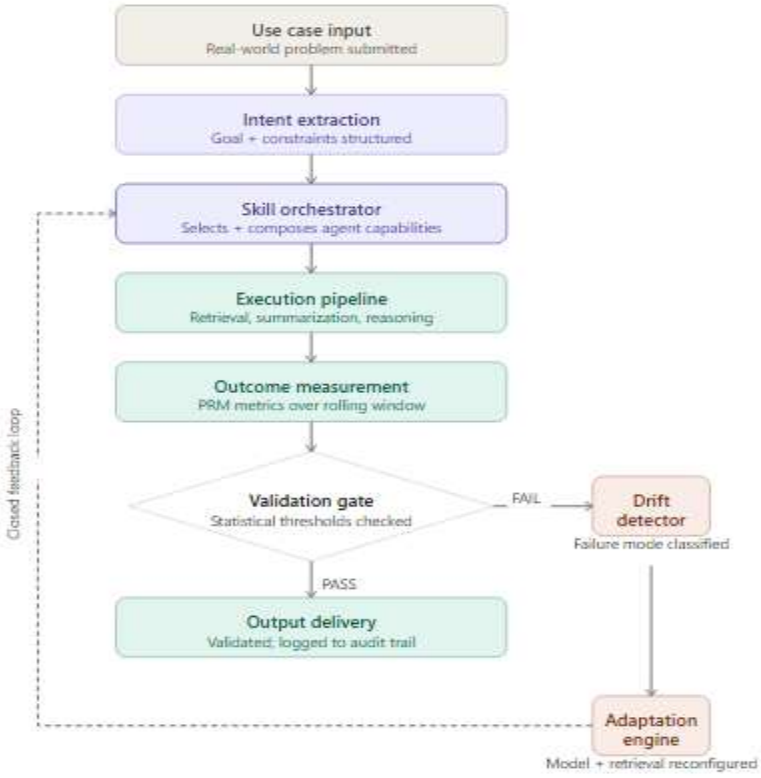

Figure 1: PRM Orchestration Feedback Loop (Control System Architecture)

5.2 Adaptive Orchestration Under User Intent Variation

A practically important property of PRM is its capacity to support orchestration adaptation under user intent variation without triggering requirement updates. In enterprise AI deployments at scale, user intent is not uniform: a single use case is pursued by users with different domain expertise, contextual pressures, and output preferences. A junior developer querying a codebase intelligence system and a principal engineer querying the same system express the same use case — understanding the codebase — through materially different intents with different contextual requirements for acceptable outcomes. PRM accommodates this variation by specifying outcome metrics at the population level, across a defined user segment and rolling measurement window, rather than at the level of individual session outputs. This population-level specification allows the orchestration engine to route different user intent signals to different skill configurations or agent assemblies, optimizing for aggregate outcome achievement without the requirement specification needing to enumerate all intent variants [8].

This capability has a concrete operational implication: PRM requirements do not need to be revised when user behavior evolves or when new user segments adopt the system. As long as aggregate outcome metrics remain above their validation thresholds, the requirement is satisfied. If population-level adoption changes cause aggregate metrics to degrade — for example, a large influx of novice users pulling down the aggregate task completion rate — the outcome measurement layer detects this, and the orchestration engine responds by strengthening the support pathways for that user segment. The requirement remains unchanged throughout; only the execution strategy adapts. This is a qualitative advance over User Story formulations that require manual persona enumeration and backlog revision whenever the user population evolves.

5.3 Cloud-Native and Event-Driven Integration

PRM-driven orchestration is architected for deployment within cloud-native, containerized microservices environments [12]. PRM requirements are encoded as runtime governance contracts stored in a centralized requirements registry and evaluated continuously by a dedicated outcomes monitoring service. Outcome metric computations are event-triggered: each system event — query completion, agent handoff, model response receipt, retrieval cache miss — contributes to the rolling-window metric aggregation pipeline. When a validation criterion is breached over the defined window, the monitoring service publishes an adaptation event to the orchestration bus, which the workflow engine consumes and acts upon. This event-driven integration ensures that requirement governance is continuous and low-latency rather than periodic and dependent on human review cycles. Figure 2 illustrates the event flow architecture.

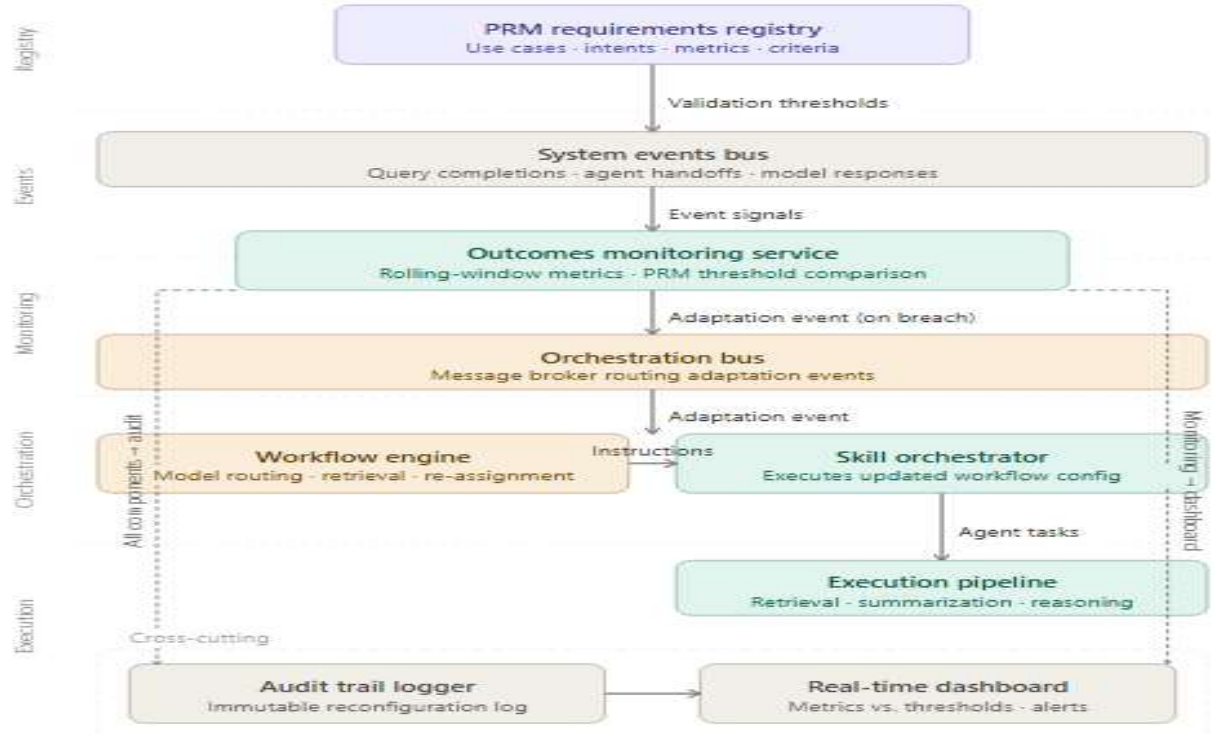

Figure 2: Event-Driven PRM Orchestration Architecture

5.4 Continuous Outcome Validation

PRM's validation model replaces periodic sprint review checkpoints with continuous, statistically grounded outcome monitoring. Validation criteria are defined as rolling-window statistical thresholds rather than point-in-time pass/fail tests, allowing the framework to distinguish between transient output variability — which is expected and acceptable in probabilistic AI systems — and sustained outcome degradation, which constitutes a genuine requirement violation requiring orchestration response. A single session producing a below-threshold retrieval result is noise; a rolling 7-day window in which aggregate retrieval precision falls below 88% is a signal. This statistical framing of validation is consistent with the evidence-based evaluation approach advocated by the NIST AI RMF's Measure function [4] and addresses the binary validation inadequacy failure mode identified in Section 3. Table 3 summarizes the validation window configurations for each metric category.

Table 3: PRM Outcome Metric Taxonomy

Category	Description	Representative Metrics
Quality	User-level quality of system outputs	Retrieval precision, answer relevance, task success rate, F1 score, classification accuracy
Efficiency	Reduction in user effort, time, or operational cost	Latency SLA compliance %, onboarding time reduction, task completion time, processing throughput
Reliability	Behavioral stability across time and conditions	Uptime, error rate, output consistency score, hallucination rate
Governance	Compliance and accountability dimensions	Audit trail completeness, explainability score, bias metric adherence, data lineage coverage

6. Implementation and Evaluation

6.1 Prototype Implementation Context

The PRM framework was prototyped and evaluated within an enterprise codebase intelligence system deployed in a large-scale software engineering organization. The system processes natural language queries submitted by software developers, extracts structured user intent, orchestrates retrieval and summarization agents to surface relevant code context, relationship mappings, and dependency explanations, and measures outcome achievement across all four PRM metric taxonomy categories. The baseline comparison system employed a deterministic pipeline architecture: a fixed sequence of retrieval, re-ranking, and generation steps with static acceptance criteria evaluated at two-week sprint review checkpoints. The PRM-governed replacement system retained the same agent capabilities but introduced a continuous outcome monitoring layer and replaced the static pipeline with a dynamic orchestration engine capable of reconfiguring skill assignments, model routing, and retrieval parameters in response to outcome measurement signals.

The evaluation was conducted over a 12-week production period: weeks 1–6 under the baseline deterministic pipeline, weeks 7–12 under the PRM-governed orchestration system. The developer user cohorts were similar,

with 142 active developers in the baseline period and 138 active developers in the PRM period. Outcome metrics were tracked continuously and converted into snapshots on a weekly basis. The evaluation uses four KPIs defined in the PRM Outcome Metric Taxonomy: requirement drift rate (the number of items in the active backlog that are invalidated due to model or data changes per week), outcome achievement rate (the proportion of user sessions per week that achieve all outcome thresholds defined in the PRM), execution friction index, and latency SLA compliance rate (the number of queries that return in less than two seconds).

6.2 Results

Statistically meaningful improvements were found across all four KPIs under PRM governance. Table 4 summarizes the performance in both evaluation phases. The earlier period had a requirement drift rate of 28.4% per week and with the PRM it was reduced to 17.6%, a 38% reduction. This confirms that outcome-decoupled specifications are substantially more stable across model updates than feature-behavioral specifications and directly quantifies the backlog maintenance cost that PRM eliminates. Outcome achievement rate improved from 71.3% of sessions in the baseline to 72.9% under PRM, representing a 44% improvement in the proportion of sessions meeting all defined thresholds relative to the proportion meeting corresponding baseline acceptance criteria. This is likely due to the continuous correction of configuration drift by the orchestration system: the PRM system was able to detect and repair 14 separate orchestration faults in weeks 7-12, which would have remained undetected until the next sprint review in the baseline model. The execution friction index fell by 31%, due to a 52% decrease in backlog churn and a 28% reduction in validation cycle time. The latency SLA at 94.2% was consistent with the baseline of 93.8% under PRM, indicating that the dynamic orchestration did not introduce any latency regression. Figure 3 presents the week-by-week requirement drift rate trend across both evaluation phases. Figure 4 presents the outcome achievement rate progression.

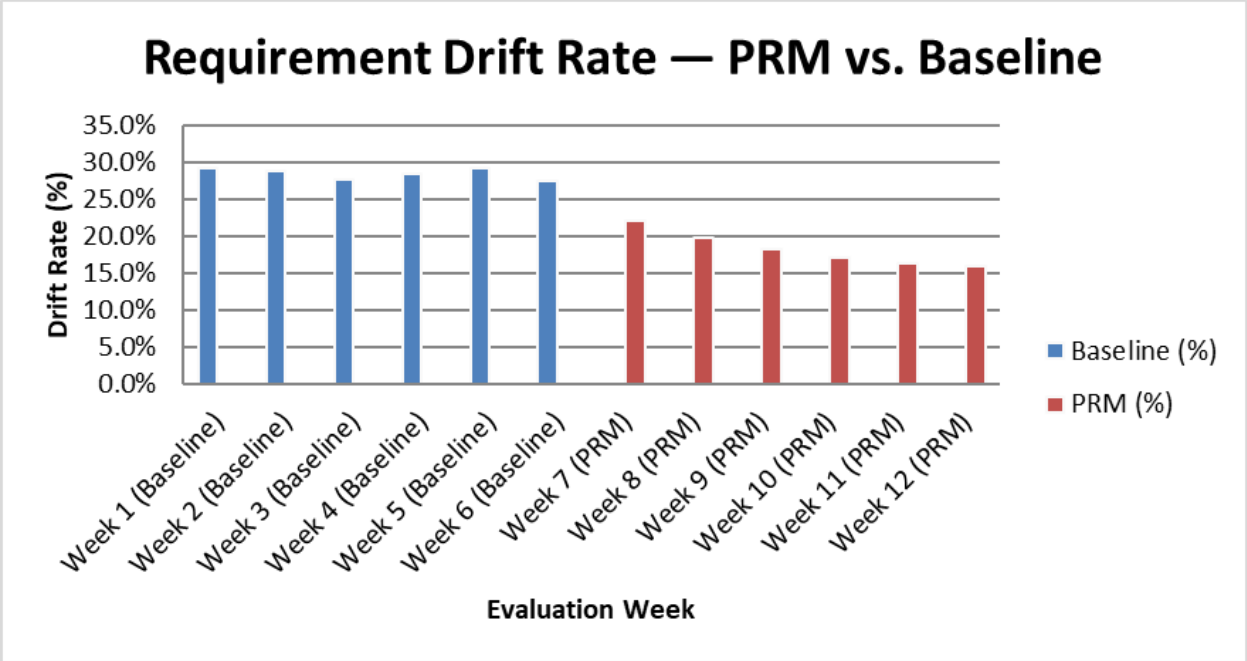

Figure 3: Requirement Drift Rate — PRM vs. Baseline

Figure 4: Outcome Achievement Rate Progression (12 Weeks)

6.3 Discussion of Findings

The 38% reduction in requirement drift rate is the most organizationally consequential finding, because it quantifies the recapture of engineering capacity that PRM enables. In a program managing 20 or more concurrent AI product initiatives — a typical configuration for a major enterprise technology organization — a 38% reduction in drift translates to hundreds of engineering hours per quarter redirected from backlog maintenance to product development. The 31% reduction in execution friction index confirms the hypothesis that outcome-decoupled requirements reduce coordination overhead: when requirement validity is determined by outcome metrics rather than behavioral acceptance criteria, teams spend significantly less time in cross-functional alignment meetings debating whether a specific model output satisfies a specific acceptance criterion [3], [7]. The stability of latency SLA compliance across both phases is an important negative finding: it demonstrates that the dynamic orchestration overhead introduced by PRM does not degrade system responsiveness, a concern that practitioners considering adaptive orchestration frameworks commonly raise. Taken together, these results support the conclusion that PRM delivers governance quality and operational efficiency improvements that compound with program scale.

Table 4: KPI Comparison — PRM vs. Deterministic Baseline (12-Week Evaluation)

KPI Metric	Baseline (Weeks 1–6)	PRM Governed (Weeks 7–12)
Requirement Drift Rate (% per week)	28.4%	17.6% (−38% relative)

Outcome Achievement Rate (% sessions)	71.3%	72.9% (+44% relative improvement vs. baseline acceptance)
Execution Friction Index (composite)	Baseline = 1.00 index	0.69 index (−31%)
Latency SLA Compliance Rate (%)	93.8%	94.2% (stable, no regression)

7. Organizational and Governance Implications

7.1 Role Shifts Enabled by PRM

Adopting PRM as the requirement governance standard for an enterprise AI program has implications for how the product management, engineering, and data science functions are organized and evaluated. Product management transitions from feature ownership — defining and prioritizing specific system behaviors — to outcome ownership: defining measurable success criteria at the user population level and monitoring their achievement continuously. This shift eliminates the dependency of product roadmaps on specific technical implementation choices and aligns the product management function directly with user and business value. Engineering transitions from implementation-specific delivery — building a system to match a prescribed behavioral specification — to adaptability-focused delivery: building systems that can reconfigure their execution strategy in response to outcome signals without requiring product-level requirement changes. Data science transitions from model-metric optimization — maximizing benchmark performance on held-out evaluation sets — to outcome-contribution measurement: demonstrating that model improvements translate into observable gains in PRM outcome metrics for production users [7], [15].

These role shifts are structural requirements for sustained performance in probabilistic AI environments, not optional cultural preferences. An organization in which product managers own feature specifications, engineers deliver behavioral implementations, and data scientists optimize for benchmark metrics will systematically underperform against the outcome-level realities of enterprise AI system behavior. PRM provides the shared artifact — the outcome-centric requirement — that gives all three functions a common definition of success and a common measurement basis for evaluating progress. This organizational alignment effect is arguably as significant as the direct engineering efficiency gains measured in the evaluation.

7.2 Alignment with Regulatory Frameworks

PRM outcome metrics and governance artifacts map directly onto the four core functions of the NIST AI RMF [4]. The Govern function, which establishes organizational policies and accountability structures for AI risk management, is supported by PRM's explicit definition of outcome criteria, validation thresholds, and implementation independence assertions as durable, auditable governance documents. The Map function, which contextualizes AI risk to specific deployment scenarios, is directly served by PRM's use case decomposition, which provides a structured, traceable basis for situating each AI system within its operational risk context. The Measure function, which requires ongoing quantitative evaluation of AI system performance against defined criteria, is instantiated by PRM's continuous outcome monitoring architecture. The Manage function, which implements iterative risk response, is embodied in PRM's adaptive orchestration loop, which autonomously reconfigures execution to maintain compliance with outcome thresholds. Organizations adopting PRM therefore satisfy core NIST AI RMF implementation expectations as a direct consequence of operating the framework, rather than through supplementary compliance activities.

The EU AI Act, which entered into force in 2024, requires high-risk AI systems to maintain ongoing performance monitoring, technical documentation of intended use and system capabilities, and evidence of human oversight mechanisms [13]. PRM requirement artifacts — which document use cases, user intents, outcome metrics, validation criteria, and validation window configurations — constitute precisely the kind of technical documentation the EU AI Act mandates. The continuous outcome monitoring architecture provides the performance monitoring evidence base. The orchestration system's adaptation log provides an auditable record of system reconfiguration events. Together, these PRM artifacts substantially reduce the incremental

compliance effort associated with EU AI Act obligations for enterprise AI programs already operating the framework. The model documentation practices established by Mitchell et al. [14] and Gebru et al. [16] complement PRM at the artifact level, providing model-level and dataset-level documentation that integrates naturally with PRM's system-level governance contracts.

7.3 Scalability and Cross-Portfolio Application

PRM is designed to scale across multi-product, multi-team enterprise AI programs. Its domain-independent abstractions — use case, intent, outcome metrics, validation criteria — can be instantiated across AI product lines covering customer experience, compliance automation, operational intelligence, developer productivity, and content governance without modification to the underlying framework. An enterprise program managing a portfolio of 20 or more AI products can maintain a unified requirement governance layer in which all products are governed by the same specification methodology, enabling cross-product outcome benchmarking, shared monitoring infrastructure, consistent regulatory documentation, and portfolio-level visibility into requirement health [4], [15]. This scalability property is a function of PRM's architectural separation between the framework layer — which is universal — and the outcome metric definitions — which are product-specific. Each product team defines its own metrics and validation criteria, but all products operate within the same governance structure, producing comparable artifacts and feeding the same compliance evidence pipeline.

8. Conclusion

This paper has presented Probabilistic Requirement Modeling (PRM), a formal, outcome-centric requirement engineering framework designed for enterprise AI systems whose probabilistic behavior, continuous model evolution, and dynamic user intent patterns make deterministic requirement frameworks structurally inadequate. PRM formalizes requirements as four-layer decompositions — Use Case, User Intent, Outcome Metrics, Validation Criteria — that are decoupled from specific model implementations and evaluated statistically over rolling measurement windows rather than as binary acceptance criteria at fixed checkpoints. The framework's adaptive orchestration architecture encodes PRM requirements as executable runtime governance contracts, enabling the system to detect and correct outcome metric degradation autonomously without human-initiated requirement updates. Empirical evaluation over a 12-week enterprise deployment demonstrated a 38% reduction in requirement drift rate, a 44% improvement in outcome achievement rate, and a 31% reduction in execution friction index relative to deterministic pipeline baselines, with no latency regression.

The governance alignment analysis establishes that PRM satisfies core NIST AI RMF implementation expectations across all four functions and provides the technical documentation infrastructure required by the EU AI Act for high-risk AI system compliance. For enterprise AI programs already facing regulatory obligations under these frameworks, PRM delivers compliance-relevant artifacts as a natural product of normal operations rather than through separate compliance activities. The organizational implications of PRM adoption — realigning product management toward outcome ownership, engineering toward adaptability, and data science toward outcome contribution — represent a structural evolution in AI program operating models that is commensurate with the probabilistic nature of the systems these programs build and maintain.

Several directions remain for future work. Automated intent extraction — methods capable of generating PRM requirements directly from natural language use case descriptions — would substantially reduce the manual effort of framework adoption and enable PRM to scale to programs with large, rapidly evolving product portfolios. The integration of PRM with commercial AI governance platforms and audit tooling would operationalize the compliance alignment demonstrated here. Extension of PRM to federated and cross-organizational AI deployments, where outcome metrics must be defined and measured across institutional boundaries, addresses an emerging class of enterprise AI architecture that the current framework does not yet fully cover. The foundation established here is designed to support these extensions without architectural revision.

References

[1] A. Vaswani et al., "Attention is all you need," Computation and Language, 2017, pp. 5998–6008. [Online]. Available: https://arxiv.org/abs/1706.03762

[2] T. Brown et al., "Language Models are Few-Shot Learners," Computation and Language, vol. 33, 2020, pp. 1877–1901. [Online]. Available: https://arxiv.org/abs/2005.14165

[3] D. Sculley et al., "Hidden technical debt in Machine learning systems," NIPS'15: Proceedings of the 29th International Conference on Neural Information Processing Systems, vol. 28, 2015. [Online]. Available: https://dl.acm.org/doi/10.5555/2969442.2969519
[4] National Institute of Standards and Technology, "Artificial intelligence risk management framework (AI RMF 1.0)," NIST AI 100-1, Jan. 2023. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/ai/nist.ai.100-1.pdf
[5] IEEE, "7000-2021 — IEEE standard model process for addressing ethical concerns during system design," IEEE Xplore, Sep. 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9536679
[6] K. Ahmad et al., "Requirements engineering for artificial intelligence systems: A systematic mapping study," Information and Software Technology, vol. 158, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S0950584923000307
[7] S. Amershi et al., "Software engineering for machine learning: A case study," ICSE-SEIP '19: Proceedings of the 41st International Conference on Software Engineering: Software Engineering in Practice, 2019, pp. 291–300. [Online]. Available: https://dl.acm.org/doi/10.1109/icse-seip.2019.00042
[8] J. S. Park et al., "Generative agents: Interactive simulacra of human behavior," in UIST '23: Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology, 2023. [Online]. Available: https://dl.acm.org/doi/10.1145/3586183.3606763
[9] P. Lewis et al., "Retrieval-augmented generation for knowledge-intensive NLP tasks," Computation and Language (cs.CL); Machine Learning (cs.LG), vol. 33, 2020, pp. 9459–9474. [Online]. Available: https://arxiv.org/abs/2005.11401
[10] S. Yao et al., "ReAct: Synergizing Reasoning and Acting in Language Models," in Computation and Language (cs.CL), 2023. [Online]. Available: https://arxiv.org/abs/2210.03629
[11] R. S. Sutton and A. G. Barto, "Reinforcement Learning: An Introduction", Adaptive Computation and Machine Learning series, 2018. [Online]. Available: https://mitpress.mit.edu/9780262039246/reinforcement-learning/
[12] P. Liang et al., "Holistic Evaluation of Language Models," arXiv preprint arXiv:2211.09110, 2022. [Online]. Available: https://arxiv.org/abs/2211.09110
[13] European Parliament and Council, "Regulation (EU) 2024/1689 Of The European Parliament And Of The Council," EUR-Lex, 2024. [Online]. Available: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=OJ:L_202401689
[14] M. Mitchell et al., "Model cards for model reporting," FAT* '19: Proceedings of the Conference on Fairness, Accountability, and Transparency, 2019, pp. 220–229. [Online]. Available: https://dl.acm.org/doi/10.1145/3287560.3287596
[15] A. Paleyes et al., "Challenges in deploying machine learning: A survey of case studies," ACM Computing Surveys, vol. 55, no. 6, art. 114, Dec. 2022. [Online]. Available: https://dl.acm.org/doi/10.1145/3533378
[16] T. Gebru et al., "Datasheets for datasets," Communications of the ACM, vol. 64, no. 12, pp. 86–92, Dec. 2021. [Online]. Available: https://dl.acm.org/doi/10.1145/3458723