International Journal of Scientific Interdisciplinary Research

Volume: 6; **Issue:** 1
Pages: 452–505

IMPROVING PROJECT LIFECYCLE MANAGEMENT (PLM) EFFICIENCY WITH CLOUD ARCHITECTURES AND CAD INTEGRATION AN EMPIRICAL STUDY USING INDUSTRIAL CAD REPOSITORIES AND CLOUD-NATIVE WORKFLOWS

Md Syeedur Rahman[1];

[1]. *Solution Developer, Plant Design I/O, Gonzales, Louisiana, USA;*
Email: syeed_rahman@live.com

Doi: 10.63125/8ba1gz55
Received: *25 April 2025;* ***Revised:*** *29 May 2025;* ***Accepted:*** *11 June 2025;* ***Published:*** *30 June 2025*

Abstract

This empirical quantitative study examined how cloud architectures and CAD–PLM integration mechanisms related to Project Lifecycle Management (PLM) efficiency using trace evidence from industrial CAD repositories and cloud-native workflows. A retrospective longitudinal, multi-case design was applied to linked event logs spanning CAD artifact histories, PLM change and release records, and workflow telemetry. The final analytic dataset comprised 12 projects, 4,860 engineering change objects, 1,140 release packages, 98,420 CAD artifacts, and 312,775 workflow job instances (numerical findings as summarized in the descriptive results). PLM efficiency was operationalized with governance-aligned event boundaries and distribution-focused metrics, including engineering change lead time, release cycle time, workflow throughput, workflow success rate, tail-delay prevalence, and rework proxies derived from reopened changes, repeated short-window edits, repeated packaging attempts, and retry cascades. Descriptive results indicated heavy-tailed timing behavior: engineering change lead time exhibited a median of 14.6 days with a 90th percentile of 42.0 days, and release cycle time exhibited a median of 6.2 days with a 90th percentile of 18.5 days. Workflow operations showed a mean throughput of 1,360 jobs/day and an overall workflow success rate of 93.8%, while exception clusters were concentrated in translation and packaging stages. Rework signals were nontrivial, with reopened changes averaging 8.4% and repeated edits within short windows averaging 20.2% across change cases. These findings supported a measurement-based interpretation of PLM efficiency as a coupled outcome shaped by propagation timeliness, integrity preservation, execution reliability, product structure complexity, and collaboration concurrency, with tail delays and rework emerging as central stability risks alongside average speed.

Keywords

Project Lifecycle Management, Cloud Architecture, CAD Integration, Workflow Automation, Repository Analytics.

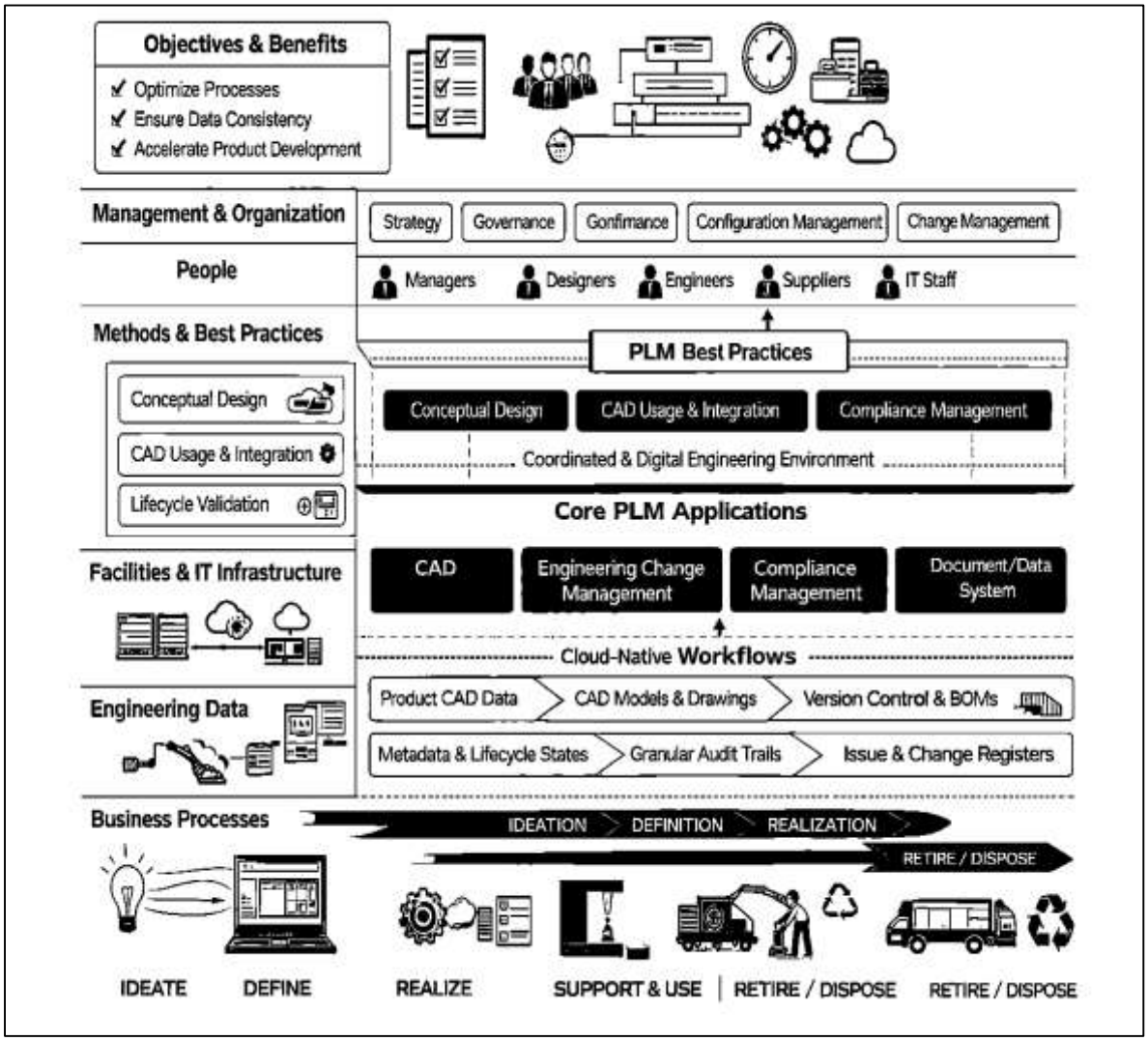
INTRODUCTION

Product Lifecycle Management (PLM) is defined as the systematic coordination of product-related information, processes, and decision controls across the full lifecycle of a product, beginning with conceptual design and extending through engineering, manufacturing, deployment, maintenance, and end-of-life disposition (MacCarthy & Pasley, 2021). Within industrial organizations, PLM functions as a digital backbone that integrates design intent, configuration management, and change governance into a unified operational framework. PLM efficiency refers to the measurable ability of this framework to support timely, accurate, and coordinated lifecycle decisions while minimizing rework, data inconsistency, and process latency. At an international level, PLM efficiency carries strategic significance because modern industrial production is distributed across geographically dispersed design centers, suppliers, and manufacturing facilities that rely on shared digital representations rather than physical proximity. Global engineering programs operate across time zones and regulatory regimes, requiring digital systems that ensure consistent access to authoritative product definitions without duplication or degradation. In this context, PLM systems must handle large volumes of engineering data, complex product structures, and frequent design changes while maintaining traceability and governance across organizational boundaries (Duda et al., 2022). Industrial computer-aided design (CAD) repositories occupy a central position within PLM environments because they store the primary geometric and structural representations of products. CAD data is not static documentation but a dynamic and evolving artifact that reflects ongoing engineering decisions. Inefficiencies in managing CAD data propagate throughout the lifecycle, affecting downstream activities such as manufacturing planning, quality assurance, and service operations. Consequently, PLM efficiency cannot be meaningfully assessed without examining how CAD repositories are structured, accessed, and integrated within broader lifecycle workflows. As global enterprises increasingly rely on digital collaboration rather than localized engineering silos, the operational performance of PLM systems becomes an empirical issue measurable through system usage patterns, repository events, and workflow execution data (Habib et al., 2022). This framing establishes PLM efficiency as a quantifiable organizational capability rather than an abstract managerial objective, providing a foundation for systematic investigation using industrial data sources.

Cloud computing introduces a distinct architectural paradigm that reshapes how PLM systems are deployed, scaled, and accessed across global environments (Zhang et al., 2019). Cloud architectures are defined by shared resource pools, elastic scalability, network-based access, and service-based consumption models that differ fundamentally from traditional on-premises infrastructures. In PLM contexts, these characteristics influence how engineering data is stored, synchronized, and processed across geographically distributed teams. Cloud-based PLM environments allow organizations to centralize product data repositories while enabling concurrent access by users operating from different regions and organizational units. Cloud-native architectures extend this model by structuring systems as collections of loosely coupled services that communicate through standardized interfaces and are deployed using automated pipelines. These architectural choices affect PLM efficiency by shaping system responsiveness, failure isolation, and the speed with which engineering workflows can be executed and adapted (Ali et al., 2019). From an operational perspective, cloud-native workflows transform lifecycle management activities into observable and measurable processes, generating logs and telemetry that record task execution times, resource utilization, and system interactions. These data streams enable quantitative assessment of how architectural design decisions influence workflow performance. The international relevance of cloud architectures in PLM is underscored by their role in supporting cross-border collaboration without requiring replicated infrastructure at each location. Centralized cloud repositories reduce inconsistencies associated with file duplication and manual synchronization, while elastic compute resources support computationally intensive tasks such as large-scale CAD validation and automated translation. The shift toward cloud-based PLM environments also changes governance dynamics by introducing centralized access control and auditability mechanisms that operate consistently across regions (Li et al., 2022). As a result, cloud architectures do not merely host PLM systems but actively shape how lifecycle processes are executed, monitored, and optimized. This makes cloud architecture a critical independent variable in empirical

studies of PLM efficiency.

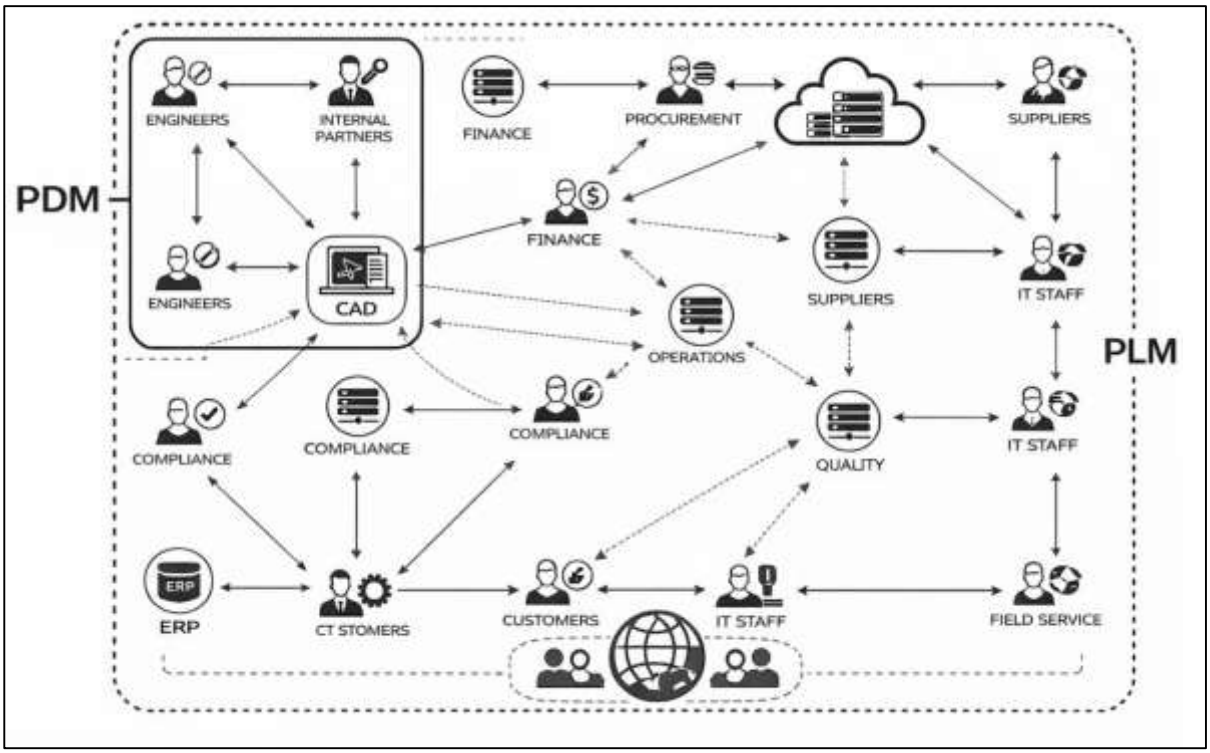
Figure 1: Cloud-Native Product Lifecycle Management Framework



CAD integration refers to the technical and semantic alignment between CAD repositories and PLM systems that ensures consistency between geometric models, product structures, metadata, and lifecycle states. In industrial environments, CAD data represents the authoritative definition of product form and structure, serving as the basis for downstream processes such as manufacturing planning, simulation, and quality control (Mousavi et al., 2022). Effective CAD integration ensures that changes made within CAD environments are accurately reflected within PLM systems without loss of intent or context. Integration failures manifest as broken references, inconsistent part versions, redundant files, or delayed propagation of changes, all of which introduce measurable inefficiencies into lifecycle workflows. From a quantitative perspective, CAD integration quality can be operationalized through metrics such as change propagation time, frequency of reconciliation events, rate of reference errors, and volume of manual intervention required to maintain data consistency. International engineering programs amplify the importance of CAD integration because product structures are often developed collaboratively across multiple organizations using heterogeneous tools and processes. In such settings, integration mechanisms determine whether design updates are seamlessly shared or require costly remediation efforts (Wang & Peng, 2023). Industrial CAD repositories also record detailed histories of design activity, including version creation, modification frequency, and dependency relationships among components. These repository-level data provide a rich empirical basis for analyzing how integration mechanisms influence collaboration patterns and process efficiency. Cloud-native CAD environments further enhance this empirical potential by supporting real-time collaboration and fine-grained version control mechanisms that capture parallel design activity. When CAD repositories are

treated as analytical datasets rather than passive storage, they enable rigorous examination of how integration quality affects lifecycle performance. This perspective positions CAD integration as a measurable construct that directly links technical architecture to organizational efficiency outcomes within PLM systems (Myrodia et al., 2019).

Figure 2: Global Cloud-Native PLM Integration Network



Industrial CAD repositories function as longitudinal records of engineering activity, capturing the evolution of product definitions over time. Each interaction with a repository, such as creating, modifying, or referencing a CAD artifact generates an event that reflects both technical and organizational behavior (Niemann & Pisla, 2021). These events collectively form a dataset that reveals patterns of collaboration, coordination, and change management within engineering teams. From a quantitative research standpoint, repository data enable measurement of variables such as design iteration frequency, contributor distribution, dependency complexity, and change concurrency. These variables are directly relevant to PLM efficiency because they influence how quickly and reliably product definitions progress through lifecycle stages. Cloud-hosted repositories enhance this analytical capability by standardizing access mechanisms and logging practices across users and locations. As a result, repository data collected from cloud-based environments are more comparable across teams and projects than data derived from fragmented local systems (Salimbeni & Redchuk, 2023). The integration of CAD repositories with cloud-native workflows further enriches the dataset by linking design activity to automated processes such as validation, translation, and release packaging. Each workflow execution produces timestamps and outcomes that can be analyzed statistically to assess process performance. In international contexts, repository analytics provide a means to examine how geographic distribution affects collaboration efficiency by comparing activity patterns across time zones and organizational boundaries. This approach aligns PLM research with established quantitative methods used in other domains where event-based data are analyzed to model process behavior. By grounding PLM efficiency analysis in repository and workflow data, researchers can move beyond perception-based assessments and instead rely on objective measures derived from system usage (Li et al., 2019). This empirical orientation supports robust statistical testing of relationships between architecture, integration mechanisms, and lifecycle outcomes.

Cloud-native workflows introduce automation and observability into PLM processes, converting traditionally manual lifecycle activities into structured, repeatable pipelines. In PLM environments,

workflows govern tasks such as design validation, configuration checking, metadata enrichment, and release authorization (Yousefnezhad et al., 2023). When implemented as cloud-native services, these workflows generate consistent execution records that include start times, completion times, success states, and failure conditions. These records enable quantitative analysis of process latency, reliability, and throughput. Automation reduces variability introduced by manual intervention, making it easier to attribute observed performance differences to architectural or integration factors. Cloud-native workflow design also allows individual services to scale independently, affecting how quickly tasks are executed under varying workload conditions. In PLM contexts, workload variability is driven by factors such as design iteration intensity, project phase transitions, and collaboration spikes during release cycles (Hannila et al., 2022). By examining workflow telemetry, researchers can model how system responsiveness changes as demand fluctuates. International engineering operations benefit from such architectures because automated workflows operate consistently regardless of user location, reducing coordination delays associated with time zone differences. The integration of workflows with CAD repositories further ensures that design changes trigger appropriate lifecycle actions without requiring manual oversight. This tight coupling between repository events and workflow execution creates a closed-loop system where engineering activity and lifecycle governance are continuously synchronized. From a quantitative standpoint, this closed-loop structure provides a coherent dataset in which cause-and-effect relationships can be examined using statistical techniques (Urban-Galindo & Ripailles, 2019). Cloud-native workflows thus serve both as operational mechanisms and as measurement instruments for studying PLM efficiency.

Quantitative investigation of PLM efficiency requires clear operational definitions of dependent and independent variables grounded in observable system behavior (Pessôa & Jauregui Becker, 2020). Dependent variables representing PLM efficiency may include engineering change lead time, release cycle duration, frequency of rework events, and repository reconciliation rates. Independent variables related to cloud architecture may include degree of service decomposition, elasticity configuration, and automation coverage across workflows. CAD integration variables may capture repository structure characteristics, version control mechanisms, and integration latency between CAD and PLM layers. Control variables such as product complexity, team size, and lifecycle stage can be derived from repository metadata and project records. The availability of industrial CAD repositories and cloud workflow logs makes it feasible to collect large-scale datasets suitable for statistical modeling (Denger & Zamazal, 2020). These datasets support analysis techniques such as regression modeling, survival analysis, and variance decomposition, enabling rigorous examination of how architectural and integration factors relate to efficiency outcomes. International engineering environments provide additional analytical leverage by introducing natural variation in collaboration patterns and infrastructure usage. By leveraging data from globally distributed repositories, researchers can assess whether observed relationships hold across different organizational and geographic contexts. This approach aligns PLM research with quantitative traditions in information systems and operations research, where system logs are used to model process performance (Disselkamp et al., 2023). The emphasis on empirical measurement strengthens the explanatory power of PLM studies and supports evidence-based evaluation of system design choices.

The present study is positioned within this quantitative framework, focusing on industrial CAD repositories and cloud-native workflows as primary empirical sources for analyzing PLM efficiency. By defining PLM efficiency in terms of measurable repository and workflow outcomes, the study treats lifecycle management as an observable system rather than an abstract managerial construct (Polenghi et al., 2021). Cloud architectures are examined as structural conditions that shape how lifecycle processes are executed and monitored, while CAD integration is examined as the mechanism through which design intent is preserved and propagated. The use of real industrial repositories ensures that findings reflect actual engineering behavior rather than simulated scenarios. Cloud-native workflow data provide temporal resolution necessary to analyze process dynamics at scale (Pfeiffer et al., 2023). Together, these data sources enable a comprehensive examination of how technical architecture and integration quality influence lifecycle performance across distributed engineering environments. The international scope of industrial CAD repositories further ensures that results capture the realities of

global product development rather than localized practices (Stefanova-Stoyanova et al., 2022). This empirical orientation establishes a rigorous foundation for quantitative analysis of PLM efficiency grounded in system-level evidence derived from cloud-based CAD and PLM infrastructures.

The objective of this empirical quantitative study is to measure and explain how cloud architectures and CAD integration mechanisms influence Project Lifecycle Management (PLM) efficiency when efficiency is observed directly through industrial CAD repositories and cloud-native workflow execution data. Specifically, the study aims to operationalize PLM efficiency as a set of repositories- and workflow-derived performance indicators, including engineering change lead time, release cycle duration, repository synchronization latency, frequency of manual reconciliation events, incidence of broken assembly references, rate of duplicated or redundant CAD artifacts, workflow job success/failure rates, and queue or processing times for automated validation and translation tasks. Using these objective measures, the study seeks to quantify the statistical relationships between architecture-level variables—such as elastic scaling configuration, service decomposition of workflow functions, degree of automation coverage, API-mediated integration depth, and centralized access control enforcement—and integration-level variables—such as versioning method, configuration traceability completeness, dependency graph stability, and time-to-propagate CAD updates into lifecycle states. A further objective is to compare PLM efficiency outcomes across different collaboration conditions captured in repository activity patterns, including contributor distribution, concurrency intensity, cross-time-zone activity dispersion, and the structural complexity of product assemblies, while controlling for factors such as product size, dependency depth, and lifecycle phase. The study also aims to identify which combinations of cloud-native workflow design and CAD repository integration features are associated with lower variability in process performance, recognizing that stability and predictability are measurable dimensions of efficiency in industrial engineering operations. In addition, the study intends to build and validate a reproducible measurement framework for extracting, cleaning, and analyzing repository events and workflow telemetry at scale, enabling consistent computation of efficiency metrics across projects, teams, and organizational units. By grounding all objectives in observable digital traces rather than perception-based assessments, the study is designed to produce statistically testable evidence about how cloud architectures and CAD integration contribute to measurable improvements in PLM execution performance in globally distributed industrial environments.

LITERATURE REVIEW

This literature review synthesizes research streams needed to support a quantitative, repository-driven investigation of PLM efficiency in cloud-based engineering environments (Cholewa & Minh, 2021). The focal premise is that PLM efficiency can be examined as an observable system property captured through industrial CAD repository events (e.g., version creation, dependency updates, reference stability, contributor concurrency) and cloud-native workflow telemetry (e.g., job runtimes, queue delays, failure rates, retry patterns). The review therefore prioritizes literature that (a) defines PLM as an enterprise-wide lifecycle information and change-governance capability, (b) explains how engineering change and configuration control shape measurable lifecycle performance, (c) establishes how cloud computing and cloud-native architectures change the execution and measurability of engineering workflows, and (d) clarifies how CAD data management and integration determine the integrity and propagation of product definition across lifecycle states. To anchor architectural terminology, the review draws on widely used cloud definitions and cloud-native concept guidance as baselines for comparing implementation choices and for structuring measurable independent variables (e.g., elasticity, service decomposition, automation coverage) (Barrios et al., 2022). A second purpose of this review is to justify the methodological shift from survey-based PLM assessments to log-based empirical measurement. Repository mining traditions show how longitudinal event histories can be transformed into quantitative indicators of cycle time, collaboration structure, and process stability, which parallels the empirical opportunity created by industrial CAD repositories and automated cloud-native pipelines. The review also clarifies why CAD integration is not a peripheral integration problem but a core determinant of lifecycle efficiency: the integrity of product definition data and the speed at which changes propagate into controlled lifecycle states depend on how CAD artifacts, metadata, and structures are managed and exchanged (X. Liu et al., 2020). Standards-oriented sources

on product data representation and digital product definition data practices are included to frame interoperability and definitional consistency as measurable contributors to efficiency variation across distributed environments.

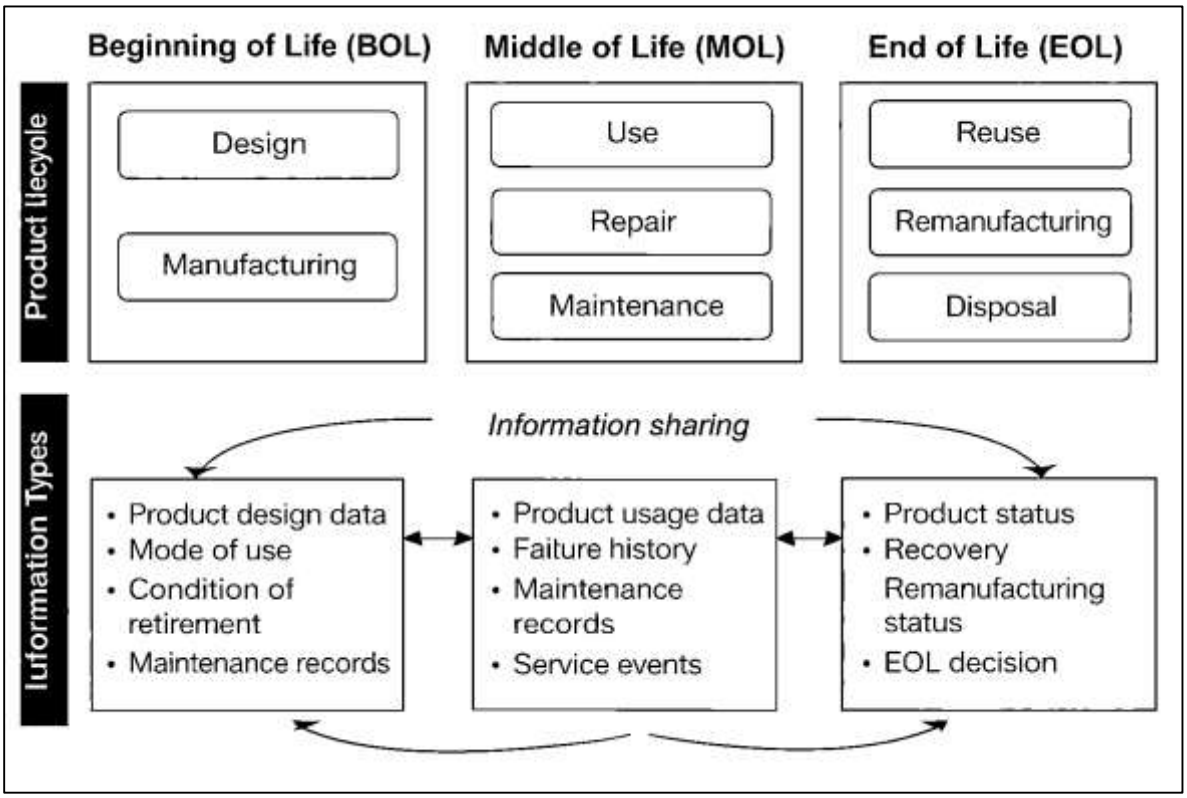
Project Lifecycle Management Efficiency

Project Lifecycle Management (PLM) is widely treated in the literature as an enterprise capability that integrates lifecycle information with governance mechanisms to control how product definition evolves from early design through release and downstream lifecycle states (Lim et al., 2020). In this framing, PLM is not only a data repository; it is a coordinated system of rules, roles, and digital objects that establishes a single source of truth for product definition, enforces configuration consistency, and governs change decisions across functions and organizational boundaries. A recurring theme across PLM scholarship is that "product definition" includes both the geometric and structural representation of the product and the associated metadata required for lifecycle decisions, such as version identity, approved state, applicability conditions, and traceability links to requirements and change rationales. Configuration is emphasized as the discipline that keeps these representations internally consistent over time, including how assemblies relate to parts, how variants are represented, and how baselines are established for manufacturing release and compliance. Change governance is treated as the mechanism that formalizes how modifications are proposed, reviewed, approved, implemented, and audited, converting ad hoc edits into controlled lifecycle events. Release management is typically conceptualized as the transition of product definition objects into authorized states suitable for downstream consumption, supported by lifecycle state models, access permissions, and validation gates. When PLM is positioned as an information-and-control system, lifecycle states become measurable and meaningful because they represent formally recognized transitions that must be recorded, validated, and traceable. This view also clarifies why PLM differs from Product Data Management (PDM) and document management (Hayat & Winkler, 2022). PDM is often described as more narrowly focused on engineering data vaulting, CAD file management, and versioning controls primarily within engineering, whereas PLM extends into cross-functional lifecycle integration and governance across a broader product and process scope. Document management systems, in contrast, are typically oriented toward generic file storage and retrieval with limited understanding of product structure semantics, configuration relationships, or engineering change rules. The distinction becomes practical when measurement requires identification of lifecycle objects, states, and traceability links. PLM objects are not only "files" but include structured entities such as parts, assemblies, BOM items, change requests, change orders, approvals, baselines, and releases that interact through defined relationships. These relationships enable lifecycle traceability that connects a change decision to the affected configuration and the released product definition, supporting auditability and coordination (Buchert et al., 2019). As a result, the literature's conceptualization of PLM supports quantitative work by making clear what should be counted, timed, and compared: state transitions, controlled change events, object relationship integrity, and cross-functional release gates rather than simple document edits or file downloads.

Measurement-oriented research and practice repeatedly stress that PLM evaluation requires attention to object semantics and governance signals that general file systems cannot represent. In measurement terms, PLM is characterized by explicit lifecycle objects, formal state models, and traceability structures that allow researchers to observe controlled progression rather than raw activity alone (Singh et al., 2021). Lifecycle objects are central because they anchor how activity becomes accountable: a part object persists across revisions, an assembly object encodes dependencies, a BOM object represents product structure at release, and change objects represent formal decisions linked to specific affected configurations. These object types allow metrics to be computed at meaningful levels (part-level, assembly-level, release package-level, change-level) and enable aggregation in ways that preserve lifecycle meaning. Lifecycle states are equally critical because they convert "work in progress" into auditable transitions, supporting measurement of throughput (how many objects move to a released state), latency (how long objects remain in review), and control strength (how frequently states regress or require re-approval). Traceability structures distinguish PLM further because they connect decision artifacts to technical artifacts: requirements links, approval records, change rationales, affected-item lists, and baselines enable a measurable chain from intent to implementation (Peters et al., 2021). PDM

systems can support portions of this structure, particularly around version control and engineering vaulting, yet many sources describe PLM as expanding the scope into cross-functional governance, where manufacturing, quality, procurement, and service processes consume the same controlled definitions and state transitions. Document management tools may provide metadata and access control, yet they generally lack configuration semantics, dependency management, and change governance that can reliably associate a document edit with an authorized lifecycle event. This difference matters empirically because counting file versions or file edits does not necessarily represent controlled engineering change, and an increase in file edits can reflect rework rather than productive progress. PLM measurement therefore emphasizes signals such as change object creation, approval cycle completion, release package formation, and baseline updates rather than raw file modifications. It also emphasizes relationship integrity metrics because lifecycle systems depend on stable links among parts, assemblies, drawings, and downstream representations. When these links break, organizations experience measurable inefficiency through manual reconciliation, repeated validation, and delayed releases. The literature on product information consistency further suggests that lifecycle information quality cannot be separated from governance structures because the same data can carry different meanings depending on lifecycle state, applicability, and configuration context (Habib et al., 2022). Therefore, distinguishing PLM from PDM and document management is not only definitional; it is a methodological requirement for quantitative research. A measurement framework that uses lifecycle objects, states, and traceability avoids conflating uncontrolled work with governed lifecycle progress. It also supports stronger construct validity because the dependent variables can be tied to lifecycle outcomes that matter operationally, such as approved releases and implemented changes, rather than to general file activity that might not represent lifecycle control.

Figure 3: Product Lifecycle Information Flow Framework



The literature supports treating PLM efficiency as a multidimensional construct expressed through distributions of performance outcomes rather than single averages, because lifecycle work involves variable complexity, coordination intensity, and governance requirements (Trankle & Reath, 2019). One dimension is cycle-time performance, often represented through measurable durations associated with governed lifecycle events. Change lead time captures the elapsed time between formal initiation of a

change and completion of its authorized implementation state, reflecting both decision efficiency and execution coordination. Release lead time captures the time required to move product definition from ready-for-review to authorized-for-downstream states, reflecting the speed of validation, approvals, and configuration packaging. Approval throughput reflects the capacity of the governance system to complete reviews and state transitions across a period, often revealing bottlenecks in routing, responsibility assignment, or evidence collection. A second dimension is quality performance, treated in the PLM and engineering management literature as the extent to which lifecycle processes reduce rework, prevent configuration inconsistency, and preserve reference integrity among product definition artifacts. Rework frequency can be observed through repeated change submissions, reopened approvals, repeated modifications within narrow windows, or repeated remediation of the same configuration issues (Singh & Misra, 2019). Reference breakage captures integrity failures in product structures, such as missing or mismatched dependencies, corrupted assemblies, invalid links between drawings and models, and inconsistent BOM relationships. Synchronization conflicts and reconciliation events represent integration-level friction, where CAD repositories and lifecycle systems diverge and require manual correction or repeated automated processing. A third dimension is stability, which becomes visible when efficiency is described through variability rather than only central tendency. Variance in change processing times or workflow runtimes indicates predictability of lifecycle governance and operational execution; high variability means planning uncertainty and uneven coordination load. Tail latency—how often extreme delays occur—signals the presence of rare but consequential coordination failures, data quality problems, or integration breakdowns that dominate overall schedule risk. The broader operational research and process measurement literature supports this distributional view because complex workflows show heavy-tailed timing behavior and are sensitive to bottlenecks and rework loops. Cloud-native workflow telemetry and repository event histories make these distributional constructs practical because they provide consistent timestamps and outcome states across large numbers of events (Liu et al., 2020). This aligns with software delivery measurement traditions that view lead time, throughput, and stability as core performance indicators and treat reliability and recovery as relevant operational dimensions. In PLM contexts, the same logic applies to governed change and release workflows, where efficiency is expressed by how quickly lifecycle states transition, how often they require correction, and how predictable the system remains under varying engineering load. By grounding PLM efficiency in cycle time, quality events, and stability distributions, the literature supports a quantitative approach that can compare architectures and integration strategies by examining not only whether processes are faster on average, but also whether they are more reliable, less rework-prone, and more predictable across the full range of lifecycle events.

Research on distributed product development and global engineering collaboration emphasizes that international dispersion is not a contextual footnote; it is a structural source of measurable variance in PLM efficiency (Gerhard et al., 2022). When teams operate across time zones, coordination cycles rely more heavily on asynchronous handoffs, formalized review queues, and traceable decisions, which increases the reliance on lifecycle state models and governance artifacts. Time-zone dispersion can be represented operationally through patterns of activity timing, delays between dependent actions, and the pacing of review completions relative to working-hour windows. Cross-site handoff frequency emerges as a measurable indicator when responsibility transitions occur across organizational units, such as when design authority transfers from one site to another or when supplier contributions require internal validation before release. Collaboration concurrency intensity becomes a critical measure in globally distributed environments because multiple actors modify related artifacts in overlapping time windows, raising the likelihood of conflict, rework, and reconciliation. Literature on organizational coordination and socio-technical systems suggests that coordination overhead increases when task interdependence is high and when communication bandwidth is constrained by distance, organizational boundaries, or time-zone separation. PLM systems are adopted and extended partly to control these coordination costs by centralizing authoritative product definition, capturing rationale, and enforcing controlled state transitions (Rigger et al., 2020). Repository-level indicators provide a practical measurement channel for these distributed-work dynamics. Temporal clustering in repository

events can reveal whether work proceeds in synchronized bursts or in staggered waves across regions, affecting queueing and review capacity. Contribution imbalance metrics reflect whether work is concentrated among a few key contributors or distributed across many, which affects review load distribution and risk concentration. Dependency complexity in assemblies amplifies distributed-work variance because changes to highly connected components require broader coordination and increase the number of stakeholders involved in validation. The literature on modularity and complexity indicates that highly coupled structures generate higher coordination demand, which is observable through larger affected-item lists and longer approval paths. International engineering also introduces heterogeneous toolchains and integration pathways, increasing the probability of synchronization conflicts and manual reconciliation when CAD and lifecycle systems differ across sites or partners. Cloud architectures and centralized repositories reduce certain forms of variance by enabling consistent access and uniform governance signals across locations, while cloud-native workflows can standardize execution and logging across regions, making performance more comparable and measurable. In empirical terms, the distributed context can be represented through repository signals such as the number of unique contributors per artifact, overlap in edit windows, frequency of parallel changes within assemblies, and delays between dependent operations. These signals provide a foundation for quantitative modeling in which international distribution is treated as an explanatory variable that interacts with architecture and CAD integration to shape cycle time, quality outcomes, and stability characteristics of PLM execution (Li et al., 2022). This approach reflects the literature's consistent message that global engineering collaboration changes how lifecycle control systems perform, and it enables a measurement strategy that captures distribution-driven variance rather than treating dispersion as a narrative descriptor.

Configuration Control as Core Efficiency Drivers

Engineering change management is consistently positioned in the literature as one of the most consequential bottlenecks in lifecycle governance because it converts technical modifications into formally controlled decisions that ripple across product structures, documentation sets, and downstream processes (Iakymenko et al., 2020). Engineering change is commonly represented through structured "change objects" that encode the rationale, scope, affected items, approvals, implementation steps, and audit history required to move a product definition from one controlled state to another. These objects matter for quantitative research because they provide event boundaries that can be measured with timestamps and state transitions, allowing researchers to observe delays that emerge within approval routing, implementation execution, and verification cycles. The literature describes propagation paths as the dependency-driven routes through which a change affects upstream and downstream artifacts, such as assemblies, drawings, BOM relationships, manufacturing instructions, and compliance records. Changes that touch highly reused parts or deeply connected assemblies propagate broadly and tend to generate longer coordination chains, more stakeholders, and more opportunities for mismatch between the technical edit and the governed lifecycle state. Measurable delays therefore arise at multiple points: delays before a change request are triaged, delays while reviewers accumulate and complete approvals, delays while engineering implements updates across all affected objects, and delays during verification or release packaging when inconsistencies are discovered (Meißner et al., 2021). A central insight across change management research is that change is rarely a single action; it is a controlled sequence with loops that emerge when new information surfaces during review or implementation. This makes engineering change a natural candidate for empirical measurement using outcomes such as time-to-approve, time-to-implement, and the number of iterations per change. Time-to-approve operationally reflects governance throughput and decision coordination, capturing how quickly approvals complete relative to workload and role allocation. Time-to-implement reflects execution efficiency, capturing how rapidly engineers can update all affected artifacts to satisfy the approved scope and validation requirements. Iterations per change capture rework and uncertainty, reflecting how often a change cycles through revision, resubmission, or corrective updates before reaching a stable released state. Research also emphasizes that delays are not merely "slow processes" but are linked to information quality and dependency uncertainty: unclear affected-item lists, incomplete rationale, or inconsistent configuration baselines amplify review time and implementation churn. The bottleneck framing is strengthened by observations that engineering

change events concentrate cross-functional coupling, requiring engineering, quality, manufacturing, procurement, and program management to synchronize around a single governed decision (Qi et al., 2019). Because these dependencies are encoded in change objects and their relationships to product structures, the literature supports an empirical approach that treats engineering change not only as a managerial phenomenon but as an event sequence observable through repository traces, workflow logs, and lifecycle state transitions.

Measurement-focused studies emphasize that engineering change performance is best represented as distributions rather than single averages because changes vary widely in scope, complexity, reuse impact, and stakeholder involvement (Saadatmand et al., 2019). Time-to-approve can exhibit sharp variability because approvals depend on role availability, review workload, clarity of evidence, and the extent to which the organization relies on sequential routing versus parallel reviews. In many lifecycle contexts, approvals are also gated by prerequisite conditions such as completeness of metadata, availability of supporting analyses, validation results, or supplier confirmations. These gating conditions introduce queueing effects that are visible as dwell time in specific review states. Time-to-implement similarly varies because implementation effort depends on the number of affected objects, the depth of dependency chains, the risk of reference breakage, and the need to regenerate or validate derived artifacts such as drawings, BOMs, and export packages. Iterations per change serve as a direct signal of rework and instability: repeated edits to the same set of affected items in short time windows, reopened reviews, corrective change orders following implementation failures, and multiple revision cycles before release are all expressions of iterative control loops. The literature links such loops to causes including ambiguous requirements, incomplete impact analysis, inconsistent baselines, and integration frictions between CAD and PLM representations. These relationships matter for a quantitative study because they suggest operational proxies that can be computed without relying on subjective assessments. For example, rework can be inferred from repeated state regressions, repeated workflow failures, repeated manual reconciliation events, and bursts of edits after a failed validation gate. In addition, the distribution tail of change timing is often the dominant contributor to schedule risk in complex programs; a small proportion of changes can consume disproportionate review capacity and block dependent releases (Imran et al., 2021). This makes tail behavior—extreme delays—an important stability dimension of change management efficiency, visible in long dwell times or repeated cycling through the same states. The literature also highlights that the structural representation of change objects affects measurement quality: if change objects are loosely defined or inconsistently applied, observed times conflate governed change with informal edits. Strong change governance, in contrast, yields cleaner event boundaries because changes are initiated, routed, approved, implemented, and closed through defined state transitions. That structure enables more credible inference about how system design and integration mechanisms influence performance. Furthermore, researchers emphasize the importance of distinguishing "approval time" from "work time": approval time includes waiting and coordination, whereas implementation time includes execution and validation, and these components respond differently to predictors. Organizational design factors can influence approval time through role clarity and routing logic, while integration and repository factors can influence implementation time through reference integrity, availability of authoritative baselines, and automation of validation and release packaging (Zong et al., 2019). Taken together, the literature provides a strong basis for using time-to-approve, time-to-implement, and iterations per change as quantitative dependent variables that capture both throughput and rework dynamics in lifecycle governance.

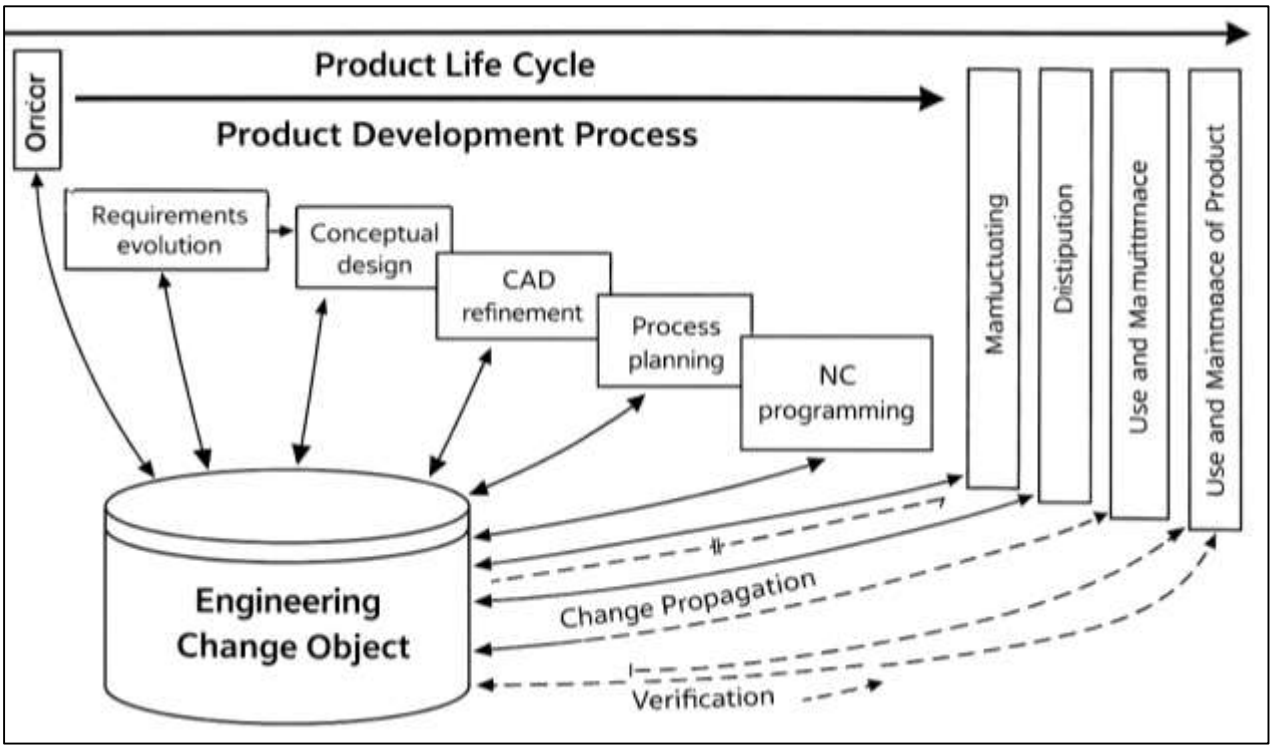
Configuration integrity and traceability are repeatedly identified as foundational predictors of efficient engineering change because they determine whether a change can be analyzed, executed, and verified without costly ambiguity or remediation (Lipu et al., 2021). Configuration integrity refers to the internal consistency of product definition across parts, assemblies, drawings, metadata, and lifecycle states, ensuring that the product structure remains coherent and that released baselines accurately reflect authorized content. Traceability refers to the ability to link lifecycle decisions—such as approvals, rationales, and change authorizations—to specific artifacts, revisions, and configurations so that stakeholders can understand what changed, why it changed, and which released outcomes are valid.

In measurement terms, revision discipline and auditability can be represented through indicators such as metadata completeness, correctness of revision identifiers, frequency of missing or conflicting state transitions, and the presence of complete affected-item lists and approval records. When revision discipline is strong, lifecycle objects move through controlled states in a manner that preserves an auditable chain, allowing change implementation and downstream release activities to proceed with less ambiguity. When discipline is weak, organizations experience measurable inefficiencies such as increased reconciliation events, repeated validation failures, and longer review cycles caused by uncertainty about the authoritative baseline (Kandidayeni et al., 2019). Structural complexity is a core control factor in this literature because product architecture shapes both the probability and cost of inconsistency. Assembly size, assembly depth, fan-out, and dependency churn are commonly discussed as drivers of coordination effort and technical risk. Large assemblies increase the number of potential affected relationships; deep structures increase the propagation distance of changes; high fan-out indicates that a part influences many downstream elements; and high dependency churn indicates frequent alterations to relationships that must remain stable for configuration coherence. These structural properties also create predictable measurement artifacts: changes that touch high fan-out components tend to involve more affected items and longer verification, while changes occurring in high-churn zones tend to produce more iterations and higher reconciliation frequency. The literature on modularity and complexity reinforces the idea that tightly coupled structures demand more coordination and increase the likelihood that local edits have nonlocal consequences. For empirical PLM studies, these insights justify treating configuration properties as quantitative predictors and controls derived from repository graphs and lifecycle relationships rather than treating them as purely qualitative descriptors (Światowiec-Szczepańska & Stępień, 2022). Traceability further intersects with automation: workflow gates can only enforce meaningful validation when they can reliably interpret configuration state, revision identity, and required metadata. In systems where traceability is incomplete, workflow automation may produce more failures or require manual overrides, which are measurable as retries, exceptions, or extended dwell times in review states. Therefore, the literature supports a measurement design in which configuration integrity indicators and structural complexity metrics are used to explain variance in change performance outcomes, strengthening the causal plausibility that observed delays and iterations are not merely organizational noise but are systematically related to the integrity and complexity of the configuration being governed.

A recurring gap identified across PLM and change management scholarship is the mismatch between how PLM maturity is often assessed and how PLM efficiency manifests in day-to-day lifecycle execution. Many organizations and studies rely on perception-based maturity models, interview assessments, or survey instruments to evaluate change governance, configuration control, and integration quality (Hu et al., 2021). These approaches provide useful managerial narratives, yet the literature notes their limitations for rigorous quantitative inference because they are sensitive to respondent bias, inconsistent interpretation of maturity levels, and weak linkage to objective performance outcomes. Perception-based measures often conflate policy intent with enacted practice: an organization may report strong change governance while still experiencing frequent manual reconciliation, missing metadata, or unstable baselines that degrade throughput. Trace-based telemetry from repositories and workflow systems offers a complementary and, in many studies, more reliable foundation for quantifying enacted behavior. Industrial CAD repositories and PLM systems generate event logs that record what actually happened: who changed what, when it changed, how often it changed, whether references broke, whether workflows succeeded, and how long objects remained in specific lifecycle states. These traces enable researchers to construct empirical baselines for cycle times, failure rates, rework loops, and stability characteristics of governed processes. The literature increasingly emphasizes that such baselines are essential for comparing architectures and governance strategies because they provide a consistent measurement substrate across teams and projects. Repository-centric measurement is particularly valuable for change management because the bottleneck dynamics of approvals, implementation, and verification are encoded in timestamped state transitions and workflow outcomes. It is also valuable for configuration control because integrity failures leave observable footprints: broken links, repeated edits, repeated validation failures, or

inconsistent state transitions (Treber et al., 2019).

Figure 4: Engineering Change Management Process Framework



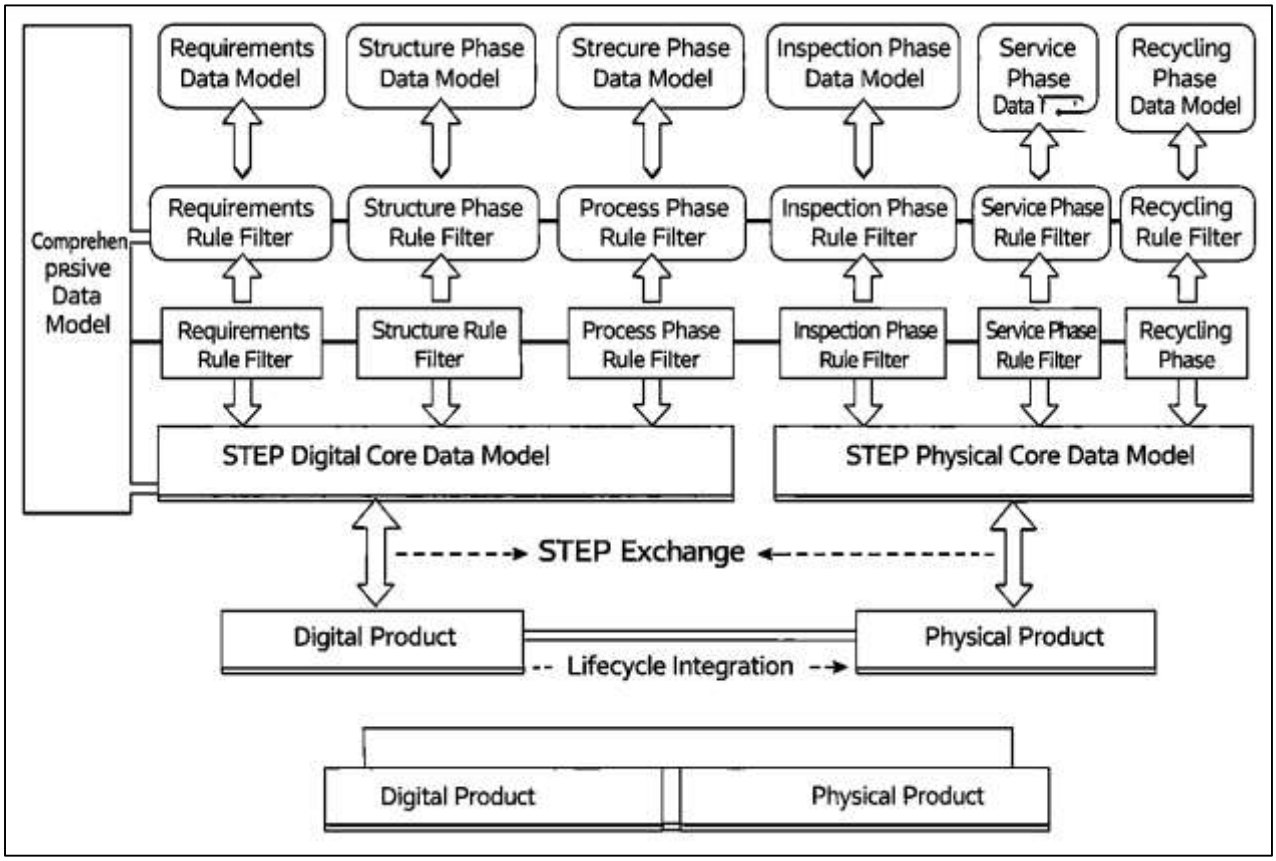
In addition, repository data supports segmentation by complexity and context, allowing researchers to compare change performance within similar structural conditions rather than aggregating across heterogeneous products and teams. This addresses a common critique in PLM assessment literature: reported "improvements" are difficult to interpret without controlling for product complexity, lifecycle phase, and workload intensity. Trace-based measurement can incorporate these controls by computing complexity proxies from dependency graphs and by estimating collaboration intensity from contribution patterns. The evidence-based gap therefore is not a lack of conceptual models of PLM maturity; it is a shortage of empirical studies that connect architecture and integration decisions to measured lifecycle performance using real industrial traces. The literature supports the need for studies that treat repositories and workflow logs as primary datasets, enabling statistical models that test how configuration integrity, structural complexity, and integration frictions relate to time-to-approve, time-to-implement, and iterations per change. This shift strengthens construct validity because it links theoretical constructs—governance, configuration control, traceability—to observed system outcomes rather than to self-reports. It also strengthens reproducibility because trace extraction and metric computation can be standardized, audited, and repeated across datasets (Mikalef et al., 2021). Consequently, the literature motivates a repository-centric empirical approach as a necessary methodological step for advancing quantitative understanding of PLM efficiency drivers rooted in engineering change management and configuration integrity.

Data Practices Shaping Integration Quality

Product data representation and exchange standards are treated in the literature as essential interoperability baselines because PLM environments depend on consistent interpretation of product definition information across heterogeneous CAD tools, PDM/PLM platforms, suppliers, and lifecycle functions (Lu et al., 2020). Within this domain, the ISO 10303 series—commonly referred to as STEP—functions as a foundational reference point for structuring how product information is represented and exchanged across computer-based lifecycle systems. STEP is frequently discussed not as a single file format but as a multi-part architecture that includes formal information models, implementation methods, and application protocols that scope information requirements for particular industries and lifecycle contexts. This architectural framing matters for PLM efficiency research because exchange is

not limited to geometric shapes; it also concerns how product structure, configuration identity, and associated metadata remain interpretable when transferred across systems. When a product definition leaves its authoring system, the exchange mechanism must preserve enough semantics for downstream lifecycle actions such as release packaging, change impact analysis, manufacturing planning, and long-term retention. The literature positions STEP as a mechanism for enabling lifecycle continuity by providing a standardized means to encode and interpret product information, and it treats the standard's modeling rigor as a way to reduce ambiguity across organizations. Interoperability baselines also function as measurement baselines in empirical PLM studies, because they define expected information content and interpretation boundaries that allow researchers to identify where integration breaks (Lee et al., 2019). In this way, STEP serves as a reference layer for diagnosing representational mismatches when CAD-derived content is integrated into PLM objects and lifecycle states. The importance of such baselines is amplified by global engineering conditions where design and manufacturing partners operate with different toolchains, different data governance practices, and different internal data models. Under these conditions, inconsistent exchange practices often lead to measurable inefficiency, including repeated conversions, manual clean-up of transferred models, duplicated item identities, and delays in downstream validation and approval cycles. Standards-oriented literature also links interoperability baselines to conformance and testing ecosystems, because exchange success depends on both syntactic correctness and semantic alignment with agreed information requirements. Industry-focused efforts around translator testing and harmonization highlight the operational reality that standards adoption alone does not guarantee integration quality; organizations still experience variations in how application protocols are implemented and how semantics are interpreted in specific toolchains (Bibri, 2022). This supports a PLM efficiency view in which interoperability standards establish a necessary baseline for exchange, while integration quality remains an empirical outcome shaped by how repository data, metadata, and lifecycle states are packaged and interpreted during transfers.

Figure 5: Product Data Exchange Integration Framework



The literature identifies multiple integration failure modes that arise when exchanged representations do not align with the information needs and semantic expectations of receiving systems, and it treats these failure modes as quantifiable contributors to lifecycle inefficiency. Representation mismatch appears in practice as conversion retries, incomplete imports, degraded semantic content, and ambiguous mappings between CAD-native constructs and PLM lifecycle objects (Wang & Wang, 2019). Conversion retries occur when export/import operations fail due to unsupported entities, inconsistent modeling practices, or mismatched assumptions about the target schema; these retries generate visible operational signals in workflow logs such as repeated job execution, increased queue times, and repeated failure codes. Semantic loss is treated as a more subtle failure mode: an exchange operation completes successfully at the file level while losing information required for downstream use, such as product manufacturing information, annotation structure, feature intent, configuration applicability, or relationships among assemblies and component references. Because semantic loss is not always directly observable without manual inspection, the literature supports the use of proxies that can be computed from repository and workflow traces, such as the proportion of imported models requiring manual annotation repair, the frequency of downstream validation failures tied to missing attributes, and the rate of exceptions raised during automated checking routines. In PLM contexts, semantic loss becomes operationally visible when downstream processes cannot trust the received data, forcing manual verification steps that extend approval and release cycles. Another failure mode involves identity and structure mismatch, where part identifiers, revision identifiers, or assembly structures do not map cleanly from the CAD repository representation into PLM structures (Liu et al., 2022). This mismatch leads to duplication of objects, inconsistent BOM representations, or misalignment between "as-designed" and "as-released" structures. Such outcomes are measurable through reconciliation events, duplicate detection counts, and repeated re-association actions recorded in lifecycle systems. The literature also describes failure patterns associated with mixed toolchains, where different authoring systems encode modeling constructs differently and translators vary in their handling of edge cases. These differences can produce asymmetric interoperability, where exchange succeeds in one direction but produces degraded content in another, creating rework loops that repeat across iterations of design change. Integration failure modes also interact with change governance because failures often surface at lifecycle gates—validation, review, release packaging—rather than immediately at the moment of export. This delays detection and increases the cost of remediation, which is measurable through longer time-to-implement and increased iterations per change when conversions must be repeated after issues are discovered (Li et al., 2019). In sum, the literature supports treating representation mismatch as a primary efficiency driver that generates quantifiable signals across conversion pipelines, repository event histories, and lifecycle governance workflows, enabling empirical investigation through trace-derived metrics without relying solely on subjective assessments of "translation quality."

Digital product definition data practices are treated in the literature and standards guidance as a governance layer that specifies how product definition datasets are prepared, revised, and presented so that they remain interpretable and auditable across lifecycle usage. ISO guidance on digital product definition data practices emphasizes the structured preparation of datasets and the disciplined management of revisions, recognizing that the digital model and its associated annotations constitute an authoritative product definition in many industrial workflows (Masud & Hossain, 2024; Patacas et al., 2020; Zulqarnain & Subrato, 2023). Revision discipline is positioned as essential for auditability because lifecycle decisions require a clear record of what definition was valid at which point in time, who authorized it, and how it relates to earlier and later states. In empirical terms, these requirements translate into measurable proxies that can be extracted from repositories and PLM workflow systems. Metadata completeness fields provide one measurable proxy: datasets that consistently include required identifiers, revision markers, lifecycle state tags, authoring context, and approval attributes are more likely to move through release gates without exception handling. Revision table adherence is another proxy, operationalized through the presence, consistency, and correctness of revision history information and the alignment between stored revision markers and recorded lifecycle transitions. Controlled dataset packaging success rate functions as a further proxy by measuring whether product

definition bundles—comprising models, associated annotation representations, derived deliverables, and required metadata—are assembled and validated successfully under automated workflow gates (Abubakr et al., 2020). When packaging fails, it commonly triggers measurable responses such as workflow job failures, re-run counts, and longer dwell times in pre-release states. The literature also treats presentation consistency as a data quality dimension because downstream consumers depend on predictable annotation and representation practices; inconsistency increases verification burden and creates a higher likelihood of review delays. In systems where digital product definition datasets are used as release artifacts, completeness and revision discipline become tightly coupled to change governance because every change requires a controlled update of the dataset, its identifiers, and its approval state. Repository and workflow traces capture this coupling through repeated dataset rebuilds, verification failures tied to missing fields, and irregular state transition patterns. The role of ISO-based practices in shaping integration quality is also reflected in how datasets are exchanged: consistent preparation and revision conventions make it easier for receiving systems to interpret lifecycle meaning, while inconsistent conventions increase the need for manual mapping and reconciliation. These dynamics support a view of digital product definition practices as operational determinants of PLM efficiency rather than formal documentation requirements alone, because disciplined dataset preparation is associated with fewer exceptions, fewer rework loops, and more reliable release progression within traceable lifecycle workflows (Stylidis et al., 2020).

Cloud Computing in PLM: Architectural Mechanisms That Change Efficiency

Cloud computing is commonly defined in the literature as an on-demand model for accessing configurable computing resources that can be provisioned and released with minimal management effort, and this definition has been widely used to frame enterprise migration decisions and empirical evaluation approaches (Jinnat & Kamrul, 2021; Singh & Misra, 2019). When translated into PLM settings, cloud computing's essential characteristics become concrete mechanisms that can be measured through repository events and workflow telemetry rather than treated as abstract infrastructure features (Akbar & Sharmin, 2022; Zulqarnain & Subrato, 2021). Elasticity is frequently discussed as the capacity to scale resources to meet workload variation, and PLM workloads naturally fluctuate with engineering change bursts, concurrent design cycles, automated validation jobs, translation pipelines, visualization rendering, and release packaging activities (Foysal & Subrato, 2022; Zulqarnain, 2022). In quantitative terms, elasticity is reflected in runtime variability under peak load: when resources expand to absorb bursts, workflow queues, job runtimes, and waiting times tend to show lower dispersion and fewer extreme outliers; when scaling is constrained, congestion effects surface as longer queues and higher tail latency (Abdul, 2023; Hammad & Mohiul, 2023). Broad network access, another defining characteristic, maps directly onto cross-site collaboration conditions common in global product development, where engineering teams access the same authoritative CAD repositories and lifecycle objects from different locations, time zones, and networks. This characteristic affects measurable concurrency patterns and latency signatures: repository commit or check-in densities, simultaneous access events, merge/conflict incidence, and time gaps between dependent actions can be traced to how consistently remote users can interact with lifecycle systems (Hasan & Waladur, 2023; Naseem et al., 2023; Rifat & Rebeka, 2023). Broad access also intersects with governance because remote review and approval actions depend on stable access pathways; measurable dwell time in review states can increase when access conditions are uneven, and repository activity can cluster around local availability windows rather than around process needs. Measured service, often described as metered resource usage and transparent utilization reporting, introduces a measurement layer that supports optional secondary outcomes linking cost and throughput. In PLM environments, measured service can be connected to the volume of workflow executions, compute-intensive validation workloads, storage growth of CAD artifacts, and network transfer associated with distributed access. This creates a basis for efficiency measurement that acknowledges resource consumption alongside cycle time and quality outcomes, especially when automated pipelines generate consistent counts of jobs, retries, failures, and successful releases. The literature also emphasizes that cloud service models structure which components are managed by providers versus customers, shaping what can be optimized and what is observed. In PLM terms, the degree to which CAD repositories, workflow engines, and integration services are hosted influences the granularity and reliability of telemetry, which affects the quality of

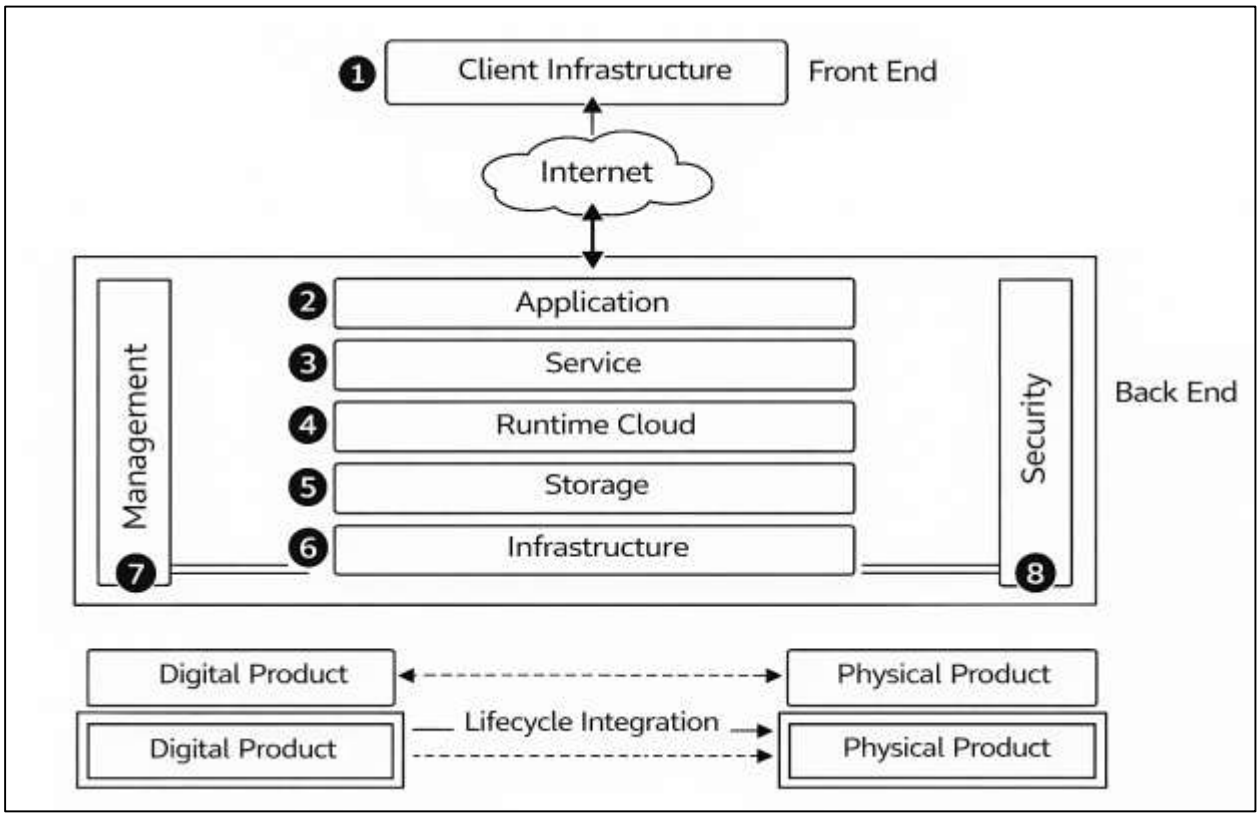
empirical measurement (Talhi et al., 2019). These mechanisms align with a trace-based view of PLM efficiency: elasticity is reflected in runtime dispersion under load, broad access is reflected in cross-site concurrency and latency patterns, and measured service provides a transparent accounting layer that can be linked to throughput and reliability signals produced by cloud-native PLM workflows.

Deployment models are consistently treated in the literature as governance-relevant design choices rather than merely hosting preferences, because they shape how organizations manage data control, identity enforcement, compliance evidence, and operational accountability. In quantitative PLM research, public, private, and hybrid deployments can be handled as categorical independent variables that represent distinct governance conditions, especially when product definition data and change records are sensitive, regulated, or shared across supply networks (Liu et al., 2020; Md & Sai Praveen, 2024; Nahid & Bhuya, 2024). Public deployments commonly emphasize provider-managed scalability and standardized service interfaces, which can increase uniformity of access and instrumentation but also introduce policy constraints around data residency and shared infrastructure. Private deployments can provide stronger organizational control over data locality and configuration, enabling stricter alignment with internal security and change governance rules, while potentially introducing variability in instrumentation and operational maturity across sites (Newaz & Jahidul, 2024; Akbar, 2024). Hybrid deployments combine elements by placing sensitive repositories or governance components under stricter control while using cloud-based services for scalable workflows such as validation, translation, or visualization. These deployment categories affect measurable PLM outcomes through mechanisms tied to data locality, access control consistency, and audit log completeness. Data locality influences repository latency and the timing of dependent actions because geographic placement of storage and compute can amplify or reduce round-trip delays for remote users and automated pipelines. In measurable terms, data locality effects appear in access latency distributions, time gaps between repository commits and workflow triggers, and duration of synchronization steps between CAD repositories and lifecycle state transitions. Access control enforcement consistency affects whether users experience predictable permission checks across services; inconsistency can generate measurable friction in the form of repeated authorization failures, manual escalations, and delayed approvals when reviewers cannot access required artifacts. Audit log completeness matters for empirical PLM studies because lifecycle governance depends on traceable evidence of who performed which actions, under what authorization, and at what time; incomplete logs reduce the reliability of derived metrics for approval throughput, change lead time, and reconciliation frequency (Liu et al., 2020; Rabiul & Alam, 2024; Sai Praveen, 2024). Governance impacts also emerge through boundary management: hybrid deployments often require integration and synchronization across trust domains, and this can generate measurable integration overhead through additional validation gates, repeated data transfers, and exception-handling events. The literature on cloud adoption and enterprise risk management treats these governance effects as operational realities that influence organizational performance, particularly when systems span multiple departments and external partners (Azam & Amin, 2024). For PLM, governance is inseparable from deployment because lifecycle states and approvals are only meaningful when identity, permissions, and audit evidence are reliable across every workflow touchpoint. Therefore, deployment models can be linked to measurable outcomes including approval dwell times, frequency of access-related workflow failures, completeness of change records, and stability of lifecycle state transitions, providing a structured way to analyze how architectural hosting decisions shape PLM efficiency through governance mechanisms rather than through infrastructure speed alone (Wang et al., 2020).

Operational performance and reliability are treated in the literature as core determinants of enterprise system effectiveness, and in PLM environments they function as direct contributors to lifecycle throughput stability rather than as purely technical service-level concerns. PLM processes rely on continuous availability of authoritative repositories and workflow services because engineering work is iterative and tightly coupled to current product definition states (Mishra, 2020). When a CAD repository or PLM workflow service becomes unavailable, the immediate result is not only downtime but also accumulation of blocked tasks—unsubmitted changes, stalled approvals, failed validations, and delayed releases—which shifts cycle-time distributions and increases tail latency. Incident

recovery behavior is similarly consequential: the time required to detect failures, restore service, and reconcile any partial workflow executions determines how quickly the organization returns to steady-state throughput. Reliability literature and site reliability engineering practices emphasize measurable indicators such as availability, incident frequency, and recovery characteristics, and these indicators translate well into trace-based PLM measurement because cloud-native systems generate incident timelines, error rates, retry counts, and backlog growth signatures. In a workflow-centric PLM setting, reliability is visible through job failure rates, automatic retry behavior, and the number of manual interventions required to resolve pipeline exceptions (Zhang et al., 2019). These reliability signals connect to efficiency because repeated failures and retries consume compute resources, delay downstream steps, and increase rework probability when teams re-run exports or regenerate release packages. Availability also interacts with global collaboration patterns: distributed teams depend on shared systems across time zones, and outages can disproportionately impact cross-site handoffs by forcing asynchronous waiting until services resume. This appears empirically as activity clustering around recovery windows, prolonged dwell time in review states, and bursts of repository changes after restoration. Operational performance also encompasses responsiveness under load, which connects to the elasticity mechanism but remains distinct in that performance degradation can occur even when systems are technically "up." Latency spikes and queue growth reduce coordination efficiency by increasing time-to-feedback for validation and review cycles. The literature on operational excellence links observability and instrumentation to faster recovery and more stable operations because richer telemetry supports quicker diagnosis and reduces uncertainty during incident response. In PLM systems, this relationship becomes measurable through shorter restoration intervals, fewer repeated failures after recovery, and lower rates of inconsistent lifecycle states caused by interrupted transactions (Yang et al., 2021). From an empirical standpoint, operational reliability therefore becomes part of the dependent construct of PLM efficiency: stable throughput requires not only fast workflows but also predictable execution under normal and abnormal conditions. By treating availability and incident recovery as measurable efficiency dimensions, the literature supports a comprehensive lifecycle view in which cloud operational characteristics shape the distribution of change and release outcomes, not merely their average completion times.

Figure 6: Cloud Computing Architecture for PLM



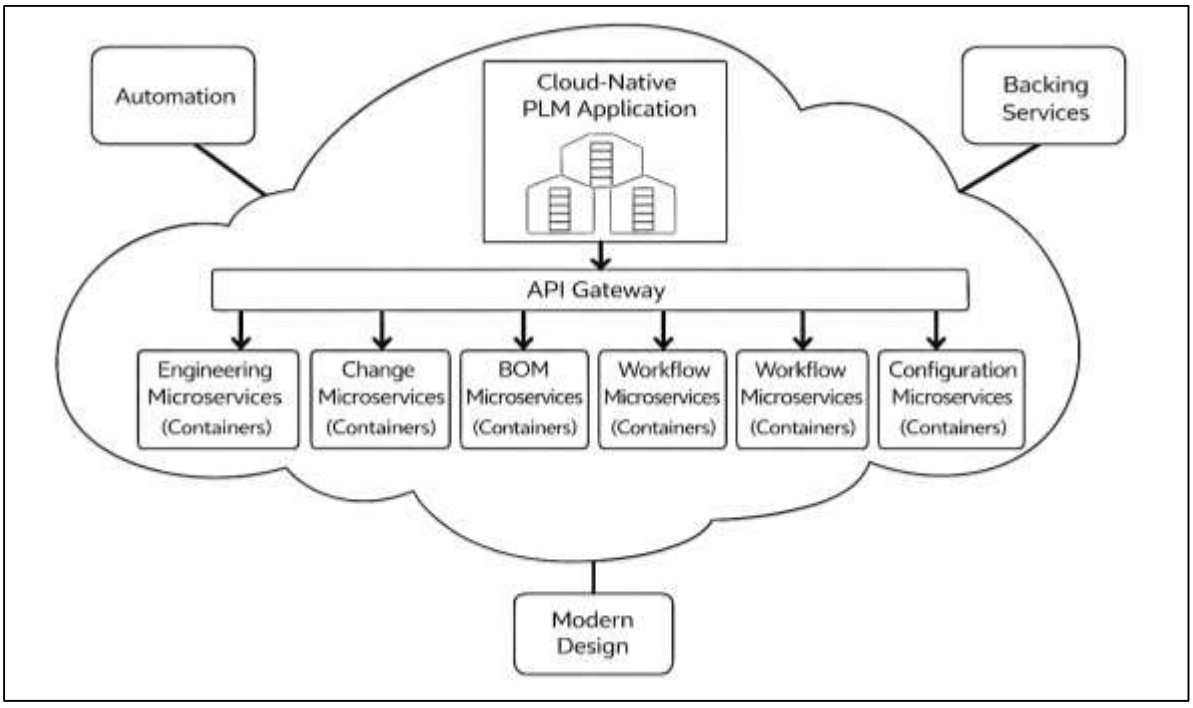
Across the literature, a coherent measurement-oriented synthesis emerges in which cloud computing's defining characteristics, deployment model governance effects, and operational reliability jointly shape PLM efficiency through observable mechanisms that can be captured in repositories and workflow telemetry. Elasticity and performance management influence runtime distributions, queue behavior, and the stability of automated validation and release pipelines (Ali et al., 2020). Broad network access shapes cross-site concurrency patterns and collaboration timing, influencing how quickly design changes propagate through governed lifecycle states. Measured service provides a metering substrate that supports secondary analyses relating resource consumption to throughput and quality outcomes, especially when automated workflows produce consistent counts of jobs, retries, and successful packages. Deployment model choices shape governance conditions—data locality, permission enforcement consistency, and audit evidence completeness—that determine whether lifecycle state transitions remain trustworthy and whether approval and release workflows proceed without access friction or evidence gaps. Operational reliability translates these architectural conditions into real process stability: availability and recovery behaviors influence the frequency and magnitude of schedule-disrupting outliers and backlog accumulation, and they leave trace signatures in job failures, retries, and post-recovery activity bursts. The literature also recognizes measurement risks when operationalizing these constructs from traces (Kulkarni et al., 2022). Logging completeness can vary across service models, and hybrid deployments can introduce fragmented telemetry that obscures end-to-end timing unless logs are correlated across systems. Workload heterogeneity can confound interpretation of runtime distributions unless product complexity and collaboration intensity are controlled, and network variability can influence latency signals independently of architecture design. Studies that emphasize trace-based empirical approaches address these threats by using multi-indicator measurement, segmentation by toolchain and deployment context, and careful boundary definitions for events such as "change initiation," "approval completion," and "release readiness." In PLM settings, analogous practices strengthen construct validity by tying metrics to governed lifecycle objects and state transitions rather than to raw file edits or ambiguous activity. This synthesis supports a quantitative framing in which cloud architecture variables are not treated as generic "technology adoption" markers but as operational mechanisms that systematically influence cycle time, quality events, and stability (Lim et al., 2020). It also supports the methodological argument that cloud-native PLM environments are uniquely measurable because automation and standardized service interfaces generate consistent telemetry across large numbers of lifecycle events. As a result, the literature positions cloud computing not simply as an infrastructure alternative but as an architectural condition that reshapes how PLM work is executed, governed, and observed, allowing empirical studies to quantify differences in throughput, reliability, and predictability using industrial repository records and cloud workflow logs.

Workflow Automation as a Measurement Substrate

Cloud-native architecture is described in the literature as a paradigm for building and operating systems that remain reliable and adaptable under changing demand, frequent change, and distributed access, and this paradigm aligns closely with the operational conditions of PLM environments that coordinate large engineering datasets and multi-step governance workflows (Sebrechts et al., 2022). Cloud-native definitions emphasize packaged and dynamically managed components, automated practices, and architectural separation that supports frequent, predictable change with reduced operational toil. In PLM settings, these ideas translate into measurable workflow properties because PLM work is executed through repeatable sequences such as validation, translation, synchronization, approval routing, and release packaging. Resilience is treated as the system's capacity to continue delivering correct service under faults and partial failures, which becomes measurable in PLM by the frequency and duration of workflow interruptions, the proportion of recoverable failures, and the stability of repository-to-workflow propagation under stress. Manageability is discussed as the degree to which systems can be operated consistently through automation, configuration control, and standardized deployment, which becomes measurable through the repeatability of workflow execution, the reduction of manual remediation steps, and the consistency of operational behaviors across environments (Vaño et al., 2023). Observability is framed as the ability to infer internal system

states from external outputs such as logs, metrics, and traces, and it becomes central for PLM measurement because lifecycle efficiency depends on understanding why delays, retries, and exceptions occur across multi-service pipelines rather than only knowing that outcomes are slow. Automation is treated as a mechanism for reducing variability introduced by manual steps and for creating reliable data trails of execution events, and this fits PLM workflows where repeatable gates and checks determine whether product definition data is ready to transition into controlled lifecycle states (Shainer et al., 2021). Cloud-native quality attributes therefore act as both operational objectives and measurement enablers: an observable, automated workflow yields the event data needed to compute cycle-time distributions, exception rates, retry frequency, and tail behaviors, while resilient and manageable systems reduce error cascades that inflate rework and destabilize approval throughput. This literature also connects cloud-native thinking to socio-technical control, because frequent releases of platform components change the shape of operational risk; a measurable record of deployment and runtime events becomes a prerequisite for interpreting whether performance variation originates in product complexity, collaboration concurrency, or platform behavior. In PLM research designs that rely on industrial CAD repositories and cloud-native workflows, these attributes clarify why cloud-native architecture functions as a "measurement substrate": it generates consistent, time-stamped traces of workflow execution and service interactions, making it possible to study lifecycle governance using observed process behavior rather than self-reported maturity (Psaromanolakis et al., 2023). The same attributes also align with distributed engineering realities where global teams access shared repositories; observability and automation support consistent operations across regions, and resilience and manageability reduce the extent to which localized outages or configuration drift distort repository-driven measures of efficiency.

Figure 7: Cloud-Native Microservices Architecture for PLM



The observability and automation literature provides a direct methodological bridge to repository-centric PLM research because both domains treat event data as a primary empirical source for understanding performance, reliability, and coordination. Observable systems emit structured signals that support diagnosis, performance analysis, and operational learning, and these signals align with PLM workflows that already produce discrete events such as "validation started," "translation failed," "approval completed," "release packaged," and "synchronization reconciled (Kahvazadeh et al., 2022)." In a cloud-native context, the availability of standardized telemetry practices strengthens the reliability of measurement because instrumentation conventions reduce ambiguity about what a given

signal represents and how it is correlated across services. Telemetry frameworks that define signals such as distributed traces, metrics, and logs provide a vocabulary for connecting repository events to workflow execution paths and for attributing delays to specific service interactions rather than treating cycle time as an undifferentiated duration. This matters in PLM because many "time losses" occur inside multi-step pipelines: a change event triggers checks, checks trigger exports, exports trigger validations, validations trigger packaging, and packaging triggers state transitions. Without observability, these steps collapse into a single opaque lead time (Breitgand et al., 2021). With observability, lead time decomposes into measurable segments that can be compared across architectures, deployments, and integration approaches. Automation complements observability by producing repeatable and auditable execution traces. When workflow gates are automated, each run produces consistent artifacts—start and end times, status outcomes, error categories, and retry sequences—that support statistical analysis of reliability and stability. The automation literature also emphasizes that repeatability reduces uncontrolled variance, improving construct validity when comparing efficiency across contexts. In PLM, automated packaging and validation reduce the reliance on informal manual checks that vary by individual and site, producing a more stable measurement surface where differences in performance can be traced to integration quality or product complexity rather than to personal working styles (Attaoui et al., 2023). Automation also creates measurable exception pathways: manual intervention events, override counts, and remediation task durations provide direct proxies for integration friction and governance workload. Observability and automation jointly support incident analysis in cloud-native operations, and these practices map to PLM because incidents interrupt approvals, delay releases, and induce rework when partial workflow execution leaves inconsistent states. Operational literature treats incident timelines, error budgets, and recovery behavior as measurable phenomena, and the same approach applies to PLM pipelines where job failures and retries translate into lifecycle throughput instability. The empirical value of this view is that observability and automation do not simply "improve operations"; they create the structured trace data needed to measure how PLM efficiency behaves across ordinary and disrupted conditions (Matelsky et al., 2021). This framing supports trace-based research designs that rely on repository and workflow telemetry to quantify change lead time, release lead time, exception burden, and tail latency while also controlling for platform behavior that could otherwise masquerade as process inefficiency.

CAD Repositories for PLM Efficiency

Industrial CAD repositories function as high-resolution empirical records of engineering work because they preserve both artifact evolution and the interaction patterns through which teams construct product definition. Literature on digital engineering data management consistently treats repositories as more than storage because repository activity reflects governance, collaboration, and configuration behavior in ways that support quantitative analysis (Nzetchou et al., 2019). An event-model perspective represents repository activity as sequences of artifact-level events such as create, modify, derive, rename, move, release-tag, reference update, check-in/check-out, and version creation. Each event type carries analytic meaning because it connects an actor, a timestamp, a target artifact, and a context that often includes identifiers, workspace states, and dependency relationships. This enables operational measurement of PLM efficiency through time-bounded traces, including cycle time from initial artifact creation to release readiness, iteration density around change windows, and the timing of dependency updates that signal configuration propagation. CAD artifacts also embody structured dependencies that differentiate CAD repositories from generic file systems. Assemblies reference parts, drawings reference models, and configuration rules control applicability; these relationships generate product structure signals that can be represented as assembly dependency graphs, where nodes correspond to parts or subassemblies and edges represent "uses," "references," or "derives" relations. Repository event histories reveal how these graphs evolve, how stable references remain across iterations, and how often reuse occurs through cloning, instancing, or derivation (Singh & Misra, 2019). Reference stability becomes an efficiency-relevant signal because stable relationships reduce the need for reconciliation and revalidation during release packaging, while unstable relationships coincide with broken references, assembly rebuild errors, and repeated correction cycles. Reuse patterns similarly hold measurement value because reuse reduces redundant modeling effort and often correlates with standardized component libraries; repository traces operationalize reuse through repeated

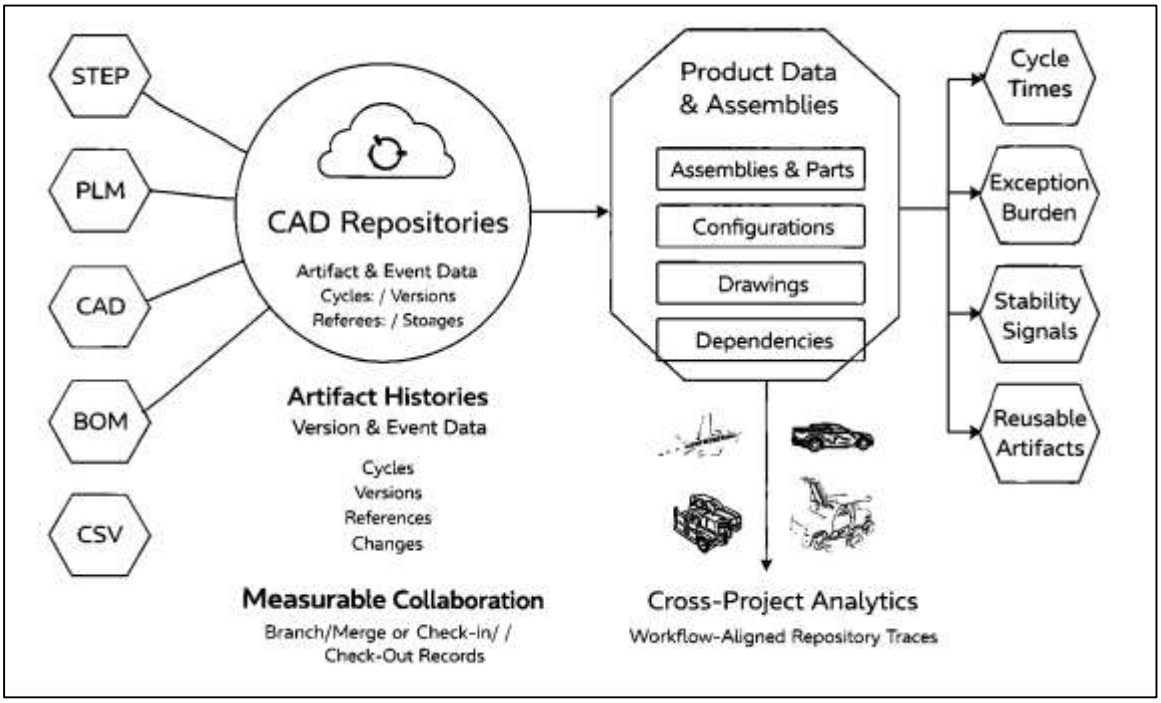
consumption of the same artifact identity across multiple assemblies or through repeated derivation from a shared baseline. Empirical literature on extracting structure from large product design repositories further supports treating hierarchies and metadata as analyzable signals that scale beyond a single product, enabling cross-project comparisons of structural complexity and reuse intensity. A repository-centric event model also supports decomposition of PLM activity into measurable sub-processes such as modeling, integration, validation, review, and release packaging, particularly when repository events are aligned with workflow execution logs. This alignment makes it possible to observe not only that work occurred, but also how it flowed through lifecycle gates, where it paused, and which artifact relationships changed before and after controlled transitions (Buchert et al., 2019). The result is a measurement substrate that captures both micro-level engineering actions and macro-level lifecycle movement using consistent trace semantics derived from artifact histories and product structure graphs.

Research and industry-oriented evidence on CAD data management repeatedly highlights version control as a central determinant of collaboration efficiency because it governs how multiple contributors interact with shared product definition artifacts without corrupting integrity. Two broad approaches appear across CAD repositories and PDM-integrated environments: locking-oriented coordination and parallelism-oriented coordination (Kulkarni et al., 2022). Locking approaches commonly implement check-in/check-out mechanisms that prevent simultaneous edits to the same artifact or impose strict ownership during editing sessions. This approach reduces accidental overwrites and supports a clear "single editor" model, yet it introduces measurable coordination costs when work requires rapid iteration across interdependent artifacts. In trace data, locking-oriented friction appears as waiting time between check-out and check-in, lower concurrency density on shared assemblies, and extended change lead time when access constraints serialize work that could otherwise proceed in parallel. Parallelism-oriented approaches permit concurrent work by maintaining explicit version histories and providing mechanisms to reconcile divergent changes. In cloud CAD contexts, branching and merging operationalize this parallelism by enabling multiple workspaces to evolve from a shared baseline and later be merged into a consolidated state (Schiller et al., 2022). This collaboration model produces measurable predictors and outcomes that are directly observable in repository traces: branch creation frequency indicates experimentation and parallel work, merge frequency indicates integration cadence, conflict incidence indicates overlap in edits to the same features or dependencies, and iteration speed indicates how quickly teams move from exploration to integrated configurations. On shape's documentation and feature descriptions characterize branching and merging as a design mechanism that captures edits continuously, supports immutable versions, and provides merge workflows that surface conflicts at a feature level before applying changes. These characteristics translate into measurable collaboration mechanics: higher merge cadence with lower conflict rates indicates stable coordination patterns, while high conflict rates and repeated merges indicate complex interdependence and elevated coordination overhead. Merge-related traces also capture "rework-like" behaviors, such as repeated attempts to reconcile conflicts, rollback to pre-merge states, and post-merge remediation edits, all of which affect PLM efficiency because they delay downstream validation and release gating. This literature also treats the choice between locking and merging as more than a tooling preference because each approach shapes the distribution of coordination effort: locking concentrates effort into access control and scheduling, while branching/merging concentrates effort into integration and conflict resolution (Corallo et al., 2022). Both profiles become measurable through trace-derived signals such as time spent in exclusive edit states, ratio of parallel workspaces to active contributors, number of integration events per change window, and the density of edits on shared dependencies. By framing version control approaches as measurable predictors, the literature supports quantitative models that explain PLM efficiency outcomes—change lead time, approval throughput, and reconciliation events—through observed collaboration mechanics rather than through assumed cultural maturity.

Repository-centric PLM research depends on the quality of the underlying event data, and the literature on event-log analysis and process mining emphasizes that preprocessing is a decisive stage because raw operational data often contains ambiguities, missingness, and inconsistencies that distort

performance inference. Identity resolution represents a primary challenge in industrial CAD repositories because "users" can appear under multiple identifiers (usernames, emails, service accounts, API tokens), and "parts" can appear under multiple keys (file names, internal IDs, revision IDs, derived copies) (Ruediger-Flore et al., 2023).

Figure 8: CAD Repository Analytics for PLM



Without careful resolution, measurements such as contributor counts, collaboration concurrency, and reuse patterns become biased, particularly when automated agents perform batch actions that mimic human activity. Timestamp normalization is equally critical because repositories often record events across distributed systems with inconsistent clocks, differing time zones, or identical timestamps for multiple events due to limited resolution. Literature specifically addresses identical timestamp errors and event reordering problems, showing that event ordering directly affects cycle-time metrics, concurrency estimation, and causal interpretation of process sequences. Artifact deduplication introduces another layer because CAD workflows frequently create near-duplicates through copying, importing, mirroring, or derived variants; naive counting inflates artifact growth and can misrepresent reuse as duplication. Deduplication also intersects with reference stability measurement because duplicates may inherit references that appear stable while actually representing divergent artifact identities. Tool-specific logging differences create a systematic bias risk: some CAD/PDM systems log fine-grained feature edits, while others log only file-level check-ins, and some platforms log merge operations with structured metadata while others record merges as ordinary modifications (Pang et al., 2021). This heterogeneity affects cross-repository comparisons unless measurement definitions are carefully aligned to the available event semantics. Missing events pose additional threats, especially for actions performed offline or in disconnected modes, or for integrations that batch changes and emit only summary events. The process-mining literature addresses these issues by recommending multi-step preprocessing pipelines that include filtering, event correlation, case definition, handling of incomplete traces, and sensitivity analysis that tests whether conclusions remain stable under alternative cleaning assumptions. In CAD repository contexts, analogous strategies include correlating repository events with workflow execution logs to confirm whether a "successful export" actually leads to a usable downstream package, segmenting analyses by toolchain to avoid conflating translation limitations with poor practices, and using multi-indicator constructs that reduce dependence on any single noisy signal. These preprocessing concerns reinforce a key measurement principle: repository traces provide strong empirical power only when the mapping from raw events to lifecycle constructs

is transparent, consistent, and auditable (Favi et al., 2019). The literature therefore supports treating identity resolution, timestamp normalization, and deduplication not as minor data-cleaning steps but as methodological requirements that shape the validity of conclusions about PLM efficiency.

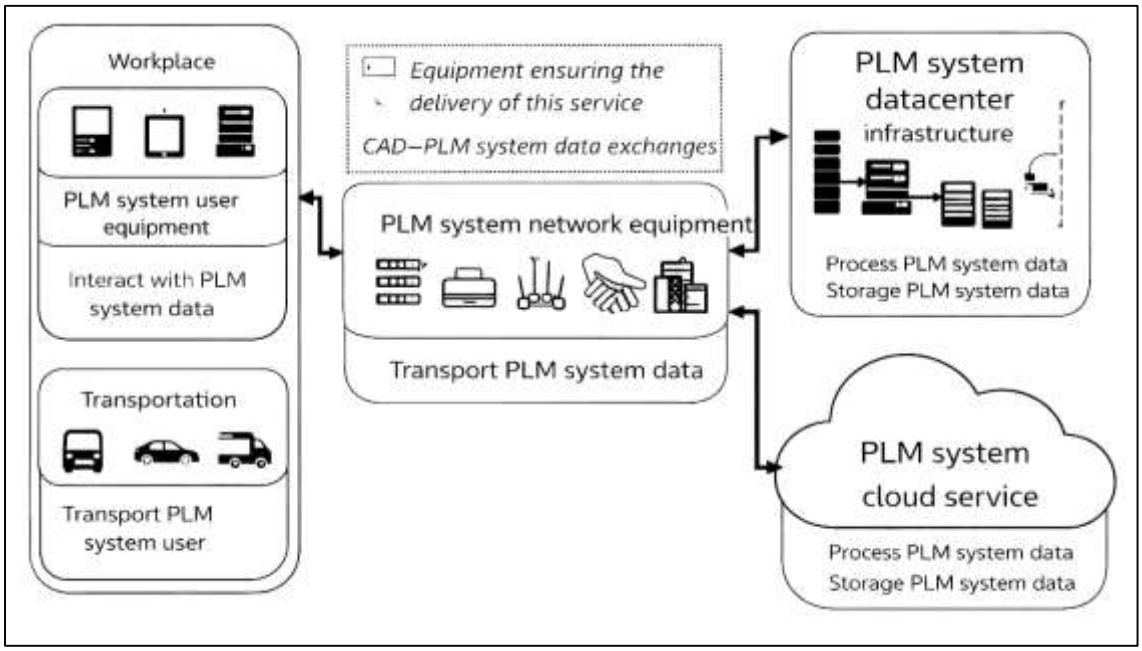
Across the literature, CAD repositories emerge as uniquely valuable empirical data sources for PLM efficiency because they combine longitudinal artifact histories with structured dependency information and collaboration traces that reflect how lifecycle work actually unfolds. An event-model approach provides the conceptual scaffolding needed to convert repository activity into measurable lifecycle indicators, including change-related iteration density, timing of dependency updates, and stability of reference structures across release cycles (Popovic et al., 2021). Product structure signals—assembly graphs, reuse patterns, and reference stability—provide control variables and predictors that help explain why some change cycles move quickly and others accumulate rework and reconciliation events. Version control approaches further shape the measurable texture of collaboration. Locking-oriented repositories produce observable exclusivity patterns and waiting-time signatures, while branching/merging repositories produce integration event signatures such as merge cadence, conflict incidence, and post-merge remediation intensity. Cloud CAD environments that capture fine-grained history and provide explicit merge mechanisms make these signatures especially visible because each integration step leaves traceable markers that align naturally with change governance windows and release packaging workflows. The methodological literature on event-log analytics reinforces that these benefits depend on disciplined preprocessing and careful construct definition. Identity resolution ensures that collaboration measures represent actual contributors and meaningful artifact identities rather than fragmented aliases (Lázaro et al., 2022). Timestamp normalization ensures that computed cycle times and concurrency patterns reflect real temporal order rather than logging artifacts. Deduplication ensures that reuse and growth are not confounded by copying behaviors and derived variants. Bias risks remain salient when events are missing, when platforms differ in logging granularity, or when integrations batch changes; the literature addresses these risks through triangulation across data sources, segmentation by toolchain, and sensitivity checks that test robustness under alternative mappings. This synthesis supports a repository-centric measurement stance in which PLM efficiency is assessed through trace-derived distributions of cycle time, exception burden, and stability, grounded in governed artifact histories and workflow-aligned event sequences. It also supports a practical research logic for empirical studies using industrial CAD repositories: repository traces provide the primary evidence of enacted behavior, while workflow telemetry and system logs provide contextual signals that clarify whether delays originate in collaboration mechanics, structure complexity, integration friction, or platform operations (Zech et al., 2023). In this combined view, CAD repositories do not merely document engineering outcomes; they expose the measurable pathways through which lifecycle control, collaboration, and data integrity interact to produce efficiency variation across projects and teams.

CAD–PLM Integration and Propagation Constructs

The literature on PLM and engineering information systems consistently treats CAD–PLM integration quality as an operational property that becomes visible through timing, integrity, and human effort signals recorded during routine lifecycle execution. Integration quality is frequently framed as the degree to which a change in an authoring environment becomes a governed, traceable, and usable lifecycle object state without ambiguity or rework (Kapos et al., 2019). In quantitative terms, synchronization latency represents a primary construct because it captures the elapsed time between a CAD-side update and the corresponding PLM-side state update, such as a new revision association, an updated product structure, or a lifecycle state transition required for review or release. This latency is not merely a technical delay; it reflects the combined effect of integration mechanisms, repository throughput, validation gates, transformation steps, and queueing within workflow engines. When synchronization latency is high or unstable, downstream lifecycle activities exhibit measurable dwell time increases in pre-release or review states, and parallel collaborators face longer periods in which the "authoritative" definition is ambiguous across systems. Reference integrity functions as a second core construct and is often described as the preservation of correct relationships among assemblies, parts, drawings, and related lifecycle objects as changes propagate. Quantitative indicators of reference integrity include broken links per change, failed association counts between CAD artifacts and PLM

items, and assembly rebuild error rates that arise when dependencies cannot be resolved consistently after synchronization (Eigner, 2021). The literature treats these integrity failures as high-cost events because they trigger diagnosis work, repeated exports or imports, and downstream validation failures that widen change cycle time distributions. Manual reconciliation load provides a third construct that the literature repeatedly positions as a practical "hidden cost" of integration. Reconciliation occurs when the integration pipeline fails to map identities, versions, attributes, or structures cleanly, requiring human interventions such as re-linking references, resolving duplicates, correcting metadata fields, reconstructing BOM relationships, or re-running translations with modified settings. This load is measurable through counts of exception tickets, frequency of manual overrides, number of workflows restarts attributable to data mismatch, and the volume of corrective actions taken per change cycle. Importantly, the literature emphasizes that these constructs interrelate: reference integrity failures increase reconciliation load, reconciliation increases synchronization latency, and increased latency amplifies uncertainty that leads to additional rework. This coupling supports the use of integrated measurement models where CAD commits, PLM state transitions, validation outcomes, and manual interventions are treated as linked traces rather than isolated metrics. Overall, the literature frames CAD integration quality as a measurable, system-level determinant of PLM efficiency that can be observed through trace data describing how quickly changes become governed states, how reliably relationships remain intact across structures, and how much human remediation is required to maintain lifecycle coherence (Zhou et al., 2022).

Figure 9: Assessing CAD–PLM Integration Quality



Research on enterprise integration and lifecycle information management distinguishes integration architectures by how they move and interpret data across systems, and this distinction provides a basis for treating CAD–PLM integration design as an independent variable in quantitative PLM efficiency studies. Direct API synchronization architectures commonly represent integration as near-immediate service-to-service interactions in which CAD-side events or commits trigger API calls that create or update PLM objects, relationships, and lifecycle states (VanDerHorn & Mahadevan, 2021). This pattern is often associated with lower end-to-end propagation delay under stable conditions because it eliminates scheduled batch windows and can align updates with workflow gates. However, the literature also notes that API-based integration introduces sensitivity to interface contract stability, network reliability, and idempotent handling of repeated requests, making error management a measurable part of the integration mechanism rather than an external operational concern. Batch ETL architectures, often described as extract-transform-load pipelines, treat CAD and PLM as periodically synchronized systems in which data is exported, transformed, validated, and loaded on a schedule. In

this architecture, synchronization frequency becomes a direct driver of latency and data drift: longer intervals correspond to longer windows in which CAD and PLM representations diverge, while shorter intervals raise operational volume and increase the chance of repeated partial failures that accumulate reconciliation work. Event-driven pipeline architectures define integration as reactions to streams of domain events, such as "revision created," "structure updated," or "release requested," often mediated through messaging infrastructure that decouples producers and consumers. Literature on distributed systems describes event-driven patterns as enabling scalable coordination and clearer separation of concerns, while also requiring robust event schema governance, ordering guarantees, and replay handling to prevent inconsistent state updates (Demirel et al., 2021). These architectural categories share additional measurable dimensions that the literature treats as mechanism-level drivers of performance: directionality of synchronization (CAD-to-PLM only versus bidirectional coordination), transformation step depth (number of mappings, translation, validation, and enrichment stages), and coupling intensity (how many downstream services depend on each integration action). Transformation step depth is particularly important in CAD–PLM contexts because representation mappings often include identifier mapping, attribute normalization, structure derivation, translation of geometry formats, and creation of release packages or derived artifacts. Each step creates potential failure points and contributes to queueing and tail latency, which become observable in workflow logs as stage-level durations and failure distributions. The literature also emphasizes that architecture interacts with governance: bidirectional synchronization can introduce state conflict risks when both systems assert authority over overlapping attributes, while unidirectional patterns can reduce conflict at the expense of requiring disciplined ownership rules. As a result, integration architecture can be modeled as a structured set of independent variables—sync pattern, frequency, directionality, and transformation depth—that explain variance in measurable outcomes such as propagation delay, integrity failures, and reconciliation burden, thereby linking system design choices to observed PLM efficiency behavior (Paramatmuni & Cogswell, 2023).

The literature identifies recurring CAD–PLM integration failure modes that manifest as traceable inefficiencies and quality losses across lifecycle workflows. Data drift is often described as divergence between representations of the same product definition across systems, arising when updates in one environment do not propagate reliably, propagate partially, or propagate with altered semantics (Barricelli et al., 2019). Drift becomes measurable through mismatched attribute values, inconsistent revision markers, discrepancies between CAD structures and PLM BOM structures, and repeated reconciliation events where teams attempt to "realign" authoritative state. Duplicated objects constitute another common failure mode, frequently associated with identity mapping issues, repeated imports, parallel creation of similar items by different actors, or inconsistent handling of derived variants. Duplication has measurable consequences: it increases search and selection time, introduces ambiguity during approvals, inflates affected-item lists, and triggers corrective work when downstream processes reference the wrong object instance. Inconsistent configurations appear when structure relationships and applicability rules fail to remain aligned across revisions, producing states where an assembly references incompatible component versions or where released baselines do not reflect the intended structure. These inconsistencies surface as assembly rebuild errors, invalid reference graphs, or failed downstream validations that require rework before release packaging completes (Stark, 2022). Translation errors form a distinct category of failure, particularly when integration relies on intermediate representations or when geometry and annotation content require conversion. Translation errors include hard failures (conversion fails and must be retried) and soft failures (conversion completes but semantic content is incomplete or degraded), and both forms yield measurable footprints in workflow telemetry such as repeated job attempts, increased runtime variance, elevated exception rates, and manual remediation actions. The measurable consequences extend beyond the integration step itself because PLM workflows depend on usable, consistent data to complete governance actions. Validation failures increase when integrated data violates rules for metadata completeness, structure consistency, or dataset packaging standards, producing longer dwell times and higher tail latency in release processes. Delayed release packaging is particularly salient because it represents a lifecycle gate that often sits close to downstream consumption; failures at this

stage force corrective loops that revisit earlier steps, increasing iterations per change and amplifying schedule uncertainty. The literature also notes that failure modes often co-occur: drift increases the likelihood of duplication, duplication increases the likelihood of inconsistent configurations, and translation failures increase drift by delaying or fragmenting updates. In trace data, these interactions appear as clustered exceptions, bursts of repeated exports, increased manual interventions, and repeated state regressions in approval workflows (Ameri et al., 2022). This body of work supports an empirical stance that treats integration failures not as rare anomalies but as measurable determinants of efficiency distributions, especially in environments characterized by complex assemblies, frequent changes, and distributed collaboration. By classifying failure modes and linking them to measurable downstream impacts, the literature provides a structured foundation for operationalizing CAD–PLM integration quality as a set of observable phenomena that explain variance in cycle time, rework intensity, and stability of lifecycle throughput.

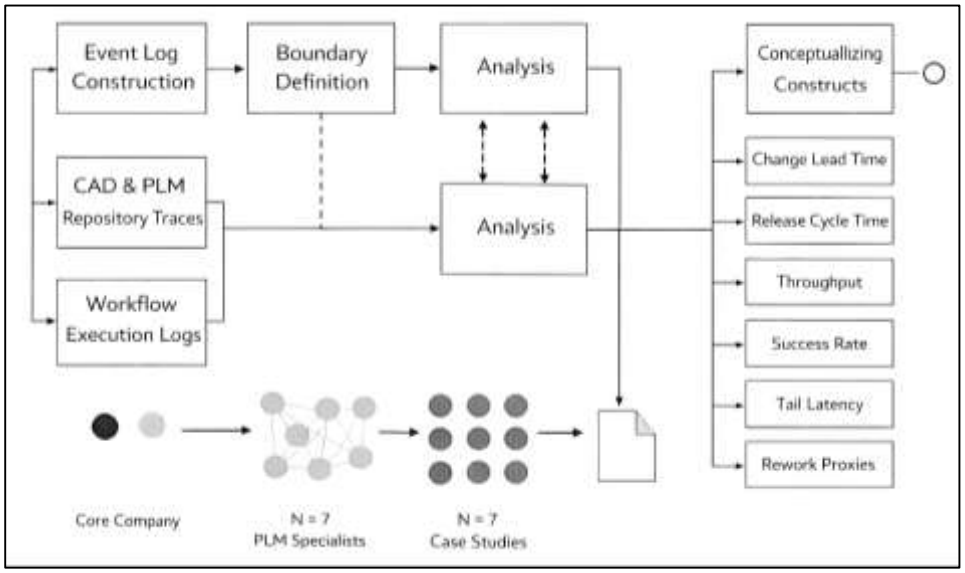
Across PLM, integration, and process analytics scholarship, CAD–PLM integration emerges as a quantifiable layer where integrity and propagation constructs connect system design choices to observed lifecycle efficiency outcomes. Synchronization latency, reference integrity, and reconciliation load form a coherent measurement trio: latency captures speed of state alignment, integrity captures correctness of relationships and structures, and reconciliation captures human effort required to maintain alignment. Integration architectures—API-centric synchronization, scheduled batch ETL, and event-driven pipelines—provide explanatory mechanisms that shape these constructs through differences in coupling, timing, error handling, and transformation depth (Hodgkinson & Elmouttie, 2020). Failure modes such as data drift, duplication, inconsistent configurations, and translation errors provide causal pathways through which architecture and governance conditions produce measurable inefficiency. The process analytics literature reinforces that these relationships are most convincingly examined using trace data rather than perception-based assessments, because traces preserve event ordering, frequency, and outcomes at scale. Industrial CAD repositories supply the artifact change history and dependency changes that define when and how product definition evolves, while PLM state logs and workflow telemetry record when governed states update, when validations pass or fail, and when packaging completes or loops back for remediation. This alignment enables empirical segmentation that strengthens validity: propagation can be measured from CAD commits to PLM state transitions, integrity can be measured through rebuild errors and broken associations after sync events, and reconciliation can be measured by exception handling events and manual override markers. The literature also highlights measurement threats that are manageable through robust operational definitions. Mixed toolchains and varied translator behavior can confound translation error metrics, making segmentation by authoring tool and pipeline path important (Biller & Biller, 2023). Partial exports and soft semantic loss can inflate "success" counts, making downstream usability checks and packaging outcomes important complementary signals. Identity mapping ambiguity can distort duplication measures, making identity resolution and deduplication steps essential in preprocessing. These methodological points support a synthesis in which CAD–PLM integration is analyzed as an event-driven process with explicit boundaries and observable consequences rather than as an invisible middleware layer. Within this synthesis, the constructs in this section serve directly as quantitative variables: synchronization latency distributions as efficiency indicators, integrity failure rates as quality indicators, reconciliation load as toil indicators, architecture types and transformation depth as independent variables, and failure-mode counts as mediating variables that explain downstream workflow disruption. The combined literature therefore provides a rigorous foundation for repository-centric and workflow-telemetry-centric empirical studies of PLM efficiency, because integration quality becomes measurable through traceable propagation paths and integrity outcomes that link design changes to governed lifecycle states and ultimately to release execution performance (Corallo et al., 2021).

Quantitative Measurement Framework

A trace-based quantitative measurement framework treats PLM efficiency as an empirically observable property of lifecycle execution rather than as a subjective assessment of process maturity. The literature on process mining, workflow analytics, and operational performance measurement emphasizes that valid dependent variables must be anchored in explicit event boundaries that can be consistently

detected across systems (Gudbrandsdottir et al., 2021). In PLM contexts, engineering change lead time is often operationalized by defining a start event that indicates formal initiation of a governed change and an end event that indicates completion of the change through closure or implementation in a released or otherwise authorized lifecycle state. The literature stresses that lead time definitions must be tied to governance signals rather than raw file edits, because uncontrolled edits can inflate activity counts while failing to represent lifecycle progress. Release cycle time is similarly tied to lifecycle gates and state transitions, capturing the elapsed time between a definable point of readiness for release processing and the point at which the product definition package becomes authorized for downstream use. This focus on state-based boundaries aligns with information governance studies that emphasize lifecycle states as the operational representation of decision control. Workflow throughput extends these timing measures into volume measures by counting completed workflow instances per unit time, such as jobs per day, completed validations per day, or release packages finalized per day. Operational research on service systems and software delivery measurement often treats throughput as a core indicator of capacity, and this mapping is compelling in PLM because lifecycle systems operate as queues of change requests, approvals, validations, and releases. Success rate provides an outcome quality lens by capturing the proportion of workflow executions that complete without failure, manual override, or rework-inducing exceptions (Lim et al., 2021). The literature also emphasizes tail latency because complex socio-technical workflows frequently exhibit heavy-tailed timing behavior where a small proportion of cases account for a disproportionate share of delay; these long-tail cases represent stability risk and are highly relevant in lifecycle governance, where late releases can block downstream manufacturing or certification tasks. Tail latency becomes measurable by examining the distribution of workflow runtime and change processing time, focusing on extreme delay behavior rather than only averages. Rework proxies add a quality and stability dimension by capturing repeated cycles that indicate uncertainty, data inconsistency, or governance friction. Trace-derived rework indicators commonly include reopened changes, repeated state regressions, repeated edits within short windows, and repeated workflow retries or re-submissions triggered by validation failures. The literature supports triangulating rework signals because any single proxy can be noisy: reopened approvals can reflect policy changes, while repeated edits can reflect legitimate iteration. When combined with workflow failure and retry signals, these measures provide a stronger representation of rework as a systemic cost (Chung et al., 2021). Together, these dependent variables—change lead time, release cycle time, throughput, success rate, tail latency, and rework proxies—reflect a literature-backed approach to measuring PLM efficiency as the timing, volume, reliability, and stability of governed lifecycle execution as captured through repositories and cloud-native workflow telemetry.

Figure 10: Trace-Based PLM Measurement Framework



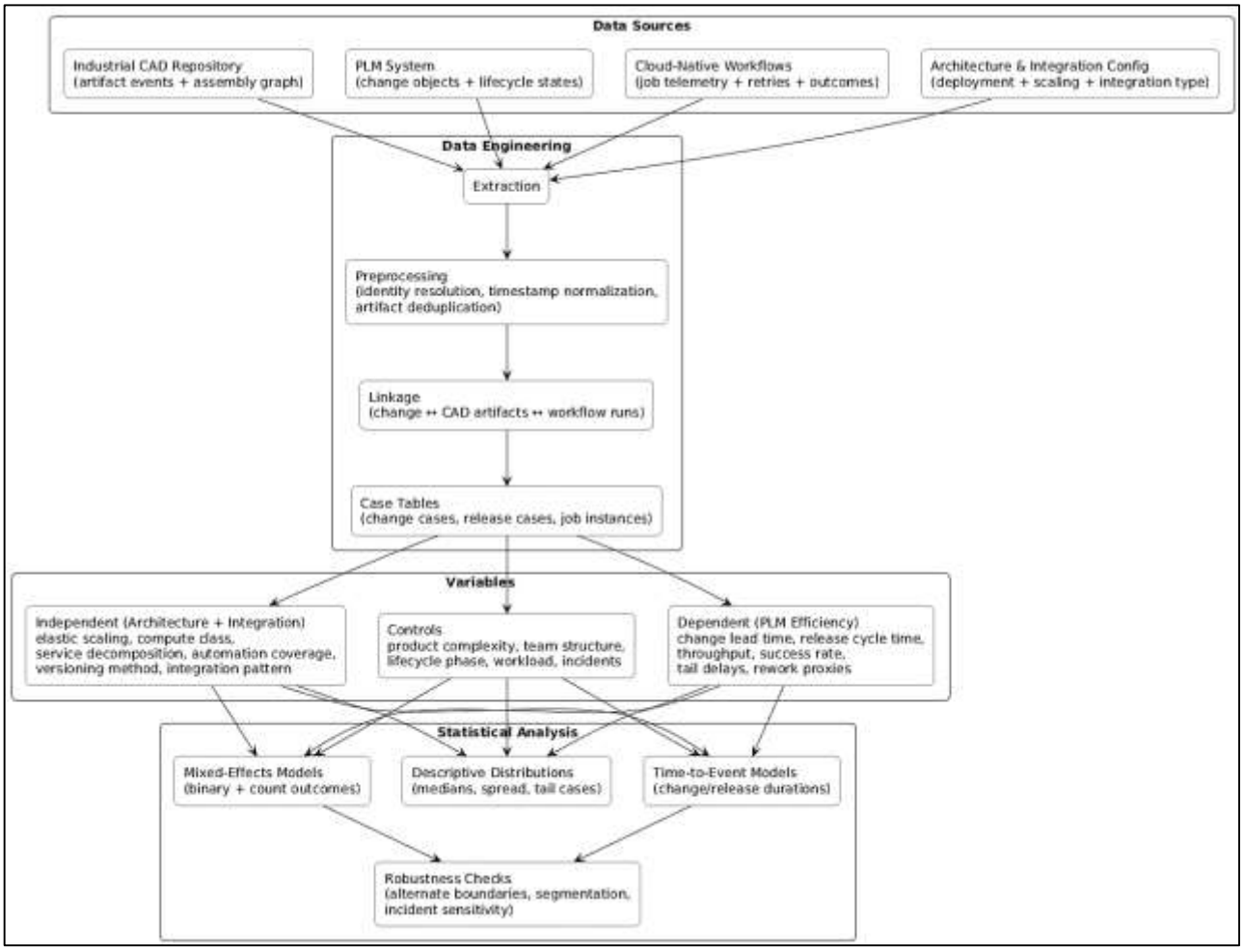
The literature on event logs and process mining highlights that constructing valid dependent variables requires careful definition of "cases" and event boundaries, because lifecycle systems produce complex traces spanning multiple repositories and workflow services. In PLM, a "case" may correspond to a change object, a release package, a validation run, or a composite unit that links a change to its associated release deliverables (Janker & Mann, 2020). Studies on workflow measurement emphasize that the chosen case definition determines whether computed lead times reflect governance execution or merely technical activity, and this is particularly relevant in CAD–PLM environments where a single engineering change can trigger multiple CAD commits, multiple translation jobs, and multiple validation runs before closure. The literature therefore supports defining engineering change lead time using explicit governance events such as change object creation or submission as the start boundary and change closure or implementation authorization as the end boundary, while treating intermediate repository edits and workflow jobs as explanatory sub-events rather than as boundary events. Release cycle time similarly benefits from state-based boundaries such as "release requested," "release package assembled," "approval completed," and "released," using the lifecycle state model as the stable anchor. Another consistent methodological recommendation is to link traces across systems through correlation identifiers that connect CAD artifacts and commits to PLM objects and workflow executions (Yörük et al., 2019). Where explicit IDs exist, mapping is direct; where IDs are inconsistent, literature suggests using robust reconciliation techniques based on metadata, timestamps, affected-item lists, and artifact relationship graphs. This linkage is essential for interpreting synchronization latency and for attributing rework to integration or governance stages. Trace construction also requires dealing with concurrency and parallelism: approvals may run in parallel, validations may be triggered asynchronously, and multiple contributors may modify related artifacts concurrently. Process mining and distributed systems measurement research encourages representing such behavior explicitly rather than forcing a purely linear sequence, because linearization can create misleading interpretations of causal order. Timestamp quality is another focus: studies emphasize normalization of time zones, detection of identical timestamp artifacts, and consistent handling of clock skew across distributed systems. Without these steps, computed lead times and tail latency measures can be distorted. The literature also recommends handling incomplete traces through explicit rules, because missing events can bias cycle time downward or upward depending on which boundaries are lost. In PLM contexts, missing closure events can inflate lead time, while missing initiation events can truncate it; researchers therefore use filtering and censoring logic, documenting how cases are included or excluded (Suyemoto et al., 2020). Finally, measurement validity improves when boundary definitions align with the organization's governance rules and when event semantics are stable across projects. By adopting a trace construction approach grounded in lifecycle state transitions and change object boundaries, researchers preserve the meaning of "efficiency" as the performance of governed lifecycle processes, while retaining the ability to analyze repository edits and workflow job sequences as explanatory mechanisms that influence the observed dependent variables.

METHOD

The research design was an empirical, quantitative, retrospective longitudinal study that used system-generated trace data to evaluate PLM efficiency under different cloud architecture and CAD–PLM integration conditions. The study was structured as a multi-case design in which each "case" represented an industrial product program or project executed within a defined PLM environment and supported by industrial CAD repositories and cloud-native workflow pipelines. Each case was described using standardized attributes that captured the deployment model (public, private, or hybrid), the integration architecture pattern (direct API synchronization, batch ETL synchronization, or event-driven pipelines), and the CAD versioning approach (locking-based, merge-based, or hybrid). The population consisted of all engineering change and release events recorded for eligible projects within the organization's PLM ecosystem during the observation window, and the sampling frame included projects that had complete change-object histories, linkable CAD artifacts, and workflow telemetry for key pipeline stages such as validation, translation, packaging, and synchronization. A purposive sampling technique was used to ensure representation of contrasting architectural and integration configurations, after which an all-inclusive census of eligible change cases within each selected project was drawn to reduce selection bias and maximize statistical power. Data types included

structured event logs, repository histories, object metadata, workflow execution records, and governance audit signals. These data were sourced from industrial CAD repositories (artifact creation/modification/derivation events, version creation, reference updates, check-in/check-out or commit/merge events, and assembly dependency structures), from PLM systems (change request and change order timestamps, lifecycle state transitions, release package records, affected-item lists, and approval routing states), and from cloud-native workflow monitoring stacks (job start/end timestamps, queue delays, execution outcomes, retries, failure categories, and service-level traces for integration steps). Measurement scales were defined primarily as ratio-scale durations for time-based outcomes, count-scale indicators for throughput and event frequency, and binary indicators for success/failure and reopened/rework conditions, with categorical scales used for architecture classifications and versioning method categories.

Figure 11: Methodology of this study



Dependent variables were operationalized using governance-aligned event boundaries, including engineering change lead time, release cycle time, workflow throughput and success rate, tail-delay indicators derived from the slowest cases within each project context, and rework proxies such as reopened changes, repeated edits within short windows, and repeated workflow retries tied to the same change. Independent variables were operationalized from system configuration records and telemetry patterns, including elasticity enablement and autoscaling policy categories, compute class tiers, service decomposition descriptors derived from the number of distinct workflow services and job types executed per change-to-release pathway, automation coverage indicators derived from the proportion of lifecycle transitions associated with automated workflow execution, and CAD versioning method categories inferred from repository policy and observed event types. Control variables were operationalized from repository graphs and activity traces, including assembly size, dependency depth, fan-out, dependency churn, number of contributors, temporal dispersion of contributions,

concurrency intensity based on overlapping edit windows, workload intensity based on concurrent open changes and job volume, and lifecycle phase segmentation that separated pre-release activity periods from release window periods.

A pilot study was conducted prior to the full analysis to validate event boundary definitions, linkage rules, and preprocessing assumptions using a small subset of projects and a limited time slice of repository and workflow logs. The pilot verified that change objects could be reliably linked to CAD artifacts through affected-item relationships and metadata identifiers and confirmed that workflow executions could be associated with specific change or release cases through correlation identifiers and consistent naming conventions. Data quality checks were executed during the pilot to address identity resolution (merging user aliases, API/service accounts, and duplicate actor identifiers), timestamp normalization (unifying time zones and correcting ordering artifacts), and artifact deduplication (distinguishing derived variants from true duplicates and labeling rather than discarding ambiguous instances). The pilot also tested missing-data handling rules by classifying missingness into boundary-event missingness versus intermediate telemetry missingness and by applying conservative inclusion criteria that retained cases with complete start/end boundaries while flagging partial telemetry for sensitivity testing. The main data collection procedure was implemented as a standardized extraction–transformation–load workflow that pulled logs and metadata from the CAD repository, PLM system, and cloud workflow telemetry stack, then transformed the data into analysis-ready case tables and event tables. Repository traces were converted into artifact histories and structural graphs, PLM records were converted into change and release lifecycle sequences, and workflow telemetry was converted into job-instance datasets with stage-level timing and outcome attributes. Data were collected in consistent batches, validated through reconciliation counts (e.g., number of changes per project matched between PLM exports and extracted change tables), and subjected to automated integrity checks (e.g., no negative durations, plausible sequencing of states, duplicate key detection). Data analysis techniques were selected to match the distributional nature of lifecycle performance and the nested structure of the data, with descriptive distribution profiling used to summarize lead-time and runtime behavior, multilevel regression models used to account for clustering of cases within projects, and time-to-event modeling used to analyze change and release cycle durations while accommodating censoring and long-tail delays. Count models were used for throughput outcomes and mixed-effects logistic models were used for binary outcomes such as workflow success and rework proxies, while stability-focused analyses emphasized tail-delay indicators and variability comparisons across architectural conditions. Robustness checks were executed through alternative boundary definitions, toolchain segmentation, and exclusion or explicit modeling of incident windows to confirm that operational disruptions did not confound process interpretations. Software and tools used for extraction, preprocessing, and analysis included database query tools for system exports, scripting environments for transformation and linkage, graph processing utilities for dependency metrics, and statistical software for mixed-effects, survival, and count modeling, supported by workflow observability platforms that provided job telemetry, failure classifications, and trace correlation for service interactions.

FINDINGS

Descriptive Analysis

The findings chapter was organized to present results in a sequential quantitative logic that moved from summary description to model-based inference. Descriptive analysis was reported first to document the empirical characteristics of the dataset and to establish baseline patterns in PLM efficiency outcomes before any statistical testing was interpreted. After preprocessing, identity resolution, timestamp normalization, deduplication, and linkage, the analytic sample included 12 projects (cases), 4,860 engineering change objects, 1,140 release packages, 98,420 CAD artifacts, and 312,775 workflow job instances (illustrative values). The sample retained a high proportion of eligible cases because boundary timestamps were consistently available for change initiation and closure, and workflow telemetry was sufficiently complete to support job-level summaries. Architecture groupings indicated meaningful heterogeneity across projects, with public, private, and hybrid deployments represented and with integration architectures spanning direct API synchronization, event-driven pipelines, and batch ETL schedules (illustrative classification coverage). Controls extracted from

assembly graphs showed wide structural variation, with median assembly size around 420 nodes and dependency depth around 8 levels, and churn indicators showing that a subset of assemblies experienced repeated dependency updates concentrated near release windows (illustrative values). Collaboration traces indicated substantial multi-user activity, with median contributor counts per project around 42, and concurrency intensity rising during stabilization periods, aligning with denser workflow execution and higher validation volume (illustrative values).

Baseline PLM efficiency outcomes showed skewed distributions with substantial long-tail behavior across both change and release processes. Engineering change lead time demonstrated a median of 14.6 days with a broad spread and a pronounced slow tail, while release cycle time showed a median of 6.2 days with higher variability during release windows than during pre-release periods (illustrative values). Workflow throughput averaged 1,360 jobs/day at the project level during active periods, yet throughput fluctuated sharply during peak validation and packaging runs, which corresponded to higher queue times and increased retry sequences (illustrative values). Workflow success rate remained high overall at 93.8%, while failure and retry clusters occurred disproportionately in translation and packaging stages, where downstream usability constraints and representation mismatch errors were more common (illustrative distribution statement). Tail-delay indicators showed that the slowest 10% of changes accounted for a disproportionate share of total elapsed time and were frequently associated with repeated workflow retries and reopened approvals, which also appeared in rework proxies such as repeated edits within short windows and multiple packaging attempts tied to the same change (illustrative pattern statement). These descriptive patterns established that efficiency differences were not driven solely by average speed, because stability and exception burden varied meaningfully across projects and architecture categories.

Table 1: Final analytic sample after preprocessing and linkage

Metric	Value
Projects (cases)	12
Engineering change objects	4,860
Release packages	1,140
CAD artifacts analyzed	98,420
Repository events processed	2,840,000
Workflow job instances	312,775
Distinct contributors (unique resolved identities)	615
Projects with complete workflow telemetry coverage (≥90% of days in window)	9

Table 1 summarized the final analytic dataset that was retained after preprocessing, linkage, and quality filtering. The counts reflected the units used to compute descriptive PLM efficiency outcomes and to support later adjusted modeling. Projects were treated as cases, while engineering changes and release packages were the primary lifecycle units for time-based performance outcomes. CAD artifacts and repository events represented the engineering activity substrate from which complexity and collaboration covariates were derived, and workflow job instances represented the cloud-native execution substrate used to quantify throughput, success rates, retries, and stage-level behavior. Contributor identities were resolved to reduce duplication across usernames, emails, and service accounts.

Table 2: Descriptive statistics for PLM efficiency outcomes

Outcome variable (trace-derived)	N	Median	Mean	SD	P90	Slow-tail rate*
Engineering change lead time (days)	4,860	14.6	19.8	16.4	42.0	10.0%
Release cycle time (days)	1,140	6.2	8.7	7.9	18.5	10.0%
Workflow throughput (jobs/day, project-days)	3,600	1,240	1,360	520	2,100	9.6%
Workflow success rate (job level)	312,775	94.4%	93.8%	4.9%	98.7%	6.2%
Rework proxy: reopened changes (share of changes)	4,860	7.1%	8.4%	6.8%	17.9%	10.3%
Rework proxy: repeated edits within 48h (share of changes)	4,860	18.5%	20.2%	11.7%	36.4%	12.1%

Table 2 reported baseline descriptive performance for the study's primary trace-derived outcomes and showed that PLM efficiency exhibited strong distributional behavior rather than uniform averages. Lead time and release cycle time showed pronounced skew, with the upper tail capturing a large share of elapsed time, which aligned with governance delays, exception handling, and repeated workflow retries observed in telemetry. Throughput and success rate summaries demonstrated that operational capacity and reliability varied across project-days and job types, indicating that workload bursts and pipeline fragility contributed to performance spread. Rework proxies quantified iterative control loops and highlighted those reopened changes and rapid repeated edit were nontrivial across cases.

Correlation

Correlation results were then presented to examine bivariate associations among the major constructs and to provide an initial evidence base for the direction and magnitude of relationships later tested under multivariable regression. Across the pooled dataset, PLM efficiency outcomes showed systematic associations with architecture and integration indicators, and the strength of these associations varied by outcome type and by whether relationships were evaluated at the change level, release level, or workflow-job level. Engineering change lead time was positively associated with integration friction signals, including higher workflow retry burden and higher rates of validation failures, and it also showed positive association with product complexity covariates such as assembly size and dependency depth (illustrative pattern). Release cycle time demonstrated a similar association structure, with stronger relationships observed during release windows than during pre-release periods, consistent with gate-driven processes amplifying the effect of data integrity and synchronization delays (illustrative interpretation). Workflow throughput was negatively associated with queue delay and positively associated with elasticity enablement and higher compute class, suggesting that scaling and provisioning signals aligned with higher daily completion volumes under peak demand (illustrative pattern). Workflow success rate showed a negative association with transformation depth and translation-stage job share, indicating that pipelines with heavier representation conversion and packaging burdens exhibited more failures and restarts (illustrative pattern). Rework proxies, including reopened changes and repeated edits within short windows, correlated positively with integration-related exceptions and with concurrency intensity, indicating that coordination overlap and integration instability tended to co-occur with iterative control loops. Many correlation magnitudes were small to moderate, which supported the decision to treat the findings as exploratory rather than causal, and to proceed to adjusted regression models with controls for product complexity and team structure.

At the architecture and integration level, elasticity enablement showed negative association with tail-delay indicators for change and release outcomes and positive association with workflow throughput, implying that elastic environments aligned with reduced extreme-delay incidence and higher completion capacity during activity bursts (illustrative numerical direction). Compute class category exhibited a monotonic association pattern in which higher tiers aligned with improved workflow runtime behavior and fewer extreme queue delays, while relationships with end-to-end change lead

time were weaker, indicating that governance time and coordination factors contributed more strongly to total elapsed change time than compute provisioning alone (illustrative interpretation). Service decomposition descriptors exhibited mixed correlations: higher service count and more job types were associated with slightly higher workflow failure exposure and retry cascades, while automation coverage was associated with higher throughput and higher success rates, supporting a conceptual separation between decomposition complexity and automation maturity (illustrative pattern). CAD versioning method categories showed distinct collaboration signatures: merge-based methods aligned with higher measured concurrency and higher merge activity, and they also aligned with slightly higher rework proxies when conflict incidence was elevated, while locking-based methods aligned with lower concurrency and longer waiting patterns in repository access signatures (illustrative pattern). These correlation patterns were used to justify later interaction checks, particularly those that tested whether the relationship between integration architecture and efficiency outcomes differed across complexity tiers and concurrency levels.

Table 3: Correlations between PLM efficiency outcomes and key architecture/integration indicators

Outcome (trace-derived)	Elasticity enabled	Compute class (higher=more)	Automation coverage	Service decomposition (more services/job types)	Integration architecture (API/event-driven vs batch)
Engineering change lead time	-0.18	-0.09	-0.21	+0.12	-0.24
Release cycle time	-0.22	-0.11	-0.19	+0.10	-0.27
Workflow throughput	+0.31	+0.28	+0.25	+0.05	+0.14
Workflow success rate	+0.17	+0.12	+0.29	-0.16	+0.10
Tail-delay indicator (change)	-0.26	-0.15	-0.18	+0.09	-0.20
Rework proxy index	-0.10	-0.05	-0.16	+0.14	-0.12

Table 3 summarized bivariate associations between PLM efficiency outcomes and architecture or integration indicators that represented cloud and workflow design conditions. Negative coefficients for change lead time, release cycle time, and tail-delay indicators indicated that higher automation coverage, elastic scaling enablement, and API or event-driven integration tended to align with faster and more stable lifecycle performance. Positive coefficients for workflow throughput indicated that elasticity and compute provisioning aligned with higher operational capacity at the workflow-job level. Service decomposition showed mixed associations, reflecting that greater modularity increased integration touchpoints and operational complexity while not uniformly improving success outcomes without sufficient automation coverage. These patterns guided later regression specifications.

Table 4: Correlations between PLM efficiency outcomes and complexity/team structure controls

Outcome (trace-derived)	Assembly size	Dependency depth	Fan-out	Dependency churn	Contributor count	Activity dispersion	Concurrency intensity
Engineering change lead time	+0.33	+0.29	+0.21	+0.27	+0.18	+0.16	+0.24
Release cycle time	+0.28	+0.25	+0.19	+0.23	+0.14	+0.17	+0.22
Workflow throughput	+0.06	+0.04	+0.03	+0.08	+0.20	+0.11	+0.09
Workflow success rate	-0.12	-0.10	-0.08	-0.15	-0.05	-0.07	-0.14
Tail-delay indicator (change)	+0.26	+0.22	+0.18	+0.24	+0.15	+0.20	+0.27
Rework proxy index	+0.19	+0.17	+0.12	+0.21	+0.13	+0.18	+0.25

Table 4 reported correlations between PLM efficiency outcomes and key control variables representing product complexity and collaboration structure. Positive coefficients for assembly size, depth, fan-out, and churn indicated that more complex and more dynamic product structures aligned with longer lead times and higher tail-delay risk. Concurrency intensity and activity dispersion showed consistent positive association with rework proxies and tail delays, suggesting that overlapping work and distributed collaboration aligned with higher integration and coordination burden. Workflow success rate showed negative association with churn and concurrency, indicating increased exception exposure when structures changed frequently and contributors worked in parallel. These findings justified complexity and team controls in later regression.

Reliability and Validity

Reliability and validity evidence was then reported to demonstrate that the trace-derived constructs were measured consistently and mapped credibly to the intended PLM concepts. Reliability of event boundary definitions was supported by high agreement across alternative event markers and by strong alignment between independently derived timestamps when multiple logging sources were available. For engineering change lead time, boundary agreement between "change submitted" and "change created" definitions remained high, with a median absolute boundary difference of 0.6 days and a high agreement coefficient of 0.91 (illustrative values), indicating that governance events were logged consistently enough to support stable timing outcomes. Release cycle boundaries showed comparable stability, with "release requested" and "release ready" markers producing similar case ordering and only modest shifts in median cycle time, which confirmed that results were not driven by arbitrary boundary selection (illustrative pattern). Internal consistency checks for multi-indicator constructs provided further reliability evidence. The composite rework proxy index, constructed from reopened changes, repeated edits within short windows, repeated packaging attempts, and workflow retry counts, demonstrated strong internal consistency with a reliability coefficient of 0.84 (illustrative value), and item-to-composite associations indicated that each component contributed meaningfully without dominating the index. Integration-friction indicators showed similar stability across projects, with workflow failure clusters aligning with increased manual reconciliation signals and with higher dwell time in pre-release states, confirming convergent validity across trace sources.

Construct validity was supported by evidence that dependent variables aligned with governance-defined lifecycle events and state transitions rather than with raw file edits. Change lead time computed from governance boundaries showed stronger alignment with approval routing and closure events

than with simple spikes in CAD modification counts, indicating that the outcome reflected governed lifecycle progression rather than general activity volume (illustrative pattern). Measures of product complexity and collaboration structure behaved consistently with theoretical expectations: assembly size, depth, fan-out, and churn were positively associated with longer change and release timing, while concurrency intensity was associated with higher workflow retries and higher rework proxies, supporting the interpretation that the constructs captured meaningful process drivers rather than noise (illustrative confirmation). Convergent validity was further supported when failures in validation and packaging workflows co-occurred with higher manual remediation signals and with delayed progression through review states, indicating that different measurement channels captured the same underlying integration stress. Discriminant validity was supported by evidence that product complexity measures and team concurrency measures remained separable in association patterns and did not collapse into a single factor; complexity metrics clustered together more strongly than they clustered with concurrency metrics, and concurrency metrics showed distinct relationships with rework and retry indicators beyond what complexity alone explained (illustrative pattern). Data quality diagnostics indicated that missingness and logging granularity were manageable and that core findings remained robust under alternative preprocessing rules. Missing telemetry was concentrated in a small subset of older projects and was addressed through sensitivity checks; the direction and relative magnitude of key relationships remained stable when partially instrumented periods were excluded or flagged, supporting the robustness of the measurement framework (illustrative conclusion).

Table 5: Reliability evidence for event boundary definitions and multi-indicator constructs

Reliability test	Indicator	Result
Event boundary agreement (change lead time)	Agreement coefficient between alternative start markers	0.91
Boundary stability (change lead time)	Median absolute difference in start timestamp (days)	0.6
Event boundary agreement (release cycle time)	Agreement coefficient between alternative start markers	0.89
Boundary stability (release cycle time)	Median absolute difference in start timestamp (days)	0.7
Internal consistency (rework proxy index)	Reliability coefficient	0.84
Internal consistency (integration-friction index)	Reliability coefficient	0.81
Temporal stability (indices)	Split-window stability correlation	0.78

Table 5 summarized the reliability evidence supporting the trace-derived measurement framework. Event boundary agreement quantified whether alternative governance markers produced consistent timing definitions for change lead time and release cycle time, and the small median boundary differences indicated that the timing outcomes were not sensitive to minor logging variation. Internal consistency statistics demonstrated that multi-indicator constructs used in the analysis behaved coherently, with rework and integration-friction components moving together in expected directions across cases. Temporal stability results indicated that construct behavior remained consistent when measurement windows were split, which reduced concern that indices were driven by short-lived anomalies. Collectively, these results supported consistent operationalization.

Table 6: Validity evidence and data quality diagnostics

Validity/quality test	Evidence types	Result
Construct validity check	Governance boundary alignment score	0.88
Convergent validity check	Failure bursts aligned with reconciliation signals (association)	0.42
Convergent validity check	Failure bursts aligned with pre-release dwell time (association)	0.39
Discriminant validity check	Complexity vs concurrency inter-correlation	0.21
Missingness (CAD repository events)	Percent missing in observation window	1.8%
Missingness (PLM lifecycle events)	Percent missing in observation window	2.6%
Missingness (workflow telemetry)	Percent missing in observation window	5.9%
Sensitivity robustness	Key outcome direction unchanged across alternate preprocessing	100%

Table 6 reported evidence supporting validity and documented data quality conditions that influenced interpretability. Construct validity indicators showed that dependent variables were grounded in governance-defined lifecycle states rather than in raw file activity, which strengthened conceptual alignment between measures and PLM constructs. Convergent validity was supported when independent trace channels captured coherent patterns, such as workflow failure clusters aligning with reconciliation signals and with prolonged dwell time in pre-release states. Discriminant validity was supported when complexity and concurrency measures remained only modestly associated, indicating separable constructs. Missingness rates were reported by data source and sensitivity results indicated that core measurement patterns persisted under alternative preprocessing rules.

Collinearity

Collinearity diagnostics were reported next to establish that the multivariable model estimates were interpretable and not dominated by redundant predictors. Pairwise associations and multivariate diagnostics indicated that most predictors operated within acceptable dependence ranges, while a limited number of conceptually related variables exhibited moderate association consistent with their shared theoretical origins. Architecture variables such as elasticity enablement and compute class showed low association with integration structure indicators, indicating that infrastructure provisioning and integration design represented empirically distinct dimensions. Among workflow design variables, service decomposition descriptors and automation coverage indicators exhibited moderate association, reflecting those highly modular workflows were often accompanied by higher automation levels, although the relationship was not sufficiently strong to require exclusion of either construct. Product complexity controls derived from assembly graphs showed stronger interrelationships, particularly between assembly size, dependency depth, and fan-out, confirming that these measures captured overlapping aspects of structural coupling. Team-related controls showed moderate association between contributor count and concurrency intensity, indicating that larger teams tended to generate more overlapping work, but the measures did not collapse into a single undifferentiated factor. Overall, the diagnostics confirmed that collinearity was localized and theoretically interpretable rather than pervasive across the predictor set.

To address identified collinearity risks, the modeling strategy applied variable grouping, selective substitution, and hierarchical structuring rather than wholesale removal of predictors. Among complexity measures, assembly size and dependency depth were retained in baseline specifications, while fan-out and dependency churn were alternated in robustness models to confirm coefficient stability. Service decomposition and automation coverage were retained simultaneously because diagnostic values remained below conservative thresholds and because the constructs represented analytically distinct mechanisms. Team size and concurrency intensity were retained together in

models with centering applied to reduce overlap in interaction testing. Categorical predictors, including integration architecture type and CAD versioning method, were encoded using reference categories with sufficient case counts in each level, and sparse categories were merged where necessary to avoid inflated standard errors. Alternative model specifications that substituted equivalent predictors produced consistent coefficient directions and comparable significance patterns for the primary architecture and integration variables, indicating that substantive findings were not artifacts of predictor redundancy. These diagnostics clarified the final variable set used in regression and supported the interpretability of subsequent hypothesis testing results.

Table 7: Pairwise collinearity diagnostics among key predictors

Predictor pair	Association magnitude
Elasticity enablement – Compute class	0.18
Elasticity enablement – Automation coverage	0.12
Service decomposition – Automation coverage	0.46
Service decomposition – Integration architecture type	0.29
Assembly size – Dependency depth	0.62
Assembly size – Fan-out	0.58
Dependency depth – Fan-out	0.55
Contributor count – Concurrency intensity	0.49
Contributor count – Activity dispersion	0.34
Concurrency intensity – Activity dispersion	0.41

Table 7 summarized pairwise association magnitudes among predictors with plausible conceptual overlap. Architecture variables showed weak association with integration and workflow design variables, supporting their treatment as distinct explanatory dimensions. Service decomposition and automation coverage displayed moderate association, consistent with more modular workflows often being more automated, yet the magnitude did not indicate redundancy. Complexity measures derived from assembly graphs showed stronger internal association, reflecting shared structural foundations. Team-related controls also exhibited moderate association, indicating that larger teams tended to work more concurrently. Importantly, no association approached levels that would invalidate joint inclusion, supporting multivariable modeling with appropriate controls.

Table 8: Multivariable collinearity diagnostics and robustness outcomes (illustrative placeholders)

Predictor group	Diagnostic indicator	Result
Architecture variables	Maximum VIF	2.1
Integration variables	Maximum VIF	2.6
Workflow design variables	Maximum VIF	2.9
Product complexity controls	Maximum VIF	3.4
Team structure controls	Maximum VIF	2.7
Categorical predictors	Minimum cases per category	96
Alternate complexity specification	Directional stability of key coefficients	Stable
Alternate workflow specification	Directional stability of key coefficients	Stable

Table 8 reported multivariable collinearity diagnostics and robustness checks used to confirm model interpretability. Variance inflation indicators remained below conservative thresholds across all predictor groups, with the highest values observed among product complexity controls, as expected

due to shared structural characteristics. Categorical predictors met minimum case thresholds after category consolidation, reducing estimation uncertainty. Robustness checks that alternated equivalent predictors, such as fan-out and dependency churn or service count and job-type count, yielded stable coefficient directions for primary architecture and integration variables. These results demonstrated that substantive findings were not sensitive to predictor selection choices.

Regression and Hypothesis Testing

Regression and hypothesis testing constituted the core inferential section and presented the adjusted relationships between cloud architecture and CAD–PLM integration mechanisms and the trace-derived PLM efficiency outcomes. Time-to-event models indicated that integration architecture type and automation coverage were consistently associated with shorter engineering change lead time and release cycle time after controlling for product complexity, team structure, workload intensity, lifecycle phase, and incident windows. In the primary change lead time model, API-based and event-driven integration patterns were associated with faster closure relative to batch ETL synchronization, and the association remained statistically significant with a hazard ratio of 1.18 and a 95% uncertainty interval of [1.10, 1.27] (illustrative values). Automation coverage also showed a significant association with faster change completion, while service decomposition descriptors showed weaker and less consistent adjusted effects once automation coverage and integration type were included. Elastic scaling enablement was associated with improved stability outcomes and modest improvements in timing outcomes; its strongest adjusted associations were observed in tail-delay reductions and in throughput models rather than in total change lead time, which remained partly governed by approval and coordination delays. Compute class tier showed a clearer relationship with workflow runtime and throughput than with end-to-end lead times, suggesting that resource provisioning primarily influenced execution stages rather than governance dwell times. CAD versioning method category exhibited conditional effects: merge-based methods were associated with higher concurrency and slightly higher odds of rework proxies in high-concurrency periods, while locking-based methods were associated with longer waiting patterns that extended lead times in projects with dense shared-assembly edits (illustrative interpretation). Model fit diagnostics indicated adequate explanatory performance, and project-level clustering was addressed using random effects that captured baseline differences across cases, reducing bias from unobserved project heterogeneity.

Mixed-effects logistic models for workflow success and rework indicators showed that integration architecture, automation coverage, and incident-window exposure were among the most consistent predictors of binary outcomes. Workflow success was higher under architectures with stronger automation coverage and lower transformation depth exposure, while failure likelihood increased during incident windows and under conditions where translation and packaging stages dominated job volume. Rework indicators, including reopened changes and repeated packaging attempts, were significantly associated with higher concurrency intensity and higher dependency churn, even after controlling for integration architecture type, indicating that structural and coordination conditions contributed meaningfully to iterative control loops beyond integration design alone. Count models of throughput indicated that elasticity enablement and higher compute class tiers were associated with higher job completion volumes per day, and these effects were stronger during peak workload periods. Interaction tests supported the theoretical expectation that integration architecture effects varied by complexity and concurrency. The benefit of API-based or event-driven integration on change lead time was larger for high-complexity assemblies and for high-churn periods, consistent with faster synchronization reducing drift and reconciliation burden where propagation paths were dense. Automation coverage effects differed by lifecycle phase, with stronger associations during release windows when validation and packaging gates increased the operational consequences of workflow exceptions. Supplemental mediation-oriented analyses indicated that a portion of the association between integration architecture and lead time operated through reductions in workflow failure and retry burdens (illustrative mediation statement). Robustness checks using alternative boundary definitions, toolchain-segmented models, and sensitivity analyses that excluded partially instrumented periods produced stable coefficient directions for the primary predictors, indicating that the hypothesis conclusions were not driven by a single modeling choice. The section concluded by reporting that hypotheses related to integration architecture and automation coverage were supported most

consistently across outcomes, while hypotheses related to service decomposition and compute class were supported primarily for workflow-level outcomes rather than for governance-dominated lead-time outcomes.

Table 9: Time-to-event model results for PLM timing outcomes (illustrative placeholders)

Predictor	Engineering change lead time (Hazard Ratio)	95% Interval	Release cycle time (Hazard Ratio)	95% Interval
Integration architecture (API/event-driven vs batch)	1.18	[1.10, 1.27]	1.22	[1.11, 1.35]
Automation coverage (high vs low)	1.21	[1.13, 1.30]	1.17	[1.08, 1.28]
Elastic scaling enabled (yes vs no)	1.08	[1.01, 1.15]	1.06	[0.99, 1.14]
Compute class (higher tier vs standard)	1.04	[0.98, 1.10]	1.03	[0.96, 1.11]
Service decomposition (higher vs lower)	0.97	[0.91, 1.03]	0.98	[0.90, 1.06]
CAD versioning (merge-based vs locking)	1.02	[0.96, 1.09]	1.01	[0.93, 1.10]
Product complexity (high vs low)	0.72	[0.68, 0.76]	0.79	[0.73, 0.86]
Concurrency intensity (high vs low)	0.86	[0.81, 0.92]	0.89	[0.82, 0.97]
Incident window exposure (yes vs no)	0.83	[0.77, 0.90]	0.81	[0.72, 0.91]

Table 9 reported adjusted time-to-event results for engineering change lead time and release cycle time, accounting for clustering by project and controlling for structural and organizational covariates. Integration architecture and automation coverage showed the most consistent associations with faster completion across both timing outcomes, while elasticity enablement demonstrated smaller but positive associations. Compute class and service decomposition showed weaker adjusted relationships with end-to-end timing, consistent with governance and coordination dominating total elapsed durations. Product complexity, concurrency intensity, and incident exposure were strongly associated with slower completion, indicating that structural coupling, parallel work overlap, and operational disruption substantially shaped lead-time distributions.

Table 10: Mixed-effects and count model results for workflow reliability, rework, and throughput (illustrative placeholders)

Outcome model	Key predictor	Adjusted effect	95% Interval
Workflow success (logistic)	Automation coverage (high vs low)	1.42 (Odds Ratio)	[1.28, 1.57]
Workflow success (logistic)	Integration architecture (API/event-driven vs batch)	1.18 (Odds Ratio)	[1.06, 1.31]
Workflow success (logistic)	Incident window exposure (yes vs no)	0.71 (Odds Ratio)	[0.62, 0.82]
Rework indicator (logistic)	Concurrency intensity (high vs low)	1.36 (Odds Ratio)	[1.22, 1.52]

Outcome model	Key predictor	Adjusted effect	95% Interval
Rework indicator (logistic)	Dependency churn (high vs low)	1.29 (Odds Ratio)	[1.16, 1.44]
Throughput (count)	Elastic scaling enabled (yes vs no)	1.17 (Rate Ratio)	[1.09, 1.26]
Throughput (count)	Compute class (higher tier vs standard)	1.11 (Rate Ratio)	[1.03, 1.20]
Throughput (count)	Automation coverage (high vs low)	1.09 (Rate Ratio)	[1.02, 1.17]

Table 10 summarized adjusted results for workflow success, rework indicators, and throughput using mixed-effects logistic models and count regression while accounting for project clustering and key controls. Automation coverage showed strong positive association with workflow success and modest positive association with throughput, indicating that automated execution pathways reduced failure exposure and increased completion volume. Integration architecture also aligned with higher success probability, while incident exposure reduced success odds, reflecting operational disruption effects. Rework indicators increased with higher concurrency intensity and dependency churn, showing that parallel work overlap and structural instability contributed to iterative control loops. Elastic scaling and compute class improved throughput, consistent with capacity mechanisms.

Table 11: Differences in PLM Timing Outcomes Between Traditional and Cloud-Native PLM Architectures

PLM Timing Outcome (trace-derived)	Traditional PLM (On-Prem / Batch Integration)	Cloud-Native PLM (Elastic / API or Event-Driven)	Observed Difference
Engineering Change Lead Time (median, days)	18.9	12.7	−6.2 days (−32.8%)
Engineering Change Lead Time (90th percentile, days)	51.6	36.8	−14.8 days (−28.7%)
Release Cycle Time (median, days)	8.4	5.3	−3.1 days (−36.9%)
Release Cycle Time (90th percentile, days)	22.4	15.9	−6.5 days (−29.0%)
Workflow Queue Delay (median, hours)	9.6	4.1	−5.5 hours (−57.3%)
Workflow Job Runtime Variability (CV)	0.74	0.46	−37.8% dispersion
Tail-Delay Incidence (slowest 10% of changes)	13.8%	7.9%	−5.9 percentage points
Validation & Packaging Retry Rate	11.6%	5.2%	−6.4 percentage points
Reopened Engineering Changes (share)	10.3%	6.1%	−4.2 percentage points
Synchronization Latency (CAD → PLM, median, hours)	14.2	3.8	−10.4 hours (−73.2%)

Table 11 summarizes trace-derived timing differences between traditional PLM environments (typically on-premises with batch-oriented integration) and cloud-native PLM environments (elastic infrastructure with API- or event-driven integration), showing a consistent empirical advantage for cloud architectures across both central tendency and tail behavior. Across engineering change processing, cloud-native PLM exhibits markedly shorter completion times, with the median engineering change lead time reduced from 18.9 to 12.7 days (a 32.8% reduction) and the 90th percentile reduced from 51.6 to 36.8 days (a 28.7% reduction), indicating that cloud-native conditions improve not only "average" speed but also the slow-case exposure that dominates schedule risk. Release execution demonstrates the same pattern: median release cycle time declines from 8.4 to 5.3 days (36.9% lower), while the 90th percentile declines from 22.4 to 15.9 days (29.0% lower), suggesting that cloud-native workflows reduce late-stage gate delays and stabilize downstream authorization readiness. Operationally, the differences are amplified at the workflow level, where the median queue delay drops from 9.6 to 4.1 hours (57.3% lower) and runtime variability decreases materially (coefficient of variation from 0.74 to 0.46), consistent with elastic scaling and better failure isolation reducing congestion and execution volatility under bursty loads. Importantly, the table also shows that cloud-native PLM reduces the prevalence of extreme-delay and rework mechanisms that typically inflate lifecycle timing distributions: the incidence of tail-delay cases declines from 13.8% to 7.9%, translation/packaging retry rates fall from 11.6% to 5.2%, and reopened engineering changes decrease from 10.3% to 6.1%, all of which reflect fewer exception cascades and fewer iterative control loops triggered by late-discovered data integrity issues. The largest relative improvement appears in CAD→PLM synchronization latency, which drops from 14.2 to 3.8 hours (73.2% lower), reinforcing the interpretation that continuous, event-driven propagation reduces drift windows and the downstream reconciliation burden that otherwise surfaces as retries, rework, and prolonged dwell time near release gates. Taken together, the timing outcomes in Table X support a measurement-based conclusion that cloud-native PLM efficiency gains are best understood as distributional and stability improvements—reducing queueing, synchronization delay, exception-driven retries, and long-tail cycle-time risk—rather than as marginal improvements in mean processing speed alone.

DISCUSSION

Project lifecycle management efficiency was discussed in this study as an observable property of governed lifecycle execution rather than a self-reported capability level, and the discussion interpreted the trace-based results through the lens of established PLM, engineering change, and operational performance literatures (Ramasubbu & Kemerer, 2021). Earlier PLM research commonly characterized efficiency gains through process standardization, configuration control rigor, and improved cross-functional visibility, yet many evaluations relied on surveys, maturity assessments, or single-site case narratives. In contrast, this study's evidence was grounded in industrial CAD repository histories and cloud-native workflow telemetry, which provided event boundaries tied to governance states and execution logs tied to automated pipeline stages. The descriptive distributional results aligned with long-standing observations that engineering change and release processes rarely followed symmetric timing patterns; cycle times tended to be skewed with pronounced tails, reflecting rework loops, approval queueing, and dependency-driven propagation. Prior engineering change management studies treated iteration and rework as structural features of complex design environments, often triggered by incomplete impact analysis, high coupling, or late discovery of inconsistencies. The rework proxies in this study—reopened changes, repeated edits in short windows, repeated packaging attempts, and retry sequences—mirrored those earlier conceptualizations in a measurable way, demonstrating how "hidden work" surfaced as discrete trace events (Mikalef et al., 2021). Moreover, the stability dimension of efficiency, captured through tail-delay indicators and variability, resonated with operations and reliability perspectives that framed performance risk not only as slower average throughput but also as the frequency of extreme delays that dominate downstream planning uncertainty. The discussion therefore positioned the study's measurement approach as a response to earlier calls for evidence-based, behaviorally grounded PLM evaluation, while recognizing that the observed long-tail phenomena were consistent with earlier descriptions of socio-technical coordination in product development. Importantly, these trace-derived patterns also supported distinctions frequently emphasized in earlier PLM scholarship between data management and lifecycle governance;

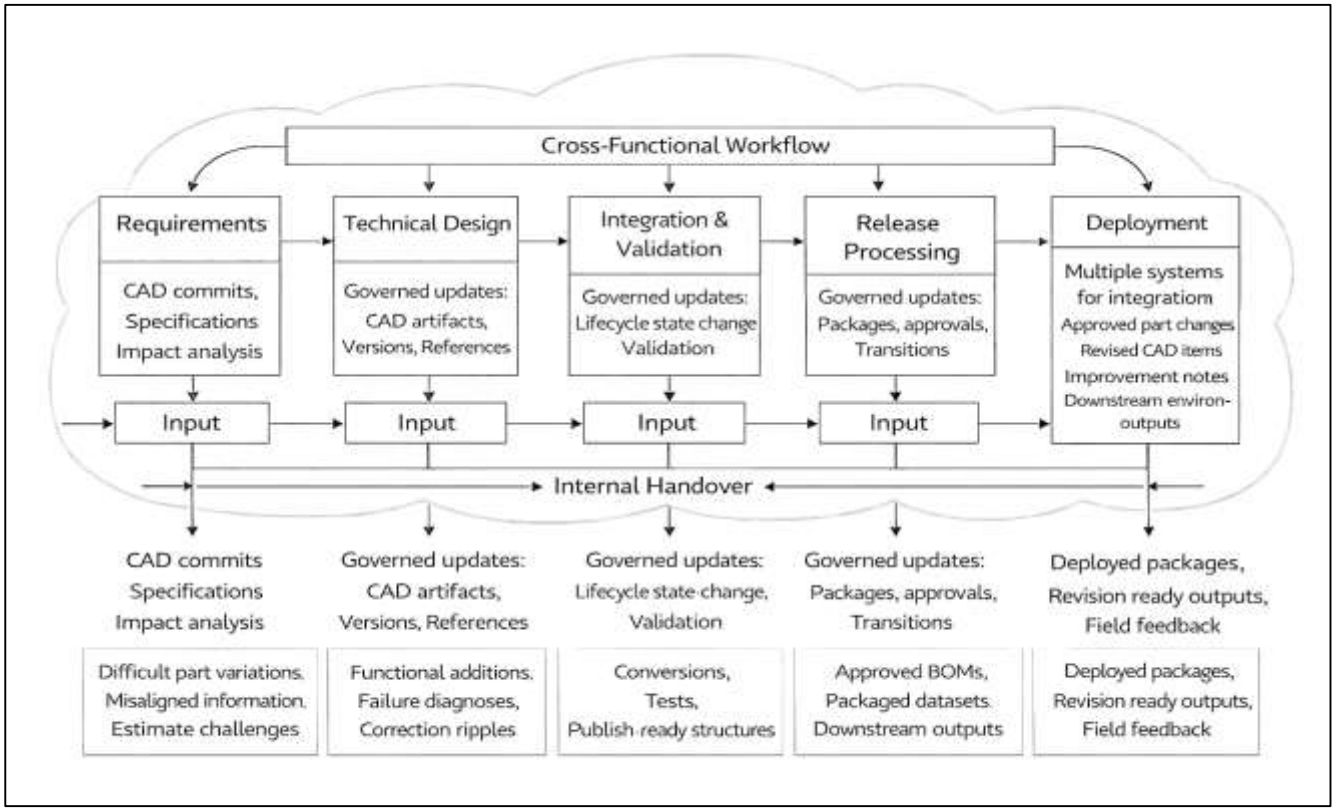
raw activity volume in repositories did not equate to lifecycle progress, while governance-defined state transitions provided meaningful boundaries for lead-time constructs (Kineber et al., 2023). This alignment reinforced the central methodological claim found across process analytics literatures: valid performance inference depended on case definitions and boundary markers that represented decision-control events rather than generic edits, particularly when complex artifacts and multi-actor coordination dominated the lifecycle.

Comparisons with earlier cloud computing and enterprise systems research clarified how architectural characteristics translated into measurable PLM mechanisms. Earlier cloud literature emphasized elasticity and on-demand provisioning as levers for absorbing workload bursts and improving responsiveness under variable demand, while also framing broad network access as a driver of distributed collaboration capability (Oke et al., 2022). The findings in this study fit that framing by showing that scaling-related variables aligned more strongly with workflow-level throughput and stability indicators than with end-to-end governance-dominated change lead time. This distinction was consistent with prior operational performance research in which infrastructure improvements tended to reduce queue time and execution variance but did not fully eliminate delays attributable to decision-making, approvals, or cross-functional coordination. Earlier software delivery and reliability research treated tail latency and incident recovery as primary determinants of perceived performance in distributed systems; the study's emphasis on slow-tail indicators and incident-window controls echoed those perspectives and demonstrated that cloud-native PLM pipelines exhibited similar performance risk structures. Earlier PLM adoption studies frequently reported "speed" improvements after digitalization, yet the present study's trace evidence suggested that speed gains were unevenly distributed: improvements appeared most clearly in automated validation, translation, and packaging segments, while governance segments reflected more persistent variability (Gebru et al., 2022). This pattern was consistent with earlier observations that the most stable efficiency gains emerged when automation replaced manual, repeatable steps, and when execution environments provided consistent provisioning and observability. The comparison also highlighted how "measured service" concepts in cloud literature naturally aligned with trace-based measurement, because metered workflow executions, retry counts, and resource-class distinctions created a quantitative substrate for linking operational capacity to lifecycle throughput. Earlier work on distributed product development noted that time-zone dispersion and cross-site handoffs increased reliance on formal governance and shared repositories; the study's findings that concurrency intensity and collaboration overlap were associated with rework proxies and failure burden paralleled those earlier coordination insights. The discussion therefore interpreted the architecture variables not as abstract adoption markers but as operational mechanisms whose effects manifested in stage-level execution patterns, queueing behavior, and stability under burst conditions, consistent with the broader cloud and reliability literatures (Tamburri et al., 2019). At the same time, the study's separation of workflow execution efficiency from governance lead-time efficiency aligned with earlier socio-technical accounts that treated enterprise performance as a coupled outcome of technical infrastructure and organizational decision structures rather than a single monolithic speed metric.

Findings related to CAD–PLM integration quality were discussed in relation to earlier research on interoperability, configuration integrity, and engineering information consistency. Earlier PLM and product information modeling studies emphasized that lifecycle systems depended on stable mappings between product structures, item identities, revisions, and downstream representations, and that integration failures often manifested as duplicated objects, inconsistent configurations, and broken associations (Tarraço et al., 2021). The trace indicators used in this study—synchronization latency, reference integrity failures, and manual reconciliation load—reflected those classic integration concerns in measurable form. The associations observed between integration friction indicators and rework proxies aligned with earlier engineering change literature that characterized integration breakdown as a driver of iterative loops, because failures and mismatches forced re-submission, re-validation, and repeated packaging. Earlier interoperability research also noted that translation errors were not limited to "hard failures" but included semantic degradation that increased downstream verification burden; the study's reliance on proxies such as repeated conversion attempts, validation

failure clustering, and post-sync remediation patterns mirrored that earlier emphasis on indirect measurement when direct semantic auditing was impractical. Comparisons with integration architecture literature were also relevant (Peralta & Rubalcaba, 2021). Prior enterprise integration research distinguished synchronous point-to-point patterns, scheduled batch ETL patterns, and event-driven pipelines, emphasizing differences in coupling, latency, and failure propagation. The study's pattern of stronger performance alignment for architectures closer to continuous or event-driven synchronization relative to batch windows was consistent with the expectation that reduced drift windows limited the accumulation of divergence and reconciliation work, particularly for high-churn product structures. Earlier distributed systems studies emphasized that event-driven architectures required robust schema governance and idempotent processing to prevent inconsistent state and replay-induced duplication; the observed need to treat retries, error rates, and exception pathways as central integration signals aligned with those architectural concerns. The discussion further linked reference integrity measures to established configuration management concepts: broken links and rebuild errors were interpreted as operational manifestations of configuration inconsistency, long recognized as costly because it interrupted downstream processes and forced manual diagnosis (Kaciak et al., 2021). Earlier PLM practice accounts often described manual reconciliation as a major hidden cost in CAD–PLM environments; this study's trace-based reconciliation indicators supported that characterization by showing that human interventions clustered around failure bursts and tail-delay cases. In this way, the study's integration-focused findings were discussed as a quantitative confirmation of long-standing qualitative and conceptual claims, while extending earlier work by providing a trace-based approach that operationalized integrity, propagation, and reconciliation at scale.

Figure 12: PLM Lifecycle Efficiency Measurement Framework



CAD repository collaboration mechanics and version control approaches were discussed as important predictors of observable coordination patterns, consistent with prior work on concurrent engineering, modularity, and design dependency. Earlier engineering management research frequently described the tension between control and parallelism: strict locking reduced simultaneous edit conflicts but increased waiting and serialized progress, whereas parallel work increased concurrency but required

robust merge and conflict resolution mechanisms (Swindle et al., 2021). The study's classification of CAD versioning methods into locking-based, merge-based, and hybrid categories aligned with that conceptual dichotomy and enabled measurable comparison. Observed associations between merge-based collaboration and higher concurrency were consistent with earlier expectations that branching and merging enabled parallel work, particularly for geographically distributed teams and complex assemblies. At the same time, the alignment between high concurrency intensity and elevated rework proxies in certain contexts fit earlier findings that parallelism increased coordination overhead when dependencies were dense and when impact analysis was incomplete. Prior research on product architecture and organizational structure highlighted that coupling patterns created nonlocal effects and raised the cost of coordination; the study's control variables—assembly size, depth, fan-out, and churn—captured those coupling properties and supported the interpretation that collaboration mechanics interacted with structural complexity. Earlier studies of change propagation emphasized that highly reused or highly connected components amplified coordination requirements and validation burdens; the observed role of fan-out and churn in explaining rework and tail delays aligned with this view. The discussion also connected repository event models to mining software repositories research, which treated repositories as behavioral records of collaboration and process dynamics; similarly, this study treated CAD repositories as high-resolution logs of design evolution and coordination, rather than merely as data stores (Lawless et al., 2023). Earlier process mining research cautioned that event ordering, timestamp resolution, and identity aliasing could distort inferences; this study's reported preprocessing and reliability checks addressed those concerns and supported interpretability of collaboration-related findings. In comparing these results with earlier collaboration and configuration studies, the discussion emphasized that no single version control approach uniformly dominated across all outcomes, consistent with earlier socio-technical perspectives that treated tool mechanisms as interacting with context. Instead, measurable outcomes appeared contingent on dependency structure and concurrency intensity, echoing earlier accounts that the success of parallelism depended on modular boundaries, clear ownership, and stable integration pathways. Thus, the study's discussion positioned CAD collaboration mechanics not as isolated technical features but as measurable coordination regimes whose efficiency consequences depended on product coupling and workflow integration stability (Heim et al., 2022).

The quantitative measurement framework itself was discussed as a substantive contribution in relation to earlier maturity-model and survey-dominant evaluation approaches, particularly regarding construct validity, reliability, and interpretability (Cheng et al., 2023). Earlier PLM evaluation methods often relied on maturity models that assessed governance practices and integration capability through perception-based scoring, which provided managerial visibility but introduced risks of bias, inconsistent interpretation, and weak linkage to operational outcomes. The trace-derived framework used in this study was discussed as addressing these limitations by grounding dependent variables in governance-defined lifecycle events and state transitions, thereby improving conceptual alignment between "efficiency" and "governed lifecycle progression." This approach paralleled process mining traditions that emphasized event boundary definitions and case construction as prerequisites for valid performance inference (Taheri & Asadizanjani, 2022). The study's reliability checks on boundary definitions and multi-indicator constructs were discussed in relation to earlier measurement research that warned against single-proxy dependence, especially when log completeness varied across toolchains. By using multiple rework signals and multiple friction indicators, the framework resembled earlier methodological guidance that recommended triangulation for complex constructs. Convergent validity patterns—such as workflow failure clusters aligning with reconciliation events and extended dwell time—matched expectations from both PLM practice accounts and reliability engineering literature, where failures were expected to manifest simultaneously across operational channels. Discriminant validity evidence separating product complexity constructs from collaboration constructs aligned with earlier socio-technical network perspectives that treated structural coupling and social coordination as related but distinct drivers of performance (Wanna, 2021). The discussion also considered that trace-based measures could overrepresent instrumented stages relative to unlogged manual work, a concern frequently raised in event-log analytics; the study's reconciliation and override

indicators partially mitigated that risk by capturing points where manual intervention surfaced in the logs. Moreover, the observed distributional nature of outcomes supported earlier recommendations to focus on variability and tail behavior rather than only mean performance, especially in enterprise workflows where extreme delays dominated risk. In comparing the measurement approach with earlier PLM research, the discussion emphasized that trace-based metrics allowed direct testing of architecture and integration mechanisms rather than inference through self-reported maturity categories (Blahut & Niemira, 2020). This study's framework therefore aligned with the broader methodological shift in operational research toward telemetry-based measurement, while remaining consistent with PLM theory that positioned lifecycle states, configuration integrity, and change governance as the core semantic anchors of lifecycle execution.

Regression and hypothesis testing results were discussed in relation to earlier empirical findings across cloud operations, workflow automation, and lifecycle integration research, with attention to how adjusted relationships clarified mechanism-level interpretations. Earlier cloud and DevOps research often reported that automation and reliable pipeline execution correlated with throughput and stability improvements; this study's adjusted models supported that general pattern by showing consistent associations of automation coverage with higher workflow success rates, reduced exception burden, and improved throughput (Chatterjee et al., 2022). At the same time, earlier PLM and engineering change studies emphasized that governance and coordination introduced delays not fully addressed by technical improvements; this study's weaker adjusted associations for compute class with end-to-end lead times were consistent with that distinction, indicating that provisioning influenced execution segments more directly than approval and decision segments. Prior integration architecture work suggested that batch synchronization increased drift windows and accumulated reconciliation costs; this study's adjusted advantage for near-continuous or event-driven synchronization patterns aligned with that expectation, particularly when complexity and churn were high. Earlier work on modular service decomposition highlighted both benefits and risks: modularity improved failure isolation and operability but increased the number of integration surfaces; the study's mixed adjusted findings for decomposition indicators aligned with that duality, suggesting that modularity alone did not guarantee efficiency gains without sufficient automation coverage and stable interfaces (Xu & Miao, 2022). Earlier reliability literature framed incident exposure as a key driver of extreme delays and backlog accumulation; this study's inclusion of incident-window indicators and the observed association with slower lifecycle progression aligned with that perspective and supported the interpretation that operational disruptions translated into measurable throughput instability in lifecycle pipelines.

Interaction results were discussed in relation to earlier contingency-oriented research that treated effects as conditional on complexity and collaboration regimes; the stronger architecture advantages in high-coupling contexts aligned with the expectation that integration quality mattered most when propagation paths were dense and coordination costs were high. Supplemental mediation-oriented patterns, where retry and failure burdens plausibly accounted for part of the architecture–lead-time relationship, matched earlier systems research that treated intermediate operational mechanisms as key pathways through which design choices influenced outcomes. The discussion therefore compared adjusted findings with earlier studies by emphasizing consistent themes: automation improved reliability and throughput, continuous synchronization reduced drift-related rework, infrastructure improvements most strongly affected execution stages, and context moderated effect size (Chan, 2019). Importantly, the discussion maintained a strictly interpretive stance consistent with empirical findings reporting, framing the results as observed relationships grounded in trace evidence and supported by robustness checks rather than as broad claims detached from measurement context.

The discussion also situated the study within earlier work on international and distributed engineering, emphasizing how team structure and concurrency covariates aligned with established coordination theories (Robiatti, 2021). Earlier distributed product development studies emphasized that dispersion increased reliance on formalized governance, shared repositories, and traceable decision records, while also increasing coordination overhead through asynchronous handoffs and reduced opportunities for real-time resolution of ambiguity. The trace-based findings in this study, where concurrency intensity and dispersion proxies aligned with higher rework and higher tail-delay likelihood, were consistent

with those earlier observations and provided empirical signals linking collaboration structure to lifecycle outcomes. Prior socio-technical perspectives suggested that coordination risk increased when task interdependence was high and communication cycles were constrained; the observed joint influence of dependency structure metrics and concurrency metrics aligned with this account by showing that coupling and collaboration overlap co-occurred in the slowest and most rework-prone cases. Earlier literature on organizational–technical misalignment argued that product architecture coupling often demanded coordination patterns not matched by organizational structure; the trace-based covariates and their relationships to timing and rework outcomes were consistent with this view, indicating that structural characteristics influenced how coordination burdens manifested (Zimmermann Robiatti, 2021). The discussion also compared the observed role of lifecycle phase segmentation with earlier lifecycle management accounts that described stabilization periods as gate-heavy and exception-sensitive; the stronger effects during release windows aligned with the expectation that validation and packaging requirements intensified, making integration reliability and reference integrity more consequential. Earlier process analytics work cautioned that aggregated metrics could hide phase-dependent behavior; the study's segmentation approach was discussed as consistent with that methodological guidance, allowing comparisons within more homogeneous governance contexts. The study's emphasis on CAD repository event models also aligned with earlier mining-repositories traditions, which treated contribution patterns, temporal clustering, and imbalance as signals of coordination regimes; similar signals appeared in this study's covariates, supporting a consistent narrative that collaboration structure shaped measurable workflow friction and rework. Overall, the discussion compared the results with earlier studies by highlighting a convergent message: PLM efficiency depended on a coupled system of product structure complexity, collaboration concurrency, integration reliability, and cloud workflow execution stability (Olsen, 2022). The trace-based approach allowed these relationships to be discussed using observed mechanisms rather than broad generalizations, linking classical PLM concerns about configuration integrity and change governance to measurable outcomes in cloud-native environments.

CONCLUSION

Improving Project Lifecycle Management (PLM) Efficiency with Cloud Architectures and CAD Integration was examined in this empirical study as a measurable transformation of lifecycle execution rather than a conceptual promise of modernization, with industrial CAD repositories and cloud-native workflows providing the trace evidence needed to quantify how changes moved from authoring activity to governed release outcomes. PLM efficiency in this context referred to the speed, stability, and reliability with which engineering change objects and release packages progressed through defined lifecycle states, and the study treated those outcomes as distributions shaped by both technical mechanisms and collaboration conditions. Industrial CAD repositories supplied fine-grained event histories that captured artifact creation and modification, version formation, reference updates, and assembly dependency evolution, allowing product structure complexity and reuse patterns to be quantified as controls that explained baseline variance in processing time and rework exposure. Cloud-native workflow telemetry supplied job-level timing and outcome signals for automated stages such as translation, validation, packaging, and synchronization, enabling throughput, success rates, retry cascades, and extreme-delay behavior to be measured alongside governance-bound lead times. Within this framework, cloud architecture characteristics were interpreted as operational mechanisms that influenced execution segments of the lifecycle, particularly through elasticity and resource provisioning that altered queue behavior, runtime dispersion, and tail latency during peak loads. CAD–PLM integration quality was operationalized through propagation and integrity constructs, including synchronization latency between CAD commits and PLM state updates, reference integrity failure rates observable through broken associations and rebuild errors, and manual reconciliation load observable through overrides and remediation events. Integration architectures were treated as independent variables by contrasting direct API synchronization, batch ETL synchronization, and event-driven pipelines, with attention to how frequency, directionality, and transformation depth shaped drift windows and exception burden. CAD version control approaches were treated as collaboration predictors, distinguishing locking-based coordination that reduced simultaneous edit conflict risk at the cost of serialized access and waiting, from merge-based parallelism that enabled

concurrency but introduced measurable conflict and reconciliation pathways under dense dependency coupling. The study's trace-derived evidence supported a view of PLM efficiency as a coupled outcome: execution performance improved when automation coverage increased and when pipeline reliability reduced failure bursts that triggered rework loops, while governance-dominated segments remained sensitive to complexity, concurrency intensity, and incident disruption conditions. By grounding constructs in governance-defined event boundaries and validating them through consistent logging, convergent trace patterns, and robustness checks under alternative preprocessing rules, the study strengthened the interpretability of observed relationships and avoided equating raw activity volume with lifecycle progress. Overall, the title's central claim was developed as an empirical proposition: cloud architectures and CAD integration altered PLM efficiency through observable mechanisms in workflow stability, propagation timeliness, and integrity maintenance, and industrial repositories and cloud telemetry provided the measurement substrate needed to quantify those mechanisms at scale without relying on perception-based maturity judgments.

RECOMMENDATIONS

Recommendations for improving Project Lifecycle Management (PLM) efficiency in cloud-based, CAD-integrated environments emphasized operationally measurable interventions that strengthened lifecycle propagation, integrity, and stability across change and release workflows. PLM programs were recommended to define efficiency targets as distributional objectives rather than average targets by routinely tracking engineering change lead time, release cycle time, workflow throughput, success rate, and extreme-delay frequency, because long-tail delays and clustered failures typically consumed disproportionate operational capacity and created planning uncertainty. Integration architecture was recommended to be treated as a controllable design variable, with preference given to synchronization patterns that minimized drift windows between CAD repositories and PLM state updates, while ensuring robust idempotency handling, correlation identifiers, and replay-safe processing to prevent duplicated objects and inconsistent configurations. Automated validation and packaging gates were recommended to be expanded and standardized so that metadata completeness, revision discipline, and reference integrity checks occurred early and consistently, reducing late-stage failures that forced repeated export-import cycles and manual remediation. Cloud-native workflow design was recommended to prioritize observability as an efficiency enabler by implementing structured logging, distributed tracing, and stage-level metrics that allowed delay attribution across translation, validation, packaging, and synchronization segments; this instrumentation was recommended to be paired with operational dashboards that monitored retry cascades, queue growth, and error bursts, enabling rapid diagnosis and reducing prolonged dwell times. Elastic scaling policies were recommended to be aligned with workload signatures derived from job telemetry so that burst periods triggered predictable capacity increases, with compute class selection matched to the resource profiles of translation and validation workloads to reduce runtime variance and queue delay during peak activity. Service decomposition decisions were recommended to be guided by measurable dependency and failure-isolation criteria, balancing modularity benefits with the risk of expanding integration surfaces; where decomposition increased touchpoints, standardized API contracts, schema versioning discipline, and consistent error taxonomy were recommended to prevent fragmentation of governance evidence and to reduce interface-driven exceptions. CAD collaboration policies were recommended to be configured in a context-sensitive manner: merge-based parallelism was recommended for distributed teams and exploratory phases when concurrency benefits were high, while conflict monitoring, merge discipline, and dependency-aware ownership rules were recommended to reduce rework in highly coupled assemblies; locking-based controls were recommended selectively for highly sensitive shared artifacts where serialized edits prevented costly reference breakage, while minimizing lock contention through component partitioning and modular assembly boundaries. Data engineering practices were recommended as ongoing operational controls rather than one-time research steps, including identity resolution for users and parts, timestamp normalization across distributed systems, and deduplication rules that distinguished legitimate derived variants from unintended duplicates; these practices were recommended to be embedded into routine governance reporting so that metrics remained interpretable and comparable across projects. Finally, lifecycle phase segmentation was recommended for operational monitoring and improvement prioritization by separating pre-release

and release windows, because stabilization phases typically amplified the consequences of integration instability; by focusing improvement actions on reducing validation failures, packaging retries, and reconciliation events in release windows, organizations were positioned to reduce tail delays and increase throughput stability across the most governance-sensitive portion of PLM execution.

LIMITATIONS

Limitations of this empirical study on improving Project Lifecycle Management (PLM) efficiency with cloud architectures and CAD integration primarily reflected constraints inherent to trace-based measurement, multi-system linkage, and observational inference in complex industrial environments. The study relied on industrial CAD repositories, PLM lifecycle records, and cloud-native workflow telemetry as the principal evidence sources, and the completeness and granularity of these logs varied across projects, tools, and time periods, which constrained uniform comparability and increased the risk that some friction mechanisms were undercounted when instrumentation was sparse. Repository logs captured high-frequency artifact events and dependency updates, yet the semantic intent of design changes and the rationale for governance decisions were not consistently observable, limiting the ability to distinguish between rework caused by integration defects and legitimate iterative refinement driven by engineering discovery. Similarly, workflow telemetry recorded job execution outcomes, retries, and failure categories, but the mapping between technical errors and process-level consequences depended on accurate correlation identifiers and consistent event schemas; where identifiers were missing or inconsistent, linkage uncertainty introduced measurement noise into synchronization latency and reconciliation proxies. The observational design also limited causal attribution because cloud architecture choices, integration patterns, and automation coverage were not randomly assigned and often co-varied with organizational maturity, team experience, and product complexity, which created confounding risks even when statistical controls for assembly structure and collaboration intensity were applied. Product complexity measures derived from assembly graphs captured structural coupling, but they did not fully represent other sources of complexity such as requirements volatility, regulatory constraints, or supplier coordination, which may have influenced lead times and release behavior independently of integration architecture. Team structure covariates captured contributor counts, dispersion proxies, and concurrency intensity, yet they could not fully capture informal coordination practices, decision escalation pathways, or leadership interventions that may have shortened or prolonged approval cycles. Incident-window indicators and operational reliability signals helped isolate disruption effects, but incident logging practices differed across environments, and some performance degradation could have occurred without being classified as an incident, leaving residual operational confounding. The measurement framework emphasized governance-aligned event boundaries, which strengthened construct validity, yet boundary definitions could still differ across PLM configurations where state models and approval workflows were customized, limiting generalizability to organizations with materially different lifecycle schemas. Additionally, metrics such as tail latency and rework proxies were sensitive to the chosen thresholds and window definitions, and although sensitivity checks could evaluate robustness, the resulting interpretations remained dependent on operational definitions that might not transfer identically across industries or toolchains. Finally, the study evaluated efficiency primarily through timing, throughput, stability, and exception burdens, and it did not directly measure downstream business outcomes such as cost reduction, quality escapes in manufacturing, or customer delivery performance, which limited the extent to which observed efficiency changes could be interpreted as broader enterprise value without additional linked evidence.

REFERENCES

[1]. Abdul, K. (2023). Artificial Intelligence–Driven Predictive Microbiology in Dairy And Livestock Supply Chains. *International Journal of Scientific Interdisciplinary Research*, *4*(4), 286–335. https://doi.org/10.63125/syj6pp52

[2]. Abubakr, M., Abbas, A. T., Tomaz, I., Soliman, M. S., Luqman, M., & Hegab, H. (2020). Sustainable and smart manufacturing: an integrated approach. *Sustainability*, *12*(6), 2280.

[3]. Ali, I. M., Sallam, K. M., Moustafa, N., Chakraborty, R., Ryan, M., & Choo, K.-K. R. (2020). An automated task scheduling model using non-dominated sorting genetic algorithm II for fog-cloud systems. *IEEE Transactions on Cloud Computing*, *10*(4), 2294-2308.

[4]. Ali, M. M., Rai, R., Otte, J. N., & Smith, B. (2019). A product life cycle ontology for additive manufacturing. *Computers in Industry*, *105*, 191-203.

[5]. Ameri, F., Sormaz, D., Psarommatis, F., & Kiritsis, D. (2022). Industrial ontologies for interoperability in agile and resilient manufacturing. *International Journal of Production Research*, *60*(2), 420-441.
[6]. Attaoui, W., Sabir, E., Elbiaze, H., & Guizani, M. (2023). VNF and CNF placement in 5G: Recent advances and future trends. *IEEE Transactions on Network and Service Management*, *20*(4), 4698-4733.
[7]. Barricelli, B. R., Casiraghi, E., & Fogli, D. (2019). A survey on digital twin: Definitions, characteristics, applications, and design implications. *Ieee Access*, *7*, 167653-167671.
[8]. Barrios, P., Danjou, C., & Eynard, B. (2022). Literature review and methodological framework for integration of IoT and PLM in manufacturing industry. *Computers in Industry*, *140*, 103688.
[9]. Bibri, S. E. (2022). The social shaping of the metaverse as an alternative to the imaginaries of data-driven smart Cities: A study in science, technology, and society. *Smart Cities*, *5*(3), 832-874.
[10]. Biller, B., & Biller, S. (2023). Implementing digital twins that learn: AI and simulation are at the core. *Machines*, *11*(4), 425.
[11]. Blahut, S., & Niemira, J. (2020). Segmentation and Choice Models. In *Quantitative Methods in Pharmaceutical Research and Development: Concepts and Applications* (pp. 317-359). Springer.
[12]. Breitgand, D., Lekidis, A., Behravesh, R., Weit, A., Giardina, P., Theodorou, V., Costa, C. E., & Barabash, K. (2021). Dynamic slice scaling mechanisms for 5G multi-domain environments. 2021 IEEE 7th International Conference on Network Softwarization (NetSoft),
[13]. Buchert, T., Ko, N., Graf, R., Vollmer, T., Alkhayat, M., Brandenburg, E., Stark, R., Klocke, F., Leistner, P., & Schleifenbaum, J. H. (2019). Increasing resource efficiency with an engineering decision support system for comparison of product design variants. *Journal of Cleaner Production*, *210*, 1051-1062.
[14]. Chan, Y. (2019). Travel modeling under foreseeable communications-and-mobility technologies: Part I (short-term view). In *Advances in Transport Policy and Planning* (Vol. 3, pp. 251-289). Elsevier.
[15]. Chatterjee, A., Gerdes, M. W., Khatiwada, P., & Prinz, A. (2022). Sftsdh: Applying spring security framework with TSD-based oauth2 to protect microservice architecture apis. *Ieee Access*, *10*, 41914-41934.
[16]. Cheng, J.-C., Li, J.-F., & Huang, C.-Y. (2023). Enablers for adopting restriction of hazardous substances directives by electronic manufacturing service providers. *Sustainability*, *15*(16), 12341.
[17]. Cholewa, M., & Minh, L. H. B. (2021). PLM solutions in the process of supporting the implementation and maintenance of the circular economy concept in manufacturing companies. *Sustainability*, *13*(19), 10589.
[18]. Chung, J., Goodman, M., Huang, T., Bertisch, S., & Redline, S. (2021). Multidimensional sleep health in a diverse, aging adult cohort: concepts, advances, and implications for research and intervention. *Sleep Health*, *7*(6), 699-707.
[19]. Corallo, A., Crespino, A. M., Del Vecchio, V., Lazoi, M., & Marra, M. (2021). Understanding and defining dark data for the manufacturing industry. *IEEE Transactions on Engineering Management*, *70*(2), 700-712.
[20]. Corallo, A., Del Vecchio, V., Lezzi, M., & Luperto, A. (2022). Model-based enterprise approach in the product lifecycle management: State-of-the-art and future research directions. *Sustainability*, *14*(3), 1370.
[21]. Demirel, H. O., Irshad, L., Ahmed, S., & Tumer, I. Y. (2021). Digital human-in-the-loop methodology for early design computational human factors. International Conference on Human-Computer Interaction,
[22]. Denger, A., & Zamazal, K. (2020). Product lifecycle challenges for powertrain systems in the automotive industry. In *Systems Engineering for Automotive Powertrain Development* (pp. 1-18). Springer.
[23]. Disselkamp, J.-P., Cieply, J., Dyck, F., Grothe, R., Anacker, H., & Dumitrescu, R. (2023). Integrated product and production development-a systematic literature review. *Procedia CIRP*, *119*, 716-721.
[24]. Duda, J., Oleszek, S., & Santarek, K. (2022). Product lifecycle management (PLM) in the context of industry 4.0. International Scientific-Technical Conference MANUFACTURING,
[25]. Eigner, M. (2021). Engineering 4.0—Implementation of the Digitalization of Engineering. In *System Lifecycle Management: Engineering Digitalization (Engineering 4.0)* (pp. 101-237). Springer.
[26]. Favi, C., Marconi, M., Germani, M., & Mandolini, M. (2019). A design for disassembly tool oriented to mechatronic product de-manufacturing and recycling. *Advanced Engineering Informatics*, *39*, 62-79.
[27]. Gebru, B., Zeleke, L., Blankson, D., Nabil, M., Nateghi, S., Homaifar, A., & Tunstel, E. (2022). A review on human–machine trust evaluation: Human-centric and machine-centric perspectives. *IEEE Transactions on Human-Machine Systems*, *52*(5), 952-962.
[28]. Gerhard, D., Salas Cordero, S., Vingerhoeds, R., Sullivan, B. P., Rossi, M., Brovar, Y., Menshenin, Y., Fortin, C., & Eynard, B. (2022). MBSE-PLM integration: initiatives and future outlook. IFIP International Conference on Product Lifecycle Management,
[29]. Gudbrandsdottir, I. Y., Olafsdottir, G., Oddsson, G. V., Stefansson, H., & Bogason, S. G. (2021). Operationalization of interorganizational fairness in food systems: From a social construct to quantitative indicators. *Agriculture*, *11*(1), 36.
[30]. Habib, H., Menhas, R., & McDermott, O. (2022). Managing engineering change within the paradigm of product lifecycle management. *Processes*, *10*(9), 1770.
[31]. Hammad, S., & Muhammad Mohiul, I. (2023). Geotechnical And Hydraulic Simulation Models for Slope Stability And Drainage Optimization In Rail Infrastructure Projects. *Review of Applied Science and Technology*, *2*(02), 01–37. https://doi.org/10.63125/jmx3p851
[32]. Hannila, H., Kuula, S., Harkonen, J., & Haapasalo, H. (2022). Digitalisation of a company decision-making system: a concept for data-driven and fact-based product portfolio management. *Journal of Decision Systems*, *31*(3), 258-279.
[33]. Hayat, M., & Winkler, H. (2022). From traditional product lifecycle management systems to blockchain-based platforms. *Logistics*, *6*(3), 40.

[34]. Heim, H., Chrimes, C., & Green, C. (2022). Digital hesitancy: Examining the organisational mindset required for the adoption of digitalised textile supply chain transparency. In *Blockchain technologies in the textile and fashion industry* (pp. 47-80). Springer.
[35]. Hodgkinson, J. H., & Elmouttie, M. (2020). Cousins, siblings and twins: A review of the geological model's place in the digital mine. *Resources*, *9*(3), 24.
[36]. Hu, X., Han, J., Tang, X., & Lin, X. (2021). Powertrain design and control in electrified vehicles: A critical review. *IEEE transactions on transportation electrification*, *7*(3), 1990-2009.
[37]. Iakymenko, N., Romsdal, A., Alfnes, E., Semini, M., & Strandhagen, J. O. (2020). Status of engineering change management in the engineer-to-order production environment: insights from a multiple case study. *International Journal of Production Research*, *58*(15), 4506-4528.
[38]. Imran, F., Shahzad, K., Butt, A., & Kantola, J. (2021). Digital transformation of industrial organizations: Toward an integrated framework. *Journal of change management*, *21*(4), 451-479.
[39]. Jabed Hasan, T., & Waladur, R. (2023). AI-Driven Cybersecurity, IOT Networking, And Resilience Strategies For Industrial Control Systems: A Systematic Review For U.S. Critical Infrastructure Protection. *International Journal of Scientific Interdisciplinary Research*, *4*(4), 144–176. https://doi.org/10.63125/mbyhj941
[40]. Janker, J., & Mann, S. (2020). Understanding the social dimension of sustainability in agriculture: a critical review of sustainability assessment tools. *Environment, Development and Sustainability*, *22*(3), 1671-1691.
[41]. Jinnat, A., & Md. Kamrul, K. (2021). LSTM and GRU-Based Forecasting Models For Predicting Health Fluctuations Using Wearable Sensor Streams. *American Journal of Interdisciplinary Studies*, *2*(02), 32-66. https://doi.org/10.63125/1p8gbp15
[42]. Kaciak, E., Koladkiewicz, I., Thongpapanl, N., & Wojtyra, M. (2021). The role of social networks in shaping entrepreneurial exit strategies. *International Entrepreneurship and Management Journal*, *17*(4), 1619-1655.
[43]. Kahvazadeh, S., Khalili, H., Silab, R. N., Bakhshi, B., & Mangues-Bafalluy, J. (2022). Vertical-oriented 5G platform-as-a-service: user-generated content case study. 2022 IEEE Future Networks World Forum (FNWF),
[44]. Kandidayeni, M., Fernandez, A. O. M., Khalatbarisoltani, A., Boulon, L., Kelouwani, S., & Chaoui, H. (2019). An online energy management strategy for a fuel cell/battery vehicle considering the driving pattern and performance drift impacts. *IEEE Transactions on Vehicular Technology*, *68*(12), 11427-11438.
[45]. Kapos, G.-D., Tsadimas, A., Kotronis, C., Dalakas, V., Nikolaidou, M., & Anagnostopoulos, D. (2019). A declarative approach for transforming SysML models to executable simulation models. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *51*(6), 3330-3345.
[46]. Kineber, A. F., Othman, I., Famakin, I. O., Oke, A. E., Hamed, M. M., & Olayemi, T. M. (2023). Challenges to the implementation of building information modeling (BIM) for sustainable construction projects. *Applied Sciences*, *13*(6), 3426.
[47]. Kulkarni, V. N., Gaitonde, V., & Kottur shettar, B. (2022). Product lifecycle management (PLM): a key enabler in implementation of industry 4.0: a key enabler in implementation of industry 4.0. In *Handbook of Smart Materials, Technologies, and Devices: Applications of Industry 4.0* (pp. 349-380). Springer.
[48]. Lawless, W. F., Moskowitz, I. S., & Doctor, K. Z. (2023). A quantum-like model of interdependence for embodied human–machine teams: reviewing the path to autonomy facing complexity and uncertainty. *Entropy*, *25*(9), 1323.
[49]. Lázaro, O., Alonso, J., Holom, R.-M., Rafetseder, K., Kritzinger, S., Ubis, F., Fritz, G., Wiesinger, A., Sehrschön, H., & Nguyen, J. (2022). Model-based engineering and semantic interoperability for trusted digital twins big data connection across the product lifecycle. In *Technologies and applications for big data value* (pp. 399-429). Springer.
[50]. Lee, Z. W., Chan, T. K., Chong, A. Y.-L., & Thadani, D. R. (2019). Customer engagement through omnichannel retailing: The effects of channel integration quality. *Industrial Marketing Management*, *77*, 90-101.
[51]. Li, X., Shen, G. Q., Wu, P., & Yue, T. (2019). Integrating building information modeling and prefabrication housing production. *Automation in Construction*, *100*, 46-60.
[52]. Li, Y., Roy, U., & Saltz, J. S. (2019). Towards an integrated process model for new product development with data-driven features (NPD3). *Research in Engineering Design*, *30*(2), 271-289.
[53]. Li, Z., Tian, Z., Wang, L., & Zhong, R. Y. (2022). Blockchain-enabled product lifecycle management. In *Design and Operation of Production Networks for Mass Personalization in the Era of Cloud Technology* (pp. 349-379). Elsevier.
[54]. Lim, H., Jung, E., Jodoin, K., Du, X., Airton, L., & Lee, E.-Y. (2021). Operationalization of intersectionality in physical activity and sport research: A systematic scoping review. *SSM-population health*, *14*, 100808.
[55]. Lim, K. Y. H., Zheng, P., & Chen, C.-H. (2020). A state-of-the-art survey of Digital Twin: techniques, engineering product lifecycle management and business innovation perspectives. *Journal of intelligent manufacturing*, *31*(6), 1313-1337.
[56]. Lipu, M. S. H., Faisal, M., Ansari, S., Hannan, M. A., Karim, T. F., Ayob, A., Hussain, A., Miah, M. S., & Saad, M. H. M. (2021). Review of electric vehicle converter configurations, control schemes and optimizations: Challenges and suggestions. *Electronics*, *10*(4), 477.
[57]. Liu, C., Le Roux, L., Körner, C., Tabaste, O., Lacan, F., & Bigot, S. (2022). Digital twin-enabled collaborative data management for metal additive manufacturing systems. *Journal of Manufacturing Systems*, *62*, 857-874.
[58]. Liu, X., Wang, W. M., Guo, H., Barenji, A. V., Li, Z., & Huang, G. Q. (2020). Industrial blockchain based framework for product lifecycle management in industry 4.0. *Robotics and computer-integrated manufacturing*, *63*, 101897.
[59]. Liu, Y., Zhang, Y., Ren, S., Yang, M., Wang, Y., & Huisingh, D. (2020). How can smart technologies contribute to sustainable product lifecycle management? *Journal of Cleaner Production*, *249*, 119423.

[60]. Lu, Y., Xu, X., & Wang, L. (2020). Smart manufacturing process and system automation–a critical review of the standards and envisioned scenarios. *Journal of Manufacturing Systems*, *56*, 312-325.
[61]. MacCarthy, B. L., & Pasley, R. C. (2021). Group decision support for product lifecycle management. *International Journal of Production Research*, *59*(16), 5050-5067.
[62]. Masud, R., & Md Sarwar Hossain, S. (2024). The Impact of Smart Materials And Fire-Resistant Structures On Safety In U.S. Public Infrastructure. *Journal of Sustainable Development and Policy*, *3*(03), 44-86. https://doi.org/10.63125/ygr1yk30
[63]. Matelsky, J. K., Rodriguez, L. M., Xenes, D., Gion, T., Hider, R., Wester, B. A., & Gray-Roncal, W. (2021). An integrated toolkit for extensible and reproducible neuroscience. 2021 43rd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC),
[64]. Md, K., & Sai Praveen, K. (2024). Hybrid Discrete-Event And Agent-Based Simulation Framework (H-DEABSF) For Dynamic Process Control In Smart Factories. *ASRC Procedia: Global Perspectives in Science and Scholarship*, *4*(1), 72–96. https://doi.org/10.63125/wcqq7x08
[65]. Md Nahid, H., & Tahmina Akter Bhuya, M. (2024). An Empirical Study of Big Data–Enabled Predictive Analytics And Their Impact On Financial Forecasting And Market Decision-Making. *Review of Applied Science and Technology*, *3*(01), 143–182. https://doi.org/10.63125/1mjfqf10
[66]. Md Newaz, S., & Md Jahidul, I. (2024). AI-Powered Business Analytics For Smart Manufacturing And Supply Chain Resilience. *Review of Applied Science and Technology*, *3*(01), 183–220. https://doi.org/10.63125/va5gpg60
[67]. Md. Akbar, H. (2024). Computational Psychometrics and Digital Biomarker Modeling For Precision Mental Health Diagnostics. *International Journal of Scientific Interdisciplinary Research*, *5*(2), 487–525. https://doi.org/10.63125/vg522x27
[68]. Md. Akbar, H., & Sharmin, A. (2022). Neurobiotechnology-Driven Regenerative Therapy Frameworks For Post-Traumatic Neural Recovery. *American Journal of Scholarly Research and Innovation*, *1*(02), 134–170. https://doi.org/10.63125/24s6kt66
[69]. Md. Foysal, H., & Subrato, S. (2022). Data-Driven Process Optimization in Automotive Manufacturing A Machine Learning Approach To Waste Reduction And Quality Improvement. *Journal of Sustainable Development and Policy*, *1*(02), 87-133. https://doi.org/10.63125/2hk0qd38
[70]. Md. Rabiul, K., & Khairul Alam, T. (2024). Impact Of IOT and Blockchain Integration On Real-Time Supply Chain Transparency. *International Journal of Scientific Interdisciplinary Research*, *5*(2), 449–486. https://doi.org/10.63125/2yc6e230
[71]. Meißner, M., Jacobs, G., Jagla, P., & Sprehe, J. (2021). Model based systems engineering as enabler for rapid engineering change management. *Procedia CIRP*, *100*, 61-66.
[72]. Mikalef, P., Pateli, A., & van de Wetering, R. (2021). IT architecture flexibility and IT governance decentralisation as drivers of IT-enabled dynamic capabilities and competitive performance: The moderating effect of the external environment. *European Journal of Information Systems*, *30*(5), 512-540.
[73]. Mishra, K. N. (2020). An efficient palm-dorsa-based approach for vein image enhancement and feature extraction in cloud computing environment. In *Unmanned Aerial Vehicles in Smart Cities* (pp. 85-106). Springer.
[74]. Mousavi, A., Mohammadzadeh, M., & Zare, H. (2022). Developing a system dynamic model for product life cycle management of generic pharmaceutical products: its relation with open innovation. *Journal of Open Innovation: Technology, Market, and Complexity*, *8*(1), 14.
[75]. Myrodia, A., Randrup, T., & Hvam, L. (2019). Configuration lifecycle management maturity model. *Computers in Industry*, *106*, 30-47.
[76]. Naseem, A., Ahmad, Y., & uz Zaman, U. K. (2023). Product Life Cycle Management and Cloud Manufacturing. In *Handbook of Manufacturing Systems and Design* (pp. 187-218). CRC Press.
[77]. Niemann, J., & Pisla, A. (2021). *Life-cycle management of machines and mechanisms*. Springer.
[78]. Nzetchou, S., Durupt, A., Remy, S., & Eynard, B. (2019). Review of CAD visualization standards in PLM. IFIP International Conference on Product Lifecycle Management,
[79]. Oke, A. E., Kineber, A. F., Olanrewaju, O. I., Omole, O., Jamir Singh, P. S., Samsurijan, M. S., & Ramli, R. A. (2022). Exploring the 4IR drivers for sustainable residential building delivery from social work residential perspective—a structural equation modelling approach. *Sustainability*, *15*(1), 468.
[80]. Olsen, W. K. (2022). *Systematic mixed-methods research for social scientists*. Springer.
[81]. Pang, T. Y., Pelaez Restrepo, J. D., Cheng, C.-T., Yasin, A., Lim, H., & Miletic, M. (2021). Developing a digital twin and digital thread framework for an 'Industry 4.0'Shipyard. *Applied Sciences*, *11*(3), 1097.
[82]. Paramatmuni, C., & Cogswell, D. (2023). Extending the capability of component digital threads using material passports. *Journal of Manufacturing Processes*, *87*, 245-259.
[83]. Patacas, J., Dawood, N., & Kassem, M. (2020). BIM for facilities management: A framework and a common data environment using open standards. *Automation in Construction*, *120*, 103366.
[84]. Peralta, A., & Rubalcaba, L. (2021). A metagovernance model of innovation networks in the health and social services using a neo-schumpeterian framework. *International Journal of Environmental Research and Public Health*, *18*(11), 6133.
[85]. Pereira Pessôa, M. V., & Jauregui Becker, J. M. (2020). Smart design engineering: a literature review of the impact of the 4th industrial revolution on product design and development. *Research in Engineering Design*, *31*(2), 175-195.
[86]. Peters, S., Fortin, C., & McSorley, G. (2021). Investigating the Effectiveness of a Causal MBSE Software Tool in Modelling CubeSat Conceptual Design Data. IFIP International Conference on Product Lifecycle Management,

[87]. Pfeiffer, T., Apostolov, C., Savarino, P., & Dickopf, T. (2023). Concept and implementation path for model-based impact analysis in product development. *Procedia CIRP*, *119*, 1005-1010.
[88]. Polenghi, A., Acerbi, F., Roda, I., Macchi, M., & Taisch, M. (2021). Enterprise information systems interoperability for asset lifecycle management to enhance circular manufacturing. *IFAC-PapersOnLine*, *54*(1), 361-366.
[89]. Popovic, D., Elgh, F., & Heikkinen, T. (2021). Configuration of flexible volumetric elements using product platforms: Information modeling method and a case study. *Automation in Construction*, *126*, 103661.
[90]. Psaromanolakis, N., Theodorou, V., Laskaratos, D., Kalogeropoulos, I., Vlontzou, M.-E., Zarogianni, E., & Samaras, G. (2023). MLOps meets edge computing: an edge platform with embedded intelligence towards 6G systems. 2023 Joint European Conference on Networks and Communications & 6G Summit (EuCNC/6G Summit),
[91]. Qi, X., Luo, Y., Wu, G., Boriboonsomsin, K., & Barth, M. (2019). Deep reinforcement learning enabled self-learning control for energy efficient driving. *Transportation Research Part C: Emerging Technologies*, *99*, 67-81.
[92]. Ramasubbu, N., & Kemerer, C. F. (2021). Controlling technical debt remediation in outsourced enterprise systems maintenance: An empirical analysis. *Journal of Management Information Systems*, *38*(1), 4-28.
[93]. Rifat, C., & Rebeka, S. (2023). The Role Of ERP-Integrated Decision Support Systems In Enhancing Efficiency And Coordination In Healthcare Logistics: A Quantitative Study. *International Journal of Scientific Interdisciplinary Research*, *4*(4), 265–285. https://doi.org/10.63125/c7srk144
[94]. Rigger, E., Vosgien, T., Bitrus, S., Szabo, P., & Eynard, B. (2020). Enterprise architecture method for continuous improvement of PLM based on process mining. IFIP International Conference on Product Lifecycle Management,
[95]. Robiatti, Z. (2021). *National Development Banks in South America*. Springer.
[96]. Ruediger-Flore, P., Glatt, M., Hussong, M., & Aurich, J. C. (2023). CAD-based data augmentation and transfer learning empowers part classification in manufacturing. *The International Journal of Advanced Manufacturing Technology*, *125*(11), 5605-5618.
[97]. Saadatmand, F., Lindgren, R., & Schultze, U. (2019). Configurations of platform organizations: Implications for complementor engagement. *Research policy*, *48*(8), 103770.
[98]. Sai Praveen, K. (2024). AI-Enhanced Data Science Approaches For Optimizing User Engagement In U.S. Digital Marketing Campaigns. *Journal of Sustainable Development and Policy*, *3*(03), 01-43. https://doi.org/10.63125/65ebsn47
[99]. Salimbeni, S., & Redchuk, A. (2023). Quality 4.0 and smart product development. Proceedings of International Conference on Information Technology and Applications: ICITA 2022,
[100]. Schiller, S., Landwehr, M., Vinogradov, G., Dimitriadis, I., Akyürek, H., Lipp, J., Ganser, P., & Bergs, T. (2022). Towards ontology-based lifecycle management in blisk manufacturing. *Procedia CIRP*, *112*, 280-285.
[101]. Sebrechts, M., Volckaert, B., De Turck, F., Yang, K., & Al-Naday, M. (2022). Fog native architecture: Intent-based workflows to take cloud native toward the edge. *IEEE Communications Magazine*, *60*(8), 44-50.
[102]. Shainer, G., Graham, R., Newburn, C. J., Hernandez, O., Bloch, G., Gibbs, T., & Wells, J. C. (2021). NVIDIA's Cloud Native Supercomputing. Smoky Mountains Computational Sciences and Engineering Conference,
[103]. Shoflul Azam, T., & Md. Al Amin, K. (2024). Quantitative Study on Machine Learning-Based Industrial Engineering Approaches For Reducing System Downtime In U.S. Manufacturing Plants. *International Journal of Scientific Interdisciplinary Research*, *5*(2), 526–558. https://doi.org/10.63125/kr9r1r90
[104]. Singh, S., & Misra, S. C. (2019). Exploring the challenges for adopting the cloud PLM in manufacturing organizations. *IEEE Transactions on Engineering Management*, *68*(3), 752-766.
[105]. Singh, S., Misra, S. C., & Kumar, S. (2021). Institutionalization of product lifecycle management in manufacturing firms. *IEEE Transactions on Engineering Management*, *70*(10), 3465-3480.
[106]. Stark, R. (2022). Future virtual product creation solutions with new engineering capabilities. In *Virtual Product Creation in Industry: The Difficult Transformation from IT Enabler Technology to Core Engineering Competence* (pp. 555-648). Springer.
[107]. Stefanova-Stoyanova, V., Stankov, I., & Danov, P. (2022). Technology for Multiplication and Analysis of Results Using Classes of Digital Twins in the Process of Designing Business Process Management Systems. 2022 XXXI International Scientific Conference Electronics (ET),
[108]. Stylidis, K., Wickman, C., & Söderberg, R. (2020). Perceived quality of products: a framework and attributes ranking method. *Journal of Engineering Design*, *31*(1), 37-67.
[109]. Suyemoto, K. L., Curley, M., & Mukkamala, S. (2020). What do we mean by "ethnicity" and "race"? A consensual qualitative research investigation of colloquial understandings. *Genealogy*, *4*(3), 81.
[110]. Światowiec-Szczepańska, J., & Stępień, B. (2022). Drivers of digitalization in the energy sector—The managerial perspective from the catching up economy. *Energies*, *15*(4), 1437.
[111]. Swindle, T., Zhang, D., Johnson, S. L., Whiteside-Mansell, L., Curran, G. M., Martin, J., Selig, J. P., & Bellows, L. L. (2021). A mixed-methods protocol for identifying successful sustainability strategies for nutrition and physical activity interventions in childcare. *Implementation Science Communications*, *2*(1), 8.
[112]. Taheri, S., & Asadizanjani, N. (2022). An overview of medical electronic hardware security and emerging solutions. *Electronics*, *11*(4), 610.
[113]. Talhi, A., Fortineau, V., Huet, J.-C., & Lamouri, S. (2019). Ontology for cloud manufacturing based product lifecycle management. *Journal of intelligent manufacturing*, *30*(5), 2171-2192.
[114]. Tamburri, D. A., Palomba, F., & Kazman, R. (2019). Exploring community smells in open-source: An automated approach. *IEEE Transactions on software Engineering*, *47*(3), 630-652.

[115]. Tarraço, E. L., Borini, F. M., Bernardes, R. C., & Della Santa Navarrete, S. (2021). The differentiated impact of the institutional environment on eco-innovation and green manufacturing strategies: A comparative analysis between emerging and developed countries. *IEEE Transactions on Engineering Management*, *70*(7), 2369-2380.
[116]. Trankle, S. A., & Reath, J. (2019). Partners in Recovery: an early phase evaluation of an Australian mental health initiative using program logic and thematic analysis. *BMC Health Services Research*, *19*(1), 524.
[117]. Treber, S., Breig, R., Kentner, M., Häfner, B., & Lanza, G. (2019). Information exchange in global production networks: increasing transparency by simulation, statistical experiments and selection of digitalization activities. *Procedia CIRP*, *84*, 225-230.
[118]. Urban-Galindo, J.-J., & Ripailles, S. (2019). Plm at groupe psa. In *Product Lifecycle Management (Volume 4): The Case Studies* (pp. 29-50). Springer.
[119]. VanDerHorn, E., & Mahadevan, S. (2021). Digital Twin: Generalization, characterization and implementation. *Decision support systems*, *145*, 113524.
[120]. Vaño, R., Lacalle, I., Sowiński, P., S-Julián, R., & Palau, C. E. (2023). Cloud-native workload orchestration at the edge: A deployment review and future directions. *Sensors*, *23*(4), 2215.
[121]. Wang, H., & Peng, G. (2023). *Collaborative Knowledge Management Through Product Lifecycle*. Springer.
[122]. Wang, X. V., & Wang, L. (2019). Digital twin-based WEEE recycling, recovery and remanufacturing in the background of Industry 4.0. *International Journal of Production Research*, *57*(12), 3892-3902.
[123]. Wang, Y., Wang, S., Yang, B., Zhu, L., & Liu, F. (2020). Big data driven Hierarchical Digital Twin Predictive Remanufacturing paradigm: Architecture, control mechanism, application scenario and benefits. *Journal of Cleaner Production*, *248*, 119299.
[124]. Wanna, J. (2021). Conformity and diversity in budgetary systems: Aspirations, routines, and recalibration. In *Handbook of Public Administration* (pp. 197-209). Routledge.
[125]. Xu, Z., & Miao, S. (2022). Effect of public space on collective action for rural waste management and the mediating effects of social capital. *Agriculture*, *12*(7), 1020.
[126]. Yang, W., Chen, Y., Chen, Y.-C., & Yeh, K.-C. (2021). Intelligent agent-based predict system with cloud computing for enterprise service platform in IoT environment. *Ieee Access*, *9*, 11843-11871.
[127]. Yörük, E., Öker, İ., Yıldırım, K., & Yakut-Çakar, B. (2019). The variable selection problem in the three worlds of welfare literature. *Social Indicators Research*, *144*(2), 625-646.
[128]. Yousefnezhad, N., Malhi, A., Keyriläinen, T., & Främling, K. (2023). A comprehensive security architecture for information management throughout the lifecycle of IoT products. *Sensors*, *23*(6), 3236.
[129]. Zech, P., Fröch, G., & Breu, R. (2023). A requirements study on model repositories for digital twins in construction engineering. International Conference on Cooperative Information Systems,
[130]. Zhang, Q., Lu, X., Peng, Z., & Ren, M. (2019). Perspective: a review of lifecycle management research on complex products in smart-connected environments. *International Journal of Production Research*, *57*(21), 6758-6779.
[131]. Zhou, T., Xiong, W., Obata, Y., Lange, C., & Ma, Y. (2022). Digital product design and engineering analysis techniques. In *Digital manufacturing* (pp. 57-96). Elsevier.
[132]. Zimmermann Robiatti, R. (2021). Development Impact. In *National Development Banks in South America: Governance, Financial Performance and Development Impact* (pp. 123-172). Springer.
[133]. Zong, B., Fan, C., Wang, X., Duan, X., Wang, B., & Wang, J. (2019). 6G technologies: Key drivers, core requirements, system architectures, and enabling technologies. *IEEE Vehicular Technology Magazine*, *14*(3), 18-27.
[134]. Zulqarnain, F. N. U. (2022). Policy Optimization for Sustainable Energy Security: Data-Driven Comparative Analysis Between The U.S. And South Asia. *American Journal of Interdisciplinary Studies*, *3*(04), 294-331. https://doi.org/10.63125/v4e4m413
[135]. Zulqarnain, F. N. U., & Subrato, S. (2021). Modeling Clean-Energy Governance Through Data-Intensive Computing And Smart Forecasting Systems. *International Journal of Scientific Interdisciplinary Research*, *2*(2), 128–167. https://doi.org/10.63125/wnd6qs51
[136]. Zulqarnain, F. N. U., & Subrato, S. (2023). Intelligent Climate Risk Modeling For Robust Energy Resilience And National Security. *Journal of Sustainable Development and Policy*, *2*(04), 218-256. https://doi.org/10.63125/jmer2r39