10.48047/jocaaa.2026.35.03.41

# Prioritize Workflow Mapping Before Any Tech Integration in Cloud Platform Engineering

**Devinder Tokas**
Independent Researcher, USA

## Abstract

The engineering programs in cloud platforms always fail to deliver good results in cases when the organizations implement the tools and platforms prior to defining the workflows that these technologies should implement. Tool-first execution codifies process ambiguity into the production infrastructure and increases the breakdown of coordination, the lack of reliability, and the cost of operation at a distributed scale. Despite the growing adoption of cloud-native architectures, platform teams lack a structured sequencing method that connects process clarity to technology selection before infrastructure is committed. Workflow mapping addresses this gap, and the architectural foundation that technology selection requires is grounded in workflow mapping, which is founded on the Business Process Model and Notation, value stream mapping, TOGAF enterprise architecture sequencing, ITIL 4 service management value streams, and the NIST Big Data Reference Architecture role-based design. End-to-end flow, ownership boundaries, failure modes, handoffs, and data contracts are made clear by platform teams before any infrastructure commitment, leading to more reliability results and quantifiably reduced integration risk. The integration method involves a workflow-focused integration, outcome definition, mapping of the current state, quantification of friction, designing the future state, and iterative governance combined with these foundations to distribute big data lifecycle management and service reliability at scale. Practically, this method equips engineering teams with a repeatable sequencing playbook that reduces integration rework, accelerates delivery cycles, and improves operational resilience. Service-level goals and error budgets transfer workflow clarity into deployment results, whereas delivery performance metrics confirm whether platform investments delivered quantifiable throughput and stability benefits in fintech, e-commerce, media, and IoT settings.

**Keywords:** BPMN, Delivery Performance Metrics, Distributed Systems, Error Budgets, ITIL 4, NIST Big Data Reference Architecture, SLOs, TOGAF, Value Stream Mapping, Workflow Mapping

## 1. Introduction

### 1.1 The Operational Reality of Distributed Cloud Platforms

Cloud platforms are systems of distributed subsystems with compute, storage, messaging, and orchestration working in unison to define quality as perceived by the user. None of these components determines the result; it is the interaction of these subsystems in the actual conditions that results in behavior that the end users will experience. On a large scale, this interdependence manifests as repeating engineering issues that cannot be simply elucidated by a single test run: partitioned data that need to be coordinated across nodes, partial failures that silently spread through dependent services, tail latency that reduces user experience at the moment of peak load, and constant change that creates regression risk across strongly coupled interfaces. The requirements of distributed task scheduling in such circumstances are that priority, fault tolerance, and resource allocation be coordinated platform concerns as opposed to

10.48047/jocaaa.2026.35.03.41

per-service afterthoughts [1]. Those organizations that absorb these realities best handle platform reliability as an emergent quality of thoughtful design, rather than a quality to be adjusted post-launch.

## 1.2 Why Workflow Mapping Precedes Technology Selection

Cloud platforms are supposed to be fast, elastic, and operationally leveraged, but integration schemes often do not actually turn into as much of a value as they are claimed to be. The failure pattern is the most common tool-first implementation. The teams implement orchestration engines, event buses, workflow schedulers, or data platforms and expect to fit organizational workflows to the tools. In the case of ambiguous ownership of underlying processes, where there are hidden queues, unspoken data contracts, or a lack of exception paths, automation amplifies the ambiguity instead of eliminating it [2]. Minor defects in the process translate into the systemic reliability and cost failures since distributed architecture increases the latency, coupling, and operational load. Committing to a technology is the last step, and by doing so, they choose the appropriate infrastructure [3]. The benefits gained directly out of this sequencing are as follows: more intuitive interfaces, more dependable automation, and reduced coordination cost throughout the whole delivery lifecycle.

Existing contributions to distributed systems scheduling, workflow optimization, and cloud-native governance each address isolated aspects of platform integration, yet none provides a unified sequencing method that connects process clarity to architecture decisions before technology is selected [1]. Workflow scheduling frameworks, BPMN-based governance models, and workflow allocation strategies collectively advance individual dimensions of platform design but leave the critical sequencing gap between process modeling and infrastructure commitment unaddressed. The workflow-first method presented here fills that gap by providing a structured, artifact-driven sequence connecting demand-to-value flow mapping directly to architecture alignment, reliability governance, and measurable delivery outcomes across distributed big data and high-scale service environments [6].

10.48047/jocaaa.2026.35.03.41

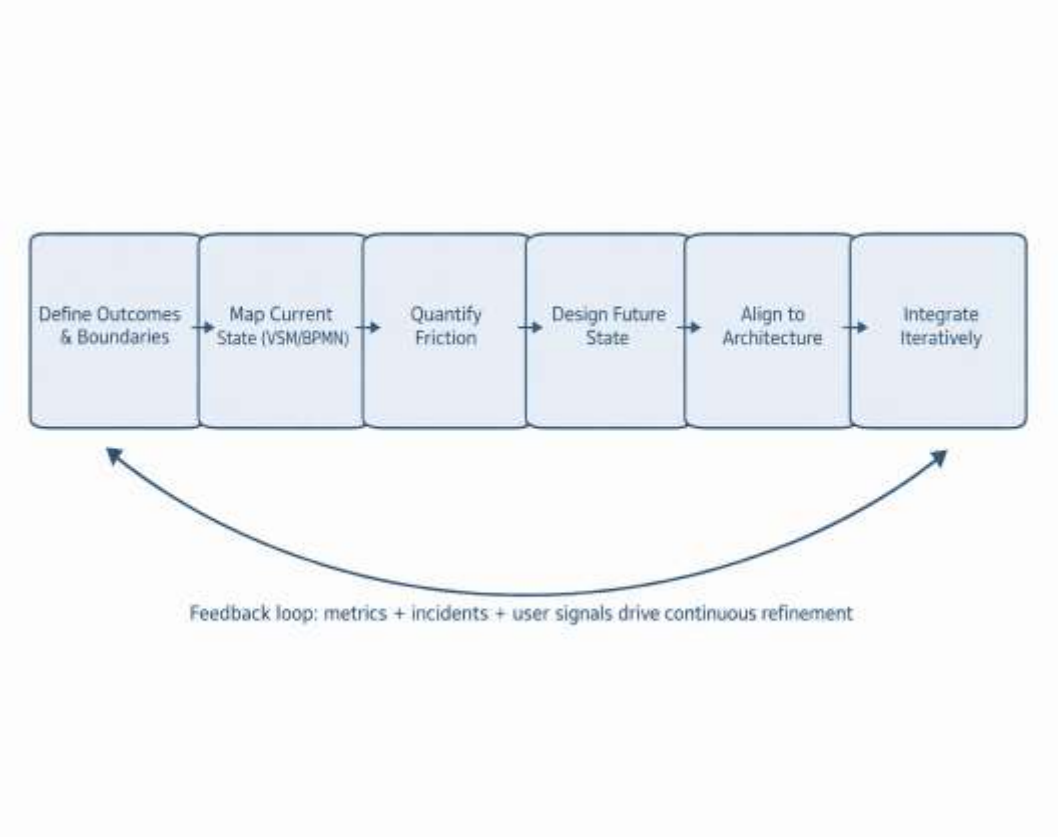


Figure 1. Workflow-first lifecycle: define → map → measure → redesign → align → integrate, with continuous feedback

## 2. Conceptual Foundations of Workflow Mapping

### 2.1 Value Stream Mapping and Bottleneck Visibility

VSM maps out all the steps and information flow necessary to provide a service, and the express purpose is to reveal waste. It is also contrasted with narrow process diagrams in that it highlights end-to-end flow so that delays, queues, rework loops, and batching are seen at the level of the system. A current-state map represents the actual operating conditions, and a future-state map represents the desired image of a better flow. Value stream mapping is used alongside BPMN in cloud integration programs, where BPMN gives the strict semantics of the processes, and value stream mapping gives insight into the system-wide throughput constraints and resource bottlenecks that have to be dealt with by workflow optimization [6].

### 2.2 Process Approach and Continual Improvement Cycles

Organizations produce results based on interconnected processes that work as a system, and this viewpoint goes well with risk-based thinking and Plan-Do-Check-Act improvement cycles. The process approach is used to steer the design of controls in cloud platform engineering to allocate risk proportionately: not all workflow transitions require a governance gate, yet high-risk steps, such as security approvals, schema evolution, and data retention, must have clear definitions [1]. This proportionality avoids over-engineering low-risk pathways and, at the same time, makes sure that consequential transitions have the safeguards in place so that their cost of failure is warranted.

10.48047/jocaaa.2026.35.03.41

### 2.3 Enterprise Architecture Sequencing Before Technology Commitment

Enterprise architecture models are ordered sequences of programs in terms of their business architecture to information systems and then technology architecture, and this sequence is important to platform programs. Technology should be capable and understandable in workflows; governance makes sure that implementations are in that architectural fashion. A workflow-first approach uses this sequencing at program and product levels: comprehend workflows and program ownership boundaries initially and implement platforms to support those workflows [4]. Companies that reverse this order often find out that organizational technology decisions are limiting instead of empowering the processes that their users require.

### 2.4 ITIL 4 Value Streams and Demand-to-Value Operationalization

ITIL 4 takes service management as an activity focused on co-creation of value and creates value streams in the form of particular combinations of activities adapted to respond to demand and create value. In the case of platform teams, value streams offer a governance-friendly framework between operational practices, such as incident management, change enablement, and problem management, and the workflows that engineers and users face in their daily lives [2]. This relation raises abstract service management principles into concrete workflow design inputs that shape how platform teams organize their operational duties.

### 2.5 NIST Big Data Reference Architecture and Role-Based Design

Distributed big data systems include specific roles that obtain, process, store, and serve data on a large scale, and a role-based reference architecture isolates these issues and defines the interfaces and cross-cutting fabrics that cross-cut across the entire system: management, security, and privacy. Workflow mapping helps fill the gap between these roles by answering the question of which actor initiates what activity, which data contracts rule interactions between actors, and what governance checks need to be run at what transition point [6]. This role-based transparency averts the proprietorship confusion that leads to the most expensive breakdowns in large-volume information settings.

| Standard | Platform Engineering Function |
|---|---|
| BPMN | Models participants, decision gateways, message flows, and exception paths across team boundaries |
| Value Stream Mapping | Exposes end-to-end delays, queues, rework loops, and batching at system level |
| NIST Big Data Reference Architecture | Clarifies actor roles, data contracts, and governance checks at each workflow transition point |

Table 1: Workflow Mapping Standards and Platform Engineering Functions [4, 6]

## 3. Why Workflow Mapping Matters in Distributed Systems

### 3.1 Socio-Technical Coupling and the Mirroring Effect

The nature of communication and coordination between organizations is what determines the system architecture and vice versa. Technical interfaces reflect the ownership ambiguity or handoff siloing of workflows, resulting in brittle coupling and slow feedback loops that cannot be fixed in retrospect regardless of how well tooled they are. A platform whose foundations are ambiguous team boundaries incorporates them as firm architectural constraints, and future restructuring becomes expensive and dangerous. Workflow mapping can be defined as a socio-technical design tool: it renders communication

10.48047/jocaaa.2026.35.03.41

channels visible, reveals latent coordination requirements, and provides an opportunity to intentionally match team tasks with service interfaces prior to their implementation [1]. This fact indicates that the quality of inner boundaries of a distributed system reflects the quality of organizational workflow design that has been adopted before the construction.

### 3.2 Failure Modes Originating in Process Gaps

In larger-scale systems, workflow breakdowns, not coding errors, are often the cause of the most expensive failures. Lack of defined paths of escalation, lack of clarity on ownership of on-call responsibilities, lack of consistency in the rollback decisions, and lack of testing of data reprocessing processes are all process gaps that result in longer outages, data corruption, or long duration of impact on customers during an incident [1]. In the event of validation failure, dependencies in the downstream timing out, a degraded latency deployment, or receiving late data out of sequence, the response of the system is entirely determined by the presence of the exception paths in the design or the latency. Workflow mapping minimizes such risks, as it forces the teams to define exception handling prior to integration to transform reactive incident improvisation into known, ownership-based response processes [3].

### 3.3 Automation Economics and Coordination Cost Reduction

Investments in cloud engineering are worth their cost when they minimize two dissimilar spending classes: coordination cost, that is, the time and energy needed to synchronize people and systems, and operational cost, that is, the time to find, identify, and rectify failure. The adoption of the tools before understanding the workflow escalates the cost of coordination instead of reducing it since teams use post-deployment cycles to work out interfaces, approvals, and incident responsibilities that ought to have been sorted during the design phase [2]. A workflow-first model preloads that coordination to the design phase, making its services easier to use and automation limits more reliable, as well as reducing operational overhead. Practically, this is the reason why the organizations that put the same level of engineering efforts into the workflow-first programs could have the same investment quickly repaid in terms of fewer incidents and shorter recovery periods [3].

## 4. Workflow-First Integration Method

### 4.1 Defining Outcomes and Boundaries

All workflow-first integrations start with the definition of what result the workflow is supposed to produce, as well as its nonfunctional requirements such as its latency, availability, correctness, and compliance. By identifying the best user journeys and how they explicitly define done, what is successful, and what is not, a resultant charter of the outcome becomes anchored, which informs all future design decisions [5]. This charter has quantifiable indicators and few hard requirements, like the retention period of regulations, personally identifiable information, and the geographic location of data. Teams that do not do this often find that they made incompatible technology decisions and only at the middle of the integration process overall find that the compliance requirements were not explicitly defined and forced them to perform expensive rework at the most inopportune time.

### 4.2 Mapping the Current State

A value stream map in a current state measures end-to-end lead time, queue time, and rework prior to any optimization. With such a system-level perspective in place, BPMN modeling of critical path participants, decision points, message flows, and exception handling can be done with the accuracy necessary for the decisions that architecture requires [4]. The science of this stage is moderation: the object is collective fact on how work flows today rather than a plan on how it will flow tomorrow. Companies that strive to

10.48047/jocaaa.2026.35.03.41

optimize when mapping the current state will always create models that are based on dreams and not reality, and these models will compromise all the downstream decisions taken that would rely on true baseline data [5].

### 4.3 Quantifying Friction and Operational Burden

Data are measured on every step of a workflow: cycle time, queue time, failure rate, and frequency of manual intervention. Platform programs also monitor operational load usage via paging events, average diagnosis time, and rates of manual rework reoccurrence [3]. The result of this step is a friction ledger, the fewest set of bottlenecks and failure modes that, should they be addressed, create the most end-to-end improvement.

### 4.4 Designing the Future State

Future-state value stream design eliminates handoffs, batching, and hidden queues that swell lead time and expand failure surface. The exception paths of rollback, replay, escalation, and quarantine are defined explicitly, according to named ownership, as opposed to informal conventions of a team [6]. In places where governance gates are needed, the ad hoc approval chains are replaced with lightweight criteria, ensuring that high-risk transitions are given due diligence without causing the bottlenecks that slow down delivery and create technical debt.

### 4.5 Mapping Workflow to Architecture Components

Each workflow stage is directly mapped into architecture tasks: what services, data stores, and control planes implement each workflow stage; who owns them; and what cross-cutting fabrics, such as management, security, and privacy, are consistently applied to the entire system [4]. Role-based reference architecture offers the structural vocabulary of this translation, and it eliminates accidental coupling since one concern is not retained, but instead, each concern is projected to an assigned architectural owner [3].

### 4.6 Iterative Integration with Governance and Feedback

Implementation is done in phases: canary pilot, rolling thunder, and standardization. Each transition is controlled by measurable gates related to the reliability targets and user outcome indicators so that degradations can trigger a rollback instead of silent continuation [5]. Workflow models receive incident learnings and metric regressions after each step; thus, the process and platform evolve together as the operational reality drifts off its initial design goal. This observation implies that teams with the longest-lasting platform stability consider workflow models as living artifacts and not a deliverable.

## 5. Workflow Mapping for Distributed Big Data Systems

### 5.1 Data Lifecycle and Operational Lifecycle as One Workflow

Distributed big data platforms incorporate two different lifecycle sequences that have to be integrated into a single flow of workflow mapping. Data lifecycle includes ingest, validate, transform, store, and serve; operational lifecycle includes observe, respond, recover, and improve. Simply mapping the data pipeline is not enough since failure management, replay mechanisms, backfill mechanisms, schema migration phases, and consumer reprocessing mechanisms constitute workflow transitions, and each has its own ownership and correctness criteria [6]. The synchronization of data producers, platform operators, and consumers on what causes each transition and what is right and timely at each stage minimizes systemic failures that occur when the roles have operational commitments that are tacit.

10.48047/jocaaa.2026.35.03.41

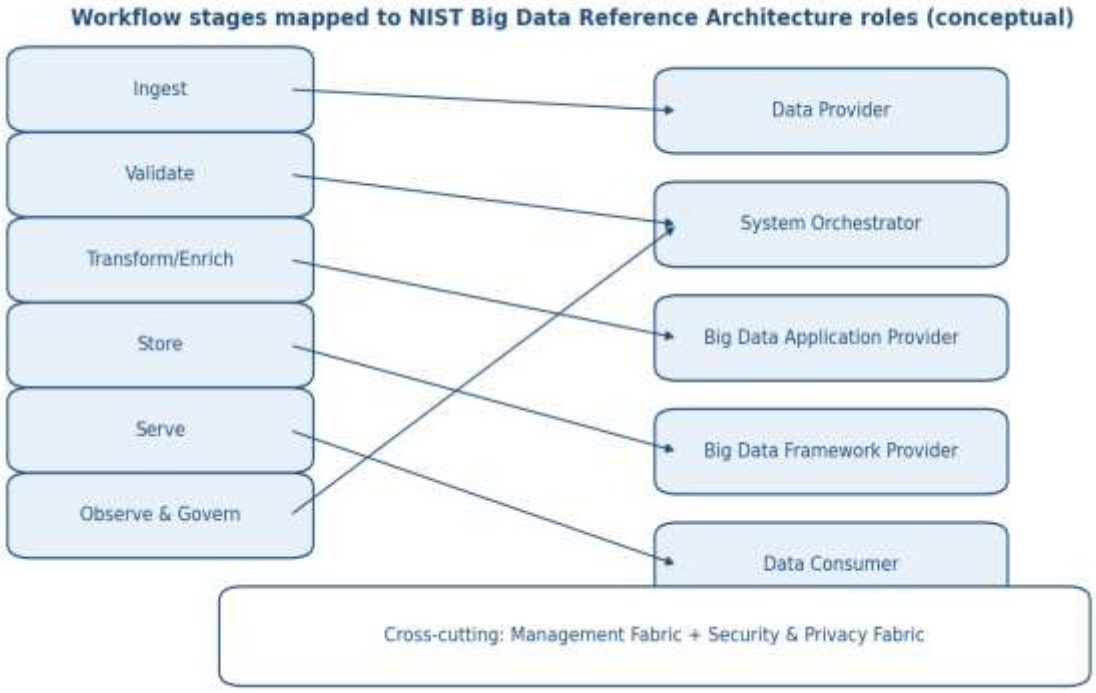


Figure 2. Conceptual mapping of workflow stages to role-based components in a big data reference architecture, including cross-cutting fabrics

### 5.2 Data Contracts and Schema Evolution as Explicit Workflow Steps

Examples of this high-risk workflow transitioning are the schema changes, versioning of the contract, and data quality verification, which require explicit entry and exit criteria and not informal coordination. A workflow-first model views data contracts as value stream elements: producers release published versioned outputs, consumer expectations are enforced through automated validation, and staged rollout rules rule adoption in services that they depend on [6]. The cost of operational failure of contracts in the late stages is always higher than the design investment to be made on contract governance as an explicit workflow step.

### 5.3 Governance, Security, and Privacy Fabric Placement

The data platforms must have stable governance controls, including access management, encryption policy, retention and deletion, lineage tracking, audit trails, and incident response. Workflow mapping does not add controls to the tooling after deployment, instead placing them at the right transition points: before ingest, before publishing, before serving, and when remediation of an incident occurs [8]. This position generates foreseeable compliance behavior and avoids the coordination friction associated with security reviews that take place in the final minute and block delivery without enhancing protection. Put simply, governance controls implemented during the design of the workflow need less enforcement effort in the long run than controls that are added to systems already in production.

### 5.4 Pipeline Observability for Freshness, Correctness, and Cost

Pipeline observability requires more than throughput metrics. Freshness indicators, correctness indicators, rates of failure in the validation part, rates of late arrival, volume of reprocessing, and cost drivers like redundant computation are all examples of workflow health indicators that provide feedback to future-state refinements and capacity planning [7]. Workflow-first design explicitly outlines what the pipeline

10.48047/jocaaa.2026.35.03.41

stages expect their telemetry to be in advance of instrumentation, avoiding the usual case where observability gaps are only realized at incidents, when the missing signals would be most needed. This observation implies that observable pipelines are not instrumented in a post facto manner but defined as workflow requirements at the beginning of the work.

## 6. Workflow Mapping for High-Scale Service Architectures

### 6.1 Service-Level Objectives and Error Budgets as Release Constraints

Service level goals convert the workflow clarity to operational commitments by stating the reliability goals as quantifiable levels at which the platform behavior is continuously checked. These thresholds are operationalized in error budgets as constraints on the number of releases: once the number of failures consumed cumulatively reaches the budget ceiling, then deployment activity is blocked until the reliability is restored, and then teams risk releasing degradations under the pressure of delivery schedules [9]. This mechanism transforms reliability governance from a reactive measure to an active delivery constraint as a part of the release workflow. SLO-defining platform teams represent an upfront commitment of infrastructure quality parameters covering latency, correctness, and availability, ensuring these architectural commitments are not aspirations after the system is deployed, and these goals are measurably more stable at scale [10].

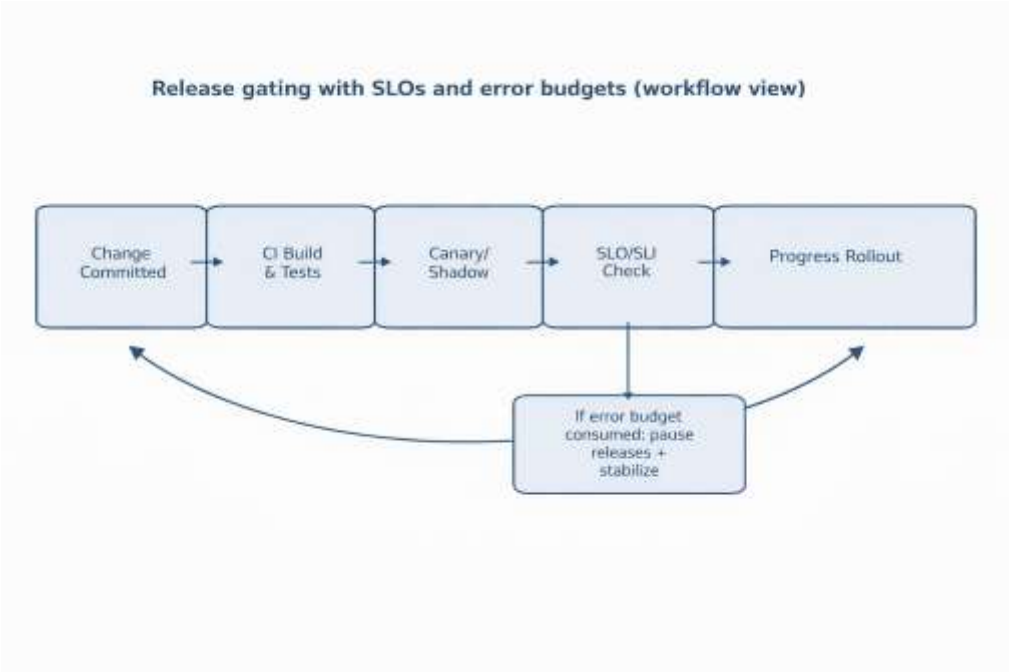


Figure 3. Example release workflow gated by SLO/SLI checks and an error-budget policy that pauses releases when risk exceeds tolerance.

### 6.2 Delivery Performance Metrics as Validation Signals

Platform investments in workflow redesign only justify their cost when outcome-oriented signals confirm that process changes produced measurable operational gains rather than surface-level adjustments. Deployment frequency, lead time for changes, change failure rate, and time to restore service each illuminate a distinct dimension of whether handoff reductions and batch size improvements translated into

10.48047/jocaaa.2026.35.03.41

tangible throughput and stability differences [9]. Reductions in batch size combined with clearly assigned ownership boundaries produce measurable contractions in lead time and recovery duration, outcomes that trace directly to the structural changes workflow mapping introduced rather than to external factors or incidental platform improvements. Persistent metric stagnation following redesign points to ownership ambiguities or unresolved coordination bottlenecks that the future-state map did not fully surface, directing subsequent refinement toward those specific workflow stages rather than broad architectural changes [10].

### 6.3 Incident, Change, and Problem Management Workflows

The operational disciplines that determine whether high-scale architectures remain stable under continuous change pressure draw as heavily from workflow design quality as from implementation quality. Incident response without role assignments, escalation criteria, and rollback decision thresholds defined in advance produces coordinated confusion during outages, extending customer impact well beyond the technical recovery window [9]. Releasing changes into production without staged validation patterns and defined governance criteria concentrates risk at the worst possible moment, when deployment pressure is highest and tolerance for failure is lowest. Postmortem findings that feed directly into workflow model updates and architecture corrections break the cycle of recurring failures at the same decision points, redirecting engineering capacity from repeated incident response toward the structural changes that prevent those incidents from materializing in the first place [10].

## 7. Illustrative Deployment Archetype: Real-Time Risk Scoring in Fintech

A real-time risk scoring platform takes streaming events, adds historical features to them, and delivers a decision under hard latency limits. Streaming infrastructure, feature stores, and decision services are tool-first integrated without prior workflow definition, resulting in a system that is functional in normal conditions but fails when schema changes happen, late-arriving data arrives, or when dependencies fail partially [6].

### 7.1 Current-State Mapping Findings

An existing analysis of the tool-first fintech implementations always reveals the same structural gaps. Schema transformations need coordination between multiple teams with no obvious approval gates, leading to monolithic deployments and a concentration of risk and blast radius of any individual failure [1]. Data quality incidents do not have a specific owner, creating inter-team discussions regarding whether this specific issue is a data problem or service problem as the customer impact continues to grow. Replay and backfill processes are also improvised in incident cases since there are no predefined runbooks, which is directly extending the duration of the outage. The decisions to release are independent of the reliability targets and create repeated regressions when the system is used in peak periods and the cost of degradation is at the highest point [3].

### 7.2 Future-State Workflow Design

A workflow-based redesign adds governance gates and explicit steps in every high-risk transition. Informal schema coordination is substituted by versioned data contracts, automated validation, and quarantine processes, and staged consumer adoption separates adoption risk in controlled increments [4]. The risk decision value stream ownership is centralized to a cross-functional team with specific interfaces to platform and security partners, eliminating the ownership ambiguity that creates incident coordination failures. The release workflow also includes canary-based deployment and SLO-gated rollback, where the latency or correctness degradations will start stabilizing instead of rolling out quietly [9].

10.48047/jocaaa.2026.35.03.41

### 7.3 Expected Outcome Signals

The redesign based on workflow-first delivers verifiable outcome indicators in all four dimensions of delivery performance. The diminished lead time is a result of the reduced batch sizes and transparent governance gates, which kill the delays in coordinating large batch deployments [10]. Reduced change failure rates are a direct result of staged rollout and automated contract validation of breaking changes before reaching full production. Swift incident recovery is due to replay and backfill workflows that are predefined with assigned ownership, and improvised processes are replaced with runbooks that can be executed. Better perceived reliability is reflected by SLO monitoring of latency and correctness, transforming reliability governance into an active delivery constraint rather than a reactive measurement [9].

| Delivery Dimension | Workflow-First Outcome |
|---|---|
| Lead Time for Changes | Reduced through smaller batch sizes and clearer governance gates eliminating coordination delays |
| Change Failure Rate | Lowered through staged rollout and automated contract validation before full production exposure |
| Incident Recovery Time | Shortened through predefined replay and backfill workflows with assigned ownership replacing improvised responses |
| Deployment Frequency | Improved through canary-based rollout and SLO-gated release workflows enabling more frequent stable deployments |

Table 2: Workflow-First Outcome Signals Across Delivery Performance Dimensions [9, 10]

## 8. Evaluation, Governance, and Operating Rhythm

### 8.1 Workflow Reviews as Architecture Reviews

Workflow models are first-class architecture artifacts, and they need to undergo the same level of disciplined review as system design documents. Quarterly reviews of current-state maps, or with each significant release, reveal drifts between planned and actual operations before those drifts become the technical debt in production infrastructure [5]. The dual role of incident postmortems in this operating rhythm is to have both confirmation that exception paths are working according to design and the identification of ownership boundaries within which repeated coordination failures are indicative of the need to revise workflow definitions. BPMN models that are continuously revised by this feedback loop do not become historical documents, which no longer reflect the systems they were initially documenting [4].

### 8.2 Lightweight Governance Gates

Governance acts as an imposition of results instead of bureaucracy, and this is the difference between governance boosting or hindering delivery. The oversight that consequential transitions seek is offered by a small number of hard gates covering security and privacy policies, regression thresholds to reliability, and data retention policies [8]. Evidence collection against these gates is automated, thereby eliminating the manual overhead coordination that converts the governance required into a bottleneck, and teams move through controlled transitions more rapidly than a team that has not and do not build unmanaged risk. All of the operations outside of these hard gates are managed by principles and metricizations, without affecting the autonomy of the teams but ensuring the audit trail that compliance and operational accountability demand.

10.48047/jocaaa.2026.35.03.41

### 8.3 Continuous Improvement Loop

Workflow-first integration is iterative in nature since constraints of the platform change with scale. New workflow transitions are arising as emerging requirements in the areas of cost optimization, geographic data residency, and partner integration are emerging [5]. In order to maintain the workflow structure and the architecture in line with the evolving operational requirements, a disciplined operating rhythm that incorporates metrics review, incident learning cycles, and periodic workflow redesign is essential in ensuring that neither the architecture nor the workflow drifts with time [9]. This observation indicates that a company where workflow mapping is an occasional program event has always faced the same failure in alignment that the original mapping was intended to avert, whereas those who practice a continuing improvement cadence have continually ensured the clarity of operations that permanent platform reliability requires [10].

## Conclusion

This article establishes workflow mapping as a foundational architectural discipline that precedes technology selection in cloud platform engineering, contributing a structured outcome sequencing method that connects process clarity to reliability governance and delivery performance outcomes. Distributed big data systems derive particular advantage in viewing data lifecycle and operational lifecycle as one unified workflow, with schema evolution and governance controls and pipeline observability having clear places as opposed to being considered after an afterthought. When service objectives operate like release gates and delivery performance indicators confirm that workflow enhancements are mirrored in tangible throughput and recovery improvements in fintech, e-commerce, media, and IoT deployments, high-scale service architectures become measurably stable. The fintech example shows that these principles yield measurable result indicators: shorter lead time, decreased change failure rates, quicker incident recovery, and increased user-perceived reliability. For industry practitioners, this method provides a directly deployable playbook: organizations adopting workflow-first sequencing reduce integration rework, lower coordination overhead, and achieve faster recovery cycles without requiring additional tooling investment beyond what existing platform programs already employ. Multi-cloud deployment environments and edge computing architectures present natural extensions for validating these principles, where coordination cost reductions can be quantified through standardized measurement frameworks applied at operational scale. AI-driven infrastructure automation introduces additional sequencing complexity that workflow-first methods have not yet been tested against, making it a productive direction for platform engineering practitioners to pursue.

## References

[1] Sheikh Umar Mushtaq, et al., "Enhanced priority-based task scheduling with integrated fault tolerance in distributed systems," Int. J. Cogn. Comput. Eng., ScienceDirect, vol. 6, pp. 152-169, Dec. 27, 2024. https://www.sciencedirect.com/science/article/pii/S2666307424000561

[2] Mustafa Ibrahim Khaleel, M. Safran, S. Alfarhood, and M. Zhu, "Workflow Scheduling Scheme for Optimized Reliability and End-to-End Delay Control in Cloud Computing Using AI-Based Modeling," Mathematics, MDPI, Oct. 18, 2023. https://www.mdpi.com/2227-7390/11/20/4334

[3] Faisal Ahmad et al., "Levelized Multiple Workflow Allocation Strategy Under Precedence Constraints With Task Merging in IaaS Cloud Environment," IEEE Access, IEEE Xplore, vol. 10, Aug. 29, 2022. https://ieeexplore.ieee.org/document/9869678

10.48047/jocaaa.2026.35.03.41

[4] Domenico Calcaterra and O. Tomarchio, "Policy-Based Holistic Application Management with BPMN and TOSCA," SN Comput. Sci., Springer Nature, vol. 4, no. 232, Feb. 23, 2023. https://link.springer.com/article/10.1007/s42979-022-01616-w

[5] Johannes Erbel and J. Grabowski, "Scientific workflow execution in the cloud using a dynamic runtime model," Softw. Syst. Model., Springer Nature, vol. 23, pp. 163-193, 2023. https://link.springer.com/article/10.1007/s10270-023-01112-6

[6] Srđan Daniel Simić, N. Tanković, and D. Etinger, "Big data BPMN workflow resource optimization in the cloud," Parallel Comput., ScienceDirect, vol. 117, Sep. 2023. https://www.sciencedirect.com/science/article/pii/S0167819123000315

[7] Francesca De Luzi, F. Leotta, A. Marrella, and M. Mecella, "On the Interplay Between Business Process Management and Internet-of-Things: A Systematic Literature Review," Bus. Inf. Syst. Eng., Springer Nature, vol. 67, pp. 265-288, Mar. 18, 2024. https://link.springer.com/article/10.1007/s12599-024-00859-6

[8] Nafiseh Soveizi, F. Turkmen, and D. Karastoyanova, "Security and privacy concerns in cloud-based scientific and business workflows: A systematic review," Future Gener. Comput. Syst., ScienceDirect, vol. 148, pp. 184-200, Jun. 16, 2023. https://www.sciencedirect.com/science/article/pii/S0167739X23001991

[9] Brennan Wilkes, A. M. P. Milani, and M. A. Storey, "A Framework for Automating the Measurement of DevOps Research and Assessment (DORA) Metrics," in Proc. 2023 IEEE Int. Conf. Softw. Maintenance Evol. (ICSME), IEEE Computer Society, pp. 62-72, 2023. https://www.computer.org/csdl/proceedings-article/icsme/2023/278300a062/1SN6qhmmIGk

[10] Ricardo Amaro, R. Pereira, and M. M. da Silva, "DevOps Metrics and KPIs: A Multivocal Literature Review," ACM Comput. Surv., ACM Digital Library, vol. 56, no. 9, Article no. 231, pp. 1-41, Apr. 25, 2024. https://dl.acm.org/doi/full/10.1145/3652508