The Role of AI in Software Release Management

Muhammad Faizan Tahir, Andrey Saltan[0000−0002−7921−986X], and Sami Hyrynsalmi[0000−0002−5073−3750]

LUT University, Lahti, Finland
faizanm88@yahoo.com, {andrey.saltan, sami.hyrynsalmi}@lut.fi

Abstract. Software release management coordinates planning, gating, deployment and monitoring in CI/CD-intensive environments where rapid iteration compresses decision windows and increases risk. This paper examines how *Analytical AI* and *Generative AI* support those decisions. Based on qualitative interviews with industry experts, we identify key benefits of AI integration, such as increased automation of repetitive tasks, more efficient resource allocation, enhanced risk forecasting, and improved over-all decision-making. These advances support faster, more reliable software deployments by streamlining processes and enabling data-driven insights. However, the study also highlights significant hurdles to effective AI/ML adoption, including integration complexity, the opacity of AI models, and organizational resistance. Addressing these challenges proves essential for realizing the full potential of AI in software release management. By examining both the practical advantages and limitations, this research offers actionable guidance for industry practitioners and contributes to broader discussions on the responsible application of AI in contemporary software product management.

Keywords: Release Management · Analytical AI · Generative AI

1 Introduction

Software release management links product intent with operational reality: it plans scope and cadence, stabilizes changes, governs deployment, and monitors outcomes so that evolving requirements are delivered reliably and on time [8, 9]. In Agile/DevOps settings, releases are smaller and more frequent, supported by CI/CD pipelines and extensive automation [6, 4, 5]. Shorter lead times enrich feedback but also push more choices toward production, intensifying coordination across engineering, operations, security, and product [3]. Under tight decision windows, release gates must integrate heterogeneous signals — quality, reliability, interoperability, security, compliance, and market timing — while remaining auditable and aligned with organizational risk tolerances.

Recent advances in Artificial Intelligence offer new instruments for these decisions. To avoid terminological ambiguity, we use *AI* as an umbrella and distinguish two classes relevant to release work. *Analytical AI* denotes predictive and diagnostic models — often labeled "ML" — that learn from historical and live telemetry to forecast failure risk, prioritize tests and fixes, detect anomalies,

and optimize schedules and resources. *Generative AI* denotes language and code models that produce human-readable text and code, useful for summarizing logs and change impact, drafting release communications and runbooks, assisting code reviews and test scaffolds. This complements long-standing SPM work that frames release management as balancing stakeholder insight with computational decision support [7].

Adopting either type of AI is nontrivial. Analytical AI models depend on representative, well-labeled data and stable telemetry schemas; Generative AI systems raise questions of accuracy, provenance, and governance. Both surface explainability and accountability concerns that shape stakeholder trust and regulatory posture [1]. Integration with existing CI/CD toolchains, decision rights, and organizational cadences is itself an engineering and change-management effort [6, 3].

This paper develops an integrated, evidence-based understanding of how AI supports decision-making in software release management and what limits its effective use. We investigate the problem empirically through a qualitative study of seven industry professionals from South Asian country (product managers, senior software engineers, and a director of engineering) using semi-structured interviews and thematic analysis. Across cases, automation/efficiency and predictive capability are the most frequently cited benefits, while integration complexity, transparency/black-box concerns, data quality, and human/organizational factors dominate the constraint set.

The research is driven by two research questions that emphasize decision quality and practical feasibility: *RQ1: In what concrete ways do Analytical and Generative AI improve decision quality in software release management? RQ2: What technical, organizational, and governance constraints impede successful adoption of Analytical and Generative AI in software release management?*. The scope of this early stage study is deliberately decision-focused: we analyze how teams select features and content for releases, schedule and coordinate deployments, manage risk and compliance, and determine readiness across environments. We do not attempt an exhaustive evaluation of all development methodologies or platforms. Findings are qualitative and context-dependent; organizations vary widely in size, industry, regulatory posture, and toolchains, and the technology landscape continues to evolve. Nevertheless, the patterns we report offer actionable guidance for practitioners and a research agenda for the community.

2 Background

Software release management is the structured coordination of planning, stabilizing, scheduling, deploying, and monitoring software changes so that evolving requirements are integrated safely and on time [8, 9]. In plan-driven lifecycles these activities are largely sequential; in Agile/DevOps they interleave to enable frequent iterations and rapid feedback [6]. Across both settings, release management links engineering and business concerns by deciding *what* is released, *when*

it is released, and *under which risk posture*, aligning code changes, operational readiness, compliance, and expectations.

Modern practice surrounds these decisions with automation. Version control coordinates parallel work; CI/CD pipelines build, test, and promote artifacts; infrastructure-as-code and containerization standardize environments; progressive delivery (canary, blue/green) limits blast radius [4–6]. These practices shorten lead times and reduce toil but also concentrate consequential choices near production: teams must pick cadence and scope, set gate thresholds, and decide when to pause, roll back, or proceed.

Decision-making remains a "wicked problem" — a pattern echoed in SPM research that calls for formalized, evidence-based processes to counter ad-hoc 'gut-feeling' choices [10]. Teams balance quality, reliability, interoperability, and security against market timing, customer commitments, and regulation [3, 11]. Classical prioritization presumes stable, comparable requirements, yet dependencies, shifting stakeholder preferences, and incident learnings render the space non-stationary [11]. In practice, decisions interleave across horizons: near-term gating (pass/fail thresholds, rollbacks), mid-term scheduling and scope, and longer-term risk posture (technical debt, observability, capacity).

Recent advances in Artificial Intelligence (AI) add new instruments. We use *AI* as an umbrella and distinguish two complementary types. *Analytical AI* — predictive/diagnostic modeling over historical and live telemetry — forecasts incident risk, prioritizes tests and fixes, detects anomalies, and optimizes schedules/resources. *Generative AI* — language/code models — summarizes change impact and logs, drafts release communications and runbooks, assists code reviews and test scaffolds, and supports conversational diagnosis. Operationally, Analytical AI models integrate at quantitative gates in CI/CD, whereas Generative AI systems augment human coordination and sense-making [2].

Adoption is gated by well-documented constraints. Analytical AI depends on reliable, representative telemetry; fragmented logs, sparse labels, and schema drift degrade signal-to-noise and yield brittle models. Integration with existing pipelines and approval flows adds engineering overhead; governance demands (auditability, privacy, safety) require explainable outputs where gate decisions affect users or regulated processes [1, 3]. Generative AI systems raise additional concerns around accuracy, provenance, and prompt-induced variability, making human-in-the-loop review essential for high-stakes steps. Organizational factors — skills, role clarity, change fatigue, and perceptions of "black-box" automation — shape trust and uptake as much as technical performance. These constraints motivate guardrails such as data governance and observability, explainability-by-design with evidence bundling, pilotization before scale, and native CI/CD integration.

In summary, release management decisions increasingly occur inside automated pipelines yet still require multi-objective trade-offs under uncertainty. Analytical AI can predict, prioritize, and detect at quantitative gates; Aenerative AI can summarize, document, and assist where language and coordination dominate. What remains under-specified — both in practice and in prior work —

Table 1. Study participants

Interview participant	Experience (years)	Role
Person A	3	Product Manager
Person B	5+	Sr. Software Engineer
Person C	5+	Director of Engineering
Person D	4	Product Manager
Person E	5+	Sr. Software Engineer
Person F	5+	Sr. Software Engineer — AI
Person G	5+	Sr. Software Engineer

is how these two types are combined in real release management, which decisions they reliably improve today, and which guardrails make adoption dependable. This study addresses these gaps empirically.

3 Methodology

This study *adopts* a qualitative, interpretivist multiple-expert design to explain how Analytical and Generative AI support decision-making in software release management and what constrains their adoption. We build on a data of seven semi-structured interviews and extend the analytic frame by tagging each AI reference as Analytical vs. Generative.

Sampling and context. We used purposeful sampling to recruit practitioners who (i) decide software release scope/cadence or own gating signals, (ii) operate with Agile/DevOps and CI/CD, and (iii) have more than 3 years of experience with software product management. Seven participants (A–G) span fintech, product, and services organizations (Table 1).

Interview protocol. Semi-structured interviews (40–45 minutes, video conference) followed a guide aligned with the literature review and the research questions. The interview covered: (1) current release management practices, cadences, and bottlenecks; (2) instrumentation and decision signals at each stage of the pipeline; (3) existing/planned use of AI; (4) perceived benefits and concrete examples; (5) constraints (data, integration, explainability, organizational readiness); (6) guardrails and adoption pathways (pilots, human-in-the-loop, audit trails). Probes elicited critical incidents (e.g., a recent rollback or canary failure) and asked respondents to walk through artifacts they use (dashboards, runbooks, release notes) so that claims could be anchored to specific practices.

Analysis approach. The empirical base comprises seven interview transcripts plus researcher memos. We performed reflexive thematic analysis in six phases, implemented in NVivo and documented with memos and versioned codebooks. First-cycle coding was largely open-coding to preserve practitioner language; code units ranged from a clause to a paragraph, with each segment optionally receiving multiple tags (e.g., *Analytical AI*, *gating*, *risk forecasting*). Second-cycle coding clustered first-order codes into conceptually distinct themes and then organized them into four reporting buckets: *benefits*, *constraints*, and *adoption practices/guardrails*.

Table 2. Weighted benefits codes by participant (A–G) and total

	A	B	C	D	E	F	G	Total
Automation	3	1	1	1	3	3	2	**14**
Efficiency	2	2	1	2	3	3	0	**13**
Decision-making (data-driven)	0	1	1	0	2	2	1	**7**
Predictive capabilities	2	1	0	1	1	1	0	**6**
Future promise of AI	3	1	1	1	0	0	0	**6**
Bug prediction	1	1	0	0	0	0	3	**5**
Prioritization	1	0	1	1	0	0	2	**5**
Strategic focus (time reallocation)	0	2	1	0	1	0	0	**4**
Code quality	1	0	0	0	0	0	3	**4**
Accuracy	1	2	0	0	0	1	0	**4**
Time-to-market	1	0	0	0	0	0	0	**1**

4 Results and Discussion

Benefits and promises of AI in software release management As depicted in Table 2, two themes dominate the coded benefits: *automation of repetitive work* in build–test–deploy and monitoring, and *efficiency* in throughput and cross-team coordination. Participants stressed that Analytical AI reduces manual checks by surfacing high-risk changes and noisy signals. "AI-powered monitoring tools can identify high-risk areas and reduce manual effort," noted Person A. Person B similarly emphasized that anomaly detection and test prioritization "enhance decision-making efficiency by processing large datasets." Beyond throughput, interviewees described improvements to *decision quality*: models help "find the real numbers" behind go/no-go and rollback calls by combining historical defects, pipeline telemetry, and user-impact indicators. Additional but less frequent codes capture *bug prediction*, *prioritization* of tests/fixes by expected risk reduction, *code-quality* gains via automated checks, and modest signals on *time-to-market*. Participants also expressed cautious optimism about the *future promise* of AI in release management, anticipating tighter CI/CD integration and more adaptive post-release monitoring. These forward-looking statements often coincided with requests for stronger telemetry and clearer rationales for recommendations.

Formally addressing RQ1 — how do Analytical and Generative AI improve decision quality in software release management? — improvements arise through three interacting mechanisms:

1. **Automation and shorter feedback cycles.** Analytical AI removes repetitive checks (test re-ordering, flaky-test suppression, canary comparisons) and turns diffuse telemetry into actionable signals at gates. This aligns with CI/CD practices and progressive delivery controls reported in industry.
2. **Predictive/diagnostic assistance.** Risk scoring based on historical incidents, change metadata, and runtime KPIs makes readiness and rollback decisions more evidence-based. This mechanism is consistent with work linking disciplined release management practices to improved delivery outcomes.

Table 3. Weighted challenges codes by participant (A–G) and total

	A	B	C	D	E	F	G	Total
Integration complexity	5	0	0	5	2	0	0	**12**
Transparency (explainability/trust)	0	2	1	2	0	4	1	**10**
Human factors (skills, resistance)	3	1	1	1	1	1	1	**9**
Pilot projects (as mitigation)	1	1	0	1	1	0	0	**4**
Cost (tools, skills, infra)	0	1	1	0	1	1	0	**4**
Data quality/coverage	0	1	0	0	1	0	0	**2**
Explainable-AI (explicit demand)	0	0	0	0	1	0	0	**1**
Over-reliance on automation	0	1	0	0	0	0	0	**1**

3. **Decision scaffolding and coordination.** Generative AI reduces cognitive and coordination overhead by drafting change-impact summaries, risk registers, runbooks, and stakeholder updates—keeping multidisciplinary actors aligned without displacing human authority.

Limitations and challenges of AI implementation Despite the strong potential, participants also identified various barriers to effective AI/ML adoption summarized in Table 3. The most salient challenges were *integration complexity* and *transparency/trust*. Teams reported substantial effort to connect model outputs to existing pipelines, dashboards, and approval flows; off-the-shelf tools "rarely fit neatly" without custom integration. "Teams are hesitant to trust AI models if they don't understand how decisions are made," said Person B. Human and organizational factors — skills, role clarity, concerns about displacement, and change fatigue — moderated scope and pace of adoption. Cost (tools, skills, infrastructure) and *data quality/coverage* were also reported, though less frequently. Some interviewees warned against *over-reliance* on automation in ambiguous situations. Interviewees described pragmatic mitigations already in use: *pilots* in low-risk contexts to demonstrate value and tune interfaces; targeted improvements to *data coverage* for high-signal metrics; and maintaining *human oversight* for high-stakes gates. Transparency was articulated in two registers: (i) traceable rationales for recommendations ("why this build should wait") and (ii) *auditability* for regulated contexts. Both were repeatedly cited as prerequisites for trust.

Addressing RQ2 — what technical, organizational, and governance constraints impede successful adoption of Analytical and Generative AI in software release management? — three constraint families dominate the evidence:

- **Integration and data readiness.** Connecting models to heterogeneous pipelines, metrics stores, and approval paths is nontrivial; weak telemetry and schema drift degrade model reliability.
- **Transparency and trust.** Stakeholders require traceable rationales and auditability—especially where decisions affect users or regulated processes. This maps to governance expectations for explainability and post-deployment monitoring.

– **Human and organizational factors.** Skills, role clarity, and attitudes toward automation shape acceptance. Interviewees favored staged adoption, preserved human-in-the-loop control for high-impact gates, and explicit "kill switches" to contain automated actions.

5 Implications and Conclusion

This study examined AI's role in software release management by distinguishing *Analytical* from *Generative* AI capabilities and situating both within contemporary Agile/DevOps practice. Across seven practitioner interviews, a clear division of labor emerged. Analytical AI is already dependable at quantitative gates — forecasting incidents and regressions, detecting anomalies in canary and post-release monitoring, prioritizing tests and fixes, and optimizing schedules and resources — matching the most frequent benefit codes (automation, efficiency, predictive capability). Generative AI adds leverage where language and coordination dominate: summarizing change impact, drafting release notes and runbooks, assisting code reviews and test scaffolds, and accelerating incident timelines. Adoption hinges on guardrails: treat telemetry as a product; prefer explainability-by-design with evidence bundling; keep humans in the loop for high-stakes gates; pilot before scale; and integrate natively within CI/CD.

Implications for practice. For practitioners, three priorities emerge. First, invest in "decision plumbing" before "decision models": reliable signals, consistent environments, and auditable runbooks improve outcomes even if model performance is static; this is where most early wins were reported. Second, pair every automated recommendation with an explanation pattern native to the available tools (dashboards, logs, diffs) and require explicit human acknowledgment for high-impact decisions; this responds directly to trust and accountability concerns. Third, adopt a quarterly "binding constraint" cadence that names the most growth- or reliability-limiting bottleneck and concentrates both engineering and AI effort there; interviewees consistently described success when AI was targeted rather than diffuse.

Implications for research. For researchers, the interview evidence suggests several testable propositions that connect AI practices to operational outcomes: (P1) greater coverage of Analytical AI signals across CI/CD and production monitoring will be associated with lower change-failure rates and faster rollback decisions, controlling for team size and baseline release frequency; (P2) use of explanation patterns that bundle underlying metrics will increase human acceptance of automated recommendations at deployment gates; (P3) staged adoption (pilot $\rightarrow$ advisory $\rightarrow$ constrained automation) will correlate with sustained use one year post-introduction; and (P4) combining Analytical AI at gates with Generative assistance for decision documentation will reduce time-to-decision in release management meetings without increasing post-release incident rates. These propositions are grounded in the weighted benefits/challenges and the lifecycle division of labor observed across cases.

Limitations. This work has the typical limitations of qualitative multiple-case designs. The sample privileges depth over breadth (seven professionals in one regional context), relies on self-report, and cannot deliver population estimates. Results should be read as analytic generalizations suitable for theory building and practice guidance, not as universal prescriptions. Future research should pair qualitative insight with quantitative metrics (e.g., change failure rate, mean time to restore, lead time for changes) to test whether teams that adopt the guardrails above realize durable improvements; it should also evaluate Generative AI interventions (release-note generation, test-suggestion prompts, ChatOps co-pilots) in controlled trials to distinguish novelty effects from sustained value.

References

1. Balasubramanian, N., Ye, Y., Xu, M.: Substituting Human Decision-Making with Machine Learning: Implications for Organizational Learning. Academy of Management Review **47**(3), 448–465 (Jul 2022). https://doi.org/10.5465/amr.2019.0470
2. Basiri, A., Hochstein, L., Jones, N., Tucker, H.: Automating Chaos Experiments in Production. In: 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP). pp. 31–40 (2019)
3. Chang, T.C., Lin, Y., Shi, K., Meen, T.H.: Decision Making of Software Release Time at Different Confidence Intervals with Ohba's Inflection S-Shape Model. Symmetry **14**(3), 593 (2022)
4. Fhang, M.C.S., Swamy, R.: Best Practices in Release Management of Large Projects. In: Proceedings of the 2018 7th International Conference on Software and Computer Applications. pp. 51–55. ACM, Kuantan Malaysia (Feb 2018)
5. Herath, I.P., Jayawardena, S., Fadhil, A., Kodagoda, N., Arachchillage, U.S.S.: Streamlining Software Release Process and Resource Management for Microservice-based Architecture on multi-cloud. In: 2023 25th International Multitopic Conference (INMIC). pp. 1–6. Lahore, Pakistan (2023)
6. Karvonen, T., Behutiye, W., Oivo, M., Kuvaja, P.: Systematic literature review on the impacts of agile release engineering practices. Information and Software Technology **86**, 87–100 (Jun 2017). https://doi.org/10.1016/j.infsof.2017.01.009
7. Kittlaus, H.B., Saltan, A.: In memoriam of professor guenther ruhe: Contributions to the software product management research and practice. Information and Software Technology **176**, 107548 (2024). https://doi.org/10.1016/j.infsof.2024.107548
8. Lahtela, A., Jäntti, M.: Challenges and problems in release management process: A case study. In: 2011 IEEE 2nd International Conference on Software Engineering and Service Science. pp. 10–13 (2011), iSSN: 2327-0594
9. Mohamed, S.I.: Software Release Management Evolution - Comparative Analysis across Agile and DevOpsContinuous Delivery. International Journal of Advanced Engineering Research and Science **3**(6) (2016)
10. Saltan, A., Jansen, S., Smolander, K.: Decision-making in software product management: Identifying research directions from practice. In: Proceedings of the International Workshop on Software-intensive Business (SiBW 2018), CEUR-WS. pp. 164–176 (2018)
11. Svahnberg, M., Gorschek, T., Feldt, R., Torkar, R., Saleem, S.B., Shafique, M.U.: A systematic review on strategic release planning models. Information and Software Technology **52**(3), 237–248 (2010)