**FEDERAL UNIVERSITY OF CEARÁ**

**CENTER OF SCIENCES**

**DEPARTMENT OF COMPUTER SCIENCE**

**PROGRAM OF MASTER AND DOCTORATE IN COMPUTER SCIENCE**

**PEDRO PAULO CORTEZ**

# THE IMPACT OF CONTINUOUS PRACTICES ON SOFTWARE MAINTENANCE AND EVOLUTION: AN INDUSTRIAL CASE STUDY

**FORTALEZA**

**2025**

PEDRO PAULO CORTEZ

# THE IMPACT OF CONTINUOUS PRACTICES ON SOFTWARE MAINTENANCE AND EVOLUTION: AN INDUSTRIAL CASE STUDY

Dissertation submitted to the Program of Master And Doctorate in Computer Science of the Center of Sciences of the Federal University of Ceará, as a partial requirement for obtaining the title of Master in Computer Science. Concentration Area: Software Engineering

Advisor: Prof. Dr. Lincoln Souza Rocha

Co-advisor: Prof. Dr. Camilo Camilo Almendra

FORTALEZA

2025

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Sistema de Bibliotecas
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

---

C858t Cortez, Pedro Paulo.
The impact of continuous practices on software maintenance and evolution : an industrial case study / Pedro Paulo Cortez. – 2025.
65 f. : il. color.

Dissertação (mestrado) – Universidade Federal do Ceará, Centro de Ciências, Programa de Pós-Graduação em Ciência da Computação, Fortaleza, 2025.
Orientação: Prof. Dr. Lincoln Rocha.
Coorientação: Prof. Dr. Camilo Almendra.

1. Manutenção e evolução de software. 2. Práticas contínuas. 3. Estudo de caso. I. Título.
CDD 005

---

PEDRO PAULO CORTEZ

# THE IMPACT OF CONTINUOUS PRACTICES ON SOFTWARE MAINTENANCE AND EVOLUTION: AN INDUSTRIAL CASE STUDY

Dissertation submitted to the Program of Master And Doctorate in Computer Science of the Center of Sciences of the Federal University of Ceará, as a partial requirement for obtaining the title of Master in Computer Science. Concentration Area: Software Engineering

Approved on: 5th August 2025

EXAMINATION BOARD

---

Prof. Dr. Lincoln Souza Rocha (Advisor)
Federal University of Ceará (UFC)

---

Prof. Dr. Camilo Camilo Almendra (Co-advisor)
Federal University of Ceará (UFC)

---

Prof. Dr. Fernando Antônio Mota Trinta
Federal University of Ceará (UFC)

---

Prof. Dr. Guilherme Horta Travassos
Federal University of Rio de Janeiro (UFRJ)

## ACKNOWLEDGEMENTS

I would like to sincerely thank my advisor, Professor Lincoln, for his guidance, encouragement, and constant support throughout this research. His experience, availability, and thoughtful feedback were essential at every stage of this work and played a decisive role in its development. I am also very grateful to my co-advisor, Camilo Almendra, whose insights, discussions, and continuous support greatly contributed to refining both the research approach and the final results.

I would also like to thank my fellow master's students for their collaboration, exchanges of ideas, and mutual support during this journey. The discussions, feedback, and shared experiences were invaluable and significantly enriched this research.

My sincere thanks go to the company that provided the data used in this study, making this empirical investigation possible. I am also grateful to everyone who contributed in any way to this work, whether through careful reviews, constructive suggestions, or meaningful discussions that helped shape and improve the quality of the research.

I would like to acknowledge the university for offering an academic environment that fostered learning, research, and critical thinking, as well as for the institutional support provided throughout the master's program.

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001.

## ABSTRACT

Context: Continuous practices (e.g., CI/CD, observability, automated testing) are widely promoted as ways to enhance software delivery and quality. However, their impact in regulated, legacy-constrained environments remains unclear.

Objective: This study investigates how continuous practices affect both delivery performance and team experience in a mission-critical system from the industry. We aim to understand both the technical outcomes and the organizational shifts that accompany the process.

Method: We conducted a mixed-methods case study combining quantitative analysis of development tasks with qualitative insights from structured interviews. The analysis included time-based metrics (e.g., active working time, completion time), descriptive statistics, and hypothesis testing. Outliers were removed using an interquartile range (IQR) strategy. The interviews explored perceptions of challenges, software quality, and workflow evolution.

Results: Statistical tests did not indicate significant differences in the time metrics. However, geometric trends and reduced variance suggest improvements. Interview responses further revealed that continuous practices improved the perceived software quality and process stability.

Conclusions: The study offers empirical evidence on the effects of continuous practices in a real-world, legacy-constrained context. Beyond tools, success depends on how well these practices fit with existing workflows and constraints. We hope this case contributes practical lessons for teams facing similar modernization journeys.



**Keywords:** software maintenance and evolution ; continuous practices ; case study.

## RESUMO

Contexto: Práticas contínuas (como CI/CD, observabilidade e testes automatizados) são amplamente promovidas como formas de aprimorar a entrega e a qualidade de software. No entanto, seu impacto em ambientes regulados e com restrições de legados ainda é pouco claro.

Objetivo: Este estudo investiga como as práticas contínuas afetam tanto o desempenho da entrega quanto a experiência da equipe em um sistema crítico da indústria. Nosso objetivo é compreender tanto os resultados técnicos quanto as mudanças organizacionais que acompanham esse processo.

Método: Conduzimos um estudo de caso com métodos mistos, combinando análise quantitativa de tarefas de desenvolvimento com insights qualitativos obtidos por meio de entrevistas estruturadas. A análise incluiu métricas baseadas em tempo (por exemplo, tempo de trabalho ativo, tempo de conclusão), estatísticas descritivas e testes de hipótese. Outliers foram removidos utilizando a estratégia do intervalo interquartil (IQR). As entrevistas exploraram percepções sobre desafios, qualidade do software e evolução do fluxo de trabalho.

Resultados: Os testes estatísticos de hipótese não indicaram diferenças significativas nas métricas de tempo avaliadas. No entanto, foi observada uma tendência de redução geométrica (média geométrica) no tempo de trabalho ativo e no tempo de conclusão após a adoção das práticas contínuas, além de uma diminuição na variabilidade. As respostas das entrevistas também revelaram que as práticas contínuas melhoraram a percepção de qualidade e previsibilidade do software.

Conclusões: O estudo fornece evidências empíricas sobre os efeitos das práticas contínuas em um contexto real, com restrições de legado. Para além das ferramentas, o sucesso depende de como essas práticas se encaixam nos fluxos de trabalho e restrições existentes. Esperamos que este caso contribua com lições práticas para equipes que enfrentam jornadas semelhantes de modernização.



**Palavras-chave:** manutenção e evolução de software ; práticas contínuas ; estudo de caso.

## LIST OF FIGURES

## LIST OF TABLES

## LIST OF ABBREVIATIONS AND ACRONYMS

| | |
|---|---|
| CD | Continuous Delivery |
| CI | Continuous Integration |
| CI/CD | Continuous Integration and Continuous Delivery |
| DevOps | Development and Operations |
| IaC | Infrastructure as Code |
| PHP | Hypertext Preprocessor |
| RQ1 | Research Question 1 |
| RQ2 | Research Question 2 |

## CONTENTS

# 1 INTRODUCTION

Software development teams are under increasing pressure to deliver high-quality software faster and more efficiently. Traditional development approaches, typically characterized by long release cycles, manual handoffs, and disconnected toolchains, often fail to meet the demands of modern software delivery (FORSGREN *et al.*, 2018; HüTTERMANN, 2012). In response, continuous practices have gained traction by promoting automation, collaboration, and rapid feedback cycles across the software lifecycle (BASS *et al.*, 2015).

Although the benefits of Development and Operations (DevOps) are well documented, adopting continuous practices in environments constrained by legacy systems presents unique challenges. Organizations with long-standing codebases, tightly coupled architectures, and siloed team structures usually face difficulties when introducing automation and modern delivery workflows (AZAD, 2022; KHAN *et al.*, 2022; MACARTHY; BASS, 2021). These challenges are amplified in large enterprises, where historical technical debt and process inertia inhibit swift transformations. Prior studies have predominantly focused on adoption in greenfield or cloud-native scenarios (ROMERO *et al.*, 2022; SU; STORER, 2023), offering limited insight into how DevOps evolves in environments shaped by historical constraints.

This dissertation addresses this underexplored area through a mixed-methods case study of continuous practices adoption in a mission-critical system from the energy sector. The target system operates under legacy and regulatory constraints and manages meteorological tower measurements, playing a critical role in the company pipeline. The transformation aimed to make the development process more efficient and consistent, while also improving the team's experience. Since the impact of continuous practices depends on context, this case offers empirical evidence from a regulated environment where modernization had to adapt to legacy infrastructure and organizational complexity.

## 1.1 Context

The software engineering community has increasingly recognized continuous practices as key enablers of delivery performance, quality assurance, and collaboration (FORSGREN *et al.*, 2018; BASS *et al.*, 2015). However, empirical evidence remains limited regarding their effects in contexts dominated by legacy systems and regulatory restrictions, where modernization efforts are constrained by legacy codebases, technical debt, and compliance demands.

This study was motivated not only by this research gap, but also by the special opportunity to access a large-scale industrial system undergoing the gradual adoption of continuous practices. Such access enabled the direct observation of real-world workflows, data, and practitioners' perspectives, a valuable condition in empirical software engineering studies.

By addressing this research need and leveraging this empirical opportunity, the research contributes to both theory and practice, offering lessons that complement the predominantly greenfield and cloud-native focus of prior work by examining a legacy system under modernization.

## 1.2 Problem Statement, Goals, and Research Questions

Despite growing advocacy for continuous practices, little is known about their measurable and perceived impact in legacy-constrained environments. The central problem investigated in this dissertation is:

> *How do continuous practices affect software maintenance and evolution tasks in a mission-critical system constrained by legacy architecture and regulatory requirements?*

The research pursued three main goals:

1. Quantitatively assess the impact of continuous practices on task performance and delivery metrics.
2. Qualitatively capture practitioner perceptions regarding benefits, challenges, and workflow transformations.
3. Obtain actionable insights to support adoption in regulated, legacy-bound contexts.

Based on these goals, the following research questions (RQs) were defined:

- **RQ1:** To what extent does the adoption of continuous practices improve software development productivity?
- **RQ2:** What are practitioners' perceptions of the benefits, challenges, and workflow changes associated with the adoption of continuous practices?

To operationalize these objectives, the study followed the Goal–Question–Metric (GQM) paradigm (RUIZ *et al.*, 2015). Table 1 provides a high-level mapping between goals, research questions, and data sources. A detailed operationalization, including specific analysis methods, is presented in Chapter 3 (Table 2).

Table 1 – High-level GQM mapping presented in the Introduction. Detailed operationalization appears in Chapter 3.

| **Goal** | **Research Question** | **Metrics / Data Source** |
|---|---|---|
| G1: Assess impact on productivity | RQ1: To what extent does the adoption of continuous practices improve software development productivity? | Task records |
| G2: Capture practitioner perceptions | RQ2: What are practitioners' perceptions of the benefits, challenges, and workflow changes associated with the adoption of continuous practices? | Structured interviews |

Out of scope for this research are: (i) a full DevOps transformation (focus is restricted to selected continuous practices), (ii) multi-case comparisons across organizations, and (iii) long-term effects beyond the period studied (2021–2023).

## 1.3 Main Contributions and Limitations

### *1.3.1 Contributions*

This subsection summarizes the key contributions of this dissertation, highlighting both its empirical and practical relevance to the study of continuous practices in software engineering.

This dissertation makes three main contributions:

1. Empirical Evidence: A quantitative analysis of software task metrics before and after the adoption of continuous practices, examining changes in productivity and workflow consistency.
2. Qualitative Insights: Developer-centered accounts describing perceived benefits, bottlenecks, and organizational challenges faced during the transformation.
3. Theoretical and Practical Guidance: Recommendations for adopting continuous practices in organizations shaped by legacy systems and regulatory constraints.

### *1.3.2 Limitations*

While the study provides valuable insights, certain limitations should be acknowledged to ensure a balanced interpretation of its findings:

1. It analyzes a single case study, which may limit the generalizability of findings.
2. The dataset comprises 181 tasks, which restricts statistical power for hypothesis testing.
3. Practitioner perceptions may be influenced by contextual or organizational biases.

## 1.4 Thesis Outline

The remainder of this dissertation is structured as follows: Chapter 2 reviews the theoretical foundations of DevOps and continuous practices, as well as empirical studies that motivate the case study. Chapter 3 describes the research design, case context, observed practices, and analysis procedures. Chapter 4 presents quantitative and qualitative results, integrating them into a broader discussion. Finally, Chapter 5 summarizes contributions, highlights limitations, and suggests directions for future research.

# 2 BACKGROUND AND RELATED WORK

This chapter situates our study within the DevOps literature and clarifies the concepts used in the dissertation. Section 2.1 introduces the foundations of DevOps and the specific continuous practices considered in this work. Section 2.2 reviews how prior studies have framed continuous practices in different organizational settings, highlighting success factors and common challenges. Section 2.3 consolidates empirical evidence on adoption and outcomes, with emphasis on legacy-constrained and regulated environments. Finally, Section 2.4 outlines the motivation for our case study by identifying gaps that motivate our research questions and the methodological choices detailed in Chapter 3.

As software systems grow in complexity, teams face more pressure to deliver updates quickly, reliably, and sustainably. DevOps has emerged as a response to these demands by integrating development and operations, fostering automation, continuous feedback, and collaborative ownership throughout the software lifecycle. The benefits of DevOps, like faster delivery, more frequent deployments, and improved reliability, are well documented in the literature (FORSGREN *et al.*, 2018; HüTTERMANN, 2012). However, applying continuous practices in established systems with high operational relevance remains a complex challenge, especially in environments shaped by legacy constraints.

## 2.1 Foundations of DevOps and Continuous Practices

DevOps is grounded in principles of automation, collaboration, and continuous improvement. By breaking down the traditional gaps between development (Dev) and operations (Ops), it enables rapid and reliable delivery of software (BASS *et al.*, 2015; AZAD; HYRYNSALMI, 2024). Core practices such as Continuous Integration (CI), Continuous Delivery (CD), and Infrastructure as Code (IaC) streamline development pipelines, reduce manual errors, and accelerate feedback loops.

To ensure terminological clarity, we adopt the definitions proposed by Ståhl et al. (STAHL *et al.*, 2017), which disentangle continuous practices from the broader DevOps paradigm. In their framework, *continuous integration* refers to a developer-level practice of frequently integrating code to detect integration issues early. *Continuous delivery* aims to keep the software in a deployable state at all times, while *continuous deployment* denotes the automated release of validated changes into production. These practices are concrete techniques within

a larger DevOps ecosystem, which also encompasses cultural, organizational, and governance dimensions. In this study, however, we do not evaluate a full DevOps transformation. Instead, we examine the adoption of a set of continuous practices, including but not limited to continuous integration and delivery, and analyze their measurable effects on software maintenance and evolution within a legacy-constrained, business-critical system.

These capabilities enhance agility while also supporting maintainability and resilience, qualities that are especially important in systems with high operational demands. Automated testing and deployment pipelines can reduce technical debt accumulation, while observability tools contribute to proactive fault detection and system stability (HüTTERMANN, 2012; DUARTE, 2021).

Despite these advantages, the integration of continuous practices is not trivial. Adopting continuous practices is especially difficult in legacy systems, where architectural complexity, fragmented tools, and resistance to change create barriers (MACARTHY; BASS, 2021; KHAN *et al.*, 2022).

### *2.1.1 Terminology and Scope*

In this dissertation, we analyze the adoption of selected practices (e.g., CI/CD, observability, secure handling of secrets, code review) rather than a full organizational DevOps transformation. For this reason we adopt the term *continuous practices* (STAHL *et al.*, 2017; AMARO *et al.*, 2023). This scope informs the research design and the operationalization adopted in Chapter 3 (GQM), where goals and research questions are mapped to measurable indicators and data sources.

## 2.2 Contextualizing Continuous Practices within DevOps Literature

Several empirical studies have demonstrated the benefits of continuous practices adoption. Forsgren et al. (FORSGREN *et al.*, 2018) reported that elite DevOps performers achieve faster deployment and greater system stability. Hüttermann (HüTTERMANN, 2012) and Bass et al. (BASS *et al.*, 2015) emphasize the role of automation in reducing lead times and improving recovery from failures.

Erich et al. (ERICH *et al.*, 2017) highlight the importance of cultural transformation for effective DevOps adoption, a finding echoed in more recent studies that frame DevOps

as a socio-technical shift requiring alignment across roles and responsibilities (WINKLER; WESTNER, 2023; SU; STORER, 2023).

However, much of the literature focuses on settings such as cloud-native platforms, startups, or large-scale enterprises where continuous practices are introduced into relatively modern architectures (ROMERO *et al.*, 2022; ALMEIDA *et al.*, 2022). These environments lack the architectural and procedural constraints of legacy systems, limiting the generalizability of findings to traditional or mission-critical contexts.

## 2.3 Empirical Foundations for Continuous Practices Adoption

A growing number of empirical studies have investigated the adoption and impact of continuous practices across a variety of organizational contexts. Case studies in traditional organizations, such as Su and Storer's analysis of a financial institution (SU; STORER, 2023) and Senapathi et al.'s study of a multinational team (SENAPATHI *et al.*, 2021), show that adoption is strongly shaped by organizational structures and team maturity.

Systematic literature reviews and multivocal studies have consolidated success factors, challenges, and metrics for evaluating DevOps outcomes (AMARO *et al.*, 2023; WINKLER; WESTNER, 2023; KUMAR *et al.*, 2023). These works emphasize that while technical enablers (e.g., CI/CD pipelines) are widely adopted, the realization of expected benefits often depends on socio-technical alignment and governance adaptation.

Studies such as Romero et al. (ROMERO *et al.*, 2022) and Almeida et al. (ALMEIDA *et al.*, 2022) report on real-world implementations of continuous practices, focusing on deployment automation and defect resolution, respectively. Complementary to these, Santos et al. (SANTOS *et al.*, 2024) and Soares et al. (SOARES *et al.*, 2021) offer empirical insights into the effects of continuous integration on task throughput, developer workflows and code quality.

Recent analyses have also called attention to the misalignment between perceived and statistically measurable benefits of continuous practices. For example, Port et al. (PORT *et al.*, 2024) and Zohaib (ZOHAIB, 2023) highlight that process improvements often coexist with persistent quality variability and developer uncertainty. In our case, combining both types of data helped explain the nuances behind the adoption process.

Together, these empirical contributions provide valuable foundations and methodological inspirations for the present study, which aims to evaluate the measurable and perceived impacts of continuous practices in a legacy-constrained, business-critical environment.

## 2.4 Study Motivation

Building on the empirical and multivocal evidence discussed above, we now motivate our case study by articulating the specific gaps that persist in legacy-constrained and regulated settings and that cannot be fully addressed by cloud-native or greenfield evidence alone.

As continuous practices mature, the software engineering community increasingly seeks empirical clarity regarding their real-world effects. While prior studies report benefits such as accelerated delivery and improved quality, the literature is not entirely consensual, especially when it comes to legacy-constrained or regulated environments. Several works indicate persistent variability in outcomes and highlight that organizational, architectural, and cultural factors (PORT *et al.*, 2024; ERICH *et al.*, 2017; SENAPATHI *et al.*, 2021) often mediate the actual benefits perceived by teams.

Most existing evidence comes from surveys or studies conducted in greenfield or cloud-native contexts, where technical and structural enablers are inherently different. There is, therefore, a growing need for detailed, context-rich primary studies that examine how continuous practices unfold in traditional enterprise systems, particularly where modernization intersects with legacy architectures, infrastructure rigidity, and compliance demands.

This study addresses that gap by presenting a mixed-methods case study in the energy sector, focusing on a mission-critical system subject to operational constraints and regulatory oversight. By combining quantitative analysis of task-level data with qualitative insights from practitioners, the study investigates how selected practices, such as Continuous Integration and Continuous Delivery (CI/CD) pipelines, observability mechanisms, and workflow adjustments, influence delivery performance and team experience.

The results contribute both methodological and practical value. They show how triangulated data can be used to study complex socio-technical settings and offer actionable insights for teams facing similar constraints. Success with continuous practices depends less on universal formulas and more on alignment with context.

## 2.5 Chapter Summary

This chapter introduced the conceptual background and situated our work within the DevOps and continuous practices literature. We clarified the terminology adopted (continuous integration, delivery, deployment) and reviewed empirical findings on adoption, benefits, and

challenges across contexts, with particular attention to legacy-constrained and regulated environments. The review highlights three points: (i) technical enablers such as CI/CD and observability are necessary but not sufficient; (ii) outcomes depend on socio-technical and governance factors; and (iii) predictability and stability usually come before consistent speed gains.

Building on these insights, Chapter 3 formalizes our *Goal–Question–Metric* (GQM) map, linking goals to research questions and concrete measures. This connection operationalizes the literature-derived expectations into an empirical design that triangulates task-level metrics (RQ1) and practitioner perceptions (RQ2) in the studied legacy-constrained system.

# 3 METHODOLOGY

This chapter presents the study design and the operationalization of our research objectives using the Goal–Question–Metric (GQM) paradigm. We first describe the research design, then present the GQM map that links goals, research questions, and metrics. Next, we detail the case context, data preparation, and the quantitative and qualitative analysis procedures. The chapter closes with a summary that links methods to the results chapters.

## 3.1 Research Design

We conducted an exploratory, single-case, mixed-methods study following established guidance for empirical software engineering (YIN, 2018; RUNESON; HÖST, 2009; KITCHENHAM; PFLEEGER, 2002). The unit of analysis included (i) work tasks (time-based performance indicators) and (ii) the project as a whole (practitioners' perceptions). Figure 1 depicts the dual-track workflow (quantitative $\leftrightarrow$ qualitative) and the triangulation step.

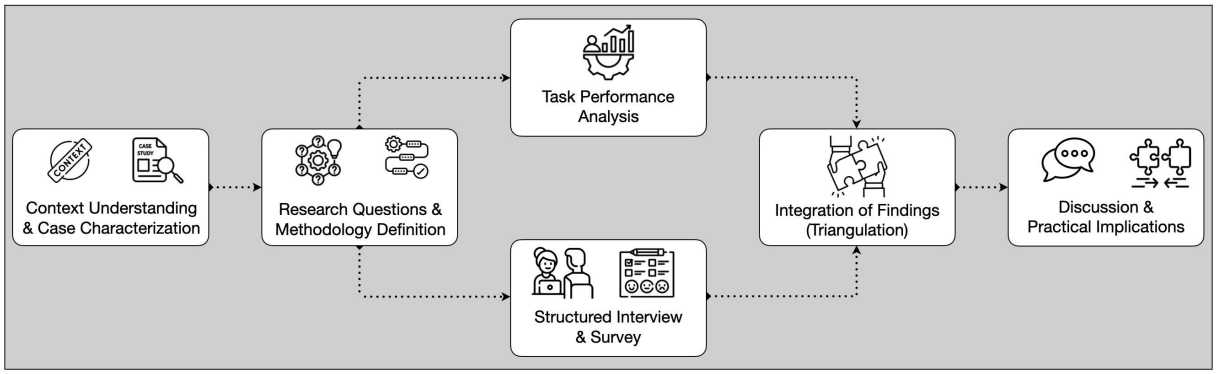


Figure 1 – Research design workflow integrating quantitative and qualitative approaches.

## 3.2 Goal–Question–Metric (GQM) Framework

We operationalized the study using the GQM paradigm, linking the objectives of the case study to research questions and concrete metrics/data sources.

### *3.2.1 Goals*

1. The first (**G1**) aims to *analyze* the maintenance and evolution process to *evaluate task efficiency and workflow time* from the *researcher's viewpoint*.
2. The second (**G2**) aims to *understand* the *perceived effects, benefits, and challenges* as-

sociated with continuous practices, focusing on the *software delivery and collaboration workflows* from the *practitioners' perspective*.

### *3.2.2 Research Questions*

**RQ1:** To what extent does the adoption of continuous practices improve software development productivity?

**RQ1.1** Does continuous practices adoption reduce the active work time per task?

**RQ1.2** Does continuous practices adoption reduce the total task completion time?

**RQ1.3** Does the adoption of continuous practices reduce the pre-work waiting time per task?

**RQ2:** What are practitioners' perceptions of the benefits, challenges, and workflow changes associated with the adoption of continuous practices?

**RQ2.1** How do developers perceive the influence of continuous practices on delivery speed?

**RQ2.2** How do developers assess the impact of continuous practices on software quality?

**RQ2.3** How do developers perceive the relationship between continuous practices and release frequency?

**RQ2.4** How do continuous practices influence developers' daily workflows?

**RQ2.5** What challenges do developers face when adopting continuous practices?

**RQ2.6** How do developers evaluate the individual relevance of the continuous practices adopted in the case study?

### *3.2.3 Metrics and Data Sources*

This subsection details the metrics and data sources defined to address each research question. Table 2 summarizes how task records and practitioner interviews were mapped to the study goals, providing a structured basis for quantitative and qualitative analysis. This mapping follows the Goal-Question-Metric (GQM) approach (BASILI *et al.*, 1994), which structures each goal according to its purpose, issue, object (process), and viewpoint.

The metrics presented in Table 2 provide an operational bridge between the study goals and the data sources used in the analysis. Quantitative indicators derived from task records are further detailed in Table 5, which consolidates the statistical definitions, aggregation levels, and measurement units applied in the study. It is important to note that, as discussed in Section 1.3, these measurements are constrained by the scope of a single industrial case and by the available

Table 2 – GQM mapping of goals, questions, and corresponding metrics and data sources.

| Goal | Research Question | Metrics and Data Source |
|---|---|---|
| G1 | RQ1.1 | Active work time per task (days)<br>Extracted from issue-tracking logs before and after adoption |
| | RQ1.2 | Total cycle time per task (days)<br>Measured from creation to closure in issue-tracking logs before and after adoption |
| | RQ1.3 | Pre-work waiting time between task creation and start (days)<br>Extracted from issue-tracking logs before and after adoption |
| G2 | RQ2.1 | (a) Likert-scale responses on perceived delivery speed<br>Survey (level of agreement)<br>(b) Open-ended comments on delivery pace and workflow adjustments<br>Interview transcripts (qualitative) |
| | RQ2.2 | (a) Likert-scale responses on perceived software quality<br>Survey (level of agreement)<br>(b) Open-ended examples of quality improvement<br>Interview transcripts (qualitative) |
| | RQ2.3 | (a) Likert-scale responses on release frequency and predictability<br>Survey (level of agreement)<br>(b) Open-ended descriptions of release process changes<br>Interview transcripts (qualitative) |
| | RQ2.4 | Open-ended descriptions of workflow and process adaptations<br>Interview transcripts (level of impact) |
| | RQ2.5 | Open-ended accounts of technical and organizational challenges<br>Interview transcripts (level of difficulty) |
| | RQ2.6 | (a) Likert-scale responses on perceived usefulness and relevance of practices<br>Survey (level of perceived usefulness and relevance)<br>(b) Open-ended reflections on practice usefulness<br>Interview transcripts (qualitative) |

volume of task data, which may limit broader generalization. Nevertheless, the mapping ensures methodological consistency between the quantitative and qualitative components of the research.

### 3.3 Case Study Context

This study investigates a mission-critical system in the energy sector responsible for managing meteorological tower measurements. The system has $\sim$ 15 years of evolution, $\sim$350k LOC in its main VM-deployed application, with supporting APIs. The stack includes Hypertext Preprocessor (PHP), JavaScript, and Python, serving 50–60 active users. The legacy, stateful architecture and regulatory constraints created barriers on automation and frequent releases.

As per Dingsøyr et al. (DINGSØYR *et al.*, 2014), the system qualifies as large-scale due to multi-team development and coordination needs. Table 3 summarizes salient attributes.

Table 3 – System characteristics at the time of analysis.

| **Aspect** | **Description** |
|---|---|
| Domain | Renewable Energy |
| Project Age | ∼15 years |
| Codebase Size | ∼350k LOC (main resource) |
| Stack | PHP, JavaScript, Python |
| Architecture | Monolithic, stateful components |
| Users | 50–60 active users |
| Regulatory Context | Compliance/stability requirements |
| Initial Model | Traditional SE, long release cycles |
| Team Structure | Multiple teams across units |
| Legacy Constraints | Limited automation, deploy friction, integration bottlenecks |

The company authorized data use and interviews; confidentiality and ethical procedures were observed (RUNESON; HÖST, 2009).

### 3.4 Continuous Practices Observed

The intervention (Dec/2022–Jan/2023) included the following continuous practices:

1. Continuous Integration and Delivery (CI/CD): Implemented with GitHub and Cloud Build, including automated tests, quality checks, and artifact promotion with approval gates.
2. Monitoring and Observability: Dashboards and alerts integrated into Chat, email, and internal tools for real-time visibility and incident response.
3. Workflow Modernization: Redefinition of roles, backlog management, and clearer team boundaries to support continuous delivery.
4. Collaboration Practices: Adoption of mandatory code reviews, pair programming, and shared deployment activities.
5. Secure Coding: Elimination of hardcoded secrets through the use of Secret Manager and supporting libraries.

These practices reflect targeted DevOps capabilities (STAHL *et al.*, 2017; DUARTE, 2021; SHAHIN *et al.*, 2017).

## 3.5 Data and Preparation

### *3.5.1 Task Dataset and Variables (G1/RQ1)*

We analyzed 181 valid tasks (Oct/2021–Dec/2023), extracted from the team's Kanban API. Each task has timestamps for creation, Doing, and resolution, enabling the computation of `timeToDone`, `timeInDoing`, and `timeInWaiting` (Table 4). Adoption occurred in Dec/2022–Jan/2023, allowing pre/post segmentation.

Table 4 – Variables used in the quantitative analysis.

| Variable | Description |
|---|---|
| `timeToDone` | Created → Resolved (end-to-end cycle time) |
| `timeInDoing` | Doing → Done (active work time) |
| `timeInWaiting` | Created → Doing (pre-work waiting) |
| `TaskType` | Bug, improvement, etc. |

### *3.5.2 Preprocessing and Outliers*

From 274 raw tasks, we removed duplicates and entries with inconsistent timestamps, retaining 181 tasks. We applied IQR-based outlier filtering per metric (not globally) to avoid cross-metric distortions. We tested IQR factors of 1.5, 2.0, and 3.0, selecting 3.0 as a balance between robustness and sample size. Sensitivity checks confirmed the stability of trends. All time variables were standardized before analysis.

## 3.6 Analysis Procedures

### *3.6.1 Quantitative Analysis (G1 / RQ1)*

Normality was assessed (Shapiro–Wilk). Given non-normality, we used Mann–Whitney U to compare pre vs. post distributions for each metric, reporting $p$-values ($\alpha$=0.05) and effect sizes. Visualizations (violin plots, histograms, time series) supported interpretation and inspection of dispersion and asymmetry. Where relevant, we performed sensitivity checks to IQR thresholds and to the inclusion of outliers.

### *3.6.2 Qualitative Analysis (G2 / RQ2)*

We conducted structured interviews (remote, recorded with consent) with core developers and adjacent roles (data engineer, infrastructure, PM). The instrument combined Likert items and open-ended questions, piloted with two participants and refined for clarity. We used a hybrid coding strategy (inductive + deductive, literature-informed), followed by thematic analysis and collaborative review to ensure consistency. Descriptive statistics summarize Likert responses; themes ground the narrative findings.

### *3.6.3 Triangulation*

We adopted an explanatory sequential design, where qualitative analysis followed the quantitative phase to interpret patterns and divergences. When metrics and perceptions diverged, we examined contextual mediators (e.g., deployment complexity, cross-team dependencies, organizational resistance) and revisited both strands to craft meta-inferences.

## 3.7 Chapter Summary

This chapter detailed how we operationalized our objectives via GQM, mapping goals to questions and to concrete metrics and data sources. We described the case context, the continuous practices under study, the dataset and preprocessing, and the analysis procedures for both quantitative and qualitative strands. The next chapter presents the results per RQ, followed by an integrated discussion and implications.

# 4 RESULTS AND DISCUSSION

This chapter presents findings per RQ and then integrates both strands. Section 4.1 reports the quantitative and qualitative results separately, organized according to RQ1 (productivity metrics) and RQ2 (practitioner perceptions). Section 4.2 integrates both perspectives, examining how measurable changes in workflow efficiency intersect with developer experiences and organizational dynamics. The chapter closes with an assessment of implications for theory and practice and a reflection on validity threats.

## 4.1 Results

This section presents the results of the study and is structured to directly address the two central research questions. To investigate Research Question 1 (RQ1), which focuses on the impact of continuous practices adoption on productivity, we conducted a task performance analysis using historical data from the project's task management system. This analysis explores how metrics such as task completion time, active work time, and waiting time evolved before and after the introduction of continuous practices.

In turn, Research Question 2 (RQ2) explores how these practices influenced development workflows and the perceived quality of the software. To capture this perspective, we carried out practitioner perception study based on structured interviews. The study aimed to understand how developers perceived and adapted to the adoption of continuous practices, including its effects on collaboration, deployment routines, and day-to-day challenges.

By combining both perspectives, the quantitative evidence from task data and the qualitative reflections of practitioners, this section offers the necessary data to understand how continuous practices have shaped software development in a mission-critical system within the energy sector.

### *4.1.1 To what extent does the adoption of continuous practices influence software development productivity?*

#### *4.1.1.1 Data Preprocessing and Statistical Approach*

To ensure reliable analyses, we cleaned the dataset by removing inconsistencies and duplicates. Outliers were identified and excluded using the interquartile range (IQR) method.

A normality test confirmed that the data did not follow a normal distribution, justifying the use of the Mann-Whitney U test as the primary statistical method. This non-parametric approach is well-suited for comparing independent groups when distributional assumptions are violated. To enhance the robustness of the analysis, we relied on a non-parametric approach and conducted visual analyses to confirm that observed differences were not due to chance alone.

All quantitative results presented here are based on data filtered using the IQR threshold of 3.0, as described in the Methodology section 3. This ensured trend analysis reflected the variability typical of real-world tasks.

#### *4.1.1.2 RQ1.1: Does continuous practices adoption reduce the active work time per task?*

To assess the impact of continuous practices adoption on the active effort invested in each task, we analyzed the `timeInDoing` metric, which captures the duration a task remained in active progress, excluding idle or waiting periods.

Following the adoption of continuous practices, we observed a small reduction in the geometric mean of active work time, decreasing from 6.48 to 6.24 days. Even though there was a slight shift in the average, it wasn't statistically or practically relevant. The Mann-Whitney U test yielded a non-significant result ($p = 0.43$). In practical terms, the difference suggests that while tooling and automation may have brought minor gains in task flow efficiency, they did not meaningfully reduce the actual development effort per task.

Interestingly, the most notable improvement was observed in the variance of the `timeInDoing` metric, which dropped considerably from 192.94 to 65.60 (days$^2$). This reduction in variability points to greater uniformity and predictability in how long developers remained engaged on tasks, even if the average effort remained roughly the same.

These patterns are illustrated in Figure 2, which shows the distribution of active work time before and after the adoption of continuous practices.

#### *4.1.1.3 RQ1.2: Does continuous practices adoption reduce the total task completion time?*

To evaluate the impact of continuous practices on the overall duration of development tasks, we analyzed the `timeToDone` metric. This measure captures the total time elapsed from the creation of a task to its completion, encompassing both active work and idle periods such as backlog waiting, review delays, or prioritization gaps.

After adopting continuous practices, the data reveals a substantial reduction in task

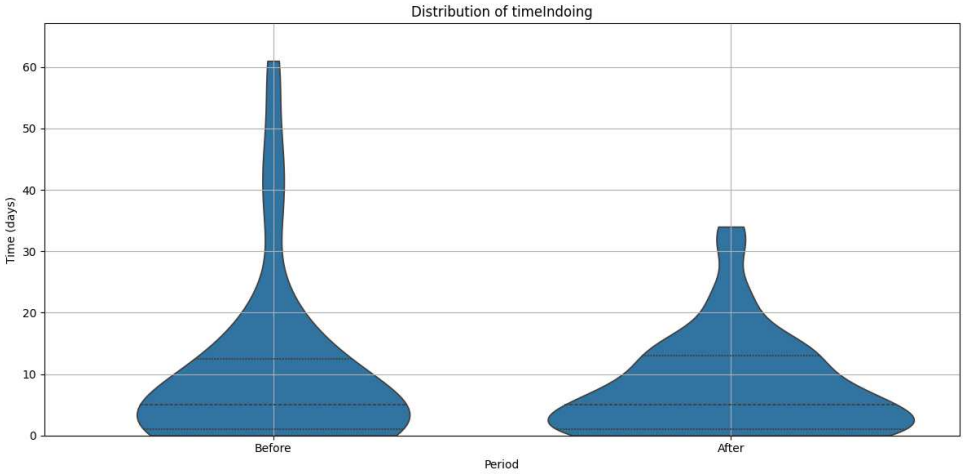


Figure 2 – Boxplot of active work time (`timeInDoing`) before and after continuous practices adoption.

completion times. The geometric mean decreased from 14.5 to 8.4 days, with a parallel reduction in the median, indicating faster and more consistent task completion across the dataset.

From a statistical standpoint, the Mann-Whitney U test did not reach the conventional 5% significance threshold, but yielded a $p$-value of 0.07, suggesting a potential trend. In addition to this central trend, there was a remarkable drop in the variance of completion times, from 2082.63 to 184.25 (days$^2$). This points to a more predictable delivery process, especially important in environments that demand cross-team coordination and careful release planning.

Overall, these findings suggest that while continuous practices may not always reduce the effort required per task, they can significantly improve system-level flow efficiency, helping teams deliver software more consistently and within shorter timeframes. These effects are illustrated in Figure 3, which shows the distribution of completion times before and after the adoption of continuous practices.

#### *4.1.1.4 RQ1.3: Does continuous practices adoption reduce waiting time within the development pipeline?*

One of the central expectations of continuous practices is the reduction of delays and inefficiencies associated with handoffs, especially those caused by long waiting periods between task creation and the start of active development. To examine this, we analyzed the `timeInWaiting` metric, which captures the duration that tasks remained idle before being picked up by developers.

The results show a clear reduction in waiting times after the adoption of continuous

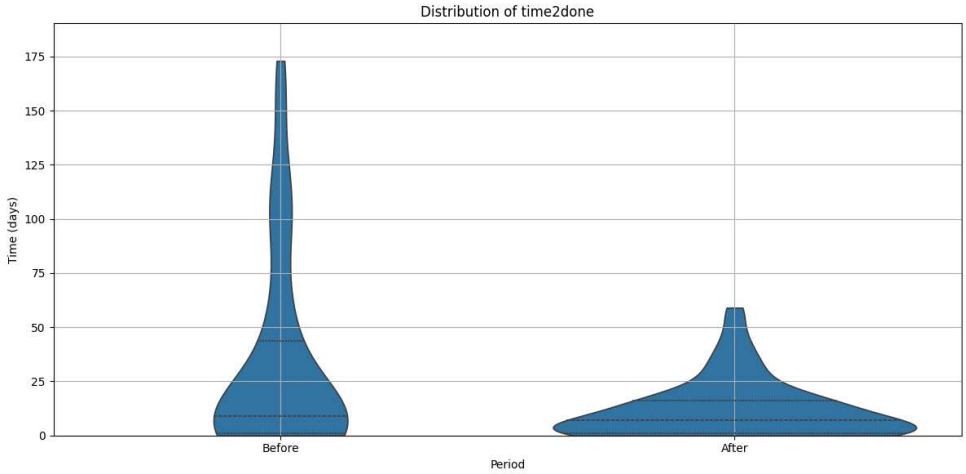


Figure 3 – Boxplot of total task completion time (`timeToDone`) before and after continuous practices adoption.

practices. The geometric mean decreased from 13.6 to 6.9 days, suggesting that tasks were entering the development phase more quickly. This improvement in intake speed indicates a potentially more responsive and organized pipeline, likely influenced by enhanced visibility and more consistent triage and prioritization.

Although the p-value (0.08) did not confirm significance, the results suggest a tendency toward improvement. Despite this, the reduction in both the geometric trend and the median provides partial support for the hypothesis of a streamlined workflow.

Interestingly, while central tendency improved, the variance in waiting time increased slightly, from 636.55 to 666.50 (days$^2$). This suggests that although many tasks entered the pipeline more efficiently, variability in task start times persisted, possibly due to differences in task complexity, priority, or external dependencies.

These observations may reflect the influence of continuous practices such as automated integration, clearer task visibility, and more structured intake mechanisms, which can reduce systematic delays but are still subject to fluctuations based on contextual factors. Figure 4 illustrates the distribution of waiting times before and after the adoption of continuous practices.

These findings highlight that continuous practices adoption contributed to a more responsive workflow, where tasks were addressed in a timelier manner. However, the lack of statistical significance suggests that while automation and visibility improve efficiency, task prioritization and resource allocation still rely on human decision-making and organizational practices.

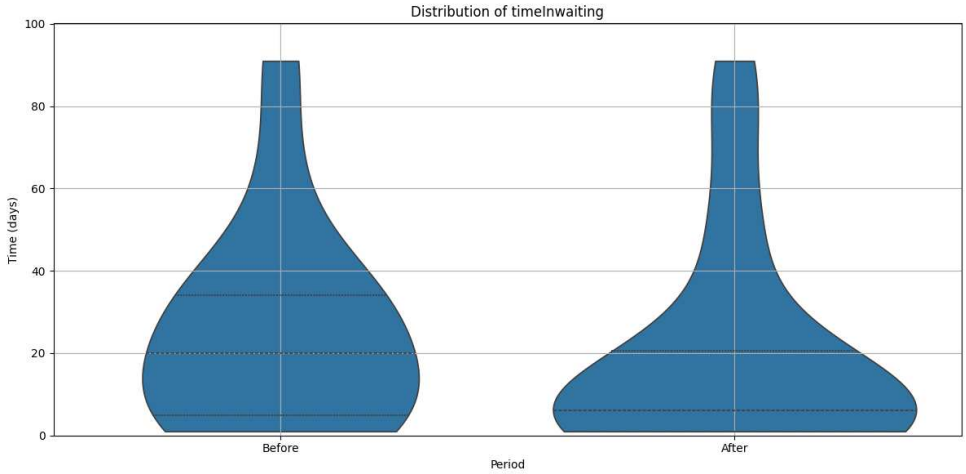


Figure 4 – Boxplot of waiting time (`timeInWaiting`) before and after continuous practices adoption.

#### *4.1.1.5 Temporal Trends*

To complement the statistical analyses presented earlier, we examined the evolution of task completion time (`timeToDone`) over the entire study period. Figure 5 presents a scatter plot of individual tasks along with a red line that represents the monthly average duration.

The plot reveals a clear downward trend in task completion time beginning shortly after the introduction of continuous practices. The monthly moving average suggests a gradual and sustained improvement in delivery flow efficiency, rather than a single-point improvement. This trend suggests a longer-term shift in team performance after adoption.

It is important to note that this visualization was generated using the full dataset, without the removal of outliers. This choice preserved the natural variability in the data and allowed us to observe long-term patterns.

In addition to delivery efficiency, we also examined the monthly appearance of bugs as an indicator of software quality throughout the same period. Figure 6 presents the number of bugs created per month (in red), a 3-month rolling average (in blue), and the average monthly values before and after January 2023.

Although there is natural fluctuation in the number of bugs, the plot shows that the average quantity remained stable or slightly reduced after the adoption of continuous practices. This observation suggests that changes in delivery workflow were not accompanied by an increase in defects, an important indicator that the gains were not achieved at the expense of quality.

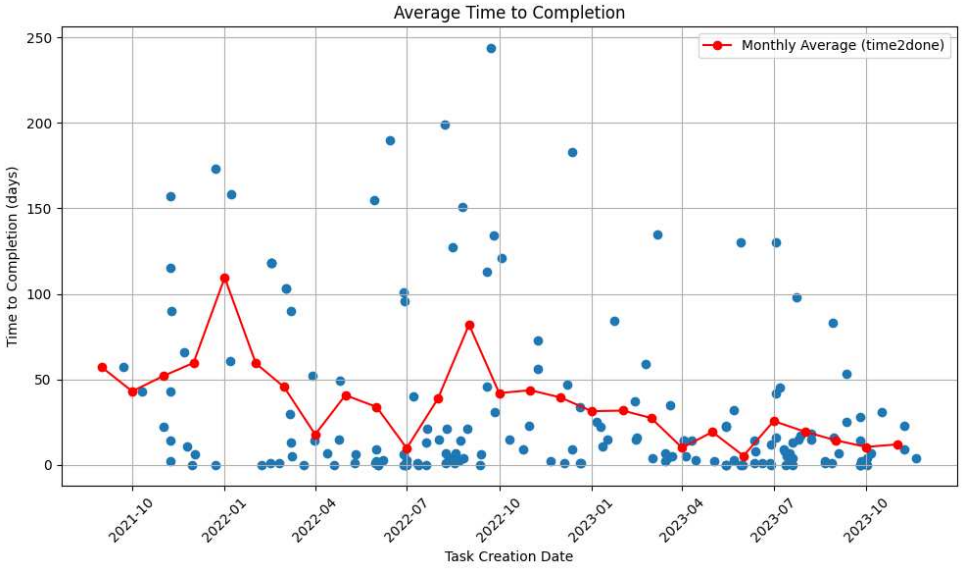


Figure 5 – Task completion time (`timeToDone`) over time with monthly average.

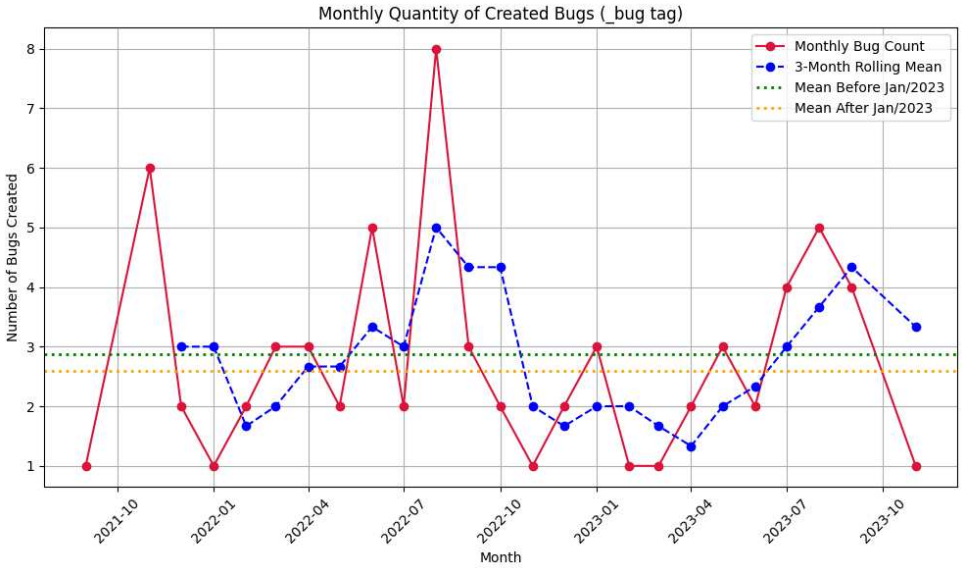


Figure 6 – Monthly number of created bugs with 3-month rolling mean and pre/post adoption averages

#### *4.1.1.6 Summary of Quantitative Findings*

Table 5 presents a consolidated view of the observed effects of continuous practices adoption on task performance metrics. The analysis reveal distinct patterns across active work time, total task duration, and waiting time within the development pipeline.

Among the metrics analyzed, total task completion time (`timeToDone`) showed the clearest improvement. The geometric mean dropped from 14.5 to 8.4 days, accompanied by a substantial reduction in variance (from 2082.63 to 184.25). Although the Mann-Whitney U test did not reach statistical significance ($p = 0.07$), the trend suggests that continuous practices

contributed to faster and more predictable task delivery.

Active work time (`timeInDoing`) also exhibited improvements, with a slight decrease in geometric mean from 6.48 to 6.24 days, and a marked decrease in variance (from 192.94 to 65.60). However, no statistical significance was observed ($p = 0.43$).

Waiting time (`timeInWaiting`) showed a notable decline in central tendency, with the geometric mean decreasing from 13.6 to 6.9 days. Yet, despite this improvement, statistical significance was not confirmed (Mann-Whitney $p = 0.08$), and variance slightly increased from 636.55 to 666.50 days². This indicates that while many tasks entered development more quickly, variability in task start times persisted, possibly due to task complexity or organizational factors.

These quantitative results provide a foundation for the qualitative analysis that follows, which helps contextualize these trends by exploring the organizational and cultural dynamics behind them.

Table 5 – Summary of Quantitative Results

| **Metric** | **Geometric Mean** | **Variance (days$^2$)** | **Mann-Whitney** |
|---|---|---|---|
| Active Work Time | $6.48 \rightarrow 6.24$ | $192.94 \rightarrow 65.60$ | $p \approx 0.43$ |
| Total Task Time | $14.5 \rightarrow 8.4$ | $2082.63 \rightarrow 184.25$ | $p \approx 0.07$ |
| Waiting Time | $13.6 \rightarrow 6.9$ | $636.55 \rightarrow 666.50$ | $p \approx 0.08$ |

Overall, the findings summarized in Table 5 suggest that while the results did not reach statistical significance, geometric trends and variance reduction point to enhanced predictability and process stability. These effects indicate that continuous practices contributed to greater consistency in delivery, even if improvements in raw speed were limited.

### *4.1.2 What are practitioners' perceptions of the benefits, challenges, and workflow changes associated with the adoption of continuous practices?*

To explore how continuous practices were experienced from the perspective of the development team, we conducted a structured study combining quantitative and qualitative elements. The instrument included Likert-scale questions to assess perceptions on delivery speed, software quality, release frequency, and workflow efficiency, along with open-ended prompts designed to elicit detailed examples, narratives, and contextual reflections from participants.

The complete questionnaire is available in Appendix A. Each research sub-question presented in this section draws on specific items from the instrument (e.g., Q7, Q11, Q14), and responses are interpreted through both statistical summaries and thematic analysis. This

mixed-method approach went beyond surface metrics by capturing the stories and experiences that contextualize the quantitative data. It also allowed for cross-referencing with findings from the Task Performance Analysis, offering a more comprehensive understanding of how continuous practices unfold in real-world projects.

By organizing the results around six sub-questions (RQ2.1 to RQ2.6), we aim to present a structured view of how practitioners perceived the effects, challenges, and individual relevance of continuous practices within the studied context. Each subsection begins by referencing the corresponding interview questions and then presents both the quantitative and qualitative insights that emerged.

#### *4.1.2.1 Participant Profile and Interview Procedures*

We interviewed 11 professionals directly involved in the studied project. The group was composed primarily of developers, but also included data engineers, a consultant, a product owner, and a manager, each contributing with different levels of experience and perspectives.

The initial questions of the study (Group 1 – Q1 to Q6) collected demographic and background information, covering participants' current roles, years of experience in software development, educational backgrounds, and familiarity with continuous practices. Participants came from diverse professional backgrounds. Some participants were in the early stages of their careers, while others had over 15 years of experience. Academic training spanned fields such as Computer Science, Mechanical and Electrical Engineering, Architecture, and Geography, with education levels ranging from undergraduate to postgraduate (some still in progress).

Familiarity with continuous practices (Q4) also varied considerably. While a few participants reported limited exposure, others described extensive experience acquired through both formal training and hands-on practice. Some respondents mentioned graduate-level courses or certifications, while others referred to online learning platforms and experiential learning within the team's workflow.

#### *4.1.2.2 RQ2.1: How do developers perceive the influence of continuous practices on delivery speed?*

This sub-question was addressed through Group 2 (Q7 to Q10) of the questionnaire, which asked participants to rate their level of agreement with the statement: "*Continuous practices have improved the speed of software delivery in the project.*".

The quantitative results suggest a generally positive, yet somewhat mixed perception. As shown in Figure 7, 54% of respondents strongly agreed with the statement, while the remaining responses were distributed across lower levels of agreement. This indicates that although many developers experienced speed improvements, others perceived more limited or conditional effects.

The open-ended responses shed light on this nuance. Developers frequently associated speed gains with automation, particularly in testing and deployment. One participant noted, "*Continuous practices help automate repetitive tasks that consume a lot of time*," emphasizing the technical benefits of reducing manual steps in the workflow.

However, several participants pointed out that delivery speed is influenced not only by automation, but also by organizational and procedural factors. As one developer explained, "*Code review often delayed delivery, as there wasn't always enough time to complete it*," while another remarked, "*It required more time for code review and alignment*." Others expressed that the impact on speed was less visible than on quality: "*Overall, the gains are more concentrated in terms of quality. When it comes to speed, the impacts may not be as clear*."

These reflections suggest that while continuous practices improve the technical conditions for faster delivery, the actual experience of speed remains shaped by coordination overhead, quality assurance steps, and team dynamics. The analysis indicates a recurring trade-off between accelerating delivery and maintaining alignment and quality in complex development environments.

#### *4.1.2.3 RQ2.2: How do developers assess the impact of continuous practices on software quality?*

This sub-question was addressed through Group 3 (Q11 to Q13) of the questionnaire, which explored developer perceptions regarding the impact of continuous practices on software quality. One of the key items in this group asked participants to rate their agreement with the statement: "*Continuous practices have improved the quality of the software produced in the project.*".

The aggregated perceptions across quality, release frequency, and delivery speed are summarized in Figure 7, which provides a comparative view of the Likert-scale responses.

Participants frequently highlighted that practices such as peer code reviews and automated testing were effective in identifying defects early, promoting best practices, and

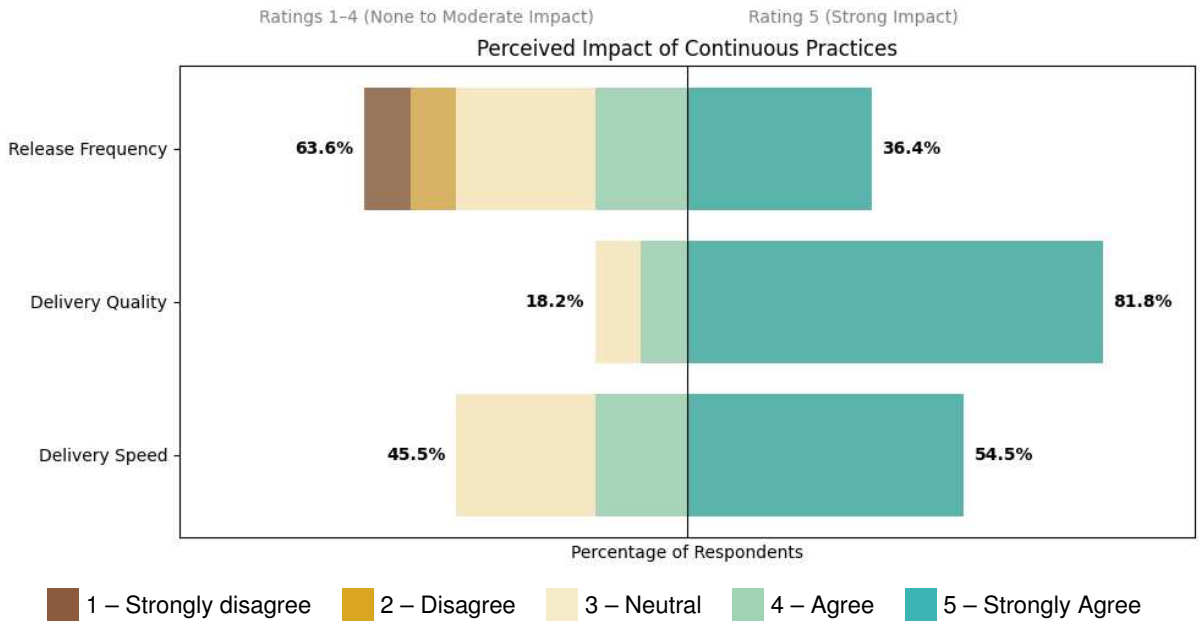


Figure 7 – Perceived impact of continuous practices on quality, release frequency, and delivery speed.

increasing confidence in production releases. One developer explained, "*It improves code quality by offering a new perspective on the implementation*," referring to how code reviews encourage collaboration and different points of view.

Many also pointed out that the improvements in quality go beyond technical aspects. As one of them put it, "*The purpose is to improve quality, not just deliver something in any way*," emphasizing a cultural shift toward shared responsibility and higher standards in the development process.

The data from the Likert-scale responses reinforces this perception. A significant 81.8% of respondents strongly agreed that continuous practices have a positive impact on the final product's quality. The remaining 18.2% rated it as a 4 on the scale, indicating that while some saw the impact as slightly less pronounced, all participants agreed that these practices contribute positively to quality. These perceptions are aligned with the quantitative trend in bug creation over time 6 , which shows that quality levels remained stable or improved despite gains in efficiency.

At the same time, developers acknowledged that focusing on quality introduces certain trade-offs. Activities like code review and validation can make the process more thorough, but sometimes slow down delivery. This tension between ensuring quality and maintaining speed appeared often in participants' reflections.

Taken together, these results reflect a strong consensus on the value of continuous

practices in improving software quality. Developers consistently saw them as essential for fostering a culture that prioritizes quality while also strengthening technical safeguards throughout the development process.

#### *4.1.2.4 RQ2.3: How do developers perceive the relationship between continuous practices and release frequency?*

This sub-question was addressed through Group 4 (Q14) of the questionnaire, which asked participants to rate their agreement with the statement: "*Continuous practices have increased the frequency of software releases in the project.*'.

Perceptions regarding the relationship between continuous practices and release frequency were varied. Several participants noted that automation facilitated more frequent deliveries by streamlining deployment routines and reducing manual overhead. As one developer explained, "*With well-established practices, the workflow becomes more automated, increasing the frequency of releases*," reinforcing the idea that technical enablers can support a more fluid delivery pipeline.

Nonetheless, many participants emphasized that the actual release frequency is shaped by organizational dynamics, not solely by technical capability. Business validation cycles, regulatory requirements, and coordination with non-technical teams were seen as significant factors influencing when releases occur. As one participant put it, "*The reason for this is that the strategy is diverse. It depends much more on the business area — how often they validate and approve things — than on the pipeline itself.*"

The Likert-scale responses reflect this complexity. While 36.4% of respondents strongly agreed that continuous practices positively impacted release frequency, the remaining answers were distributed across lower agreement levels. This dispersion suggests that developers' perceptions vary depending on their exposure to the decision-making processes of releases and the organizational structures in which they operate. Although most agreed that automation simplifies the path to deployment, they also recognized that the release cadence ultimately depends on broader strategic and organizational considerations.

In summary, continuous practices were seen as enablers of frequent releases, but not as determiners. The frequency of actual deployments remained contingent on external decision layers, highlighting once again that technical improvements alone do not guarantee faster or more continuous delivery in complex environments.

#### *4.1.2.5 RQ2.4: How do continuous practices influence developers' daily workflows?*

This sub-question was addressed through Group 5 (Q15 to Q18) of the questionnaire, which invited participants to reflect on how continuous practices have influenced their day-to-day work routines. Responses included both a Likert-scale evaluation and open-ended comments on perceived changes in workflow efficiency, structure, and team dynamics.

Most developers reported that the adoption of continuous practices brought improvements to their daily workflows. Automation helped reduce manual effort and improved visibility into task progress and system status. CI/CD pipelines, in particular, were seen as valuable tools for increasing transparency and promoting alignment across teams. As one developer noted, "*They help identify problems and speed up resolution*," emphasizing how structured workflows support clarity and responsiveness in daily tasks.

Many participants also pointed out that continuous practices introduced a higher level of procedural rigor. This added structure was seen as a mechanism to improve quality and ensure consistency across the delivery pipeline, but it also required greater coordination between teams. Developers acknowledged that standardizing key stages of the workflow, such as testing, code review, and deployment, helped create predictable processes and reduce errors.

At the same time, some developers expressed concerns that too much structure could limit flexibility. As one participant remarked, "*I don't believe it's positive to adhere too rigidly to pre-defined delivery flows*," highlighting that strict processes may hinder responsiveness in fast-changing environments.

In summary, developers recognized that continuous practices improved workflow control, visibility, and stability. However, they also emphasized that the benefits depend on how well teams adapt to the additional structure and balance standardization with the need for flexibility.

#### *4.1.2.6 RQ2.5: What challenges do developers face when adopting continuous practices?*

This sub-question was addressed through Group 6 (Q19 to Q24) of the questionnaire, which asked participants to evaluate the level of challenge associated with different aspects of continuous practices adoption.

While developers generally acknowledged the benefits introduced by continuous practices, their adoption was accompanied by a range of challenges that spanned cultural,

technical, and organizational dimensions.

The most frequently reported barrier was the cultural shift required to embrace new ways of working. As shown in Figure 8, about 45% of participants perceived cultural change as a large or great challenge. This perception was echoed in open-ended responses, where developers described the difficulty in changing ingrained habits, adapting team dynamics, and fostering cross-role collaboration. Several participants emphasized the effort required to build mutual trust and redefine responsibilities in light of increased automation and visibility.

Another commonly reported difficulty was related to the learning curve of CI/CD adoption. While a portion of the team considered the transition smooth, 30% of respondents rated it as a large or great challenge. This difference likely reflects varied levels of previous experience. Developers already familiar with CI/CD concepts adapted more quickly, while others needed time, support, and onboarding guidance. Some suggested assigning a dedicated CI/CD advocate to help spread good practices across the team.

In addition to cultural and learning barriers, participants pointed to more structural challenges that significantly influenced the adoption process. Legacy systems, in particular, brought compatibility issues, especially in areas such as automated testing, environment provisioning, and deployment orchestration. These limitations often forced teams to rely on manual steps or mixed approaches, which ended up weakening the benefits of continuous integration and delivery. In several cases, developers mentioned that the existing architecture made it hard to automate key parts of the workflow, adding friction to tasks that should have been smooth.

Organizational policies and governance processes also emerged as critical constraints. Security protocols, audit requirements, and compliance validations added procedural overhead to the delivery process, often delaying releases even when the pipeline was technically ready. Pre-existing workflows and hierarchical approval chains further limited the autonomy of teams, reducing their flexibility to fully benefit from the agility introduced by continuous practices.

Taken together, these findings reinforce the notion that adopting continuous practices is not just about tools or automation. It also involves helping people adapt to cultural changes, supporting the team's development of technical skills, and navigating the practical constraints of legacy systems and organizational policies.Effective adoption, therefore, requires a socio-technical perspective that acknowledges both the human and systemic dimensions of transformation.

The extent to which each type of challenge was perceived is presented in Figure 8,

which consolidates participant responses across cultural, technical, and organizational dimensions.

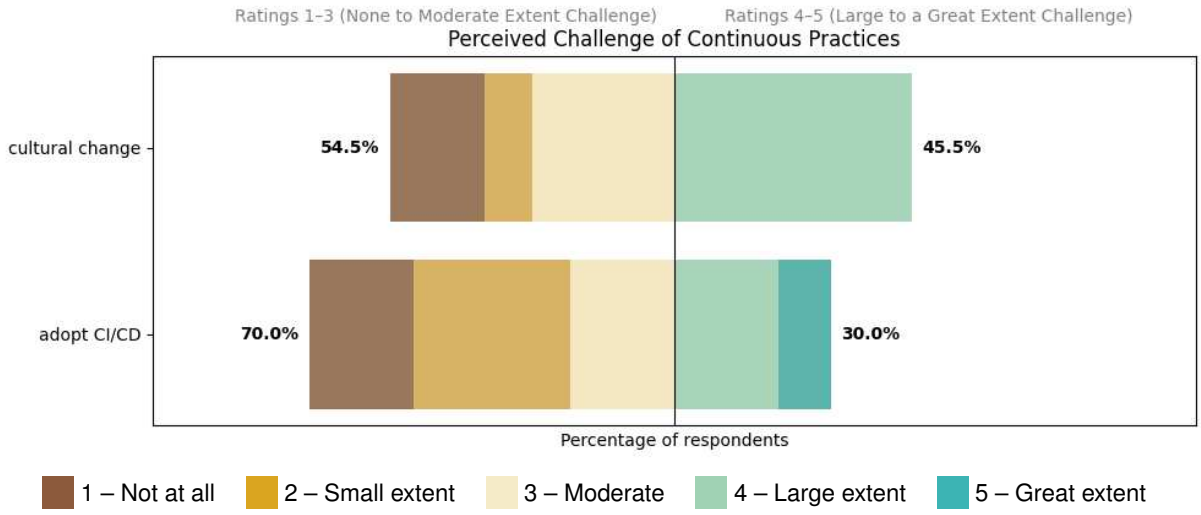


Figure 8 – Perceived level of challenge in adopting continuous practices.

#### *4.1.2.7 RQ2.6: How do developers evaluate the individual relevance of the continuous practices adopted in the case study?*

This sub-question was addressed through Group 7 (Q25) of the questionnaire, which asked participants to rate the relevance of each continuous practice adopted in the project. The responses were collected using a five-point Likert scale and aimed to assess how each practice was perceived when considered individually.

In addition to understanding the perceived effects and challenges of continuous practices, the study also sought to evaluate how individual practices are evaluated by developers in terms of relevance and utility.

Figure 9 presents the perceived relevance of five key continuous practices: CI/CD pipelines, monitoring and observability, cultural change initiatives, security practices, and code review with pair programming. The responses reveal clear differences in how these practices are valued within the development context.

CI/CD pipelines emerged as the most widely recognized enabler, with 81.8% of respondents rating them as highly relevant. This result reflects a strong consensus around the centrality of automation in achieving process efficiency and delivery consistency. Developers frequently highlighted CI/CD as a foundational layer that supports other improvements by

reducing manual effort and providing structured, repeatable workflows.

Monitoring tools and observability dashboards followed closely, with 72.7% of participants assigning high relevance. These practices were praised for improving system visibility and supporting proactive issue identification, particularly in environments where stability and traceability are critical. Their prominence suggests that beyond automation, developers value tools that enhance transparency and control over runtime behavior.

Cultural change initiatives and security practices were also seen as relevant, though with slightly less consensus (63.6% and 54.5%, respectively). The lower levels of strong agreement may reflect the challenges associated with implementing cultural shifts and navigating security requirements in legacy systems. Nonetheless, a majority of respondents acknowledged their importance, especially in ensuring sustainable adoption and regulatory compliance.

Interestingly, code review and pair programming received the lowest level of strong agreement, with only 45.5% of respondents rating them as highly relevant. This variation suggests that their perceived importance may depend more heavily on contextual factors, such as team size, collaboration culture, or project criticality. While still considered valuable by many, these practices may not be uniformly applied or prioritized across all settings.

These findings offer a nuanced view of how developers evaluate continuous practices individually. They reinforce the central role of CI/CD and observability as the most universally appreciated components, while highlighting that the relevance of other practices may be more context-sensitive.

Finally, Figure 9 summarizes how developers rated the individual relevance of each continuous practice, highlighting the practices considered most critical in the studied context.

#### *4.1.2.8 Summary of Qualitative Findings*

The qualitative analysis revealed that developers generally perceive continuous practices as beneficial to software development workflows, particularly in enhancing software quality and reducing manual effort through automation. Practices such as automated builds and structured deployment pipelines were associated with greater consistency, early defect detection, and increased confidence in releases. This results align with the findings in (DHAKAD, 2023; VASILESCU *et al.*, 2015; FEIJó *et al.*, 2024).

Despite the technical enablers introduced by automation, the impact of continuous practices on delivery speed has received mixed perceptions. Developers acknowledged reductions

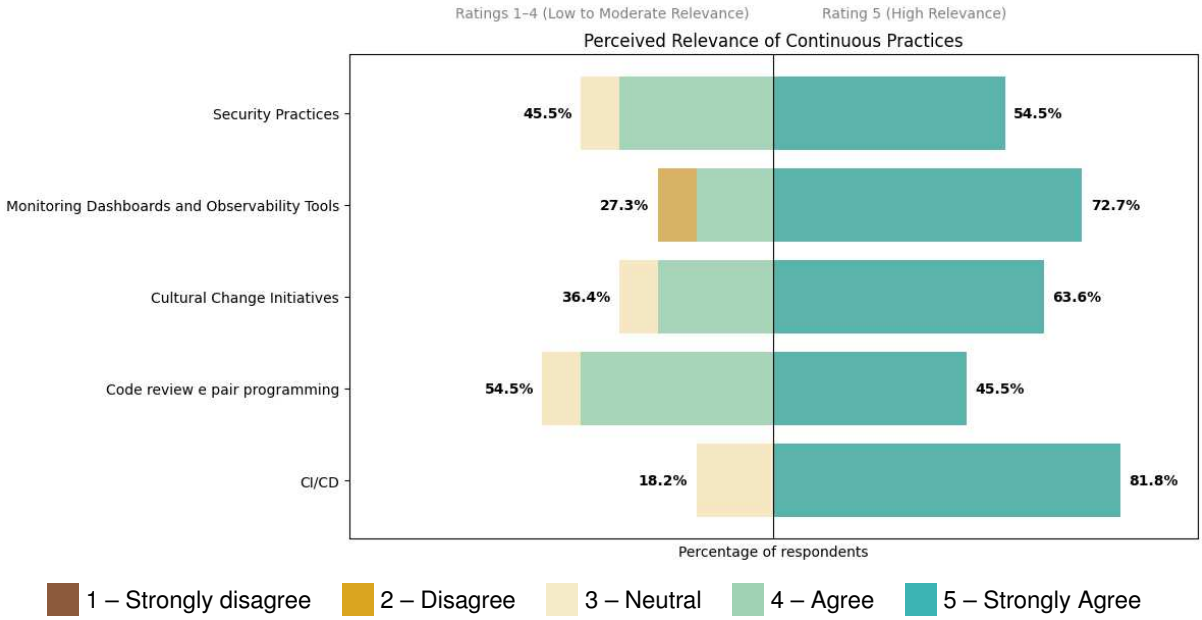


Figure 9 – Perceived relevance of different continuous practices according to developer responses.

in manual effort and an increased ability to release more frequently. However, they also pointed out that procedural demands such as code reviews, compliance checks, and alignment with non-technical stakeholders often introduced delays. These constraints highlight a recurring trade-off between delivery acceleration and the need to maintain governance, quality assurance, and coordination across teams (LEITE *et al.*, 2019; KHAN *et al.*, 2022; KREY, 2022). In regulated and legacy-constrained environments, these competing aspects influence how teams realize the benefits of continuous practices.

Beyond technical and cultural factors, participants consistently emphasized the influence of organizational and legacy constraints. Limitations imposed by legacy systems, rigid approval workflows, and hierarchical decision-making structures were frequently cited as barriers to fully realizing the potential of continuous practices.

To complement these narratives, Figure 7 provides an overview of how developers perceived the impact of continuous practices across key delivery dimensions. While improvements in software quality were consistently recognized, perceptions related to speed and release frequency were more diverse. This contrast reinforces the notion that quality gains are more immediately perceived, whereas changes in delivery pace and cadence tend to be mediated by organizational structures and decision-making processes.

Overall, these results highlight that the effectiveness of practice adoption extends beyond the availability of tools. It is fundamentally shaped by team collaboration, workflow

maturity, and the organization's readiness to embrace new delivery models and cultural change. These findings also set the stage for a deeper discussion on how continuous practices interact with technical, cultural, and organizational factors to shape delivery outcomes.

## 4.2 Discussion and Implications

### *4.2.1 Overall Discussion*

This section discusses the study findings in light of existing literature on continuous practices adoption. It focuses on how these practices are implemented in regulated, legacy-constrained environments, identifying areas of convergence and divergence with prior studies, and discussing their implications for theory and practice.

The findings indicate that adopting continuous practices in the context studied led to moderate improvements in delivery flow and greater process stability. These improvements, however, were not uniform, as they were shaped by factors such as organizational structures, legacy constraints, and the cultural readiness of the teams involved.

Quantitative analyses showed a reduction in variance for key metrics, pointing to a more consistent development flow. This result reinforces prior work that highlights how structured automation, especially through continuous integration and delivery pipelines, can help reduce variability in software delivery (SU; STORER, 2023; WINKLER; WESTNER, 2023; AMARO *et al.*, 2023).

However, while geometric trends and temporal analyses suggest potential improvements in overall task flow, the observed reductions in median task durations were not statistically significant according to the Mann-Whitney tests. This limitation aligns with prior findings that highlight the high variability and skewed nature of software development data (PORT *et al.*, 2024; ZOHAIB, 2023). These results reinforce the argument that performance should be evaluated not only in terms of speed gains but through broader indicators such as predictability, process stability, and organizational resilience.

The qualitative findings provide additional depth to these insights. Developers acknowledged that continuous practices reduced manual effort and increased visibility, yet these improvements did not always translate into faster delivery. Bottlenecks such as reviewer availability, compliance checks, and coordination across roles continued to impact the delivery pipeline. As one developer explained, even with automated pipelines in place, releases were frequently

delayed due to pending reviews or approvals. This perspective reinforces the notion that the benefits of DevOps are shaped not only by technical infrastructure but also by organizational dynamics, in line with prior research (KREY, 2022; LEITE *et al.*, 2019; KHAN *et al.*, 2022).

Overall, the findings suggest that while continuous practices can foster delivery efficiency and process control, their impact is largely conditioned by the surrounding organizational context. The following subsections examine how these results align with or differ from prior studies and explore their implications for theory and practice.

### *4.2.2 Integration of Findings*

By comparing and contrasting RQ1 and RQ2, we observe that the most consistent outcome of continuous practices was not raw acceleration of delivery, but rather greater reliability and smoother workflows. Quantitative data showed reduced variance and temporal improvements, while developers emphasized quality, visibility, and workflow control. Together, these complementary findings highlight that in regulated and legacy-constrained environments, continuous practices primarily act as stabilizers and enablers of collaboration, rather than as linear accelerators of throughput.

Combining numbers with developer perspectives helped us build a more complete picture of how continuous practices shaped the workflow. Triangulating both perspectives revealed key points of convergence and showed how measurable changes in metrics intersect with developer perceptions and organizational dynamics. In addition, this integration allows for a contextualized comparison with prior studies, positioning our findings within the broader landscape of DevOps and continuous practices research. This approach reinforces the study's objective of uncovering not just whether continuous practices work, but how and under what conditions they lead to meaningful improvements, a principle aligned with good practices in empirical software engineering research (STOREY *et al.*, 2025; RUNESON; HÖST, 2009).

#### *4.2.2.1 Productivity and Perceived Efficiency*

A clear alignment emerged between the quantitative and qualitative results regarding productivity improvements (RQ1 and RQ2), although this alignment was not absolute. The quantitative data suggested a downward trend in total task duration and waiting time, particularly when analyzing geometric means and temporal patterns, but did not reach statistical significance in hypothesis testing. Meanwhile, developer narratives highlighted perceived gains in efficiency,

often attributed to automation and streamlined workflows.

These results are consistent with prior studies that associate continuous practices with efficiency gains by reducing manual work and promoting coordination across tasks (FORSGREN *et al.*, 2018; SU; STORER, 2023; WINKLER; WESTNER, 2023). Our observations of reduced task duration and waiting time echo findings from (ROMERO *et al.*, 2022) and (DUARTE, 2021), which indicate that automation and standardization help developers focus on value-generating tasks and mitigate bottlenecks introduced by handoffs.

For example, reductions in task duration appeared consistent with developers' perceptions of shorter delivery cycles. Similarly, lower waiting times were associated with the view that automation helped mitigate delays. However, developers also emphasized that tooling alone was not sufficient to ensure faster delivery. In fact, while some tasks were accelerated, others became more complex due to the addition of new steps, such as automated checks, required reviews, and integration gates, which in some cases introduced delays or procedural overheads.

These insights resonate with the tensions observed in the literature between technical capability and realized delivery gains. As noted by (SHAHIN *et al.*, 2017) and (SANTOS *et al.*, 2024), performance improvements in DevOps are often difficult to validate statistically due to noisy data and skewed distributions. Our findings follow this pattern, showing a disconnect between perceived gains and formal significance in task duration improvements, a divergence that calls for more holistic evaluation lenses, such as variance reduction and qualitative perceptions.

Additionally, studies such as (PORT *et al.*, 2024) and (AMARO *et al.*, 2023) emphasize that improvements in productivity often manifest not as uniform speed gains, but as enhanced consistency, transparency, and control in the delivery process. This perspective helps contextualize our results: while automation enabled technical acceleration, the benefits were shaped, and sometimes limited, by organizational complexity, reflecting a broader shift from performance-centric to capability-centric interpretations of DevOps success.

These perceptions mirror the quantitative trends: a general sense of improvement, but not a uniform or definitive acceleration of delivery. The insights reinforce that while continuous practices provide technical enablers for productivity, the realized benefits are shaped by organizational structures, collaboration dynamics, and decision-making processes and may involve trade-offs between automation gains and process complexity.

#### *4.2.2.2 Workflow Predictability and Quality Reinforcement*

Another key area of convergence was workflow stability (RQ1 and RQ2). The task performance analysis showed a marked reduction in the variance of task completion times, indicating greater consistency in delivery and supporting the notion of process stabilization. Developers confirmed these patterns by describing increased visibility into the system, improved monitoring, and the establishment of structured release routines.

For instance, the observed reduction in variance aligned with developers' perceptions of a more predictable and stable workflow. Many participants reported that having clearer visibility into pipeline stages and system status helped detect issues earlier, contributing to smoother delivery processes. At the same time, while some developers recognized improved alignment across teams, others highlighted new procedural overheads and the need for closer coordination between roles.

These results are consistent with prior studies such as (SU; STORER, 2023; WINKLER; WESTNER, 2023), which associate automation and structured delivery pipelines with greater predictability and stability in software processes. Likewise, the observed decrease in work time variance resonates with findings from (ROMERO *et al.*, 2022; DUARTE, 2021), which demonstrate that standardization supports more consistent and controlled workflows, even in heterogeneous or evolving environments.

In parallel, the qualitative responses revealed that continuous practices were also strongly associated with perceived improvements in software quality (RQ2.2). Developers emphasized how peer code reviews and automated testing increased confidence in releases, enabled earlier defect detection, and encouraged the adoption of better implementation practices.

These quality-related perceptions are supported by previous research linking continuous integration practices to improvements in code quality and reliability (DHAKAD, 2023; VASILESCU *et al.*, 2015). Notably, Feijó et al. (FEIJó *et al.*, 2024) found that CI adoption contributed to reducing confusion atoms in open-source projects, reinforcing the role of CI/CD not only in accelerating delivery, but also in fostering more maintainable and robust software.

This perspective reinforces the view that assessing the outcomes of DevOps transformations requires multiple complementary lenses, statistical variance, temporal trends, and practitioner perception, as proposed by Shahin et al. (SHAHIN *et al.*, 2017), particularly in contexts where data distributions are skewed or noisy.

Finally, these insights align with Forsgren et al. (FORSGREN *et al.*, 2018), who

argue that high-performing teams should prioritize system reliability and software quality before achieving consistent gains in throughput or speed.

Overall, these findings suggest that continuous practices contribute to workflow predictability and perceived quality by improving system observability, enforcing process consistency, and encouraging collaborative refinement. However, the effectiveness of these outcomes depends on the ability of teams to balance automation and procedural structure with adaptability and shared ownership of the delivery process.

#### *4.2.2.3 Constraints and Organizational Readiness*

Beyond individual metrics, the findings highlight the central role of organizational and legacy factors in shaping the outcomes of continuous practices. Developers frequently pointed to limitations imposed by legacy systems, inflexible processes, and hierarchical decision chains that slowed or restricted their adoption and evolution.

Technical challenges, such as difficulties in integrating legacy components, were cited as a major barrier to fully implementing CI/CD pipelines. Participants also noted that organizational structures, particularly in regulated environments, introduced significant overhead through rigid approval workflows and strict compliance requirements. Additionally, cultural resistance and partial team readiness were seen as sources of friction, often limiting the potential benefits of practices.

These results echo the findings of (KHAN *et al.*, 2022) and (AZAD, 2022), which emphasize that DevOps adoption in traditional or highly regulated contexts is frequently constrained by legacy architectures and governance mechanisms that inhibit the flexibility needed for rapid delivery cycles. Our findings also align with (FORSGREN *et al.*, 2018) and (ALMEIDA *et al.*, 2022), who observed that while tooling can provide the foundation for continuous practices, the realization of benefits is often gated by organizational culture and decision-making agility.

Moreover, Shahin et al. (SHAHIN *et al.*, 2017) stress that evaluating DevOps performance must account for socio-technical complexity, where process friction, uneven maturity, and conflicting priorities can mask or moderate expected outcomes. In this study, several participants reported that even when pipelines were technically functional, bureaucratic validation steps and lack of autonomy delayed releases and limited responsiveness, highlighting the tension between technical capability and organizational readiness.

This aligns with the concept of “people maturity” proposed by Akbar et al. (AKBAR

*et al.*, 2024), which emphasizes the importance of cultural alignment, individual competence, and shared responsibility as key enablers of secure and sustainable DevOps adoption. Similarly, (PORT *et al.*, 2024) argue that maturity models for DevOps should consider legacy constraints and compliance overhead as first-order variables in adoption trajectories.

These insights underscore that the success of continuous practices is not simply a matter of tool adoption. It depends on how well technical enablers are supported by organizational alignment, process maturity, and cultural evolution.

#### *4.2.2.4 Methodological Reflection and Synthesis*

Taken together, the findings of this study highlight that the adoption of continuous practices in complex, regulated environments is not a linear path toward efficiency, but rather a multifaceted transformation influenced by both technical and organizational conditions. While automation and pipeline integration can yield measurable gains in predictability and perceived efficiency, realizing these benefits depends on how mature the environment is, from system architecture to compliance rules, and how ready the team is to change.

The analysis suggest that organizations seeking to adopt continuous practices should calibrate expectations: gains in speed may be incremental or uneven, particularly when existing processes and cultural dynamics are not fully aligned. Instead of focusing solely on throughput acceleration, efforts should prioritize building organizational resilience, reducing variance, and enhancing transparency across the delivery process.

From a theoretical standpoint, this study reinforces emerging views that DevOps success cannot be captured only through traditional performance metrics like lead time or deployment frequency. It contributes to a more systemic understanding of continuous practices, where delivery stability, socio-technical fit, and the ability to adapt under constraints become central dimensions. By triangulating quantitative data with practitioner perceptions, this case adds empirical grounding to frameworks that advocate for context-sensitive, capability-driven models of DevOps adoption (AMARO *et al.*, 2023; PORT *et al.*, 2024).

The synthesis of findings reveals that continuous practices led to important advances in automation, visibility, and workflow standardization. The most consistent impact, however, was observed in delivery predictability and perceived quality. Reductions in variance across key metrics pointed to more stable processes, while participants reported clearer expectations, reduced rework, and improved coordination across teams.

Still, delivery speed did not increase uniformly. Organizational bottlenecks, legacy system constraints, and uneven team maturity directly shaped the outcomes. Rather than accelerating all workflows, they created a more supportive environment for improvement. The realized benefits largely depended on the alignment between organizational structure, culture, and technical maturity. These insights suggest that the success of continuous practices and DevOps is less about adopting tools and more about how they are embedded within the sociotechnical fabric of the organization.

### *4.2.3 Theoretical and Practical Implications*

The findings of this study offer relevant contributions both to the academic understanding of continuous practices adoption within the DevOps context and to the practical strategies employed by organizations implementing continuous practices in complex environments.

#### *4.2.3.1 Theoretical Implications*

From a theoretical perspective, this study reinforces the need for a multidimensional view of success when adopting continuous practices, one that goes beyond throughput to include predictability, coordination, and adaptability. While many prior studies have emphasized delivery speed as a key metric of performance (FORSGREN *et al.*, 2018; WINKLER; WESTNER, 2023), our findings suggest that in legacy-rich or compliance-driven contexts, stability is often more realistic and valuable short-term outcomes.

Although continuous practices are frequently discussed within the broader scope of DevOps, this study adopts a more focused perspective by analyzing specific practices, such as continuous integration, delivery, and monitoring, without requiring a full DevOps transformation. This view aligns with a growing body of research advocating for refined success models that consider organizational enablers, sociotechnical integration, and domain-specific constraints (AMARO *et al.*, 2023; PORT *et al.*, 2024). By triangulating quantitative data with practitioner perceptions, our study adds empirical depth to these frameworks and highlights that results depend on the context, and technical metrics alone don't tell the whole story.

Furthermore, the observed gaps between perceived improvements and the lack of statistical significance in median task durations challenge linear assumptions of cause and effect often present in DevOps literature. These findings underscore the importance of interpreting

the adoption of continuous practices as a process of organizational learning and capability development, rather than merely a technical upgrade, in line with perspectives from (SU; STORER, 2023; ZOHAIB, 2023; KHAN *et al.*, 2022).

#### *4.2.3.2 Practical Implications*

For practitioners, the findings emphasize that the adoption of continuous practices demands more than just the implementation of technical tools, it also requires structural alignment, procedural adaptation, and cultural readiness. In safety-critical or regulated environments, expectations of rapid transformation should be moderated, with a stronger focus on the gradual and iterative integration of these practices.

A key recommendation is to prioritize predictability over speed in the initial phases of adoption. Indicators such as variance reduction and flow stability can serve as more meaningful metrics of progress than raw delivery throughput. Additionally, automation efforts must be aligned with governance requirements, ensuring that CI/CD pipelines are compatible with compliance and audit processes.

This need for deliberate organizational-wide adaptation aligns with prior studies that frame DevOps adoption as a progressive learning process rather than a purely technical upgrade. For instance, Rafi et al. (RAFI *et al.*, 2020) highlight the importance of structured improvement paths in DevOps initiatives, underscoring that organizational change, rather than tooling alone, often defines the trajectory of successful implementation.

Organizations should also invest in structured change management strategies that go beyond technical enablement. Initiatives such as onboarding programs, cross-role facilitation, and stakeholder alignment play a crucial role in supporting sustainable adoption. Furthermore, evaluating the outcomes of DevOps initiatives requires a broader set of indicators. Rather than relying solely on median task duration or deployment frequency, assessments should incorporate trend evolution, variance analysis, and practitioner perceptions to reflect the complexity of the implementation context.

These insights are particularly relevant for large-scale or traditional organizations undergoing continuous practices applications. They echo findings from (LEITE *et al.*, 2019; KREY, 2022; MISHRA; OTAIWI, 2020), which argue that success depends as much on organizational maturity, adaptability, and cultural engagement as on the sophistication of pipeline automation.

Overall, this study contributes to the broader discourse on DevOps by offering an empirical, context-aware view of what continuous practices adoption achieves and what it requires in environments where speed must coexist with stability, compliance, and legacy integration.

To conclude, it is important to reflect on how the specific continuous practices evaluated in this study relate to the observed outcomes. The task performance analysis, supported by participant narratives, consistently highlighted CI/CD pipelines and observability tools as the main enablers of process efficiency and predictability, with reduced variance and perceived improvements in delivery flow. Code review and pair programming were also associated with positive perceptions of quality, although their impact on throughput was more limited, often constrained by reviewer availability and coordination overhead. In contrast, practices related to cultural change revealed a more mixed perception. While participants recognized their importance, many described difficulties in aligning them with existing structures and routines. The visual analyses, including the stacked bar charts, reinforce these findings by showing strong support for technical enablers like CI/CD and observability, while also pointing to areas where perceived benefits did not consistently translate into measurable performance gains. Overall, these findings underscore the importance of a context-aware approach to implementing continuous practices, especially in settings where technical progress must coexist with structural complexity and organizational constraints.

## 4.3 Threats to Validity

While this study provides valuable insights into the adoption and effects of continuous practices, several threats to validity must be considered to ensure a balanced and credible interpretation of the results. Following established methodological guidelines for case studies in software engineering (RUNESON; HÖST, 2009; STOREY *et al.*, 2025; LAGO *et al.*, 2024; YIN, 2018), we structure this discussion according to four types of validity threats: construct validity, conclusion validity, internal validity, and external validity. Where applicable, we describe the mitigation strategies employed to address these risks.

### *4.3.1 Construct Validity*

Construct validity refers to whether the operationalization of concepts accurately reflects the theoretical constructs being studied. In our quantitative phase, task metrics (e.g., timeToDone, timeInDoing, timeInWaiting) were derived from structured state transitions recorded in the team's Kanban system and operationalized as temporal metrics. The dataset spanned over two years, minimizing selection bias and providing a longitudinal view of task behavior. This choice is supported by prior empirical work on DevOps performance indicators (PORT *et al.*, 2024; AMARO *et al.*, 2024).

For the qualitative phase, we designed a structured interview guide based on a targeted literature review. The guide underwent iterative refinement with contributions from the research team and was pilot-tested with two participants to improve clarity and logical flow. The use of a hybrid inductive-deductive coding process (STOREY *et al.*, 2025) also helped ensure systematic identification of both emergent insights and theory-aligned constructs.

### *4.3.2 Conclusion Validity*

Conclusion validity relates to the extent to which conclusions drawn from the data are credible and statistically sound. To mitigate this threat, we selected a non-parametric test (Mann-Whitney U) due to the high variance and non-normal distribution of the data. Robustness was further reinforced through sensitivity analysis and parameter variation, such as different thresholds for outlier exclusion (IQR = 3.0).

Triangulation with qualitative evidence helped to confirm the direction and practical relevance of observed effects, even when statistical significance varied between tests. Descriptive shifts in median and variance, together with visualizations (e.g., violin plots, histograms, time series), offered converging evidence that supported the observed performance shifts.

### *4.3.3 Internal Validity*

Internal validity concerns the degree to which observed effects can be attributed to the studied intervention. While we observed reductions in task cycle time and waiting periods post-adoption, alternative explanations (e.g., parallel process changes, team restructuring, or organizational initiatives) may have also influenced the outcomes.

To mitigate this, we employed explanatory sequential triangulation. Quantitative

findings were complemented by developer narratives, which contextualized or challenged observed trends. Discrepancies between metrics and perceptions were further explored through mediating factors (SU; STORER, 2023; SENAPATHI *et al.*, 2021) such as cross-team coordination, infrastructure readiness, or adoption resistance, thus supporting more nuanced causal interpretation.

### *4.3.4 External Validity*

External validity addresses the generalizability of our findings. As a single-case study conducted in a regulated energy-sector organization, the results may not be directly transferable to all software development contexts. Specific factors such as organizational culture, system criticality, and legacy constraints may limit applicability elsewhere.

However, our explanatory mixed-methods approach, integrating objective metrics with practitioner perspectives, offers insights that are theoretically transferable and empirically grounded. Detailed contextualization supports analytical generalization. Future replications in diverse domains, including large-scale and safety-critical systems (DINGSØYR *et al.*, 2014; LAGO *et al.*, 2024), would help broaden external relevance.

By addressing these four types of validity and applying mitigation strategies aligned with best practices in software engineering research (RUNESON; HÖST, 2009; STOREY *et al.*, 2025; YIN, 2018), our study provides a rigorous and transparent foundation, enhancing the credibility and replicability of the findings.

## 4.4 Chapter Summary

This chapter reported the empirical results of the study. Quantitative analyses (RQ1) revealed moderate improvements in task completion times and waiting periods, with the most robust gains observed in reduced variance and improved delivery predictability. Qualitative findings (RQ2) reinforced these trends by showing that developers consistently valued improvements in software quality, workflow visibility, and collaboration, while acknowledging challenges related to legacy constraints, compliance overhead, and cultural adaptation.

By integrating both strands, the results suggest that the primary contribution of continuous practices in complex environments lies in enabling stability and transparency, rather than uniform acceleration. These insights prepare the ground for the conclusion chapter, where

we synthesize contributions, limitations, and directions for future research.

# 5 CONCLUSION

This study investigated the adoption and impact of continuous practices on software maintenance and evolution tasks in a mission critical system within the energy sector. Using a mixed-methods approach that combined task-level statistics with practitioner insights, we offered a contextual view of how these practices unfold in legacy-constrained environments.

## 5.1 Main Contributions

This dissertation offers three primary contributions to software engineering research and practice:

1. **Empirical Evidence:** A quantitative assessment of task performance before and after the adoption of continuous practices, one of the few empirical accounts from a regulated, legacy-constrained industrial system.
2. **Qualitative Insights:** An in-depth account of practitioner perceptions regarding benefits, challenges, and cultural adaptations required for adoption, adding context to the quantitative results.
3. **Practical Guidance:** Actionable recommendations for organizations aiming to adopt continuous practices in complex settings, showing that predictability and collaboration often matter more than raw speed.

The results indicate that continuous practices can improve workflow predictability and process stability. Developers reported perceived gains in software quality, visibility, and coordination. Quantitative effects on delivery speed were limited. Reductions in variance across key metrics, together with consistent reports of improved consistency, suggest that in regulated contexts predictability is a more realistic and valuable near-term outcome than pure speed.

However, the realization of benefits was often limited by organizational and structural factors. Compliance requirements, coordination overhead, and technical debt in legacy systems were frequently cited as barriers. These findings emphasize that successful adoption depends not only on technical tools, but also on cultural readiness, governance alignment, and the maturity of organizational processes.

In sum, this study contributes to a contextualized view of continuous practices adoption. Rather than a universal delivery accelerator, continuous practices play a more nuanced role. In complex environments, continuous practices seem to work best when used to support

learning, collaboration, and gradual adaptation, rather than just speeding things up. Ultimately, our findings suggest that continuous practices should not be seen merely as acceleration resources, but as catalysts for cultivating sustainable, resilient, and adaptive software delivery in complex environments.

## 5.2 Limitations

As with any empirical study, this research has limitations that should be acknowledged. First, it relies on a single-case study, which limits generalizability across different organizational contexts. Second, the dataset analyzed 181 tasks limits the statistical power for detecting small effects. Third, practitioner perceptions may reflect local bias from culture, regulatory pressure, or timing. These limitations do not diminish the value of the findings, but they frame the extent to which conclusions can be transferred to other settings.

## 5.3 Directions for Future Research

Although this study offers a detailed account of the impacts of continuous practices in a specific organizational setting, several research avenues remain open. One promising direction is the expansion of empirical studies across different industries with varying levels of regulation and legacy constraints, such as healthcare, finance, or defense. These comparative efforts could help clarify how domain-specific characteristics shape the adoption, adaptation, and effectiveness of continuous practices.

Another direction is to investigate transformations in very large-scale systems, as defined by Dingsøyr et al. (DINGSØYR *et al.*, 2014). These environments involve additional complexity, including distributed teams, diverse toolchains, and coordination overhead, which may significantly influence both implementation strategies and outcomes. Increasing the quantity and temporal range of task data could also support more robust longitudinal analyses, enabling researchers to track the evolution of DevOps maturity over time.

Qualitatively, broadening participant roles can uncover organizational dynamics and local adaptations that are often overlooked. This is especially relevant for understanding resistance to change, cultural mismatches, and the conditions that support a sustainable DevOps mindset.

Furthermore, future studies could benefit from isolating specific practices, such

as automated testing, deployment pipelines, or observability tools, to better understand their individual contributions. This granular perspective may help clarify causal mechanisms and support more targeted improvements in practice.

Lastly, emerging paradigms like DevSecOps deserve closer scrutiny. Understanding how security-oriented practices integrate with continuous delivery pipelines, particularly in mission-critical and regulated contexts, can refine frameworks for secure and efficient DevOps adoption. A systematic review by Rajapakse et al. (RAJAPAKSE *et al.*, 2022) reinforces this need by identifying critical challenges that remain unresolved in current adoption efforts.

Together, these research directions can help to build a broader and more nuanced understanding of how continuous practices shape software engineering processes, technically, organizationally, and culturally. Overall, this dissertation highlights that the impact of continuous practices is not universal, but deeply contextual. In legacy and regulated environments, their greatest value lies in fostering stability, transparency, and collaboration rather than accelerating delivery at all costs. By documenting both measurable outcomes and lived experiences, this work contributes to a more realistic understanding of DevOps adoption and offers a foundation for researchers and practitioners to build upon.

## REFERENCES

AKBAR, M. A.; RAFI, S.; HYRYNSALMI, S.; KHAN, A. A. Towards people maturity for secure development and operations: A vision. *In:* **Proceedings of the 28th International Conference on Evaluation and Assessment in Software Engineering**. Association for Computing Machinery, 2024. p. 528–533. Available at: https://doi.org/10.1145/3661167.3661238. Accessed on: 10 feb. 2025.

ALMEIDA, C. D. A. de; FEIJó, D. N.; ROCHA, L. S. Studying the impact of continuous delivery adoption on bug-fixing time in apache's open-source projects. *In:* **19th International Conference on Mining Software Repositories (MSR '22)**. *[s. n.]*, 2022. p. 403–413. Available at: https://doi.org/10.1145/3524842.3528049. Accessed on: 20 aug. 2024.

AMARO, R.; PEREIRA, R.; SILVA, M. M. da. Capabilities and practices in devops: A multivocal literature review. **IEEE Transactions on Software Engineering**, v. 49, p. 883–901, 2023. Available at: https://api.semanticscholar.org/CorpusID:248140206. Accessed on: 18 nov. 2024.

AMARO, R.; PEREIRA, R.; SILVA, M. M. da. Devops metrics and kpis: A multivocal literature review. **ACM Computing Surveys**, 2024. Available at: https://doi.org/10.1145/3652508. Accessed on: 08 mar. 2025.

AZAD, N. Understanding devops critical success factors and organizational practices. *In:* **2022 IEEE/ACM International Workshop on Software-Intensive Business (IWSiB)**. *[s. n.]*, 2022. p. 83–90. Available at: https://doi.org/10.1145/3524614.3528627. Accessed on: 08 mar. 2025.

AZAD, N.; HYRYNSALMI, S. Multivocal literature review on devops critical success factors. *In:* **Proceedings of the 28th International Conference on Evaluation and Assessment in Software Engineering**. Association for Computing Machinery, 2024. p. 520–527. Available at: https://doi.org/10.1145/3661167.3661236. Accessed on: 10 feb. 2025.

BASILI, V. R.; CALDIERA, G.; ROMBACH, H. D. The goal question metric approach. **Encyclopedia of software engineering**, p. 528–532, 1994.

BASS, L.; WEBER, I.; ZHU, L. **DevOps: A Software Architect's Perspective**. *[S. l.]*: Addison-Wesley Professional, 2015.

DHAKAD, K. Adopting continuous integration practices to achieve quality in devops. **International Journal of Advanced Research in Science, Communication and Technology**, 2023. Available at: https://api.semanticscholar.org/CorpusID:256975037. Accessed on: 16 apr. 2025.

DINGSØYR, T.; FÆGRI, T. E.; ITKONEN, J. What is large in large-scale? a taxonomy of scale for agile software development. *In:* JEDLITSCHKA, A.; KUVAJA, P.; KUHRMANN, M.; MÄNNISTÖ, T.; MÜNCH, J.; RAATIKAINEN, M. (Ed.). **Product-Focused Software Process Improvement**. *[S. l.]*: Springer International Publishing, 2014. p. 273–276.

DUARTE, R. M. **DevOps Capabilities and Metrics**. 2021. Available at: https://api.semanticscholar.org/CorpusID:253965647. Accessed on: 05 sep. 2024.

ERICH, F. M.; AMRIT, C.; DANEVA, M. Devops adoption in practice: Patterns and anti-patterns. **Journal of Software: Evolution and Process**, Wiley, v. 29, n. 6, p. e1885, 2017. Available at: https://doi.org/10.1002/smr.1885. Accessed on: 08 mar. 2025.

FEIJó, D. N.; ALMEIDA, C. D. A. d.; ROCHA, L. S. Studying the impact of ci/cd adoption on atoms of confusion distribution and prevalence in open-source projects. **Journal of Software Engineering Research and Development**, v. 12, n. 1, 2024. Available at: https://doi.org/10.5753/jserd.2024.4118. Accessed on: 16 apr. 2025.

FORSGREN, N.; HUMBLE, J.; KIM, G. **Accelerate**: The science of lean software and devops. *[S. l.]*: IT Revolution Press, 2018.

HüTTERMANN, M. **DevOps for Developers**. Apress, 2012. Available at: https://doi.org/10.1007/978-1-4302-4570-4. Accessed on: 08 mar. 2025.

KHAN, M. S.; KHAN, A. W.; KHAN, F.; KHAN, M. A.; WHANGBO, T. K. Critical challenges to adopt devops culture in software organizations: A systematic review. **IEEE Access**, v. 10, p. 14339–14349, 2022. Available at: https://doi.org/10.1109/ACCESS.2022.3145970. Accessed on: 05 sep. 2024.

KITCHENHAM, B. A.; PFLEEGER, S. L. Principles of survey research: part 5: populations and samples. **ACM SIGSOFT Software Engineering Notes**, Association for Computing Machinery, v. 27, n. 5, p. 17–20, 2002. Available at: https://doi.org/10.1145/571681.571686. Accessed on: 22 apr. 2025.

KREY, M. Devops adoption: Challenges & barriers. *In:* **Hawaii International Conference on System Sciences**. *[s. n.]*, 2022. Available at: https://api.semanticscholar.org/CorpusID:245884929. Accessed on: 18 nov. 2024.

KUMAR, A.; NADEEM, M.; SHAMEEM, M. Systematic literature review of metrics for measuring devops success. **Computational Intelligence and Network Security**, 2023. Available at: https://api.semanticscholar.org/CorpusID:258414313. Accessed on: 05 sep. 2024.

LAGO, P.; RUNESON, P.; SONG, Q.; VERDECCHIA, R. Threats to validity in software engineering - hypocritical paper section or essential analysis? *In:* **Proceedings of the 18th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)**. Association for Computing Machinery, 2024. p. 314–324. Available at: https://doi.org/10.1145/3674805.3686691. Accessed on: 16 mar. 2025.

LEITE, L.; ROCHA, C.; KON, F.; MILOJICIC, D.; MEIRELLES, P. A survey of devops concepts and challenges. **ACM Computing Surveys**, v. 52, n. 6, p. 127:1–127:35, 2019. Available at: https://doi.org/10.1145/3359981. Accessed on: 20 aug. 2024.

MACARTHY, R. W.; BASS, J. M. The role of skillset in the determination of devops implementation strategy. *In:* **2021 IEEE/ACM Joint 15th International Conference on Software and System Processes (ICSSP) and 16th ACM/IEEE International Conference on Global Software Engineering (ICGSE)**. *[s. n.]*, 2021. p. 50–60. Available at: https://api.semanticscholar.org/CorpusID:235637460. Accessed on: 16 apr. 2025.

MISHRA, A.; OTAIWI, Z. Devops and software quality: A systematic mapping. **Computer Science Review**, v. 38, p. 100308, 2020. Available at: https://doi.org/10.1016/j.cosrev.2020.100308. Accessed on: 10 feb. 2025.

PORT, D.; TABER, B.; EMKANI, P. Investigating effectiveness and compliance to devops policies and practices for managing productivity and quality variability. **Journal of Systems and Software**, v. 213, p. 112030, 2024. Available at: https://doi.org/10.1016/j.jss.2024.112030. Accessed on: 05 sep. 2024.

RAFI, S.; WU, Y.; AKBAR, M. A. Rmdevops: A road map for improvement in devops activities in context of software organizations. *In:* **Proceedings of the 24th International Conference on Evaluation and Assessment in Software Engineering**. Association for Computing Machinery, 2020. p. 413–418. Available at: https://doi.org/10.1145/3383219.3383278. Accessed on: 08 mar. 2025.

RAJAPAKSE, R. N.; ZAHEDI, M.; BABAR, M. A.; SHEN, H. Challenges and solutions when adopting devsecops: A systematic review. **Information and Software Technology**, v. 141, p. 106700, 2022. Available at: https://doi.org/10.1016/j.infsof.2021.106700. Accessed on: 10 feb. 2025.

ROMERO, E. E.; CAMACHO, C. D.; MONTENEGRO, C. E.; ACOSTA, Ó. E.; CRESPO, R. G.; GAONA, E. E.; MARTÍNEZ, M. H. Integration of devops practices on a noise monitor system with circleci and terraform. **ACM Transactions on Management Information Systems (TMIS)**, v. 13, p. 1–24, 2022. Available at: https://doi.org/10.1145/3505228. Accessed on: 20 aug. 2024.

RUIZ, J. M. S.; MAYO, F. J. D.; ORIOL, X.; CRESPO, J. F.; BENAVIDES, D.; TENIENTE, E. A benchmarking proposal for devops practices on open source software projects. **Journal of Software Engineering Research and Development**, 2015. Available at: https://arxiv.org/abs/2304.14790. Accessed on: 06 jan. 2025.

RUNESON, P.; HÖST, M. Guidelines for conducting and reporting case study research in software engineering. **Empirical Software Engineering**, Springer, v. 14, n. 2, p. 131–164, 2009. Available at: https://doi.org/10.1007/s10664-008-9102-8. Accessed on: 16 apr. 2025.

SANTOS, J.; COSTA, D. A. da; MCINTOSH, S.; KULESZA, U. On the need to monitor continuous integration practices. **Empirical Software Engineering**, v. 30, 2024. Available at: https://doi.org/10.48550/arXiv.2409.05101. Accessed on: 10 feb. 2025.

SENAPATHI, M.; BUCHAN, J.; OSMAN, H. Devops capabilities, practices, and challenges: Insights from a case study. *In:* **Proceedings of the 2021 International Conference on Software Engineering and Knowledge Engineering (SEKE)**. *[s. n.]*, 2021. p. 1–10. Available at: https://doi.org/10.1145/3475215.3475231. Accessed on: 06 jan. 2025.

SHAHIN, M.; BABAR, M. A.; ZHU, L. Continuous integration, delivery and deployment: A systematic review on approaches, tools, challenges and practices. **IEEE Access**, 2017. Available at: https://doi.org/10.1109/ACCESS.2017.2779538. Accessed on: 20 aug. 2024.

SOARES, E.; SIZILIO, G.; SANTOS, J.; ALENCAR, D.; KULESZA, U. The effects of continuous integration on software development: a systematic literature review. **Empirical Software Engineering**, v. 27, 2021. Available at: https://doi.org/10.48550/arXiv.2103.05451. Accessed on: 20 aug. 2024.

STAHL, D.; MARTENSSON, T.; BOSCH, J. Continuous practices and devops: beyond the buzz, what does it all mean? *In:* **2017 43rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA)**. *[s. n.]*, 2017. p. 440–448. Available at: https://doi.org/10.1109/SEAA.2017.8114695. Accessed on: 22 may. 2025.

STOREY, M.-A.; HODA, R.; MILANI, A. M. P.; BALDASSARRE, M. T. **Guiding Principles for Using Mixed Methods Research in Software Engineering**. 2025. Available at: https://arxiv.org/abs/2404.06011. Accessed on: 16 apr. 2025.

SU, L.; STORER, T. A case study of devops adoption within a large financial organisation. *In:* **2023 IEEE International Conference on Software Maintenance and Evolution (ICSME)**. *[s. n.]*, 2023. p. 403–413. Available at: https://api.semanticscholar.org/CorpusID:266199839. Accessed on: 12 oct. 2024.

VASILESCU, B.; YU, Y.; WANG, H.; DEVANBU, P.; FILKOV, V. Quality and productivity outcomes relating to continuous integration in github. *In:* **Proceedings of the 10th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering**. Bergamo, Italy: Association for Computing Machinery, 2015. p. 1–10. Available at: https://doi.org/10.1145/2786805.2786850. Accessed on: 06 jan. 2025.

WINKLER, F.; WESTNER, M. A systematic literature review of devops success factors and adoption models. *In:* **Proceedings of the 12th International Symposium on Information and Communication Technology**. Association for Computing Machinery, 2023. p. 525–532. Available at: https://doi.org/10.1145/3628797.3628883. Accessed on: 05 sep. 2024.

YIN, R. K. **Case Study Research and Applications**: design and methods. *[S. l.]*: SAGE Publications, 2018.

ZOHAIB, M. **What Practitioners Really Think About Continuous Software Engineering**: A taxonomy of challenges. 2023. Available at: https://api.semanticscholar.org/CorpusID:257833780. Accessed on: 18 nov. 2024.

# APPENDIX A – SURVEY INSTRUMENT

This appendix presents the complete instrument used to collect responses from software professionals. The questionnaire was organized into thematic groups aligned with the research questions addressed in this study. Some items were adapted from prior empirical studies on continuous practices to ensure consistency and comparability with established research.

All Likert-scale questions used a five-point scale. The labels varied slightly depending on the focus of the question:

- Level of agreement: 1 = Strongly disagree, 2 = Disagree, 3 = Neutral, 4 = Agree, 5 = Strongly agree
- Level of perceived challenge, or extent: 1 = Not at all, 2 = Slightly, 3 = Moderately, 4 = Considerably, 5 = To a great extent
- Level of perceived usefulness: 1 = Not important, 2 = Slightly important, 3 = Moderately important, 4 = Important, 5 = Very important
- Level of perceived relevance of practices: 1 = Strongly disagree, 2 = Disagree, 3 = Neutral, 4 = Agree, 5 = Strongly agree

## Group 1 – Participant Profile

Q1. How many years of professional experience do you have in software development? *[Open text]*
Q2. What is your academic background and in which field? *[Open text]*
Q3. What is your highest academic degree completed? *[Open text]*
Q4. For how long have you been working with continuous practices (in years)? *[Open text]*
Q5. Have you completed any training or courses in continuous practices? Please describe. *[Open text]*
Q6. What is your primary role in the organization? *[Open text]*

## Group 2 – Perception of Productivity and Delivery Speed Improvements

The following questions were designed to assess participants' perceptions regarding the impact of continuous practices on productivity, particularly in terms of time spent in

development phases (RQ1), as well as perceived improvements in delivery speed (RQ2.1). These questions explore how participants evaluate the influence of continuous practices on work pace, task flow, and release frequency.

Q7. Do you agree that the evaluated practices positively impacted delivery speed? *[Likert]*

Q8. Can you share specific examples where this did or did not affect task completion time? *[Open-ended]*

Q9. Did the practices impact the workflow, such as time spent working on tasks (time in doing) or waiting time (from card creation to being moved to doing)? *[Likert]*

Q10. How did this impact manifest? *[Open-ended]*

**Group 3 – Perceived Impact on Software Quality**

The following questions aim to investigate whether participants observed improvements in software quality as a result of adopting continuous practices. In particular, they focus on perceived effects on the quality of deliveries and the role of observability mechanisms such as monitoring dashboards and alerts.

Q11. Do you agree that the evaluated practices positively impacted delivery quality? *[Likert]*

Q12. Can you share specific examples where this did or did not affect the quality of the tasks? *[Open-ended]*

Q13. What is your perception of the usefulness of monitoring dashboards (e.g., Grafana) and email alerting mechanisms? *[Likert]*

**Group 4 – Perceived Impact in Release Frequency**

This question addresses whether practitioners observed a variation in the frequency of software releases as a result of adopting continuous practices, offering insights into the perceived acceleration of the delivery pipeline.

Q14. Do you agree that the implemented practices led to an increase in release frequency in your team's context? *[Likert]*

**Group 5 – Influence of Continuous Practices on Developers' Daily Workflows**

This section explores how continuous practices affect developers' daily routines and operational workflows. It includes both structured and open-ended questions aimed at capturing perceptions about practical impacts and opportunities for improvement or complementarity.

Q15. Did the practices impact the workflow, such as time spent working on tasks (time in doing) or waiting time (from card creation to being moved to doing)? *[Likert]*

Q16. How did this impact manifest? *[Open-ended]*

Q17. What other practices could be adopted? *[Open-ended]*

Q18. Is there anything else you would like to share about the impact of these practices on your daily work or your team as a whole? *[Open-ended]*

**Group 6 – Challenges Faced During the Adoption of Continuous Practices**

This section investigates the main obstacles encountered by practitioners when adopting continuous practices. It includes both Likert-scale and open-ended questions focused on technical complexity, cultural resistance, integration with legacy systems, and other contextual barriers.

Q19. How challenging was it to adopt CI/CD? *[Likert]*

Q20. Please justify (e.g., technical complexity, team adaptation). *[Open-ended]*

Q21. Did cultural change represent a major challenge? *[Likert]*

Q22. What kinds of resistance or difficulties emerged? *[Open-ended]*

Q23. What specific challenges do you face when integrating continuous practices? *[Open-ended]*

Q24. Were there difficulties in integrating tools or adapting legacy code to the new pipeline? *[Likert]*

**Group 7 – Perceived Relevance of Continuous Practices**

This section aims to understand how practitioners perceive the relative relevance of specific continuous practices in their development and maintenance contexts. Participants were asked to rate each practice individually based on their perceived importance and utility.

Q25. How do you evaluate the relevance of each practice? *[Likert – individual scale for each*

*item]*

- CI/CD
- Code review and pair programming
- Cultural changes
- Monitoring dashboards and observability tools
- DevSecOps