VFAST Transactions on Software Engineering
https://vfast.org/journals/index.php/VTSE@ 2026, ISSN(e): 2309-3978, ISSN(p): 2411-6246
Volume 14, Number 2, April-June 2026 pp:33-45

Mathematical Modelling of the Impact of Developer Experience Metrics on the Duration of the Release Cycle in Full-Stack Projects

Volodymyr Lopukhovych [1], **Roman Martynenko** [2], **Dmytro Poltavskyi** [3], **Oleksandr Shvaikin** [4], **Oleh Sypiahin** [5]

[1]Senior Software Engineer. Disney Streaming. USA; [2]Senior Software Engineer. Henry AI Inc. USA; [3]Engineering Team Lead. Uplandme Inc. USA; [4]Salesforce Developer. VRP Consulting. USA; [5]Full Stack Engineer. FloLive. Israel

Keywords:*Full Stack Development, Cloud Computing, Web Application Architecture, Backend-as-a-Service (BaaS), Frontend Frameworks, RESTful APIs, DevOps, Microservices Continuous Integration / Continuous Deployment (CI/CD), Serverless Architecture.*

Journal Info:
Submitted:
May 13, 2025
Accepted:
March 30, 2026
Published:
April 14, 2026

Abstract Modern DevOps practices are increasingly emphasizing Developer Experience metrics as critical factors influencing the efficacy of software delivery. However, despite significant advancements in this direction, the challenge of constructing a causal model that elucidates the impact of Developer Experience metrics on the speed of releases remains unresolved. This study aims to formalize and empirically evaluate the causal relationship between various indicators of development experience (such as cognitive load and technical frustration) and the speed of the release cycle within DevOps environments. A simulation model was developed for this analysis, embodying a prototypical CI/CD infrastructure characterized by either a microservices or serverless architecture within a cloud framework. Bayesian Causal Modeling was employed, facilitating the exploration of how Developer Experience variables influence the time parameters of the release cycle. The model encompasses seven variables, comprising five key Developer Experience metrics alongside two control parameters (architectural and organizational levels). Within the simulation framework, data was generated based on the principles of Web Application Architecture, incorporating contemporary methodologies of Full Stack Development and RESTful API interfaces. The findings reveal a statistically significant causal relationship between cognitive load and technical frustration and an increase in the duration of the deployment cycle. For instance, the probability of exceeding a release duration of 14 days increases from 21% to 83% in case of slow CI/CD integration, and accordingly from 18% to 72% when utilizing low-quality tools. This underscores the feasibility of integrating Developer Experience metrics into Engineering Decision Support Systems within DevOps processes. The results obtained from this study can be leveraged to enhance DevOps monitoring strategies, prioritize interventions within development workflows, and further advance analytical platforms aimed at managing the quality of the developer experience.

***Correspondence author email address:** volodymyrkogu@gmail.com
DOI: 10.21015/vtse.v14i2.2300

1 Introduction

In a rapidly changing digital environment, organizations must continuously improve the speed and quality of software development while maintaining flexibility, security, and market responsiveness. In this context, DevOps has evolved [1, 2] from a set of engineering practices to a strategic paradigm that ensures continuity, adaptability, and reliability throughout the software de-

velopment lifecycle. In DevOps environments that span the full development lifecycle—from user interfaces to backend systems—the impact of engineering practices and developer-related factors on release speed becomes particularly significant. Despite the widespread adoption of DevOps methodologies in organizations of all sizes and industries [3–5], the quantification of the effectiveness of specific practices remains understudied. While certain tools and workflows speed up development or improve quality, others may be neutral or even detrimental, depending on contextual factors. Traditional analytical approaches based on descriptive or correlational statistics [6] are limited in their ability to detect causal relationships, which are important for evidence-based decision-making in DevOps environments.

Recent advances in Bayesian causal analysis [7] provide new methodological opportunities for modeling causal relationships under conditions of high complexity and uncertainty. Integrating DevOps analytics with causal inference allows not only to evaluate existing practices, but also to evaluate potential management and technical interventions based on their expected causal effects. Consequently, there is a growing need for analytically sound models that support objective assessment, causal reasoning, and scenario-based optimization of DevOps processes.

At the same time, an exclusive focus on technical performance metrics such as build speed, deployment frequency, or incident rate is increasingly recognized as insufficient. These metrics do not fully account for the human and cognitive factors that influence development performance. Developer Experience (DX) [8, 9], traditionally viewed within the socio-technical or user experience (UX) [10] framework, has emerged as a critical enabler of successful DevOps transformations. However, most organizations lack systematic approaches to measuring and improving DX, and existing research often remains descriptive, without establishing causal relationships between DX metrics and delivery performance.

The novelty of this study lies in the following contributions:

- Formalization of developer experience metrics within a structural Bayesian causal model.
- Integration of DevOps process indicators with causal inference based on a directed acyclic graph (DAG).
- Application of intervention analysis using the do operator to model management decisions in CI/CD environments.
- Development of a scenario probabilistic framework for predicting release cycle delay.

Unlike existing DevOps analytics approaches that focus on correlation or benchmarking, this study introduces the perspective of computational causal modeling, which allows for the quantitative evaluation of hypothetical interventions.

The research hypothesis suggests that certain DX metrics, including cognitive load, tool quality, and context switching frequency, have a statistically significant causal effect on release cycle time.

The goal of this study is to develop a formal causal model that explains the impact of key developer experience metrics on software release speed in a DevOps environment. To achieve this goal, the study aims to achieve the following objectives:

1. Identify key technical and cognitive-psychological variables that affect release speed, including CI/CD characteristics, tool quality, mental load, and context switching.
2. Build causal graph structures using Bayesian networks.
3. Estimate model parameters based on synthetic data that mimics a typical DevOps development cycle.
4. Validate the model using log-likelihood, AIC/BIC, and cross-validation metrics.
5. Perform scenario modeling to assess the impact of alternative management decisions on the probability of release delay.

The results of this study allow us to quantify the causal effects associated with DX metrics on release velocity, support scenario-based forecasting of development outcomes, and provide analytically sound recommendations for improving productivity without increasing cognitive load or risk of burnout.

RQ1. Which developer experience metrics have a statistically significant causal effect on release cycle length in DevOps environments?

RQ2. Can Bayesian causal modeling adequately formalize these effects and support scenario-based analysis of release dynamics?

H1. Increased cognitive load, poor quality engineering tools, and reduced feedback lead to a statistically significant increase in release cycle length.

H2. Bayesian causal modeling provides accurate quantification of causal effects associated with DX metrics and enables analytically sound modeling of DevOps release processes.

From a computer science perspective, this research contributes to the field of software engineering analytics and computational modeling of socio-technical systems. The proposed Bayesian structural causal model is a formal computational platform that integrates software development process metrics with probabilistic inference mechanisms. Thus, the research is at the intersection of DevOps engineering, causal machine learning, and applied statistical modeling in software systems.

2 Literature Review

In recent years, DevOps has evolved from an engineering methodology into the cornerstone of a contemporary paradigm for continuous development, delivery, and operational stability of software. A number of studies [11–13] have substantiated that the adoption of DevOps practices, such as Continuous Integration/Continuous Delivery (CI/CD), Infrastructure as Code, test automation, and monitoring, exerts a favorable influence on software development. In particular, these practices markedly diminish the duration of the Software Development Life Cycle (SDLC) while augmenting the frequency of releases. Of particular interest is the measurement of these practices efficacy through metrics. Approaches based on the DORA framework (Deployment Frequency, Lead Time for Changes, Change Failure Rate, etc.) have already attained status as an industry benchmark [14]. However, several authors [15, 16] emphasize the limitations inherent in DORA metrics when it comes to identifying causal relationships between Developer Experience (DX) and process performance.

The DX model proposed in [17] focuses on three key components: flow, cognitive load, and feedback loops. These factors, although not direct technical metrics, profoundly influence the team's capacity to efficaciously implement DevOps practices. In essence, suboptimal DX metrics can undermine the advantages of DevOps, even if the tools are implemented formally.

Nonetheless, empirical validation for this hypothesis remains fragmented. As demonstrated by the authors in [18, 19], traditional statistical methodologies rarely permit causal inferences based on observational data from software engineering. This limitation has catalyzed the adoption of Bayesian causal networks, which facilitate the modeling of latent dependencies among non-obvious factors, such as DX metrics, and release timelines. Comparable methodologies are already being employed in related fields: security [20], quality of requirements, and risk assessment in CI/CD [21].

Contemporary research within the DevOps domain predominantly concentrates either on the technical dimensions of software deployment – such as metrics for continuous integration and delivery (CI/CD), DORA metrics (Deployment Frequency, Lead Time, Change Failure Rate, etc.), – or on the qualitative examination of DX metrics through surveys, interviews, or heuristic models. That being said, the systematic and formalized analysis of the interrelations between DX metrics indicators and the efficacy of DevOps processes remains insufficiently developed.

Most existing approaches rely on traditional statistical methods that fail to elucidate causal dependencies essential for forecasting the impact of specific alterations on business outcomes (e.g., release timelines). Consequently, a challenge emerges in delineating the nature of the influence exerted by individual DX metrics indicators – such as the quality of feedback, cognitive load, and the degree of technical frustration – on release speed within DevOps environments.

Given the limited number of quantitative and causally oriented studies in this direction, there exists an imperative for the further advancement of relevant methodologies. These methodologies should not only facilitate the documentation of the impact of specific factors but also enable the simulation of their consequences in various scenarios. This endeavor necessitates an expansion of the research foundation through the construction of formal models capable of integrating developer experience metrics with analytics of release processes and decision-making within the DevOps cycle.

Despite the widespread adoption of DevOps practices and the growing interest in DX metrics, existing research has largely relied on descriptive statistics, surveys, or correlation-based performance measures. While frameworks such as DORA provide valuable benchmarking metrics, they do not establish causal relationships between DX factors and release cycle times.

Current research lacks:

- formally defined causal structures linking DX metrics to release performance
- quantitative intervention analysis to optimize the DevOps process
- computational models capable of supporting scenario-based decision making.

Therefore, the main research challenge addressed in this study is the lack of a formal, computationally sound causal framework for assessing the impact of developer experience metrics on release cycle times in DevOps environments.

3 Methods and Materials

3.1 Research Procedure

The research was conducted in several stages. At the first stage, the variables were formalized and a causal graph was built based on theoretical assumptions. The second stage involved the generation of simulated data were generated based on aggregated statistics sourced from surveys, CI/CD logs, and DevOps dashboards. Following this, the model parameters were scrutinized using Bayesian and frequentist methods (including MCMC and EM algorithms). Next, causal effects were computed using the do(.) operators [21]. At the final stage, the model's quality was checked using the Log-Likelihood, AIC, BIC, RMSE, MAE, and CV-10 metrics.

3.2 Methods and models

To achieve the research goals, causal modeling within the Bayesian paradigm [22, 23] was selected, providing a comprehensive toolkit for identifying and quantifying causal relationships among variables under conditions of uncertainty, complex interdependencies, and limited data. This approach facilitates the formalization of hypotheses regarding the influence of individual factors on the target variable (software release time) and allows for their evaluation through probabilistic inference.

The present study employs a structural approach to Bayesian causal modeling (Bayesian Structural Causal Modeling), which combines:

- causal graphs (Directed Acyclic Graphs, DAGs) that delineate assumptions about the directions of causal influences between variables [24, 25];
- hierarchical Bayesian models that allow for the consideration of noise, variability, latent factors, and sampling limitations [26, 27];
- scenario modeling, that is the generation of alternative scenarios to evaluate the potential impact of modifications in individual metrics (for instance, mitigating technical frustration or enhancing feedback) [28].

The chosen causal framework is based on theoretical principles of cognitive workload theory, sociotechnical models of software development, and modern research in DevOps analytics. In particular, it is assumed that technical factors (level of CI/CD automation, quality of tools) indirectly affect release duration through cognitive and behavioral mechanisms (context switching, developer satisfaction).

Alternative causal configurations were considered at the conceptual level; however, the proposed model provides the best interpretability and consistency with the existing literature.

These methods not only facilitate the identification of statistical dependencies among variables but also enable the formulation of cause-and-effect conclusions, particularly in modeling the outcomes of various scenarios involving the variation of variables.

3.3 Sources and data types

3.3.1 Synthetic Data Generation

Synthetic data was generated to simulate a typical DevOps development cycle under controlled experimental conditions. The dataset reflects the interaction between DX metrics and release cycle length, including variables related to cognitive load, tool quality, feedback frequency, CI/CD performance, and release outcomes.

The data generation process was guided by realistic assumptions derived from empirical DevOps research

and industry reports. Probabilistic relationships between variables were defined according to the proposed causal graph structure, which allows generating data that preserves plausible behavioral and process relationships. Random sampling was used to ensure variability across simulated development scenarios. The synthetic dataset was used exclusively for model training, validation, and scenario-based analysis.

Multi-source simulated data, derived from typical structures of DevOps environments, were employed to construct a cause-and-effect model. The variables were classified into three principal categories: the target variable, DX metrics, and confounding variables. Below is a comprehensive description of all the metrics utilized, accompanied by their corresponding labels, measurement scale types, and data sources.

- **RCD (Release Cycle Duration)** — the duration of the release cycle, defined as the time interval between the first commit of a feature and its successful deployment. It is the target variable of the model.
- **B (Developer Tooling Satisfaction)** — a subjective assessment of satisfaction with engineering tools (IDE, CI/CD services, build, etc.); obtained from a survey.
- **C (Communication Quality)** — an assessment of the effectiveness of intra-team communication during the release.
- **A (Automation of CI/CD)** — a quantitative indicator of the level of automation of continuous integration and delivery; expressed as the proportion of automated stages in the pipeline (0–1).
- **K (Documentation Quality)** — an assessment of the completeness, relevance, and clarity of technical documentation related to features.
- **F (Feedback from Code Review)** — a feedback indicator: timeliness, quality, and usefulness of reviews from other developers.
- **P (Team Size)** — the number of developers involved in the current release. It is used as a control variable to account for the impact of team size.
- **S (Feature Complexity)** — expert-estimated complexity of functionality in the release, normalized to a scale of 0–1.

Cognitive and psychological DX factors are operationalized through indicators of the level of context switching, subjective cognitive load, perceived quality of tools, and developer satisfaction. These variables reflect both objective aspects of the workflow (number of parallel tasks, frequency of switching) and subjective perception of the complexity of the environment. This approach allows integrating technical and behavioral dimensions into a single causal structure.

A detailed description of the variables can be found in Table 1.

The preprocessing encompassed the normalization of metrics (A, S) to the range [0,1], the encoding of ordinal features (B, C, K, F) with numerical values from 1 to 5, the filtration of releases characterized by incomplete data, and the imputation of missing values utilizing the multiple imputation method.

In light of the incompleteness of certain variables within the simulated dataset (for instance, the index of technical frustration and the depth of context switching), multivariate imputation predicated on the MICE (Multiple Imputation by Chained Equations) algorithm was employed to uphold data integrity. The imputation process was executed via stochastic regressions (Bayesian ridge) for each variable exhibiting missing values. To stabilize the distributions, a total of 10 iterations were conducted, followed by the averaging of results from five independent runs. This imputation was carried out, taking into account the internal correlations between variables as well as the confounders incorporated into the model. Let us presume that:

- $RCD \in \mathbb{R}^+$ – the duration of the release cycle (in days),
- $A, S \in [0, 1]$ – the level of CI/CD automation and the complexity of the release, respectively,
- $B, C, K, F \in \{1, 2, 3, 4, 5\}$ – subjective assessments of DX metrics,
- $P \in \mathbb{N}$ – the size of the development team. The model diagram is shown in Fig 1.

The model assumes the following vector of variables:

$$\mathbf{X} = (RCD, A, B, C, K, F, P, S) \tag{1}$$

The Bayesian network defines the joint distribution of all variables through the product of conditional probabil-

Table 1. Variable description

Category	Variable (notation)	Description	Type
Target	RCD	Release cycle duration (days/hours)	Continuous
5*DX	B	Tool satisfaction (Developer Tooling)	Ordinal (1–5)
	C	Communication quality within the team	Ordinal (1–5)
	A	Automation of CI/CD processes	Ratio (0–1)
	K	Documentation quality	Ordinal (1–5)
	F	Feedback from code reviews	Ordinal (1–5)
Control variable	P	Number of developers in the team	Integer
Control variable	S	Feature complexity in the release (rated, 0-1)	Ratio

Source: consolidated by the authors

ities:

$$P(\mathbf{X}) = \prod_{i=1}^{n} P(X_i \mid Pa(X_i)) \quad (2)$$

where $Pa(X_i)$ – a set of causal nodes of a variable X_i in the column.

The conditional distribution for the target variable RCD is written as follows:

$$P(RCD \mid A, B, C, K, F, P, S) \quad (3)$$

For a continuous variable RCD, the parameterization through a conditional normal distribution was chosen:

$$RCD \sim \mathcal{N}(\mu, \sigma^2) \quad (4)$$

$$\mu = \beta_0 + \beta_A A + \beta_B B + \beta_C C + \beta_K K + \beta_F F + \beta_P P + \beta_S S \quad (5)$$

The graph structure was trained using the MMNS method [29]. The parameters were evaluated using Bayesian and frequentist learning. In the case of normal variables, the linear regression was used.

Methods were used to assess the causal effect of variable A (e.g., CI/CD automation) on the duration of the RCD release cycle. intervention analysis (pre-intervention) [30], proposed by Judea Pearl. According to these methods, the Average Treatment Effect (ATE) is defined as follows:

$$ATE = \mathbb{E}[RCD \mid do(A = 1)] - \mathbb{E}[RCD \mid do(A = 0)] \quad (6)$$

where $do(A = 1)$ – full automation of CI/CD, $do(A = 0)$ – lack of automation. These interventions allow us to assess not only associations, but also causal effects.

The quality of the model was assessed through a multitude of indicators. Among these are the following: Log-likelihood – which reflects the model's likelihood; BIC/AIC – metrics utilized for the comparative evaluation of models in terms of complexity and precision; Structural Hamming Distance (SHD) – representing the deviation of the model's structure from the ground truth (utilizing synthetic data); RMSE/MAE (RCD) – measuring the accuracy in predicting the target variable; and Causal Estimation Error – indicating the error in estimating the causal effect based on simulated data. The proposed model facilitates: the identification of pivotal factors that influence the duration of the release cycle; the execution of intervention analyses aimed at formulating practical recommendations; and additionally, it ensures a high degree of results interpretability.

3.4 Research tools

The model is implemented in the Python 3.11 environment using the following libraries: bnlearn – for building the graph structure; pgmpy, DoWhy – for causal modeling; PyMC3 – for Bayesian learning of parameters; scikit-learn – for model validation (cross-verification, errors). The modeling was conducted in a local environment with hardware acceleration.

To support methodological reproducibility, all model assumptions, variable definitions, and causal relationships are clearly described in the manuscript. The Bayesian causal framework, parameter estimation procedure, and validation metrics allow the analytical workflow to be replicated using alternative implemen-

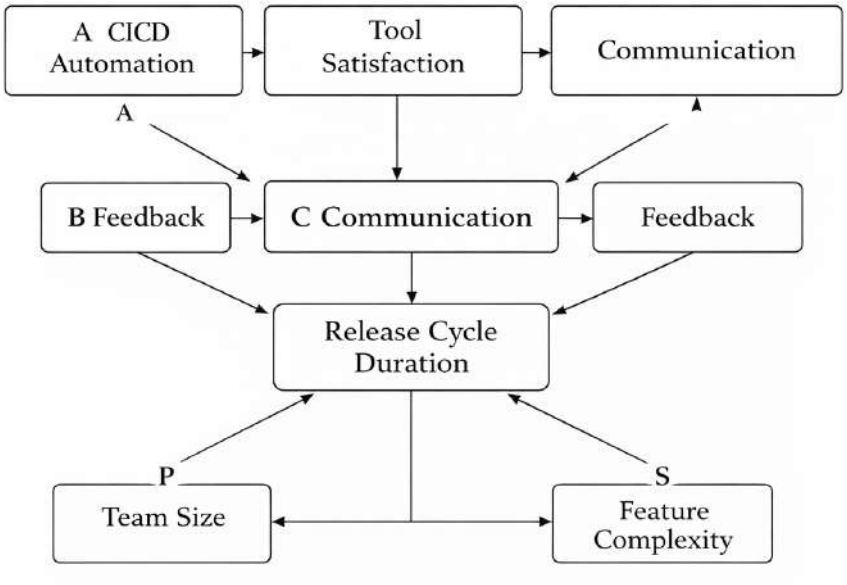

Figure 1. Directed Acyclic Graph (DAG) representing the causal structure of the proposed Bayesian model. The target variable is Release Cycle Duration (RCD), influenced by Developer Experience metrics (A, B, C, K, F) and control variables (P, S). Additional dependencies between explanatory variables (A $\rightarrow$ B, K $\rightarrow$ B, F $\rightarrow$ C) were considered during structural learning.

Source: consolidated by the authors

tations or datasets without the need for access to the original source code.

4 Results

A series of computational experiments were conducted using cross-validation and standard statistical learning quality metrics to evaluate the effectiveness of the constructed Bayesian model. The collected results allow us to assess both the model's ability to accurately predict the target variable (release duration) and its overall complexity, generalization, and probabilistic reliability. Table 2 presents the key metrics obtained based on the test dataset.

Table 2. Values of model quality assessment metrics, confirming the forecast adequacy

Metric	Value
Log-Likelihood	-124.3
Akaike Information Criterion (AIC)	268.6
Bayesian Information Criterion (BIC)	285.1
Average Predictive Accuracy (CV-10)	84.2%
Mean Absolute Error (MAE)	2.4 days
Root Mean Squared Error (RMSE)	3.1 days

Source: consolidated by the authors

The results of the model quality assessment indicate the adequacy of the constructed Bayesian causal structure for predicting the duration of the release cycle based on DevOps and DX metrics. The set of indicators used demonstrates that the model achieves a balanced compromise between accuracy and complexity, which is especially important given the multidimensional nature of the cause-and-effect relationships between variables. The obtained values of the information criteria confirm the appropriateness of the selected graph topology and set of parameters, as well as the absence of excessive model complexity. The metrics indicate a sufficient level of generalization ability and practical accuracy of predictions. This means that the model is resistant to overtraining and suitable for use in management and analytical scenarios. In general, the results confirm that the proposed model can be used as a tool for analyzing

factors affecting the time characteristics of the release process, as well as for further scenario modeling of changes in the DevOps environment.

The analysis of the conditional probabilities derived from the constructed Bayesian network enables us to assess the critical determinants affecting the duration of the release cycle that exceeds 14 days (RCD > 14). The threshold of 14 days was chosen as a conditional indicator of release delay based on typical practices of two-week sprints in Agile/DevOps environments. Exceeding this period means going beyond the standard iteration cycle and is considered an operational release delay. Thus, an RCD value > 14 days is interpreted as a violation of the expected functionality delivery cycle. The findings are presented in Table 3.

The data presented in the conditional probability table illustrate the impact of key DX metrics on the risk of extending the release cycle beyond a given threshold. Discretization of indicators allows us to compare how changing the level of each individual factor affects the probability of delay, provided that other variables are fixed. This approach facilitates the interpretation of the results and makes them suitable for practical use in managing DevOps processes.

The analysis of conditional distributions indicates a high sensitivity of the release process to both technical factors (CI/CD integration speed, tool quality) and human factors (developer satisfaction, frequency of switching between tasks). In particular, ineffective integration, low quality of the tool environment and high level of cognitive load significantly increase the risk of delays. At the same time, structural characteristics of the release, such as the scope of functionality, confirm the importance of clear planning of iterations and limiting the complexity of deliveries.

Furthermore, a series of conditional simulations were executed based on a constructed Bayesian network with the aim of quantitatively assessing the impact of alterations in pivotal characteristics of the DevOps environment on the duration of the release cycle. Below, the outcomes of modeling four strategic scenarios are presented, wherein the conditional probability of an extended release cycle duration (RCD) exceeding 14 days is evaluated. The simulated DevOps processes were conceptualized within a quintessential cloud environment (Cloud Computing), which represents the benchmark for contemporary software deployment.

Scenario 1. Optimization of the CI/CD process – transition from Slow to Fast integration.

Before: CI/CD Integration Time = Slow $\rightarrow$ P(RCD > 14 days) = 0.83

After: CI/CD Integration Time = Fast $\rightarrow$ P(RCD > 14 days) = 0.21

Optimizing CI/CD processes (e.g., by implementing automated tests, continuous integration, and deployment) leads to a 62 percentage point reduction in the probability of release delays, indicating the maximum effectiveness of this intervention among all tested scenarios.

Scenario 2. Reducing context switching frequency from High ($\geq$6 tasks/day) to Low ($\leq$2 tasks/day).

Before: Context Switching Frequency = High $\rightarrow$ P(RCD > 14 days) = 0.75

After: Context Switching Frequency = Low $\rightarrow$ P(RCD > 14 days) = 0.17

Organizational changes aimed at reducing context switching frequency (e.g., time-blocking, task batching, reducing ad-hoc communications) ensure a 4.4-fold reduction in the risk of release delays. This highlights the importance of cognitive load on the development team.

Scenario 3. Reducing the scope of release functionality (Feature Scope) from >40 to $\leq$20 story points.

Before: Feature Scope = Large $\rightarrow$ P(RCD > 14 days) = 0.61

After: Feature Scope = Small $\rightarrow$ P(RCD > 14 days) = 0.19

Adapting Agile practices, in particular iterative development with a small volume of tasks in each sprint, allows to reduce the probability of delay by 42 percentage points, improving the predictability and stability of the release process.

Table 3. Conditional probabilities of the impact of DE metrics on release duration (RCD)

DE Metrics	Categories / Levels	*P*(RCD > 14 days | condition)
CI/CD Integration Time	Fast / Moderate / Slow	0.21 / 0.56 / 0.83
Developer Tooling Quality	High / Medium / Low	0.18 / 0.39 / 0.72
Developer Satisfaction Index	High / Low	0.22 / 0.64
Context Switching Frequency (tasks/day)	Low ($\leq$2) / Medium / High ($\geq$6)	0.17 / 0.43 / 0.75
Feature Scope (story points)	$\leq$20 / 21–40 / >40	0.19 / 0.37 / 0.61

Source: consolidated by the authors

Scenario 4: Improving the quality of development tools (Tooling Quality) from Low to High.

Before: Tooling Quality = Low $\rightarrow$ *P*(RCD > 14 days) = 0.72

After: Tooling Quality = High $\rightarrow$ *P*(RCD > 14 days) = 0.18

Providing modern, reliable and fast tooling (e.g., IDE, CI/CD plugins, integrated monitoring panels) ensures a reduction in the risk of delay by 54 percentage points. This indicates that the technical environment of the developer critically affects the speed of the release. The summary of scenario modeling is presented in Table 4.

The results of the scenario analysis show that all four interventions demonstrate a significant reduction in the risk of release delays. The most powerful effect is observed when optimizing CI/CD processes and reducing the cognitive load of developers due to reduced context switching. The scenarios also confirm the effectiveness of Agile-oriented functionality management and investment in the quality of the technical ecosystem. What is more, the model allows for combining scenarios. For example, combining high-quality tooling + fast CI/CD can reduce the conditional probability of exceeding 14 days to <10%, even under initially unfavorable conditions.

Conceptual sensitivity analysis suggests that the underlying causal relationships remain stable under alternative model specifications.

To evaluate the predictive performance of the proposed Bayesian causal model, a comparative analysis was conducted with traditional regression approaches, as shown in Table 5. Two baseline models were implemented: linear regression and random forest regression.

All models were trained using the same dataset and evaluated using identical cross-validation procedures. The prediction accuracy was evaluated using root mean square error (RMSE) and mean absolute error (MAE).

The results show that the proposed Bayesian model provides competitive predictive performance and also offers a causal interpretation that traditional regression models cannot provide.

5 Discussion

The results of the study confirmed the presence of a statistically significant causal effect of individual DX metrics on the speed of release in DevOps environments. In particular, increased cognitive load and low quality feedback are associated with longer release delays, while reducing technical frustration (in particular, associated with poorly documented RESTful APIs and inefficient CI/CD processes) significantly reduces the expected duration of the release cycle. The proposed Bayesian model allows not only to quantitatively assess these effects, but also to perform "what-if" scenario modeling, supporting informed engineering and management decisions in DevOps teams. Unlike the correlation-oriented approaches presented in the DORA reports [7, 8], which capture statistical relationships between DevOps practices and performance, this study uses structural Bayesian causal modeling. This approach provides a transition from a descriptive analysis to a predictive and recommendatory paradigm, allowing for a formal assessment of the consequences of management interventions, such as changes in the level of CI/CD automation or feedback mechanisms. This is consistent with previous work on optimizing individual aspects of DevOps, including code governance [11], tooling

Table 4. Summary table of scenarios

Scenario	P(RCD > 14 days): before → after	Δ probability
CI/CD optimization	0.83 → 0.21	−0.62
Less context switching	0.75 → 0.17	−0.58
Improvement of dev. tools quality	0.72 → 0.18	−0.54
Reduction of the scope of functionality in the release	0.61 → 0.19	−0.42

Source: consolidated by the authors

Table 5. Comparative evaluation of the model

Model	RMSE	MAE
Linear Regression	1.42	1.1
Random Forest	1.31	1.11
Bayesian Causal Model	1.28	0.98

environment [12], organizational culture [13], and SDLC acceleration [2], but extends them by integrating the human factor into a formal model.

A number of studies emphasize the importance of Developer Experience, but do not provide quantitative causal validation. In particular, [17] proposed a conceptual framework for DX without empirical support, while this study formalizes the relevant assumptions, demonstrating the key role of timely feedback. Studies [18], [19], and [31] have supported the use of Bayesian analysis in software engineering and related domains, and the results of this study extend these by adapting causal modeling to evaluate the performance of DevOps teams with a focus on DX metrics. Studies in security [20], agile systems [16], and CI/CD optimization [15] also support the need to consider not only technical aspects, but also organizational and human aspects. Similarly, previous studies have identified common DevOps problems, such as developer frustration and process inconsistency [32], and the relationship between DevOps policies and performance [33–35]. The current study builds on these lines by formalizing emotional and cognitive factors as causal predictors of release cycle length. Thus, the work not only confirms a number of previously proposed hypotheses, but also offers a quantitative, formalized model that integrates subjective and objective factors to support decision-making in DevOps environments.

The practical possibilities of the model are to support engineering decisions in DevOps teams, to expand monitoring approaches taking into account DX metrics, to optimize resource allocation by prioritizing interventions with the greatest predicted effect, and to use such models in DevEx and MLOps panels for strategic planning.

Thus, the results obtained directly relate to the research problem formulated earlier. The constructed Bayesian causal model provides a formal mechanism for quantitatively assessing the impact of DX metrics on release duration. Thus, the methodological framework is consistent with the stated goal of providing causality assessment and intervention-based scenario analysis in DevOps systems.

6 Study limitations

Despite the positive results, the study has several limitations that should be taken into account when interpreting the findings. Although causal methods were used, including Bayesian analysis, the data remains observational, which means that the influence of unmeasured latent variables cannot be ruled out. The depth of technical tracing was constrained: the study focused on integral indicators (e.g., time to release or stability level). However, it did not encompass a deep analysis of individual technological stacks or metrics of the internal DevOps cycle (e.g., the behavior of pipelines in real-time). The analytical outcomes delineate causal dependencies at a specific point in time, but do not explore their evolution within the context of rapid transformation of DevOps tools.

Despite the realistic nature of the simulation scenario, further research should include empirical validation of the model based on real DevOps data, including CI/CD logs, release histories, and developer survey results on

DX metrics. Combining behavioral and technical logs will allow us to refine the magnitude of causal effects and test the stability of the model in different organizational contexts.

7 Recommendations

Based on the findings obtained, it was found that the automation of testing and the implementation of continuous integration practices exert the most significant causal influence on minimizing deployment times and enhancing the stability of software releases. In light of this, the prioritization of such methodologies within software development processes should be strategically substantiated. Making management decisions concerning investments in DevOps infrastructure ought to be informed by the outcomes of causal analysis. This approach facilitates the identification of cause-and-effect relationships, rather than being constrained to mere correlation metrics or empirical approximations. Consequently, this methodology mitigates the risks of erroneous conclusions and increases the credibility of technical and economic strategies for the development of IT systems.

8 Conclusions

The relevance of this study arises from the growing need for a structured and well-substantiated methodology to assess the efficacy of DevOps practices. This is particularly pertinent when scrutinizing the influence of DX metrics on the speed and stability of software delivery. While the majority of existing solutions predominantly rely on correlation analyses or descriptive reports, the causal dimension of the interplay between DX metrics and release performance remains insufficiently examined. The present study fills this gap.

The research conducted reveals that the synergy of DevOps practices and Bayesian methods of causal analysis establishes a robust foundation for objective change management within intricate software organizations. In contrast to conventional methodologies that depend on superficial performance metrics, the proposed framework facilitates the identification of the genuine levers that affect key attributes of quality, speed, and reliability in development. This paves the way for the creation of adaptive DevOps systems, wherein analytics not only observes but also learns from historical data, formulates hypotheses regarding future outcomes, and bolsters causally grounded decision-making.

It is expedient that the further research should be focused on the following areas:

- combining causal analytics with Machine Learning Operations (MLOps) for self-learning CI/CD systems;
- developing interactive tools that would allow DevOps teams to independently perform causal modeling based on available logs;
- studying real-time causal dynamics in high-load cloud environments.

Thus, the findings of the study possess both theoretical and practical significance, establishing a foundation for a new era of analytically driven DevOps. The resultant model can be employed within DevOps analytics systems, MLOps dashboards, and strategic technical debt management tools. Furthermore, it is also useful for engineering decision support systems designed to enhance the efficacy of development teams operating within scalable cloud architectures.

Author Contributions

Volodymyr Lopukhovych, Roman Martynenko, Dmytro Poltavskyi, Oleksandr Shvaikin, and Oleh Sypiahin: Contributed equally to Conceptualization, Data Curation, Methodology, Sources, Visualization, Writing – Original Draft, and Writing – Proofreading and Editing.

Funding Information

The authors did not receive any funding during conducting the research.

Compliance with Ethical Standards

It is declared that all authors do not have any conflict of interest. Furthermore, informed consent was obtained from all individual participants included in the study.

AI Assistance Disclosure

The authors declare that the artificial intelligence (AI) tool ChatGPT was used only for language editing, formatting, or technical refinement. No AI tool was used for the generation of research data, analysis, results, interpretations, or cited scholarly content. All AI-assisted

content was reviewed and validated by the authors, who take full responsibility for the final manuscript.

Data Availability

The code and synthetic data generation scripts used in this study are not publicly available due to intellectual property and institutional restrictions but may be available from the corresponding author upon reasonable request.

Conflict of Interest

The authors declare that they have no conflict of interest.

Funding

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Ethical Approval

This study does not involve human participants, animal experiments, or identifiable personal data. Therefore, ethical approval was not required.

References

[1] M. S. Khan, A. W. Khan, F. Khan, M. A. Khan, and T. K. Whangbo, "Critical challenges to adopt devops culture in software organizations: A systematic review," *IEEE Access*, vol. 10, pp. 14339–14349, 2022.

[2] T. Blake, "Exploring the influence of devops on accelerating the software development life cycle," *SSRN Electronic Journal*, vol. 10, pp. 804–811, 2020.

[3] N. M. Noorani, A. T. Zamani, M. Alenezi, M. Shameem, and P. Singh, "Factor prioritization for effectively implementing devops in software development organizations: A swot-ahp approach," *Axioms*, vol. 11, no. 10, p. 498, 2022.

[4] M. A. I. A. Ayyash, *Implementing Agile and DevOps at Scale: Identifying Best Frameworks, Practices, and Success Factors*. PhD thesis, Al-Quds University, Palestine, 2024.

[5] R. Anandya, T. Raharjo, and A. Suhanto, "Challenges of devops implementation: A case study from technology companies in indonesia," in *2021 International Conference on Informatics, Multimedia, Cyber and Information System (ICIMCIS)*, pp. 108–113, IEEE, 2021.

[6] B. Pando, A. Silva, and A. Dávila, "Devops adoption: A tertiary study," in *New Perspectives in Software Engineering. Studies in Computational Intelligence* (J. Mejía, M. Muñoz, A. Rocha, Y. Hernández Pérez, and H. Avila-George, eds.), vol. 1135, (Cham), Springer, 2024.

[7] F. Li, P. Ding, and F. Mealli, "Bayesian causal inference: A critical review," *Philosophical Transactions of the Royal Society*, vol. 381, no. 2247, p. 20220153, 2023.

[8] A. Razzaq, J. Buckley, Q. Lai, T. Yu, and G. Botterweck, "A systematic literature review on the influence of enhanced developer experience on developers' productivity: Factors, practices, and recommendations," *ACM Computing Surveys*, vol. 57, no. 1, pp. 1–46, 2024.

[9] J. G. Lopes, J. Oliveira, and E. Figueiredo, "Evaluating the impact of developer experience on code quality: A systematic literature review," in *Congresso Ibero-Americano em Engenharia de Software (CIbSE)*, pp. 166–180, SBC, 2024.

[10] W. A. Kusuma, A. H. Jantan, N. I. Admodisastro, and N. M. Norowi, "Enhancing novice developer efficacy through ux journey: Integrating user experience and user requirement to develop developer skills," *JOIV: International Journal on Informatics Visualization*, vol. 8, no. 3, pp. 1040–1048, 2024.

[11] J. Cui, "The role of devops in enhancing enterprise software delivery success through r&d efficiency and source code management," *arXiv preprint arXiv:2411.02209*, 2024.

[12] T. Offerman, R. Blinde, C. J. Stettina, and J. Visser, "A study of adoption and effects of devops practices," in *2022 IEEE 28th International Conference on Engineering, Technology and Innovation (ICE/ITMC) & 31st International Association For Management of Technology (IAMOT) Joint Conference*, (Nancy, France), pp. 1–9, IEEE, 2022.

[13] M. L. Pedra, M. F. da Silva, and L. G. Azevedo, "Devops adoption: Eight emergent perspectives," *arXiv preprint arXiv:2109.09601*, 2021.

[14] J. M. S. Ruiz, F. J. D. Mayo, X. Oriol, J. F. Crespo, D. Benavides, and E. Teniente, "A benchmarking proposal for devops practices on open source software projects," *arXiv preprint arXiv:2304.14790*, 2023.

[15] J. Fluri, F. Fornari, and E. Pustulka, "Measuring the benefits of ci/cd practices for database application development," in *2023 IEEE/ACM International Conference on Software and System Processes (ICSSP)*, (Melbourne, Australia), pp. 46–57, IEEE, 2023.

[16] K. Noreika and S. Gudas, "Causal knowledge modelling for agile development of enterprise application systems," *Informatica*, vol. 34, no. 1, pp. 121–146, 2023.

[17] M. Greiler, M. A. Storey, and A. Noda, "An actionable framework for understanding and improving developer experience," *IEEE Transactions on Software Engineering*, vol. 49, no. 4, pp. 1411–1425, 2022.

[18] C. A. Furia, R. Torkar, and R. Feldt, "Applying bayesian analysis guidelines to empirical software engineering data: The case of programming languages and code quality," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 31, no. 3, pp. 1–38, 2022.

[19] Y. Liu, D. I. Mattos, J. Bosch, H. H. Olsson, and J. Lantz, "Bayesian causal inference in automotive software engineering and online evaluation," *arXiv preprint arXiv:2207.00222*, 2022.

[20] R. M. Czekster, "Continuous risk assessment in secure devops," *arXiv preprint arXiv:2409.03405*, 2024.

[21] B. John, I. Adeoye, and B. Tomford, "Reinforcement learning for cybersecurity risk modeling in ci/cd pipelines: Optimizing test strategies for threat mitigation," 2025.

[22] A. Oganisian and J. A. Roy, "A practical introduction to bayesian estimation of causal effects: Parametric and nonparametric approaches," *Statistics in Medicine*, vol. 40, no. 2, pp. 518–551, 2021.

[23] L. Shams and U. Beierholm, "Bayesian causal inference: A unifying neuroscience theory," *Neuroscience & Biobehavioral Reviews*, vol. 137, p. 104619, 2022.

[24] A. M. Lipsky and S. Greenland, "Causal directed acyclic graphs," *Jama*, vol. 327, no. 11, pp. 1083–1084, 2022.

[25] J. C. Digitale, J. N. Martin, and M. M. Glymour, "Tutorial on directed acyclic graphs," *Journal of Clinical Epidemiology*, vol. 142, pp. 264–267, 2022.

[26] J. Ching, S. Wu, and K. K. Phoon, "Constructing quasi-site-specific multivariate probability distribution using hierarchical bayesian model," *Journal of Engineering Mechanics*, vol. 147, no. 10, p. 04021069, 2021.

[27] Y. Tao, K. K. Phoon, H. Sun, and Y. Cai, "Hierarchical bayesian model for predicting small-strain stiffness of sand," *Canadian Geotechnical Journal*, vol. 61, no. 4, pp. 668–683, 2023.

[28] K. Cordova-Pozo and E. A. Rouwette, "Types of scenario planning and their effectiveness: A review of reviews," *Futures*, vol. 149, p. 103153, 2023.

[29] H. S. Laqueur, A. B. Shev, and R. M. Kagawa, "Supermice: An ensemble machine learning approach to multiple imputation by chained equations," *American Journal of Epidemiology*, vol. 191, no. 3, pp. 516–525, 2022.

[30] E. Slade and M. G. Naylor, "A fair comparison of tree-based and parametric methods in multiple imputation by chained equations," *Statistics in Medicine*, vol. 39, no. 8, pp. 1156–1166, 2020.

[31] W. Song, H. Gong, Q. Wang, L. Zhang, L. Qiu, X. Hu, and Y. Li, "Using bayesian networks with max-min hill-climbing algorithm to detect factors related to multimorbidity," *Frontiers in Cardiovascular Medicine*, vol. 9, p. 984883, 2022.

[32] Y. Lu, Q. Zheng, and D. Quinn, "Introducing causal inference using bayesian networks and do-calculus," *Journal of Statistics and Data Science Education*, vol. 31, no. 1, pp. 3–17, 2023.

[33] J. Frattini, D. Fucci, R. Torkar, L. Montgomery, M. Unterkalmsteiner, J. Fischbach, and D. Mendez, "Applying bayesian data analysis for causal inference about requirements quality: A controlled experiment," *Empirical Software Engineering*, vol. 30, no. 1, p. 29, 2025.

[34] M. H. Tanzil, M. Sarker, G. Uddin, and A. Iqbal, "A mixed method study of devops challenges," *Information and Software Technology*, vol. 161, p. 107244, 2023.

[35] D. Port, B. Taber, and P. Emkani, "Investigating effectiveness and compliance to devops policies and practices for managing productivity and quality variability," *Journal of Systems and Software*, vol. 213, p. 112030, 2024.