International Journal of Computer (IJC)

ISSN 2307-4523 (Print & Online)

https://ijcjournal.org/index.php/InternationalJournalOfComputer/index

Maturity Metrics for Engineering Teams in AI-First Application-Development Projects

Satyashil Awadhare*

Engineering Lead at Google,Burlingame, California, USA

Email:satyashil@gmail.com

Abstract

This study is conducted to define a practical maturity-metrics model for AI-First engineering that links delivery and ML lifecycle to business outcomes, enabling reproducible, responsible, scalable operations across process, platform, team, and product. Comparative analysis of maturity models and delivery indicators (DORA Four Keys) combined with ML-specific metrics (retrain time, drift alerts, explainability coverage). Systematic review of MLOps practices and enterprise cases. SMART-based operationalization, automated collection via unified observability and event telemetry; explicit addition of a product plane to Microsoft's MLOps model. Triangulation against McKinsey and DORA survey data. The model formalizes maturity across four planes—process, platform, team/culture, and product—mapped to five levels from no MLOps to fully automated operations. It integrates delivery indicators (lead time, deployment frequency, change failure rate, MTTR) with ML metrics (retrain time, drift alerts, explainability coverage) and requires SMART, automated, actionable collection. It further extends the metric set to cover generative AI and LLM-specific behavioural indicators such as factuality, instruction-following, retrieval effectiveness, safety violations and cost per useful token, and introduces repeatable proxy measures for people and culture—including knowledge diffusion, psychological safety and role-ratio tracking—so that human and organisational factors are observable and auditable. Evidence from DORA and McKinsey indicates that teams instrumenting both delivery and ML metrics achieve higher productivity, steadier SLA adherence, and reduced burnout; meanwhile, rapid AI tool uptake coexists with low trust in generated code (only 24% fully trust), underscoring an adoption–resilience gap the model addresses. Clustering metrics by change delivery, ML lifecycle, user/business value, reliability and quality, governance/ethics, and people/culture focuses investment. Operational enablers—event-based telemetry, unified observability, infrastructure as code, and SLO-prioritized alerting—support uninterrupted releases and faster recovery. The framework functions as an audit checklist and portfolio-planning instrument, translating measurements into managed, value-linked actions. Unifies DORA delivery metrics with ML lifecycle indicators across four planes, adding a product layer and operational enablers (event telemetry, IaC, SLO-prioritized alerting) to make metrics actionable for investment.

Keywords: maturity metrics; MLOps; delivery metrics; drift; observability.

Received: 10/3/2025
Accepted: 12/3/2025
Published: 12/13/2025

** Corresponding author.*

1. Introduction

The rapid spread of generative and analytical AI systems has placed engineering teams in a new strategic position: they are now responsible not only for code delivery speed but also for the model lifecycle, data quality, and ethical compliance. The scale of change is hard to ignore: according to a global McKinsey survey, 78% of companies use AI in at least one business function, with the figure up six points in just a year—a sign that AI-First is ceasing to be a niche approach and is becoming the norm for enterprise products [1].

However, integrating models by itself does not guarantee value. In an AI-First scenario, any data deterioration, feature drift, or technical model regression quickly impacts users and finances, which means engineering maturity becomes a matter of survival. This is why the DORA-2024 survey recorded a paradox: more than half of developers already regularly rely on AI tools inside the IDE and CI/CD, yet only 24% fully trust the quality of generated code, pointing to a gap between the speed of adoption and the resilience of practices [2].

The link between engineering maturity and business results is increasingly evident. The same DORA studies show that teams that systematically measure delivery metrics together with ML-specific indicators (standardized retrain time, drift alerts, explainability coverage) achieve higher productivity, less burnout, and more stable SLA adherence than groups where metrics are absent or fragmented. When transparent indicators are embedded in the OKR loop, companies more often report revenue growth and a higher share of active users of AI features; McKinsey notes the corresponding bottom-line impact among those that introduced KPI monitoring for models and processes simultaneously [1].

The metrics model proposed herein serves as a navigational framework across these concerns. It may be operationalized as a checklist for baseline audits, as a structure for investment planning in an MLOps platform, and as a template for retrospectives in which teams, using the same model, select one or two priority metrics for targeted improvement. The framework stipulates separate measurement of process, platform, product, and culture; examination of their correlations with business indicators; and a staged elevation of maturity—progressing from ad hoc experimentation to sustainable, reproducible, and ethically responsible AI delivery.

2. Materials and Methodology

The study is based on an interdisciplinary approach combining analysis of academic and industry sources, empirical data from global surveys, and practical cases of implementing the AI-First approach in an enterprise engineering environment. The theoretical foundation consists of results from McKinsey’s global survey reflecting the dynamics of AI penetration into business functions and its impact on corporate strategies [1], as well as DORA reports recording the relationship between engineering maturity, delivery quality, and trust in tools [2]. Another element includes methodological frameworks developed in Microsoft Learn research on the MLOps maturity model, which enabled the structuring of the technical and organizational parts of the analysis [4].

Methodologically, the work relies on several research techniques. First, a comparative analysis was applied to existing maturity models and delivery indicators, including classic software-delivery performance metrics (lead time, deployment frequency, change failure rate, MTTR), as well as extended indicators for ML systems such as retrain time, explainability coverage, and drift alerts [3]. This approach made it possible to establish the difference between product-oriented and process-oriented metrics and to identify their influence on the resilience of engineering practices.

Second, a systematic review of MLOps practices and related methodologies was carried out. Use of Microsoft's maturity model [4] made it possible to highlight key planes of analysis—process, platform, team, and product—and relate them to business outcomes recorded in McKinsey reports [1]. The principles of SMART metrics [5] were applied to test how far the proposed indicators can be operationalized, embedded in OKRs, and used to assess progress over time.

Third, the study used content analysis of empirical cases and public reports of companies applying AI-First practices. Comparing their practical experience with DORA data [6] made it possible to identify the key factors in moving from experimental AI implementations to scalable and reproducible processes, and to confirm that maturity increases correlate directly with improved productivity and reduced operational risks.

3. Results and Discussion

Maturity metrics are understood as indicators that assess not the final characteristics of the model or product but the organization's ability to consistently, quickly, and ethically produce those results. Classic product metrics measure, for example, user retention or revenue growth, while model metrics measure F1-score or MAE. However, even a perfect model-quality metric says little about how long retraining will take, how quickly the team can roll back an unsuccessful release, or how reliably data are collected. This is why DORA researchers single out a separate class of delivery indicators—lead time, deployment frequency, change failure rate, and MTTR—and show that they correlate with organizational effectiveness and team well-being. In contrast, product and model metrics do not possess such properties [3].

In addition to the metrics already discussed, it is advisable to explicitly introduce indicators that capture the behavioural profile of generative systems: a hallucination rate, a refusal rate, and a consolidated toxicity/safety score. The hallucination rate denotes the proportion of responses that contain verifiably incorrect or misleading factual content and should be estimated by combining automated fact-checking procedures with stratified human evaluation. The refusal rate records the share of interactions in which the model declines to perform a legitimate and safe instruction or returns an uninformative refusal; tracking this metric helps detect both excessive suppression and inadequate filtering of outputs. The toxicity/safety score aggregates the outcomes of a standardized suite of safety assessments that detect PII leakage, hateful or abusive language, dangerous or unlawful advice, and other policy violations, producing a calibrated signal of the model's adherence to safety requirements.

These indicators must be operationalised within the delivery pipeline so that they are observable, auditable and enforceable. Specifically, the metrics should be computed automatically as part of a standardized evaluation suite executed in the CI/CD pipeline prior to release, and should include adversarial and product-centric test scenarios alongside routine probes. Evaluation artefacts must be versioned together with model binaries and retained in the observability repository; release acceptance criteria and SLO-oriented alerting should explicitly reference these signals so that threshold breaches block deployment or trigger automated rollback and incident logging. Instrumenting the checks with deterministic identifiers linking results to model artifact, prompt variant and retrieval configuration ensures that failures can be traced rapidly to their root causes and that factuality and safety requirements are transformed from abstract desiderata into concrete operational controls.To work with this layer systematically, the measurement object must be expanded. MLOps practice shows that maturity consists of at least four interrelated planes. The first is process, i.e., how much and how well CI/CD, CI/CT pipelines are automated and reproducible. The second is a platform – cloud infrastructure, observability, and data versioning tools. The third is the team: its competencies, how roles are distributed, and the culture of interaction. And finally, fourth is the product itself: how ready ML features are to operate and what kind of feedback mechanisms exist. A similar multidimensional logic is confirmed by Microsoft's MLOps maturity model (Table 1), where quality is assessed separately across people/culture, processes/structures, and technology; we merely add an explicit product layer to link engineering efforts with user value [4].

[Table 1]

When selecting concrete metrics for each plane, it is essential to adhere to three principles. First, they must meet SMART criteria: be specific, measurable, achievable, relevant, and time-bound; otherwise, the team will not be able to verify progress or link the numbers to OKRs [5]. Second, collection should be automatic: if an indicator requires manual entry into a spreadsheet, it will quickly go stale and lose trust. Third, a metric must be actionable—its deterioration should unambiguously trigger an action, whether retrain, rollback, or sprint reprioritization. Practice shows that such indicators close the observe—understand—respond loop and turn maturity from a fine slogan into daily managed work.

The next step is to describe how exactly this capability evolves from chaotic experiments to an industrial scale. It is taken as a basis a five-level scale similar to the one formulated by Microsoft Learn's MLOps research—from no MLOps to fully automated operation; each level adds technical and cultural mechanisms, reducing risk and increasing speed [4].At the initial stage (zero–first), called Initial, models live as a by-product of data analysis. Code and data are stored in notebooks, releases are done manually, and there is hardly any monitoring. The team can prototype quickly, but each product integration turns into night releases and manual rollbacks. It is here, according to observations from the annual DORA report, that the concentration of incidents is maximal and the relationship between throughput and stability is weakest, confirming that fast chaotic change does not lead to sustainable advantages [6].

When basic CI processes for code and a single Git branch appear, the team moves to Emerging: container builds, unit tests, and manual but repeatable model deployment. Operational risk decreases, but feedback about data drift or response degradation still lags. The typical pain of the level is we see the metrics only post factum and respond with a delay.

Scaling begins where a shared ML platform, automatic feature versioning, centralized experiment tracking, and the first continuous-training pipeline appear. Processes are already codified (IaC, Terraform/Helm), and delivery metrics are reconciled with product KPIs. In the DORA report, precisely those teams that achieved such standardization show synchronous growth in both deployment speed and reliability, disproving the old move fast—break things myth, as shown in Figure 1 [6].

[Figure 1]

At the Optimizing level, automation covers the full ML lifecycle: an A/B orchestrator, drift alerts with auto-training, and SLO-oriented alerting. Infrastructure scales horizontally, and compute cost is controlled through auto-pause policies and spot instances. The focus shifts to reducing technical debt and complying with Responsible AI: the team's actions include regular audits for explainability and bias.

Finally, Leading describes organizations where the platform has become a product in its own right: real-time ML, zero-downtime releases, automatic rollback in minutes, and recommendation services and generative models trained online on streaming events. At this maturity level, engineering and ethical metrics are included in the corporate OKR portfolio on a par with financial ones, and cross-functional teams operate under the principle model = feature. Microsoft notes that such automation requires not only technical but also cultural maturity: incidents are treated as opportunities to improve processes rather than as blame for individual specialists [4].

Thus, the five-level scale shown in Figure 2 sets a clear trajectory: from manual, hard-to-reproduce experiments to self-healing, business-oriented systems. It allows efforts to be aligned with real value: transitions between levels occur not when another trendy tool is introduced but when indicators for process, platform, team, and product simultaneously rise to the next stable plateau.

[Figure 2]

Dividing metrics into logical clusters helps the team focus on the most vulnerable parts of the pipeline and choose where to apply effort depending on the current maturity level. The first cluster relates to change delivery. It includes indicators showing how smoothly code and models travel from the repository to production. The shorter the average time from commit to successful deployment and the more frequent the deployments, the easier it is for the business to experiment without significant outage risk. A necessary counterweight to speed is the average time to restore the system after an incident, as it shows whether the team can move fast safely. The second cluster is made up of the machine-learning lifecycle. It asks how quickly the team reacts to data drift, the delay between new samples arriving and being used for training, and how controlled retraining is. If data gets into the feature store with little lag and model retraining is triggered automatically on finding a big swing, then the product stays current and strong against environmental change.

In recognition of the qualitatively distinct operational profile of generative systems and large language model architectures, the metrics framework for the machine-learning lifecycle should be augmented to capture behaviours that are not well represented by traditional supervised-learning indicators. These systems tend to exhibit factual hallucination, variability in instruction compliance, acute sensitivity to prompt formulation and retrieval quality, drift in reward signals under interactive training regimes, and degradation across multi-turn context. Consequently, assessment must include measures of factuality, instruction fidelity, retrieval effectiveness, safety and operational cost, and these measures must be instrumented so that each response can be traced back to the exact model artifact, prompt variant and retrieval configuration; persistent telemetry fields and automated check results linked to any human evaluation are necessary to render such signals auditable and actionable.

The third cluster is in user and business value. It reports the proportion of active audiences using AI-based features and the increase in key financial indicators directly attributable to those features. When changes in the model or algorithm are supported by growth of engagement and revenue, it is a proof for the team that technical efforts are materialized into company results.

The fourth cluster merges indicators of reliability with quality in meeting the terms of service agreements, stability of prediction latency, scope of automatic drift signaling, and speed of rollback to the last version. In simple words, this metric builds consumer trust in the service, hence helping avoid reputational losses when scaling load.

The fifth cluster covers governance and ethical aspects. It shares models of providing interpretability on auditor request, frequency in detecting and removing bias, as well as completeness of decision documentation. With responsible development being regulators' increasing interest, these indicators are gradually moving from the reputational plane to the category of mandatory ones.

In addition to the governance and ethical indicators already described, it is advisable to incorporate economic and operational metrics that are specific to large language model deployments, since their budgetary and operational impact differs qualitatively from that of conventional ML products. Practically, this requires systematic measurement of cost per query or per user session, average tokens per response, and latency per output token, with these signals collected automatically, aggregated by product scenario, and integrated into budgetary and service-level controls. Thresholds on cost and token consumption should function as predicates for throttling, quota enforcement and alerting, and breaches of operational budgets should elicit predefined mitigation actions—such as controlled quality degradation, failover to less expensive models, or temporary suspension of high-cost features—to prevent catastrophic financial exposure.

In retrieval-augmented generation contexts interpretability must be given an auditable, provenance-oriented formulation. The practical test of a RAG system's explainability is the system's ability to identify the corpus fragments on which specific answers are based; this capability should therefore be evaluated using citation precision and citation recall metrics computed on standardized question sets with known relevant chunks. To enable formal verification and post-hoc audit, every generated assertion and its associated retrieval identifiers and scoring diagnostics must be recorded in telemetry and persisted alongside the model version and retrieval configuration, thereby preserving the full chain of evidence from query through retrieval to generation.

Bringing cost, operational and provenance metrics into the governance and ethics cluster converts a set of reputational desiderata into enforceable operational controls. These metrics should be included in release acceptance criteria and maturity assessments, instrumented as automated checks in the CI/CD pipeline and surfaced continuously in the unified observability layer. Concurrent monitoring of financial, behavioural and provenance signals thus provides an integrated capability to manage security and compliance risks while protecting the economic sustainability of LLM-based services.

It speaks to the human factor and culture. How well the team absorbs shocks emanating from the exit of key specialists, what roles are played by engineers vis-à-vis researchers, and how much time is spared for learning and knowledge sharing. Without these intangibles, no matter how advanced a platform is, it cannot bear long-term growth pressure; hence, it should be tracked by mature organizations together with technical and product metrics.

The people and culture plane should be rendered measurable through repeatable proxy indicators that can be incorporated into governance gates and team objectives, rather than remaining an unquantified margin note. Knowledge diffusion can be observed through signals that reflect how rapidly engineers acquire productive fluency with the platform and how broadly model-integrated features are delivered across non-specialist teams; psychological safety can be monitored by combining anonymised pulse assessments concerning incident reporting with operational traces such as the elapsed time from internal detection of a model failure to formal reporting and the proportion of incidents initiated by staff rather than only by automated detectors; and structural evolution can be tracked by reporting the balance of capacity across research, applied engineering and platform engineering as maturity advances. These cultural proxies should accompany behavioural measures for generative systems in acceptance criteria for progression between maturity levels, and safeguards against gaming, privacy erosion and evaluator drift must be implemented through cross-validation of signals, guaranteed anonymity and aggregation of survey responses, and standardisation of human evaluation procedures.

To make metrics move from being abstract ideas to a daily management tool, an end-to-end loop of data collection, storage, and visualization has to be built. A unified observability layer is what it all starts with: service and container system metrics are collected by Prometheus exporters; training and inference parameters are registered in MLflow; features with their versions are maintained in Feast, then this data gets consolidated in a repository, e.g., a cloud analytical warehouse. This separation of roles simplifies scaling: low-level monitoring remains onboard the orchestrator, and ML-cycle artifacts, including models and feature snapshots, are available through the platform API for quick experiment reproduction and audits.

Observability quality increases significantly when event-based telemetry tags accompany each transaction in the product. Experiment identifiers, the model version, and the hash of the feature set are passed along with the user request and attached back to logs in the data layer. This enables them to piece together the entire story between tweaks in hyperparameters, output predictions, and product KPIs without some convoluted SQL probe. Where metrics collection environments are highly deterministic, that is where reliability in metrics is born. Code infrastructure allows us to describe clusters, roles, and network policies declaratively—no more it works only on the developer's machine effect. When the configuration deviates from the baseline, the system signals this just as it would a latency drop, so environment drift ceases to be a hidden cause of random incidents.

The collected data require an accessible visualization. Operations teams use graph dashboards with panels for time to restore and release frequency. At the same time, executives prefer aggregates by maturity clusters and a trend line of the impact of AI features on revenue. One single source of truth, different vantage points, remove the gap in interpretation, and foster more confidence in the numbers.

At long last, the whole situation isn't whether there is sufficient signaling. SLO-class levels group alerts: the first two classes cover availability threats and severe drift, the next cover degradation of user experience, and the rest cover less critical deviations. Such a prioritized channel reduces noise, allows on-call engineers to focus on incidents that genuinely affect the business, and sustains a healthy balance between delivery speed and reliability.

4. Conclusion

The analysis shows that for teams working in the AI-First paradigm, the key source of manageability is not the final quality metrics of models or product KPIs by themselves, but maturity metrics that assess an organization's ability to deliver change consistently, quickly, and ethically. The fast uptake of AI tools with a lack of trust in their outputs shows an inherent gap between the speed of adoption and the resilience of practice; bridging this is enabled by shifting from disparate measures to a coherent metric system that links engineering processes to business value and responsibility requirements.

A systemic framework that comes from separating the process, platform, product, and team culture planes proves practical: it creates a shared language for audits, investment planning in any component of the platform, and retrospectives where priority improvements have to be selected. Traditional delivery metrics fused with ML-specific metrics, and the way they tie into the goals-and-key-results loop, are what correlate productivity, sustained adherence to quality of service, and team well-being. It moves the maturity discussion from something declarative to being able to do it.

A five-stage scale throws up a clear path of journeying from early fumbling steps right through to complete automatic working. Changes between stages happen when signs in all areas get better together: making and using pipelines become steadier and automatic; watching and tracking data and models grow easier; a habit of careful testing and learning from mistakes gets stronger; and product feedback tools help check the user value of changes. The result is the capability for uninterrupted releases, rapid recovery from failures, and responsible handling of data and models.

To illustrate how the four planes form a closed loop, consider a concrete incident: a measurable decline in an "Answer Faithfulness" metric within the governance cluster coincides with a statistically significant increase in user "down-vote" events captured by product telemetry. Those two signals, when correlated by the unified observability layer and linked via deterministic identifiers (model_version, prompt_hash, retrieval_ids), should automatically escalate to a P1 reliability alert against the service's SLOs and trigger an operational remediation such as an automated rollback to the last validated prompt-version in the change-delivery pipeline. The rollback is immediately logged, alerted, and routed into a blameless root-cause analysis that updates the platform's telemetry, the product experiment configuration, and the team's OKRs or backlog, thereby demonstrating how governance, product, reliability and change-delivery interact to convert an observed behavioural regression into an auditable, automated, and value-focused corrective cycle.

Clustering metrics by change delivery, machine-learning lifecycle, user and business value, reliability and quality, governance and ethics, as well as human factor and culture, shows where exactly to concentrate efforts at each stage of development. The principles of specificity, automatability, and actionability turn measurement into a management lever: deterioration in a metric unambiguously initiates an action, whether retraining, version rollback, or task reordering.

It executes on a shared perspective strata and occasion telemetry, binding model variants, highlight collections, and experimental settings with client demands and item measurements. A code-based framework eliminates environment drift as an underlying reason for occurrences. Visualizations' single wellspring of truth set in everybody's hands guarantees choices at the group and administration levels. Alerts focused on by SLO classes eliminate commotion and keep the center away from dangers that affect clients and finances.

It should be stressed that the study's methodological design is conceptual and synthetic, as it seeks to develop a coherent framework for AI-First engineering based on existing survey data, established industrial maturity models, and publicly available case material, rather than drawing causal conclusions from primary experimental data. As a result, the proposed planes, levels, and metric clusters should be interpreted as a structured reference architecture whose practical value lies in guiding implementation and investment decisions. At the same time, the precise effect sizes of individual indicators remain subject to empirical calibration in specific organisational settings. The focus on vendor-agnostic patterns, observability, DORA metrics, and their application above the toolchain, as well as domain-specific tailoring required in regulated domains, for small enterprises or in on-premises or non-cloud environments, is intended to provide coverage for those areas. The signals for generative AI/LLM-specific metrics, including factuality, retrieval effectiveness, safety, and cost, are designed to capture the state of the art, practice, and regulations at the time of this writing in that domain and may change over time. In that sense, the framework can be thought of as a strong baseline of constructs rather than the final word.

The proposed maturity-metrics model is practically a navigator for engineering teams under AI-First conditions: narrowing the gap between experimentation and operation reducing operational risk accelerating safe iterations helping convert technical progress into verifiable user and business value by embedding this model into day-to-day planning and feedback cycles making maturity not a one-off initiative but rather a continuous discipline that sustains scalable reproducible and ethically responsible delivery of AI features.

References

[1]. Singla A, Sukharevsky A, Yee L, Chui M. The state of AI: How organizations are rewiring to capture value [Internet]. McKinsey & Company. 2025 [cited 2025 Jul 19]. Available from: https://www.mckinsey.com/capabilities/quantumblack/our-insights/the-state-of-ai

[2]. DeBellis D, Storer KM. DORA Research: 2024 [Internet]. DORA. 2024 [cited 2025 Jul 20]. Available from: https://dora.dev/research/2024/ai-preview/

[3]. Harvey N. DORA's software delivery metrics: the four keys [Internet]. DORA. 2025 [cited 2025 Jul 21]. Available from: https://dora.dev/guides/dora-metrics-four-keys/

[4]. Microsoft Learn. Machine Learning Operations Maturity Model [Internet]. Microsoft Learn. [cited 2025 Jul 22]. Available from: https://learn.microsoft.com/en-us/azure/architecture/ai-ml/guide/mlops-maturity-model

[5]. SMART Metrics Guide [Internet]. Sopact. [cited 2025 Jul 24]. Available from: https://www.sopact.com/guides/smart-metrics

[6]. Stephens R. DORA Report 2024 – A Look at Throughput and Stability [Internet]. RedMonk. 2024 [cited 2025 Jul 25]. Available from: https://redmonk.com/rstephens/2024/11/26/dora2024/

Table 1: Maturity model: five levels of technical capability [4]

Level	Description	Highlights	Technology
0	No MLOps	Difficult to manage the whole machine learning model lifecycle The teams are disparate, and releases are painful Most systems exist as black boxes, with little feedback during/and post-deployment	Manual builds and deployments Manual testing of the model and the application No centralized tracking of model performance Training of the model is manual
1	DevOps but no MLOps	Releases are less painful than No MLOps, but rely on the Data Team for every new model Still limited feedback on how well a model performs in production Difficult to trace/reproduce results	Automated builds Automated tests for application code
2	Automated Training	The training environment is fully managed and traceable Easy to reproduce model Releases are manual, but low-friction	Automated model training Centralized tracking of model training performance Model management
3	Automated Model Deployment	Releases are low-friction and automatic Full traceability from deployment back to the original data Entire environment managed: train > test > production	Integrated A/B testing of model performance for deployment Automated tests for all code Centralized tracking of model training performance
4	Full MLOps Automated Operations	The complete system is automated and easily monitored Production systems are providing information on how to improve and, in some cases, automatically improve with new models. Approaching a zero-downtime system	Automated model training and testing Verbose, centralized metrics from the deployed model

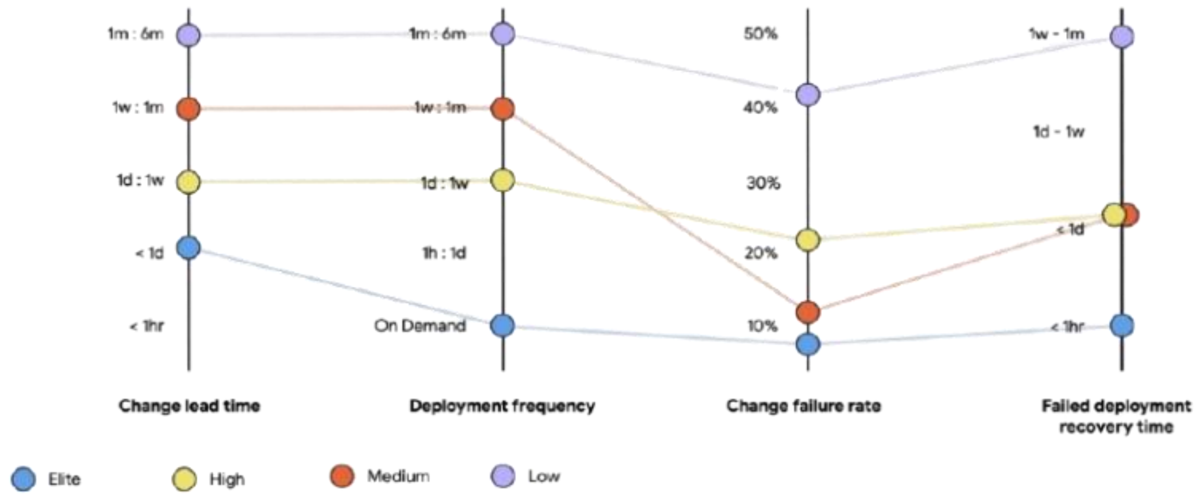

Figure 1: Software delivery performance level [6]

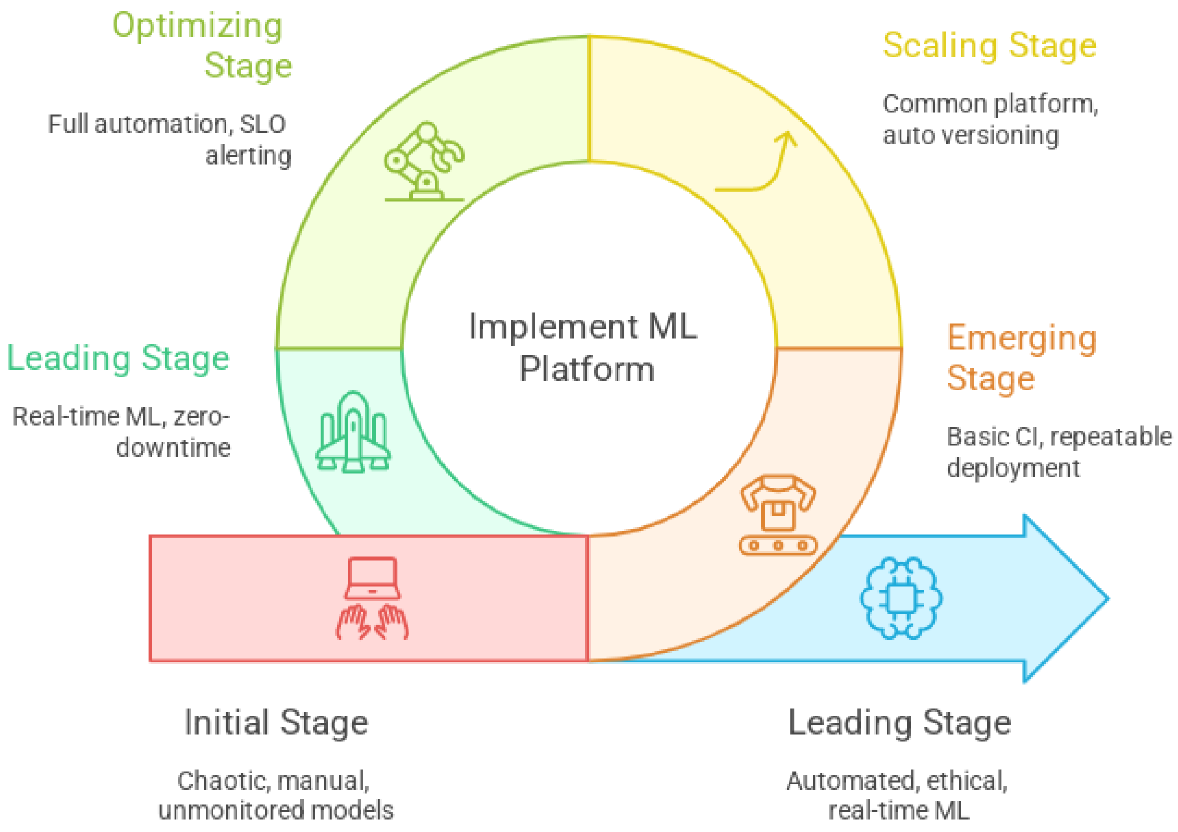
Optimizing Stage
Full automation, SLO alerting
Scaling Stage
Common platform, auto versioning
Implement ML Platform
Leading Stage
Real-time ML, zero-downtime
Emerging Stage
Basic CI, repeatable deployment
Initial Stage
Chaotic, manual, unmonitored models
Leading Stage
Automated, ethical, real-time ML

Figure 2: Five-Stage Metric Model (compiled by author)