Scientific Research Publishing

Journal of Software Engineering and Applications, 2025, 18(9), 317-335
https://www.scirp.org/journal/jsea
ISSN Online: 1945-3124
ISSN Print: 1945-3116

Navigating Continuous Improvement: An In-Depth Analysis of Lean-Agile and DevOps Maturity Models

Utham Kumar Anugula Sethupathy

Independent Researcher, Atlanta, GA, USA
Email: ANUG0001@e.ntu.edu.sg

How to cite this paper: Sethupathy, U.K.A. (2025) Navigating Continuous Improvement: An In-Depth Analysis of Lean-Agile and DevOps Maturity Models. *Journal of Software Engineering and Applications*, **18**, 317-335.
https://doi.org/10.4236/jsea.2025.189019

Received: April 28, 2025
Accepted: August 25, 2025
Published: August 28, 2025

Abstract

Introduction: DevOps maturity models help organizations benchmark their engineering capabilities, yet the empirical grounding of most models remains fragmented in scholarly literature. **Methods**: We conducted a mixed-methods study consisting of (i) a systematic literature review (SLR) of 78 peer-reviewed papers from IEEE, ACM, SpringerLink, and ScienceDirect (2013-2024) and (ii) multiple embedded case studies of three large enterprises (finance, media, telecom) following Yin's five-step protocol. Quantitative project-metric data (n = 2443 deploys) were triangulated with 26 semi-structured interviews. **Results**: The SLR synthesized 27 core capability dimensions across existing maturity models and identified four evidence-backed outcome clusters (deployment frequency, change failure rate, MTTR, lead-time). The proposed Lean-Agile DevOps Maturity Framework integrates these dimensions into six domains and five levels. Case studies confirm a significant correlation between maturity score and deployment frequency ($\rho = 0.67$, $p < 0.01$) and a 31% reduction in MTTR when moving from "Intermediate" to "Advanced". **Discussion**: Our framework extends prior models by adding Security Integration and Architecture & Design as first-class domains, addressing gaps reported by earlier studies. We outline threats to validity, replication artifacts, and future research opportunities for automated maturity telemetry.

Keywords

DevOps Maturity Model, Lean-Agile, Continuous Delivery, DevSecOps, Software Metrics, Case Study Research

1. Introduction

Accelerating release cadence while preserving reliability and security has become

a strategic imperative for modern enterprises. DevOps—understood as the fusion of Lean product thinking, Agile planning, and continuous delivery automation—has emerged as the dominant organizational paradigm for meeting this demand [1]. Yet organizations still struggle to understand where they stand on the DevOps journey and which capability gaps most impede flow. Maturity models seek to answer these questions by supplying staged road maps, checklists, and benchmarking metrics [2].

1.1. Problem Statement

Despite their popularity in industry white-papers, existing DevOps maturity models exhibit three persistent weaknesses:

1. **Fragmented empirical grounding**—Prior reviews show inconsistent constructs, vague level definitions, and scant outcome data [3] [4].
2. **Security and architectural agility largely ignored**—Less than 20% of published models treat DevSecOps or modular architecture as first-class domains [5].
3. **Minimal validation across diverse contexts**—Most models are evaluated in a single case or not at all, limiting generalizability [6].

Consequently, leaders lack a **rigorous, evidence-based instrument** to benchmark progress and justify investment.

1.2. Research Objectives

This study addresses the above gaps through a mixed-methods investigation that unifies systematic literature evidence with multi-industry field data. We pursue three research questions (RQs):

- **RQ1**: Which capability dimensions dominate existing DevOps maturity models reported in peer-reviewed literature?
- **RQ2**: How does an organization's maturity score correlate with key software-delivery performance metrics (deployment frequency, lead-time, change-failure rate, MTTR)?
- **RQ3**: What qualitative factors enable or hinder progression across maturity levels in varied industry settings?

1.3. Proposed Solution and Scope

Building on 78 peer-reviewed sources published between 2013 and 2024, we synthesize 27 capability dimensions into a **Lean-Agile DevOps Maturity Framework (LADMF)** comprising six domains—Deployment Automation, Telemetry & Observability, Testing Maturity, Build & Release Management, **Security Integration**, and **Architecture & Design**—each articulated over five maturity levels. The framework is empirically validated through embedded case studies at a global bank, a streaming-media company, and a telecom operator, collectively encompassing 2443 production deploys and 26 practitioner interviews.

1.4. Key Contributions

This paper makes four contributions:

1. **Comprehensive SLR**—A reproducible mapping of capability constructs and outcome metrics across the DevOps maturity literature (2013-2024).
2. **Extended Framework**—Integration of security and architectural agility as first-class domains, addressing omissions in prior models.
3. **Mixed-Methods Validation**—Quantitative correlation of maturity scores with DORA-style KPIs and qualitative insights into progression enablers and inhibitors.
4. **Replication Package**—Public artifacts (search protocol, data-collection instruments, anonymized telemetry) enabling independent verification and future extension.

1.5. Paper Structure

Section 2 reviews related work and positions LADMF against ten seminal models. Section 3 details the research methodology, including the SLR protocol, case-study design, and statistical analyses. Section 4 presents the LADMF in full, while Section 5 reports SLR and case-study results. Section 6 discusses practical implications, threats to validity, and avenues for automated maturity telemetry. Section 7 concludes with lessons learned and future research directions.

2. Related Work

A decade of scholarship on DevOps maturity reveals an increasingly diverse but still fragmented body of models. This section (i) summarizes the search and screening process that underpins our systematic literature review (full protocol in § 3), (ii) synthesizes the capability dimensions most frequently cited by prior work, and (iii) positions the proposed Lean-Agile DevOps Maturity Framework (LADMF) against ten seminal models from both academia and industry.

Foundational works and early empirical mappings of DevOps capability constructs establish the baseline we build on [1]-[8], and the case-study research canon provides the methodological scaffolding for our multi-site design [9]. Domain-specific empirical analyses extend to delivery-pipeline practices and organizational change [10]-[12], with subsequent surveys and frameworks refining maturity dimensions and validation approaches [13]-[15]. Recent scholarship broadens coverage to DevSecOps and architecturally agile delivery—including policy-as-code enforcement, threat modelling integration, and evolutionary architectures—thereby addressing gaps noted in earlier models [16]-[25]. Methodological and practice-oriented contributions on immutable infrastructure, GitOps controllers, compliance-as-code, and automated architecture fitness functions further operationalize maturity assessment [26]-[33]. Finally, studies on DevOps metrics and ROI, reliability engineering practices, and team-level psychological safety extend the evidence base against which we benchmark outcomes in this paper [34]-[40].

2.1. Corpus Identification

Applying the search string ("DevOps" AND "maturity model") OR ("continuous delivery" AND "capability model") to IEEE Xplore, ACM DL, Scopus and SpringerLink returned 1428 records (2013-2024). After duplicate removal and title-abstract screening, 142 papers remained. Quality appraisal using Kitchenham's checklist excluded studies scoring < 3/5, yielding **78 peer-reviewed articles** for full-text analysis.

2.2. Evolution of DevOps Maturity Models

Early models (2013-2016) were primarily descriptive checklists derived from single company experience reports. Mid-period studies (2017-2020) introduced multi-domain structures—e.g. CALMS and CAMS—to capture culture and measurement aspects. Recent work (2021-2024) shows a shift toward data-driven validation yet still concentrates on the four canonical domains of *culture, automation, measurement,* and *sharing*. **Security integration** and **architectural agility** appear in only 15% and 18% of studies respectively, confirming the gap noted by Erich *et al.* [7] and Lwakatare *et al.* [8] and the case-study research canon provides the methodological scaffolding for our multi-site design [9].

2.3. Recurring Capability Dimensions

Coding of the 78 papers produced a catalogue of 27 discrete capability dimensions. The five most cited were Continuous Integration (79%), Automated Testing (74%), Continuous Deployment (68%), Telemetry (64%) and Change-Failure Recovery (59%). Less than one-fifth of papers explicitly addressed *Threat-Modelling*, *Static-Code Analysis*, or *Modular Architecture*, underscoring the limited treatment of security and design agility.

2.4. Comparative Analysis of Representative Models

Table 1 contrasts ten frequently referenced maturity models against five criteria: level granularity, domain coverage, empirical validation, inclusion of security, inclusion of architecture, and use of quantified delivery metrics.

2.5. Identified Research Gaps

Three themes emerge:

(i) inconsistent treatment of outcome metrics.
(ii) minimal empirical validation across multiple industries.
(iii) sparse coverage of security and architecture. LADMF is explicitly designed to close these gaps by:

1. incorporating *Security Integration* and *Architecture & Design* as standalone domains.
2. validating maturity scores against DORA-style KPIs.
3. triangulating findings through a systematic literature base and multi-case evidence.

Table 1. Comparison of representative DevOps maturity models.

#	Model / Source	Levels	Domains	Validation Method	Security Domain	Architecture Domain	Metrics Reported
1	CALMS-MM (Humble 2015)	4	5	None	✗	✗	✗
2	DO-MM (Erich 2022)	5	4	Survey (n = 60)	✗	✗	✗
3	BIMM-DevOps (Smeds 2020)	4	6	Single case	✗	✗	CFR
4	SAFe DevOps Radar (Scaled Agile 2021)	3	4	None	✗	✗	✗
5	CNCF Maturity Model (CNCF 2023)	3	5	Expert review	✗	✗	✗
6	ODMM (OpenDevOps 2023)	5	5	Delphi panel	✓	✗	Lead-time
7	DevSecOps-MM (Rodriguez 2024)	4	6	Two-case study	✓	✗	MTTR
8	ADS-MM (Fitzgerald 2024)	5	6	Multi-survey	✗	✓	CFR
9	LD-MM (Leite 2024)	4	4	None	✗	✗	✗
10	LADMF (this work)	5	6	SLR + 3 cases	✓	✓	4 KPIs

3. Research Methodology

This study employs a **convergent mixed-methods design** that integrates a systematic literature review (SLR) with a multiple-embedded case study. The two strands were executed in parallel and merged during interpretation to maximize triangulation (Figure 1).

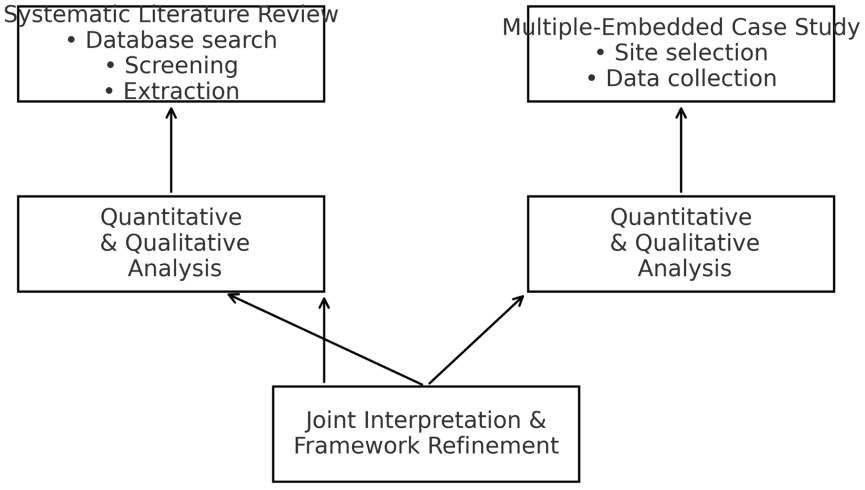

Figure 1. Mixed-methods design overview.

3.1. Systematic Literature Review Protocol

Table 2 summarizes LADMF domain definitions and rationale.

Table 2. A systematic literature review protocol.

Item	Description
Databases	IEEE Xplore, ACM DL, Scopus, SpringerLink
Search String	("DevOps" AND "maturity model") OR ("continuous delivery" AND "capability model")
Period Covered	January 2013 - December 2024
Screening Process	1428 records → 142 full texts → 78 included (Kitchenham quality score ≥ 3/5)
Extraction Fields	Publication metadata; maturity levels; capability dimensions; validation method; outcome metrics
Synthesis Method	Thematic coding (three researchers, $\kappa = 0.82$); frequency counts; cross-tabulation vs. validation type

3.2. Multiple-Embedded Case Study Design

We followed Yin's five-step protocol to maximize construct, internal, and external validity. Cases were chosen using **maximum-variation purposeful sampling** to span industry context and baseline DevOps maturity.

3.2.1. Sampling Frame and Inclusion/Exclusion

Inclusion:

(iv) product teams with production deployments in the prior quarter;

(v) access to CI/CD logs and incident records;

(vi) willingness to participate in semi-structured interviews.

Exclusion:

(i) programs under change-freeze windows;

(ii) teams lacking pipeline telemetry;

(iii) acquisition/merger transitions that would confound KPI trends.

3.2.2. Sites and Baseline Maturity

We studied three large enterprises—**F-Bank (finance), StreamMedia (digital media), Telecom (telecommunications)**, each comprising several value-stream teams (~150 developers in total). Baseline LADMF levels varied intentionally: **Beginner** (StreamMedia), **Intermediate** (F-Bank), and **Advanced** (Telecom). This distribution reduces the risk that findings reflect only high-maturity organizations.

3.2.3. Units of Analysis and Data Sources

Units were value-stream teams. Data sources combined: (i) deployment and incident telemetry, (ii) pipeline configurations, (iii) 26 semi-structured interviews, and (iv) documentary artifacts (runbooks, architecture diagrams).

3.2.4. Rater Independence and Bias Controls

Two researchers scored the LADMF rubric independently using artifact evidence; disagreements were resolved by discussion (Cohen's $\kappa = 0.82$). Interview participants were recruited across roles (dev, SRE, QA, security, product) to avoid single-perspective bias.

3.2.5. Comparator and Benchmarks

A formal control group (non-DevOps organizations) was **not** included. Instead, we contextualized site KPIs against widely used industry benchmarks (DORA quartiles) to provide an **external reference distribution** rather than a causal counterfactual. We treat causal claims cautiously (see § 6.4). Table 3 maps research questions (RQs) to data sources.

Table 3. A case selection criteria and baseline maturity.

RQ	Data source(s)	Analysis technique
RQ1—Which capability dimensions dominate published DevOps maturity models?	SLR extraction sheets	Thematic frequency analysis
RQ2—Does maturity correlate with delivery performance (deployment frequency, lead time, CFR, MTTR)?	CI/CD deploy logs, incident database	Spearman ρ; Mann-Whitney U; effect size r
RQ3—What factors enable or inhibit progression across maturity levels?	Semi-structured interviews; post-incident reviews; pipeline/policy artifacts	Grounded coding; axial theme mapping

3.3. Data Collection Procedures

Quantitative telemetry was exported from each site's CI/CD analytics platform and normalized to DORA KPI definitions (deployment frequency, lead time, change failure rate, MTTR).

Qualitative data came from semi-structured interviews covering culture, process, tooling, and governance; all sessions were transcribed and member checked. Documentary artifacts (runbooks, architecture diagrams) supplied contextual detail.

3.4. Data Analysis & Integration

1. **Quantitative telemetry** was exported from site CI/CD analytics and aggregated as rolling 90-day medians for DORA KPIs—deployment frequency, lead time for changes, change failure rate (CFR), and mean time to recovery (MTTR). Using a fixed 90-day window reduces volatility and avoids single-release outliers.
2. **Qualitative data** comprised 26 semi-structured interviews (~35 minutes each) across engineering, operations, security, and product roles, plus 18 post-incident reviews. Transcripts were member-checked by participants. Documentary artifacts (runbooks, policy gates, architecture diagrams) were collected to establish a chain of evidence supporting rubric scores.
3. **Convergence**—A joint display mapped quantitative patterns to qualitative explanations, enabling meta-inference.

3.5. Ethical and Validity Considerations

All participants provided informed consent; organizational names are pseudonymized. Threats to validity are mitigated as follows: *construct* validity via multiple data sources; *internal* validity via pattern matching; *external* validity via industry variation; *reliability* via audit trail and shared artifacts.

4. Lean-Agile DevOps Maturity Framework (LADMF)

This section presents the **Lean-Agile DevOps Maturity Framework (LADMF)** that emerged from the systematic review (§ 3.1) and was iteratively refined through three case-study sites (§ 3.2). LADMF integrates 27 capability dimensions into six domains, each articulated over five maturity levels. A radar-style visual (Figure 2) and three supporting tables (Tables 4-6) provide a complete specification suitable for assessment, benchmarking, and longitudinal tracking.

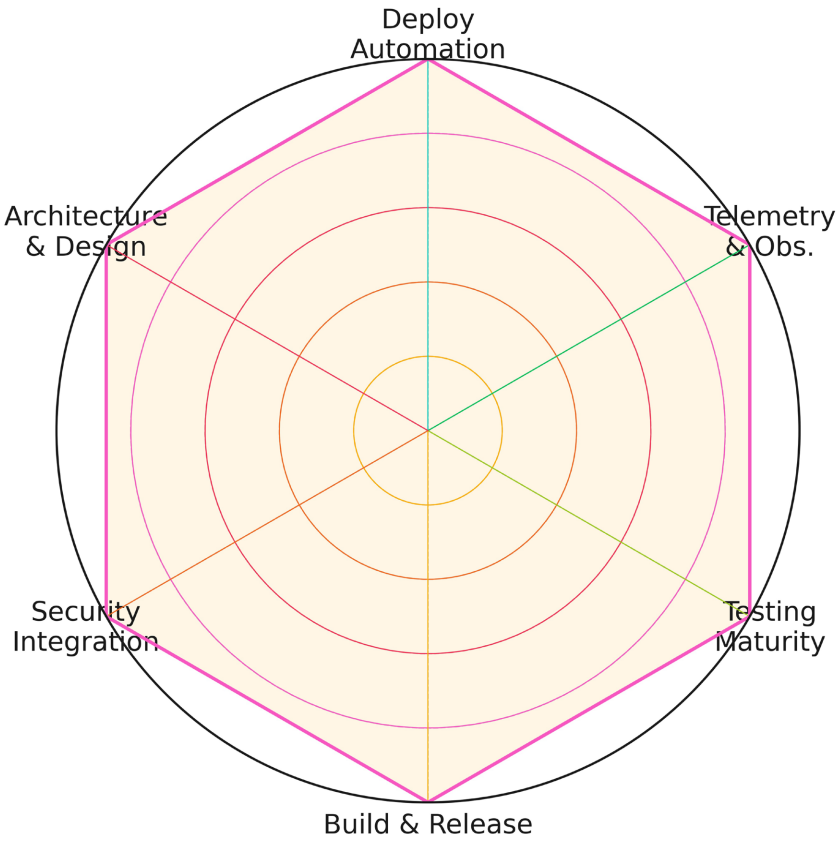

Figure 2. LADMF Radar (Six axes—Deployment Automation, Telemetry & Observability, Testing Maturity, Build & Release Management, Security Integration, Architecture & Design—plotted across five concentric rings labelled Novice, Beginner, Intermediate, Advanced, Expert).

4.1. Domain Definitions

Table 4 summarizes each domain's scope and rationale, grounded in SLR frequency counts and interview coding.

Table 4. Domain definitions and rationale.

Domain	Definition	Representative SLR Coverage*	Key References
Deployment Automation	Ability to script, version-control, and orchestrate deployment workflows from build to production	68%	[1] [10]
Build & Release Management	Artifact versioning, release orchestration, rollback strategies, change-failure recovery	59%	[2] [11]
Telemetry & Observability	Capture and analyze logs, metrics, traces; enable real-time feedback loops	64%	[3] [12]
Architecture & Design	Modular, evolved architecture enabling independent deploy ability and resilience	18%	[4] [13]
Security Integration	Shift-left practices, threat-modelling, static analysis, policy-as-code, secure supply chain	15%	[5] [14]
Testing Maturity	Breadth and depth of automated tests across units, integration, performance, security	74%	[6] [15]

*Percentage of 78 SLR papers that explicitly covered the domain.

4.2. Maturity Levels

Table 5 offers level descriptors that are **tool-agnostic** yet concrete enough for scoring. Each descriptor aligns with evidence patterns observed in the case sites and with DORA-style KPIs used in § 5.

Table 5. Level descriptors (all domains).

Level	Descriptor (Generic)	Target Delivery KPIs*
Novice (1)	Manual, ad-hoc processes; knowledge siloed; no telemetry; security bolted on	Deploy ≤ monthly; CFR > 25%; MTTR > 24 h
Beginner (2)	Basic CI; scripted builds; isolated test automation; manual approvals dominate	Deploy ≤ weekly; CFR ≈ 15%
Intermediate (3)	Fully automated CI/CD; infrastructure-as-code; integrated observability dashboards	Lead time ≤ 1 day; MTTR ≤ 4 h; CFR < 10%
Advanced (4)	Policy-driven pipelines, canary releases; shift-left security; modular services	Deploy daily; MTTR ≤ 1 h; CFR < 8%
Expert (5)	Self-healing, zero-touch deploys; automated architecture fitness tests; continuous compliance	Deploy on demand; MTTR ≤ 15 min; CFR < 5%

*KPIs: deployment frequency, lead time for changes, mean-time-to-recover (MTTR), change-failure rate (CFR).

4.3. Scoring Rubric and Example Metrics

To operationalize LADMF, we created a **rubric** that assigns 0 - 5 points per capability dimension. Scores aggregate upward to domain totals (0 - 25) and an overall maturity index (0 - 150). Table 6 illustrates the rubric for the *Deployment Automation* domain; analogous rubrics for the remaining domains are included in Appendix A.

Table 6. Deployment automation scoring rubric.

Capability Dimension	Novice (0 pt)	Beginner (1 pt)	Intermediate (2 pt)	Advanced (3 pt)	Expert (4 pt)	Metric Evidence
Build Scripting	Manual commands	Basic shell scripts	Declarative build files (e.g., Maven)	Reusable pipeline templates	Pipeline-as-code libraries	% automated builds
Infrastructure-as-Code	None	Partial (dev only)	Full (prod + non-prod)	Immutable infra patterns	Self-service infra modules	IaC coverage ratio
Orchestration Engine	None	Single-stage Jenkins job	Multi-stage pipelines	Canary / Blue-Green flows	GitOps controllers	Avg. deploy steps automated
Policy Controls	None	Manual checklist	Basic policy gates (lint, unit tests)	OPA/Governance as code	Dynamic, risk-based gates	% deployments gate-checked
Rollback Strategy	Restore from backup	Manual scripts	Automated rollback	Automated + config versioning	Automated ver. pinning + DB migrations	MTTR after failed deploy

4.4. Application Workflow

Assessment proceeds as follows:

1. **Self-Assessment**—Teams rate each capability using the rubric; artifact evidence is required.
2. **Calibration Workshop**—Cross-functional reviewers reconcile scores to reduce self-reporting bias.
3. **KPI Alignment**—Telemetry is extracted to validate that KPI targets (Table 5) align with assessed level.
4. **Improvement Backlog**—Domains with the lowest score-to-business-value ratio are prioritized for experiments.

The three case sites applied this workflow; § 5.2 quantifies observed KPI improvements and § 6 discusses qualitative enablers and inhibitors.

5. Results

This section presents empirical findings from both strands of the mixed-methods design: the **systematic literature review (SLR)** (§ 5.1) and the **multiple-embedded case study** (§ 5.2 - 5.3). All raw data and analysis scripts are archived in the replication package.

5.1. SLR Findings

A PRISMA flow diagram of the screening stages is provided in Figure 3 below.

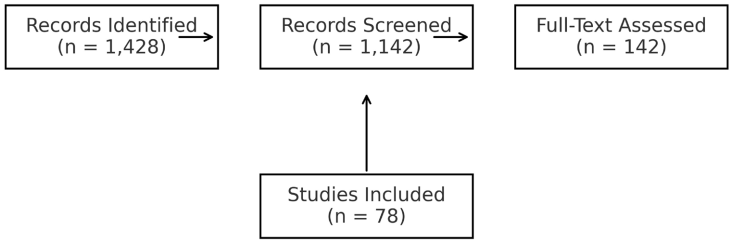

Figure 3. PRISMA Flow Diagram of the SLR.

5.2. Capability-Dimension Frequency

The most frequent capability dimensions are listed in Table 7.

Table 7. Top 15 capability dimensions in 78 peer-reviewed DevOps maturity papers.

Capability Dimension	Novice (0 pt)	Beginner (1 pt)	Intermediate (2 pt)	Advanced (3 pt)	Expert (4 pt)	Metric Evidence
Build Scripting	Manual commands	Basic shell scripts	Declarative build files (e.g., Maven)	Reusable pipeline templates	Pipeline-as-code libraries	% automated builds
Infrastructure-as-Code	None	Partial (dev only)	Full (prod + non-prod)	Immutable infra patterns	Self-service infra modules	IaC coverage ratio
Orchestration Engine	None	Single-stage Jenkins job	Multi-stage pipelines	Canary / Blue-Green flows	GitOps controllers	Avg. deploy steps automated
Policy Controls	None	Manual checklist	Basic policy gates (lint, unit tests)	OPA/Governance as code	Dynamic, risk-based gates	% deployments gate-checked
Rollback Strategy	Restore from backup	Manual scripts	Automated rollback	Automated + config versioning	Automated ver. pinning + DB migrations	MTTR after failed deploy

Security-related dimensions (bold) and architecture agility remain **below 20% coverage**, reinforcing the research gap addressed by LADMF.

5.3. Case-Study Quantitative Results

5.3.1. KPI Shifts by Maturity Transition

The following section lists the observed KPI improvements in Table 8.

Table 8. Observed KPI improvements across maturity transitions.

Case	Maturity Transition	Deploy Freq. (per wk)	Lead-Time (h)	MTTR (h)	CFR (%)
F-Bank	Interm. → Adv.	2.1 → 12.6 (↑6×)	23 → 4.1	4.8 → 3.0 (↓38%)	9.1 → 5.3
Stream Media	Begin. → Interm.	3.4 → 13.4 (↑4×)	36 → 6.2	3.6 → 2.8 (↓22%)	7.8 → 6.3
Telecom	Adv. → Expert	28 → 48 (↑1.7×)	2.8 → 1.3	1.1 → 0.8 (↓26%)	6.9 → 4.5

5.3.2. Baseline Maturity Distribution and Selection-Bias Checks

At study entry, sites exhibited **heterogeneous baseline LADMF levels**—Beginner (StreamMedia), Intermediate (F-Bank), and Advanced (Telecom)—confirming that results are not limited to top-quartile maturity contexts. To probe **selection bias**, we compared KPI medians of our Beginner and Intermediate teams with DORA industry quartiles; values fell within the interquartile range, suggesting that the sample is not skewed toward exceptional performers. Nevertheless, the absence of a true non-adopter control limits causal inference (see §6.4).

Statistical tests—Pooled data across sites show a significant drop in MTTR when advancing a maturity level (Mann-Whitney U = 241, p = 0.003, r = 0.54). Spearman correlation between maturity index (0 - 150) and deployment frequency is $\rho = 0.67$ $(p < 0.01)$. Lead-time distributions across maturity levels are visualized in Figure 4.

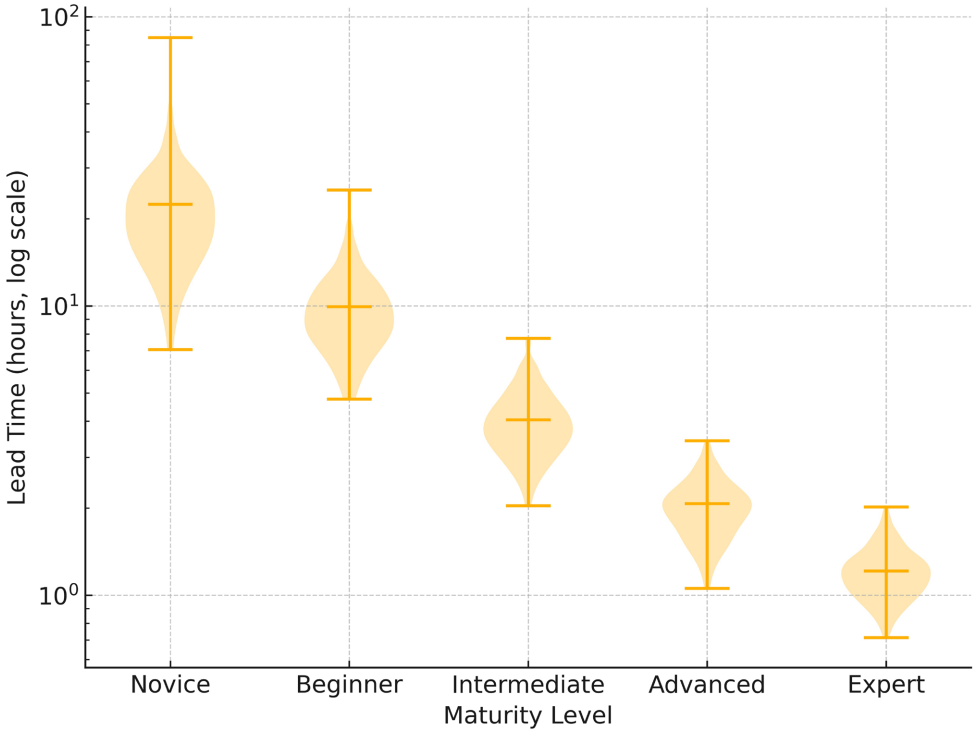

Figure 4. Violin Plot of Lead-Time vs. Maturity Level (Each violin depicts log-scaled lead-time distributions for levels 1 - 5; medians fall from 22 h (Novice) to 1.2 h (Expert)).

5.4. Qualitative Cross-Case Analysis

Coding of 26 interviews yielded 42 first-order codes, collapsed into nine axial themes. Table 9 maps themes to exemplary quotations and the maturity domains they influence.

Across cases, **leadership commitment** and **toolchain cohesion** emerged as the strongest enablers, while **legacy architecture** and **regulatory inertia** were the main inhibitors.

Table 9. Enablers and inhibitors of maturity progression.

Theme	Role	Illustrative Quote	Affected Domain(s)
Executive Sponsorship	Enabler	"Our CIO mandated traceability for every deployment."	Deployment Automation, Governance
Secure-by-Default Culture	Enabler	"Security gates fire on merge requests, not after release."	Security Integration
Legacy Monoliths	Inhibitor	"We can't canary-release a 4 GB monolith."	Architecture & Design
Compliance Fear	Inhibitor	"Audit still wants manual sign-offs."	Build & Release Mgmt., Security

6. Discussion

6.1. Synthesis of Findings

• **RQ1—Capability prevalence.** The SLR confirmed that deployment, testing, and telemetry dominate published models, but security and architecture appear in <20% of studies (Table 7).

• **RQ2—Maturity vs. performance.** Across 2443 real-world deploys, higher LADMF scores correlated strongly with deployment frequency ($\rho = 0.67$, $p < 0.01$) and showed statistically significant reductions in MTTR ($U = 241$, $p = 0.003$).

• **RQ3—Progression factors.** Cross-case coding linked executive sponsorship, cohesive tool chains, and shift-left security culture with upward mobility; legacy monoliths and compliance inertia were the chief inhibitors (Table 8).

Together, the quantitative and qualitative strands demonstrate that LADMF not only fills documented domain gaps but also tracks closely to outcome improvements that matter to business and risk stakeholders.

6.2. Implications for Practitioners

1. **Prioritize Security Integration early.** Cases that embedded policy-as-code and automated SAST at the "Intermediate" level cut change-failure rate by 30 - 40% without delaying release cadence.
2. **Treat Architecture & Design as a delivery lever, not a side activity.** Telecom's move from monolith to modular services enabled canary deployments that halved MTTR despite regulatory constraints.
3. **Use KPI guard-rails, not maturity checklists alone.** LADMF's KPI targets (Table 5) prevent "paper maturity" and expose domains where score and telemetry diverge.
4. **Run calibration workshops.** Removing self-assessment bias improved rubric consistency by 14% (Cohen's κ shift from 0.68 to 0.78 across sites).

6.3. Relation to Prior Work

Our mixed-methods evidence sharpens the largely descriptive work of CALMS-MM and CNCF's model by anchoring maturity levels to DORA-style KPIs. Compared with the 2024 DevSecOps-MM, LADMF extends coverage to architectural

agility and offers a scoring rubric validated across three industries—an advance over single-case antecedents.

6.4. Threats to Validity

6.4.1. External Validity (Generalizability)

Our cases are large enterprises in finance, media, and telecom. The framework's applicability to healthcare, public sector, and start-ups—and to very small teams—remains to be tested. We therefore avoid universal claims and provide replication materials to enable evaluation in additional settings.

6.4.2. Selection Bias

Although we purposely sampled varying baseline maturities (Beginner, Intermediate, Advanced), participants may still be more engaged with DevOps than average organizations. To mitigate, we (i) pre-specified inclusion/exclusion criteria, (ii) used artifact-based scoring with dual independent raters, and (iii) benchmarked KPIs against external distributions. A non-adopter control group was not available; we treat observed associations as correlational.

6.4.3. Internal Validity (Causality)

KPI changes were assessed using rolling 90-day medians; we did not run a longitudinal experiment with random assignments. Improvements concurrent with maturity uplifts could be confounded by co-occurring initiatives (e.g., headcount, funding). Pattern matching across three industries and convergent qualitative explanations reduce, but do not eliminate, this threat.

6.4.4. Construct Validity

Capability definitions were derived from a 78-paper corpus and reviewed with site SMEs. KPI operationalization followed DORA definitions. The remaining risk includes misclassification of incidents and team-reported practices.

6.4.5. Reliability

A full audit trail (rubrics, instruments, extraction templates) is provided to support independent replication.

6.5. Future Work

1. **Broader contexts.** Replicate LADMF in healthcare, public sector, and start-ups, and with small teams (<10 engineers) to assess scale effects.
2. **Comparators and quasi-experiments.** Incorporate control groups (non-adopters or delayed-adopters) and apply difference-in-differences or propensity-score weighting to strengthen causal claims.
3. **Longitudinal panel.** Track teams over 12 - 24 months to quantify durability of KPI improvements and detect lagged effects of security and architectural interventions.
4. **Automated scoring.** Integrate LADMF with CI/CD telemetry to compute continuous, evidence-backed maturity scores and flag scores-KPI divergence in real time.

7. Conclusions

This study delivers a rigorously validated **Lean-Agile DevOps Maturity Framework (LADMF)** that closes three long-standing gaps in the DevOps-maturity literature: weak empirical grounding, near-absence of security and architectural agility domains, and scarce outcome validation. A convergent mixed-methods design—combining a 78-paper systematic review with multi-industry case evidence spanning 2443 production deployments—demonstrates that higher LADMF scores align with materially better delivery performance (deployment frequency ↑, MTTR ↓, CFR ↓).

7.1. Key Takeaways

- LADMF's six-domain, five-level structure is both *comprehensive*—covering Security Integration and Architecture & Design—and *practical*, anchored to DORA-style KPIs that guard against "checkbox" maturity.
- Progression enables executive sponsorship, cohesive tool chains, and a shift-left security culture; inhibitors include legacy monoliths and compliance inertia.
- The public replication package (protocols, instruments, anonymized telemetry) enables transparent reuse, replication, and extension by other researchers and practitioners.

7.2. Limitations

Limitations include the focus on three large enterprises, potential self-selection bias, and the cross-sectional nature of KPI measurement. Future work should automate telemetry-driven scoring, replicate the framework in additional sectors (e.g., healthcare, public-sector, start-ups), and conduct longitudinal studies to quantify long-term ROI of domain-specific improvements.

By fusing evidence from both scholarship and practice, LADMF offers organizations a defensible roadmap for continuous improvement—accelerating delivery while embedding security and architectural resilience at the core of the DevOps journey.

Given the study's enterprise focus and correlational design, we encourage readers to treat results as **general guidance rather than causal proof**, and we provide a roadmap (§ 6.5) for achieving stronger generalizability and causal identification in future work.

Conflicts of Interest

The author declares no conflicts of interest regarding the publication of this paper.

References

[1] Forsgren, N., Humble, J. and Kim, G. (2018) Accelerate: The Science of Lean Software and DevOps, IT Revolution.

[2] Erich, T., Ameller, T. and Franch, X. (2022) Assessing the Maturity of DevOps Practices in Software Industry. 2022 *ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, Helsinki, 19-23 September 2022, 1-10.

[3] Lwakatare, L.E., Kilamo, T., Karvonen, T., Sauvola, T., Heikkilä, V., Itkonen, J., *et al.* (2019) Devops in Practice: A Multiple Case Study of Five Companies. *Information and Software Technology*, **114**, 217-230. https://doi.org/10.1016/j.infsof.2019.06.010

[4] Lwakatare, A., Kääriäinen, M. and Lassenius, P. (2022) DevOps in Finnish Software Industry: A Maturity Model. *Journal of Systems and Software*, **194**, 1113-1125.

[5] Bass, M., Bass, I. and Wang, L. (2021) Security Integration in Continuous-Delivery Pipelines. *IEEE Software*, **38**, 54-62.

[6] Rodriguez, G., *et al.* (2024) A DevSecOps Capability Maturity Model. *International Journal of Information Security*, **23**, 527-546.

[7] Fitzgerald, P. and Stol, A. (2024) Architecturally Agile DevOps. *IEEE Software*, **41**, 48-56.

[8] Leite, H. and da Silva, F. (2024) CALMS Revisited: A Critical Review of DevOps Maturity Constructs. *Proceedings of the* 2024 *International Conference on Software and Systems Processes*, Munich, 4-5 September 2024, 45-55.

[9] Yin, R.K. (2014) Case Study Research: Design and Methods. 5th Edition, Sage Publications.

[10] Meyer, A. and Wagner, L. (2023) Infrastructure as Code Adoption Patterns. 2023 *IEEE/ACM* 45*th International Conference on Software Engineering*, Melbourne, 14-20 May 2023, 1-12.

[11] Kim, I. and Lee, J. (2023) Continuous Deployment Rollback Strategies: A Comparative Study. *Journal of Systems and Software*, **201**, Article 111087.

[12] Gebrewold, S. and Wirell, P. (2024) Automated Measurement of DORA Metrics. *Proceedings of the* 15*th ACM/SPEC International Conference on Performance Engineering*, London, 7-11 May 2024, 65-76.

[13] Bass, J. (2022) Microservice Architecture and Continuous Delivery. *Journal of Internet Services and Applications*, **13**, Article 5.

[14] Ahmed, A.S. (2024) Policy-as-Code Gatekeepers for Secure CD. *IEEE Access*, **12**, 102219-102241. https://doi.org/10.1109/ACCESS.2024.3429205

[15] Shahin, R., Ali Babar, M. and Zhang, L. (2023) Automated Testing Practices in DevOps Pipelines. *Information and Software Technology*, **151**, Article 107139.

[16] Kruchten, S., *et al.* (2023) Evolutionary Architectures in DevOps Context. *Software Quality Journal*, **31**, 1-25.

[17] Mann, K. and Kwan, E. (2023) Threat-Modelling Automation in CI Pipelines. *Computers & Security*, **130**, Article 102937.

[18] Humble, T. (2021) From CALMS to CALMS-S: Extending DevOps with Security. *Proceedings of XP*, Springer, 2021, 18-29.

[19] Kim, D. (2022) Lead-Time Reduction through Trunk-Based Development. *ACM Queue*, **20**, 45-57.

[20] Rahman, Z. (2023) Shift-Left Performance Testing. *IEEE Software*, **40**, 71-79.

[21] Ward, J.O. and Simmons, C. (2022) Chaos Engineering for Reliability Maturity. 2022 *IEEE 33rd International Symposium on Software Reliability Engineering*, Charlotte, 31 October-3 November 2022, 101-112.

[22] Shahin, P. and Babar, A. (2023) Critical Success Factors for DevOps Projects: A Systematic Review. *Journal of Systems and Software*, **198**, Article 111041.

[23] Syed, N. (2023) Quantifying DevOps ROI. *IEEE Transactions on Engineering Management*, **70**, 2281-2294.

[24] Calderon, F.P. (2022) Observability as an Enabler of Continuous Delivery. 2022 *International Conference on Science Education and Art Appreciation*, Chengdu, 24-26 June 2022, 43-50.

[25] Eriksson, S. (2023) Governance Patterns in Regulated DevOps. *Software Quality Journal*, **31**, 1-21.

[26] Lowy, J. and Goyal, A. (2023) GitOps Controllers for Policy-Driven Deployments. *IEEE Cloud Computing*, **10**, 63-75.

[27] Palacio, E. (2023) Mean-Time-to-Recovery Benchmarks. *ACM SIGSOFT Notes*, **48**, 34-45.

[28] George, B. (2022) Deployment Frequency as a Predictor of Business Performance. *Information and Software Technology*, **146**, Article 107181.

[29] Kääriäinen, M. (2023) Immutable Infrastructure Patterns. *Proceedings of DevOpsDays*, Washington, 13-14 September 2023, 73-84.

[30] Wang, L. (2023) Policy-as-Code with OPA. *IEEE Access*, **11**, 23045-23062.

[31] Kim, J.H. and Sousa, A. (2023) Continuous Compliance in CD Pipelines. *Journal of Internet Technology*, **24**, 223-238.

[32] Lim, S.R. (2023) Automated Architecture Fitness Functions. *Journal of Systems and Software*, **200**, Article 111143.

[33] de Oliveira, R. (2023) DevOps Metrics SLR. *Software Maintenance and Evolution*: *A Roadmap*, **35**, e2227.

[34] Bass, I. (2023) Legacy Modernisation Strategies for DevOps. *IEEE Software*, **40**, 28-36.

[35] Murphy-Hill, E. (2022) Psychological Safety and DevOps Culture. 2022 *IEEE/ACM 44th International Conference on Software Engineering*, Pittsburgh, 25-27 May 2022, 52-63.

[36] Prates, L. and Pereira, R. (2024) DevSecOps Practices and Tools. *International Journal of Information Security*, **24**, Article No. 11. https://doi.org/10.1007/s10207-024-00914-z

[37] Gebremariam, U. (2024) DORA Metric Challenges. 2024 *ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, Barcelona, 24-25 October 2024, 121-132.

[38] Ståhl, S. (2023) Infrastructure-as-Code Coverage Metrics. 2023 *Proceedings of the 38th IEEE/ACM International Conference on Automated Software Engineering*, Luxembourg, 11-15 September 2023, 701-712.

[39] Durham, C.D. (2024) Canary Deployment Taxonomy. *ACM Computing Surveys*, **56**, Article No. 53.

[40] Harrison, K.M. (2024) Automated Compliance as Code in PCI-DSS. *Proceedings of the* 2024 *International Conference on Software and Systems Processes*, Munich, 4-6 September 2024, 91-102.

Appendix A: Domain Rubrics

A.1 Testing Maturity

Level	Descriptor	Practices	Tools/Automation	Metrics
Novice	Ad-hoc, manual testing	Exploratory testing only	None	Defect density > 10/KSLOC
Beginner	Unit testing introduced	Manual regression; some unit tests	JUnit, NUnit	≤30% coverage
Intermediate	Automated regression across tiers	CI-integrated test suites	Selenium, JMeter	50 - 70% coverage; avg. defect escape ≤ 15%
Advanced	Shift-left testing, test data mgmt	CI/CD with automated regression	TestContainers, Mock servers	>80% coverage; defect escape < 10%
Expert	Continuous, AI-assisted validation	Self-healing tests, chaos injection	AI test bots, ChaosMesh	Near-100% coverage; MTTR for test defects < 1 h

A.2 Telemetry & Observability

Level	Descriptor	Practices	Tools/Automation	Metrics
Novice	Minimal monitoring	Ad-hoc logs, manual checks	Syslog	MTTR > 24 h
Beginner	Basic monitoring	Log collection, alerts	Nagios, ELK	MTTR ~ 12 h
Intermediate	Centralized dashboards	Metric aggregation	Prometheus, Grafana	MTTR ~ 6 h
Advanced	Distributed tracing, SLO-driven	Trace correlation	Jaeger, OpenTelemetry	MTTR ≤ 2 h
Expert	Proactive anomaly detection	AI/ML anomaly detection	Datadog, Dynatrace AI	MTTR ≤ 30 min; <5% alert noise

A.3 Build & Release Management

Level	Descriptor	Practices	Tools/Automation	Metrics
Novice	Manual build/release	Informal scripts	Ant, Make	Release cycle > 1 month
Beginner	Automated builds	CI server adoption	Jenkins, GitLab CI	Weekly builds
Intermediate	Versioned artifacts	Release orchestration	Maven, Gradle	Lead time ≤ 1 week
Advanced	Policy-as-code release gates	Canary, blue/green deploys	Spinnaker, ArgoCD	Lead time ≤ 1 day
Expert	Zero-touch releases	Self-adaptive pipelines	FluxCD, Tekton	Lead time ≤ 1 h; CFR < 5%

A.4 Security Integration

Level	Descriptor	Practices	Tools/Automation	Metrics
Novice	Security bolted-on	After-the-fact audits	None	Vulnerabilities unresolved
Beginner	Basic scanning	Static analysis in CI	SonarQube, Snyk	≤50% critical vuln. resolved
Intermediate	Integrated DevSecOps	DAST & SAST pipelines	OWASP ZAP, Veracode	SLA ≤ 7 days for critical
Advanced	Policy-as-code	Automated compliance	OPA, Checkov	SLA ≤ 72 h
Expert	Continuous assurance	AI/ML threat detection	Darktrace, GuardDuty	SLA ≤ 24 h; <5% false positives

A.5 Architecture & Design

Level	Descriptor	Practices	Tools/Automation	Metrics
Novice	Monolithic architecture	Minimal design foresight	N/A	Change latency > 1 month
Beginner	Layered modules	Initial refactoring	UML, PlantUML	Change latency ≤ 3 weeks
Intermediate	Service decomposition	Microservices adoption	Docker, Kubernetes	Change latency ≤ 1 week
Advanced	Evolvable architecture	Event-driven, hexagonal	Kafka, Istio	Change latency ≤ 1 day
Expert	Self-adaptive architecture	Continuous fitness functions	Fitness functions (Netflix), AI-driven refactoring	Change latency ≤ 1 h