Deanship of Graduate Studies

Al-Quds University

Implementing Agile and DevOps at Scale: Identifying Best Frameworks, Practices, and Success Factors

Mohammad Adnan Abu Ayyash

M.Sc. Thesis

Jerusalem - Palestine

1445 AH / 2024 AD

Implementing Agile and DevOps at Scale: Identifying Best Frameworks, Practices, and Success Factors

Prepared By:

Mohammad Adnan Abu Ayyash

B.Sc. Management Information Systems from the Arab American University - Palestine.

Supervisor: Dr. Raid Zaghal

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Science in Computer Science / Department of Computer Science Faculty of Science & Technology / Deanship of Graduate Studies, Al-Quds University.

1445 AH / 2024 AD

Al-Quds University
Deanship of Graduate Studies
Department of Computer Science
Masters in Computer Science

Thesis Approval

Implementing Agile and DevOps at Scale: Identifying Best Frameworks, Practices, and Success Factors

Prepared By: Mohammad Adnan Abu Ayyash

Registration No: 21820244

Supervisor: Dr. Raid Zaghal

Master thesis submitted and accepted, Date: 27 / 5 / 2024.

The names and signatures of the examining committee are as follows:

1. Head of Committee: Dr. Raid Zaghal signature .....
2. Internal Examiner: Dr. Radwan Qasrawi signature .....
3. External Examiner: Prof. Yousef Abuzir signature .....

Jerusalem - Palestine

1445 AH / 2024 AD

Declaration

I certify that this thesis submitted for the degree of Master, is the result of my research, except where otherwise acknowledged, and that this study, or any part of the same, has not been submitted for a higher degree to any other university or institution.

Mohammad Adnan Abu Ayyash

Signature: 

Date: 27 / 5 /2024

Dedication

To my father, Dr. Adnan Abu Ayyash, whose unwavering support, wisdom, and belief in me have been my guiding light throughout this journey. And to my dear mother, Lila, you have always offered emotional, spiritual, and moral support at every step of this academic journey, for her boundless love, patience, and encouragement, which have been my greatest source of strength.

Thank you for always being there for me.

Acknowledgment

All praise is to Allah. I extend my heartfelt gratitude to whose unwavering support and encouragement have been instrumental in completing this thesis. Primarily, I express my deepest appreciation to my beloved wife, Asma, and our daughters, Farah, Rahaf, Naya and Asma, your support, patience, and unconditional love have been a driving force for me. Your strength, understanding, and boundless love have empowered me to focus on my studies and achieve my academic goals with determination and perseverance. I am deeply grateful for your love, encouragement, and unwavering faith in me. I would also like to express my sincere gratitude to my academic supervisor, Dr. Raid Zaghal, for his guidance, expertise, and mentorship throughout the research process. Your insightful feedback, constructive criticism, and dedication to excellence have significantly contributed to refining and succeeding in this thesis. I am profoundly thankful for your guidance and support. To all those who have supported, believed in, and inspired me throughout this journey, I express my deepest gratitude. Your contributions, whether big or small, have played a significant role in shaping this work and my academic endeavours.

Abstract

Agile methodologies are widely acknowledged for their capacity to improve project outcomes, accelerate delivery schedules, and enhance team productivity within the software development domain. However, a noticeable gap remains in understanding the global implementation of Agile practices alongside DevOps and the factors contributing to their collective success, particularly in Agile-DevOps integration. This study aims to address this gap by examining the deployment of Agile methodologies, particularly when integrated with DevOps, across the international software industry.

Using a mixed-methods approach, we conducted surveys and interviews with 53 software development professionals from 93 companies across various regions and analyzed their responses alongside a comprehensive dataset from these companies. We identified prevalent challenges, optimal practices, and critical success factors associated with Agile and DevOps integration, including organizational culture, leadership support, and continuous improvement practices.

Furthermore, the investigation aimed to identify an appropriate organizational change framework grounded in Agile principles and develop strategies to optimize outcomes. The insights garnered from this research effort are expected to inform practitioners on the effective implementation of Agile and DevOps methodologies, thereby improving processes for existing adopters. Additionally, the study explored the potential contributions of computer science to scaling Agile and DevOps initiatives, leveraging methodologies such as PROMETHEE-II for informed decision-making.

Identified success factors underwent validation and prioritization through consultation with industry experts, providing valuable insights for enhancing Agile and DevOps processes. Recommendations for companies within this domain were proposed and

validated through real-world application in software projects. Based on our analysis, the Triangulation method was suggested as particularly suitable for small software companies. Adoption of these methodologies and frameworks can address challenges associated with scaling Agile and DevOps, promoting more efficient software development practices aligned with organizational objectives. While various frameworks such as SAFe, LeSS, DEAM, ITIL, and Kanban offer scalability solutions, careful consideration should be given to selecting the most suitable

framework aligned with organizational requirements and objectives.

Main results indicated that companies implementing Agile-DevOps integration with strong leadership support and a culture of continuous improvement saw the most significant improvements in productivity and delivery schedules.

Keywords

Agile, methodologies, framework, software industry, software development, DevOps, Agile-DevOps integration, scaling, SAFe, LeSS, challenges, best practices, success factors, organizational change, computer science, decision-making.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

Abbreviation	Abbreviation Full Name
MTBF	Mean Time Between Failures
MTTD	Mean Time To Detect
CI/CD	Continuous Integration/Continuous Delivery
ARTs	Agile Release Trains
IaC	Infrastructure as Code
LPM	Lean Portfolio Management
TDD	Test-Driven Development
MTTR	Mean Time to Restore
ITIL	Information Technology Infrastructure Library
DAM	Digital Asset Management
ORM	Object-Relational Mapping
DSM	Daily Stand-up Meetings
CFT	Cross-Functional Teams
GAs	Genetic Algorithms
KPIs	Key Performance Indicators
SOA	Service-Oriented Architecture
FPA	Function Point Analysis
VSM	Value Stream Mapping
SDLC	Software Development Lifecycle
DOI	Digital Object Identifier
CAMSOPPRTPT	C= Culture A= Automation M= Measurement S= Sharing O=Organizational P= People PR=Process T =Technical PT=Project
CSFs	Critical Success Factors

Chapter 1

Introduction

This chapter introduces the thesis. It describes the problem statement, purpose, research questions, limitations, contributions, methodology and organization of the thesis.

1.1 Overview of Agile and DevOps Integration

The integration of Agile and DevOps methodologies has become indispensable in contemporary software development, facilitating efficiency, collaboration, and the seamless delivery of high-quality software products. This amalgamation accelerates development timelines and maximizes the inherent advantages of each methodology, marking a transformative phase in the software development lifecycle.

In the domain of software development, Agile methodologies are crucial for improving project outcomes, accelerating delivery schedules, and enhancing team productivity. These methodologies prioritize adaptability, teamwork, and iterative development cycles, fundamentally reshaping conventional software development paradigms. Simultaneously, DevOps principles have gained prominence, emphasizing collaboration between development and operations teams to achieve streamlined and efficient software delivery.

Despite the evident benefits of Agile and DevOps, their integration, commonly referred to as Agile-DevOps integration, presents significant opportunities for businesses seeking to optimize processes and enhance agility. By combining Agile's iterative approach with

DevOps' focus on automation and continuous delivery, this integration facilitates rapid innovation and deployment.

However, a notable gap exists in understanding the implementation of Agile-DevOps integration across the software industry and the factors contributing to its success. This research aims to address this gap by examining the utilization of Agile methodologies, particularly in conjunction with DevOps. Through comprehensive surveys, interviews, and data analysis, we aim to identify common challenges, best practices, and critical success factors associated with Agile-DevOps integration.

By delving into the complexities of Agile-DevOps integration, this study aims to empower businesses to refine their software development processes and maintain competitiveness in the digital landscape.

Despite the increasing demand for Agile DevOps integration in software organizations, numerous studies have explored various aspects such as tools, definitions, concepts, factors, practices, and characterizations of Agile DevOps integration. However, none of these studies have thoroughly investigated the conceptual mapping and relationships between success factors. This research gap has motivated me to highlight the logical relationship between success factors and develop a taxonomy to assist Agile DevOps experts in designing strategies for better implementation.

To address this gap, I have leveraged a previous study on the identification of success factors for Agile DevOps integration, published in the EASE'20 conference. I have mapped the identified success factors to the CAMSOPPRTPT criteria (principles of Agile DevOps integration) to develop a taxonomy. Additionally, I have analyzed the logical relationship between these success factors and ranked them using the PROMETHEE-II

method within the domain of Agile DevOps. This ranking and understanding of logical relationships will aid Agile DevOps experts in enhancing their decision-making capabilities by considering specific success factors.

Various approaches exist for measuring ranks in different domains of technology. For instance, methods such as TOPSIS, ELECTRE, WSM, AHP, and PROMETHEE are commonly used for rank measurement. However, PROMETHEE stands out as an approach that assesses ranks through pairwise comparisons of alternatives, determining the strength of one alternative relative to another. This approach has been widely adopted across various scientific fields where decision-making is paramount.

The results derived from the PROMETHEE approach are straightforward and consistent. For example, researchers like Siahaan and Mersan utilized the PROMETHEE approach to select the best student in college based on criteria such as skills, class performance, grading, and attendance. Similarly, Liu and Guan evaluated the quality of railway services provided to passengers using the PROMETHEE technique, transforming linguistic terms into numerical values through a fuzzy triangular scale. Zhao et al. also employed an upgraded version of the PROMETHEE approach for the assessment of incident management plans, such as those for earthquakes and floods.

The integration of Agile's iterative approach with DevOps' continuous integration and deployment processes yields several notable advantages:

- Enhanced Collaboration and Communication: The fusion of Agile's emphasis on team collaboration with DevOps' focus on communication streamlines workflows, fostering tighter integration among development, operations, and other stakeholders.

- Accelerated Delivery Cycles: The combination of Agile's iterative development cycles with DevOps' automation expedites software delivery, ensuring faster time-to-market without compromising quality.
- Improved Quality and Stability: Agile's iterative testing combined with DevOps' automated testing and deployment leads to enhanced product quality and increased stability through continuous monitoring and feedback loops.
- Agility in Adapting to Change: Agile's flexibility and DevOps' rapid deployment capabilities enable teams to respond swiftly to evolving requirements and market demands, ensuring adaptability and responsiveness.
- Optimized Resource Utilization: The alignment of Agile's incremental delivery with DevOps' automated provisioning of resources ensures efficient resource utilization throughout the development lifecycle.

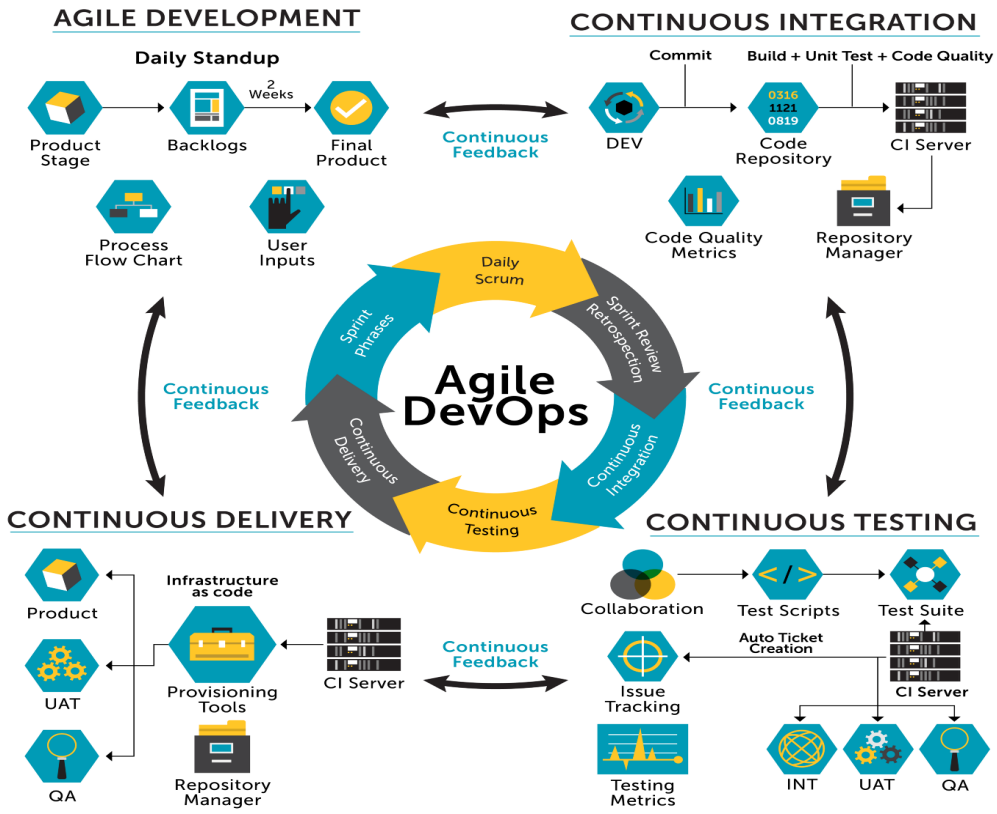

Figure 1.1. A diagram of software development Agile/DevOps integration. Adapted from "Agile – DevOps Implementation" by M. August 2023, AgileFirst [Web]. Retrieved June 21, 2024, from https://agilefirst.io/agile-devops/

1.1.2 Implementing Agile and DevOps Together (Agile DevOps – Integration)

The integration of Agile and DevOps transcends their individual strengths, amplifying collective benefits in software development. By harmonizing iterative development, continuous deployment, and collaborative processes, this unified approach catalyses the delivery of robust, high-quality software products while enhancing team productivity and responsiveness.

In the dynamic landscape of software development, the evolution of methodologies has profoundly influenced the industry's trajectory. Agile methodologies have emerged as a guiding principle, emphasizing iterative development, collaboration, and adaptability [8]. Concurrently, the shift towards DevOps, which prioritizes automation, continuous integration, and infrastructure orchestration, presents both opportunities and challenges alongside Agile frameworks [12]. This convergence marks a pivotal moment in software engineering, necessitating a deeper understanding of Agile-DevOps integration, especially on a large scale.

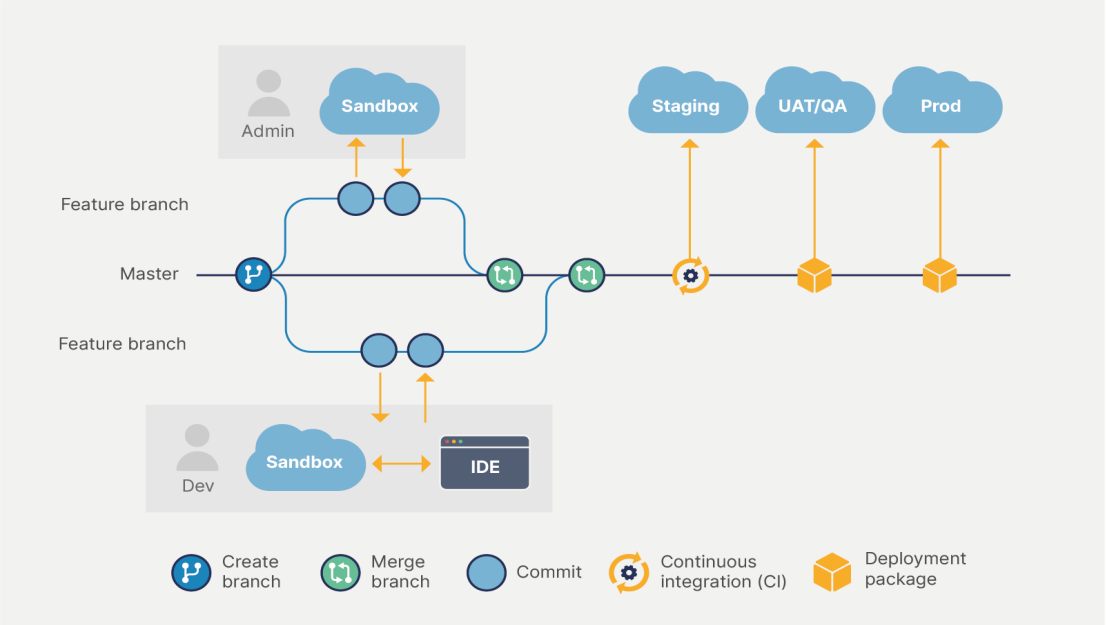

Figure 1.2 Implementing Agile and DevOps Together - Agile DevOps CI/CD Workflow. Adapted from "Agile – DevOps Implementation" by M. August 2023, AgileFirst [Web]. Retrieved June 21, 2024, from https://agilefirst.io/agile-devops/

The landscape of software development has undergone significant transformation with the advent of Agile methodologies and subsequent integration of DevOps practices. This paradigm shift from traditional, linear development approaches to iterative, collaborative models has fundamentally redefined project management and product delivery strategies in the software industry.

Agile methodologies, characterized by iterative development and adaptive planning, have gained prominence for enhancing team productivity, responsiveness to changing requirements, and ensuring the delivery of high-quality software products. Simultaneously, the emergence of DevOps practices underscores the synergy between development and operations, aiming to streamline workflows, expedite deployment cycles, and enhance overall efficiency.

However, despite their transformative potential and widespread adoption, the implementation and adaptation of these methodologies at scale, particularly within the DevOps context, present intricate challenges that require comprehensive exploration. There remains a pressing need for nuanced insights into implementation nuances, success factors, and challenges associated with Agile and DevOps integration within the global software development landscape.

This study aims to address these gaps by conducting a taxonomy study focused on Agile methodologies within the framework of DevOps, encompassing an international perspective. Through an extensive exploration involving surveys and interviews with industry practitioners across diverse regions, this research seeks to identify common challenges, extract best practices, and uncover critical success factors essential for the effective implementation of Agile methodologies in conjunction with DevOps practices.

1.2 Research Problem

The adoption and optimization of Agile and DevOps methodologies present intricate challenges for organizations seeking to scale these practices. This research aims to elucidate these challenges, with a particular focus on integration complexities, adaptation strategies, and the determinants of successful implementation. Understanding and categorizing these challenges are pivotal for developing strategic frameworks that facilitate effective Agile-DevOps integration across diverse organizational contexts.

Moreover, the scarcity of comprehensive studies investigating the global amalgamation of Agile practices within the DevOps framework underscores the urgency and significance of this research effort. The absence of a definitive taxonomy outlining critical success factors, best practices, and challenges in this domain impedes organizations aiming to enhance project outcomes through systematic adoption or optimization.

1.2.1 Statement of the Problem

The amalgamation of Agile methodologies and DevOps practices has emerged as a transformative approach in software development, promising accelerated delivery, enhanced collaboration, and improved software quality. However, the effective integration of these methodologies encounters multifaceted challenges that necessitate a comprehensive examination. The literature underscores the pivotal role played by organizational culture, leadership dynamics, communication patterns, and collaboration frameworks in the successful adoption and implementation of Agile and DevOps practices. Despite these insights, empirical studies bridging the gap between these methodologies remain limited, leaving a void in understanding the nuanced interplay between Agile and DevOps in diverse organizational and cultural contexts.

The modern software development landscape has embraced Agile and DevOps

methodologies due to their potential to enhance productivity, quality, and time-to-market. However, despite their widespread adoption, there exist critical gaps and challenges in the effective implementation and integration of these methodologies at scale within various organizational contexts. This section aims to elucidate the fundamental challenges faced by organizations in the successful deployment and synchronization of Agile and DevOps practices.

1.2.2 Key points to address

- The growing popularity and adoption of Agile and DevOps in software development.
- The persistent challenges hinder seamless integration and implementation.
- The influence of these challenges on project outcomes, team collaboration, and organizational efficiency.
- The need for a comprehensive investigation into these challenges to enhance the efficacy of Agile and DevOps methodologies in diverse organizational contexts.

1.3 Research Objectives and Hypotheses

This research aims to conduct a thorough taxonomy study focusing on Agile methodologies within the DevOps framework by exploring international perspectives through surveys and interviews with industry practitioners globally. It seeks to categorize common integration challenges and best practices, aligning these insights with decision-making parameters for selecting Agile, DevOps, or Integration methodologies in software development. Additionally, the study aims to delineate critical success factors in Agile-DevOps integration and correlate these factors with methodology selection criteria. Methodologically, the research will develop and distribute a survey instrument, conduct

in-depth interviews, and analyze empirical data from software company datasets, synthesizing these findings into a comprehensive taxonomy to provide actionable insights for enhancing Agile and DevOps integration efforts in organizations. The primary objective of this study is to investigate the complexities associated with integrating Agile and DevOps methodologies within the software development domain.

This investigation involves analyzing the effectiveness of various scaling frameworks and their alignment with organizational strategic goals. The study proposes hypotheses to explore correlations between successful Agile-DevOps integration, organizational transformation, and the utilization of specific scaling frameworks.

Research Objectives

1.3.1 Primary Objectives

This section will explicitly delineate the specific objectives of the study:

- Taxonomy Study of Agile and DevOps:

  Conduct a comprehensive taxonomy study of Agile methods implementation, emphasizing the intersection with DevOps on a global scale.
- Identification of Success Factors: Identify common challenges, best practices, and success factors in the international implementation of Agile methods, particularly in the context of DevOps.
- Framework Selection and Proposal: Select and propose an adaptable framework for Agile organizational change and adaptation strategies, especially within the context of DevOps, enhancing implementation outcomes.
- To investigate the complementarity between Agile and DevOps methodologies in large-scale software development projects.
- To propose and recommend key metrics for measuring the success of large-scale

Agile and DevOps implementations.

- To analyze the critical success factors contributing to large-scale Agile and DevOps implementations across various industries.

1.3.2 Secondary Objectives

- To explore and assess the potential contributions of computer science to scaling Agile and DevOps projects.
- To explore and define best practices for effective communication and alignment across multiple teams in large-scale Agile implementations.
- To assess the impact and contribution of infrastructure as code and automation tools in DevOps methodologies for scalability.

Hypothesis

1.3.3 Primary Hypotheses:

- **Hypothesis 1**: There is a significant correlation between effective communication and successful scaling of Agile and DevOps methodologies in large-scale organizational settings.
- **Hypothesis 2**: Organizational culture significantly influences the successful adoption and implementation of Agile and DevOps practices. Rationale: Previous studies have highlighted the impact of cultural factors on the success or failure of Agile and DevOps implementations [33, 15, 26].
- **Hypothesis 3**: The impact of cultural context on the adoption of Agile and DevOps practices varies significantly across regions and industries, influencing the methodologies' efficacy.
- **Hypothesis 4**: Theoretical frameworks play a pivotal role in guiding the implementation and alignment of Agile and DevOps methodologies, significantly

impacting their success.

- **Hypothesis 5**: Effective leadership and team dynamics are critical for the successful adaptation of Agile and DevOps methodologies. Rationale: Research indicates the pivotal role of leadership and team collaboration in successful Agile and DevOps adoption [39, 15].

1.3.4 Secondary Hypotheses:

- **Hypothesis 6**: The integration of Agile and DevOps practices tends to exhibit variances across industries, leading to different outcomes and challenges.
- **Hypothesis 7**: Scaling Agile methodologies across multiple teams and projects introduces complexities related to communication, coordination, and alignment with organizational goals.

  Rationale: Studies have identified challenges in scaling Agile practices due to difficulties in aligning with organizational objectives and ensuring effective communication [26, 16].
- **Hypothesis 8**: The criticisms and challenges surrounding Agile and DevOps methodologies stem from contextual disparities and organizational peculiarities.
- **Hypothesis 9**: Regional and industry-specific factors significantly contribute to the effectiveness of Agile and DevOps implementation.
- **Hypothesis 10**: Regional variations significantly impact the adoption and implementation of Agile and DevOps practices.

  Rationale: Studies have shown that cultural differences influence the success and implementation strategies of Agile and DevOps methodologies across different regions [22, 15].

1. Hypotheses related to the Organization dimension: H 1. The existence of a strong management commitment is a critical success factor that contributes to the successful agile software development projects in terms of (a) Quality, (b) Scope, (c) Time, and (d) Cost. H 2. The presence of agile-friendly organizational environment is a critical success factor that contributes to the successful agile software development projects in terms of (a) Quality, (b) Scope, (c) Time, and (d) Cost. H 3. The existence of agile-friendly project team environment is a critical success factor that contributes to the successful agile software development projects in terms of (a) Quality, (b) Scope, (c) Time, and (d) Cost. 2. Hypotheses related to the People dimension: H 4. Having a team of high caliber is a critical success factor that contributes to the successful agile software development projects in terms of (a) Quality, (b) Scope, (c) Time, and (d) Cost. H 5. Having a strong customer involvement is a critical success factor that contributes to the successful agile software development projects in terms of (a) Quality, (b) Scope, (c) Time, and (d) Cost. 3. Hypotheses related to the Process dimension: H 6. The practice of agile project management process is a critical success factor that contributes to the successful agile software development projects in terms of (a) Quality, (b) Scope, (c) Time, and (d) Cost. H 7. The practice of a methodical project definition process is a critical success factor that contributes to the successful agile software development projects in terms of (a) Quality, (b) Scope, (c) Time, and (d) Cost. 4. Hypotheses related to the Technical dimension: H 8. The practice of agile software engineering techniques is a critical success factor that contributes to the successful T. Chow, D.-B. Cao / The Journal of Systems and Software 81 (2008) 961–971 969 Author's personal copy agile software development projects in terms of (a) Quality, (b) Scope, (c) Time, and (d) Cost. H 9. The execution of a correct delivery strategy is a critical success factor that contributes to the successful agile software development projects in terms of (a) Quality, (b) Scope, (c) Time, and (d)

Cost. 5. Hypotheses related to the Project dimension: H 10. Limiting only to non-life-critical projects is a critical success factor that contributes to the successful agile software development projects in terms of (a) Quality, (b) Scope, (c) Time, and (d) Cost. H 11. Limiting only to projects of variable scope with emergent requirements is a critical success factor that contributes to the successful agile software development projects in terms of (a) Quality, (b) Scope, (c) Time, and (d) Cost. H 12. Limiting only to projects with dynamic, accelerated schedule is a critical success factor that contributes to the successful agile software development projects in terms of (a) Quality, (b) Scope, (c) Time, and (d) Cost.

1.4 Significance of Study

This research holds immense significance in addressing the critical gaps and challenges encountered in Agile and DevOps implementation. Understanding the complex interplay between cultural, organizational, and regional factors and their impact on these methodologies will provide valuable insights for organizations.

The outcomes of this study aim to provide actionable recommendations and strategies specifically tailored to facilitate the integration of Agile and DevOps practices. By bridging these gaps and aligning Agile and DevOps principles, organizations can enhance project outcomes, improve team productivity, and foster a culture of continuous improvement within their software development contexts.

This research makes substantial contributions to academic research and industry within the field of software development methodologies. It enhances understanding of Agile methodologies within the context of DevOps, benefiting various stakeholders:

- Contribution to Practitioners and Industry Professionals: The study offers actionable insights based on key hypotheses:
  - H1: Successful Agile-DevOps integration relies on organizational alignment.
  - H2: Cultural adaptability and flexibility are crucial for integration success.
  - H3: Effective tool integration significantly impacts Agile-DevOps outcomes.

  These insights guide practitioners in adopting, adapting, or optimizing Agile methodologies within DevOps practices to improve project outcomes, team dynamics, and organizational agility.

- Contribution to Academic and Research Community: By addressing critical gaps in existing literature, this study lays a foundation for future research into Agile-DevOps integration at scale. It advances scholarly understanding of theoretical frameworks, methodological approaches, and empirical findings.
- Contribution to Organizational Transformation: The research facilitates organizational transformations by deepening understanding of Agile-DevOps integration dynamics. Organizations leverage these insights to refine implementation strategies, foster cultural alignment, and enhance cross-functional team collaboration.
- Researcher's Contribution

  The researcher conducted a comprehensive literature review to identify gaps and opportunities in Agile and DevOps integration, setting a solid foundation for the study. A robust research framework was developed, incorporating customized surveys, interviews, and advanced data analysis techniques to gather detailed insights from industry practitioners. Utilizing the triangulation method, the study integrated multiple data sources to enhance the validity and reliability of the findings. The PROMETHEE method was applied to develop a decision-making framework, selecting a type-1 preference

function to facilitate comparisons of integration strategies. The researcher identified critical success factors for Agile-DevOps integration, providing actionable recommendations for overcoming integration challenges and best practices for enhancing organizational efficiency and product delivery. The thesis was structured coherently, addressing advisor feedback to ensure clarity and flow, effectively communicating complex concepts to a broad audience.

1.5 Scope

This study primarily focuses on Agile-DevOps integration across diverse industries, emphasizing challenges and strategies for large-scale implementations. It includes a comprehensive taxonomy study of Agile methodologies within the DevOps framework, gathering international perspectives through surveys and interviews with industry practitioners.

1.5.1 Comprehensive Taxonomy Study - Agile within DevOps Framework

This study delves into international perspectives through surveys and interviews with industry practitioners, focusing on Agile methodologies within the DevOps framework. Its goals include identifying common challenges, categorizing best practices, and illuminating critical success factors in Agile-DevOps integration.

1.6 Limitations

While offering valuable insights, this research acknowledges limitations such as data scope and potential biases inherent in survey and interview methods. The evolving nature of Agile and DevOps practices may impact the relevance of findings over time, and generalization across all organizational sectors and regions remains challenging.

1.7 Research Questions

1.7.1 Primary Research Questions:

- **RQ1.** What are the success factors of Agile DevOps integration reported in state of the art and state of the practices?

- **RQ2.** How to enhance decision-making capabilities of Agile DevOps experts for considering the particular success factor?
- **RQ3.** What are the prevalent challenges faced by organizations in scaling Agile and DevOps methodologies, particularly in larger, multi-team environments?
- **RQ4.** How do cultural variations and regional disparities influence the adoption and adaptation of Agile and DevOps practices?
- **RQ5.** What key theoretical frameworks underpin the successful implementation and alignment of Agile and DevOps methodologies?
- **RQ6.** How does organizational culture influence the successful assimilation of Agile and DevOps practices?

The investigation aims to delve into the impact of organizational culture on the successful integration of Agile and DevOps. The study seeks to explore how cultural values, norms, and practices within an organization either facilitate or impede the adoption and effective implementation of combined Agile and DevOps methodologies.

1.7.2 Secondary Research Questions:

- **SRQ1.**How do Agile and DevOps methodologies complement or conflict with each other in different organizational settings?
- **SRQ2.**What are the variations in Agile and DevOps adoption in different industries and regions, and what factors contribute to these variations?

- **SRQ3.**What criticisms and challenges exist concerning the practical implementation of Agile and DevOps methodologies, and how can these be addressed?
- **SRQ4**.To what extent does leadership contribute to fostering effective communication and collaboration within combined Agile and DevOps environments? This research question seeks to examine the role of leadership in cultivating an environment conducive to successful integration. It aims to explore how leadership styles, communication strategies, and collaborative frameworks employed by organizational leaders influence the synergy between Agile and DevOps practices.
- **SRQ5.**What are the primary challenges encountered by organizations in diverse cultural and regional contexts when implementing Agile and DevOps together? This question focuses on uncovering the distinct challenges faced by organizations operating in diverse cultural and regional environments when simultaneously implementing Agile and DevOps methodologies. It aims to identify common barriers, unique contextual challenges, and strategies employed to overcome these hurdles.
- **SRQ6.**What impact do comprehensive training and support mechanisms have on facilitating the adoption of Agile and DevOps practices?

1.8 Structure of the Thesis

This thesis comprises five chapters designed to comprehensively address the research objectives:

- **Chapter 1 – Introduction:** Provides an overview of the research background, problem statement, objectives, hypotheses, significance, scope, and limitations related to Agile-DevOps integration.

- **Chapter 2 – Background & Literature Review:** Explores foundational concepts of Agile methodologies and DevOps practices, including key principles, adoption trends, challenges, criticisms, regional variations, and the theoretical framework underpinning the study.
- **Chapter 3 – Research Methodology:** Discusses the research design, data collection methods, analysis techniques, and ethical considerations crucial for conducting the taxonomy study and gathering insights.
- **Chapter 4 – Findings and Discussion:** Presents empirical findings from surveys and interviews, analyzes data to identify common challenges, best practices, and success factors, and discusses implications for practitioners and the industry.
- **Chapter 5 – Conclusion , Recommendations & Future Work :** Summarizes key findings, contributions, and actionable recommendations for organizations aiming to integrate Agile and DevOps. It suggests future research directions based on study findings and limitations.

Chapter 2

Background & Literature Review

This chapter is divided into two parts; The first part provides a general background of the concepts needed to understand the rest of this research and covers the most important techniques used in implementing Agile DevOps integration. It covers basic concepts of it in software engineering and concepts related to the chosen technique which is Triangulation.

The second part provides different related works and studies. The second part is divided into two sections, in section 2.1 we will review some related work about Agile DevOps and integration studies in Software Development, in section 2.2 we will give some conclusions about the second part.

2.1 Agile and DevOps in Practice

While Agile and DevOps methodologies offer significant benefits for software development, successful implementation requires careful planning and adaptation to organizational contexts. A study by Hoda, Salleh, and Grundy (2018) examining Agile adoption in a large financial institution highlights the challenges of scaling Agile practices across multiple departments. The study found that fostering collaboration between development and operations teams through cross-functional teams was crucial for overcoming communication silos. This aligns with the findings of Kim, G., Debois, P., Willis, J., Humble, J., & Allspaw, J. (2016) who emphasizes the importance of

breaking down silos between development and operations in their *The DevOps Handbook:*. " *How to Create World-Class Agility, Reliability, & Security in Technology Organizations*. IT Revolution Press ".

Additionally, a case study by Bass, L., Weber, I., & Zhu, L. (2015). *DevOps:" A Software Architect's Perspective"*. Addison-Wesley Professional. Explores how a software company utilized DevOps practices to achieve faster delivery times. The case study emphasizes the importance of continuous integration and continuous delivery (CI/CD) pipelines in automating software builds and deployments, leading to more frequent releases and improved software quality.

Here are some specific examples of how Agile and DevOps practices have been successfully implemented in different contexts:

- E-commerce Platform: A rapidly growing e-commerce platform adopted Scrum, a popular Agile framework, to manage its development process. The company established cross-functional teams consisting of developers, testers, and product owners. This facilitated closer collaboration and faster response times to customer needs (e.g., Schwaber, K., & Sutherland, J. (2017). Scrum case study.
- Manufacturing Company: A large manufacturing company implemented DevOps practices to improve collaboration between its development and operations teams. The company adopted tools for continuous integration and continuous delivery (CI/CD), allowing for automated testing and deployment of software updates to production environments. This resulted in faster delivery cycles and fewer deployment issues (e.g., Humble, J., & Farley, D. (2010). *Continuous Delivery*).

These examples demonstrate the potential of Agile and DevOps practices to enhance software development processes. However, it's crucial to consider the specific needs and context of an organization when implementing these methodologies.

A. Background

Integrating Agile and DevOps methodologies requires careful planning and implementation. This general background provides an overview of Agile, DevOps, Agile/DevOps Integration, and the research technique of Triangulation in software engineering. Understanding these concepts is essential for comprehending the rest of the research and the techniques used in Agile, DevOps, or Integration practices [27]. Here are some methods and practices to facilitate the integration effectively: Agile-DevOps Integration Methods

Numerous methods and practices facilitate the integration of Agile and DevOps approaches. These include:

- Continuous Integration (CI): Automates the process of integrating code changes from multiple developers into a central repository frequently (e.g., Fowler, 2006). This helps to identify and fix bugs early in the development process, Implements CI/CD pipelines to automate the build, test, and deployment processes. Continuous integration ensures that code changes are regularly merged into a shared repository and tested [27], while continuous delivery automates the deployment of code changes to production environments.

- Continuous Delivery (CD): Enables the seamless deployment of code changes to production environments (e.g., Humble & Farley, 2010). This streamlines the software delivery pipeline and reduces release cycles.
- Infrastructure as Code (IaC): Treats infrastructure as code, allowing for automated provisioning and configuration management (e.g., Morris & Kief, 2016). This ensures consistent infrastructure across development, testing, and production environments, Treats infrastructure as code by using tools like Terraform, Ansible, or Puppet to automate the provisioning and management of infrastructure resources. IaC enables teams to version control, test, and replicate infrastructure changes, leading to greater consistency and reliability.
- Shared tooling and automation: Utilizing common tools and automation scripts for tasks like building, testing, and deploying software promotes consistency and efficiency across development and operations teams (e.g., Kim et al., 2016).
- DevOps culture: Fostering a culture of collaboration, shared responsibility, and continuous improvement is crucial for successful Agile-DevOps integration (e.g., Forsgren, Humble, & Kim, 2018).
- Cross-Functional Teams: Forms cross-functional teams composed of members from development, operations, testing, and other relevant areas. These teams work together throughout the software development lifecycle, promoting collaboration and shared responsibility.
- Feedback Loops: Establishes feedback loops between development, operations, and stakeholders to gather insights and identify areas for improvement. Use monitoring, logging, and analytics tools to track application performance, user behaviour, and operational metrics [27].

- Toolchain Integration: Integrates Agile and DevOps tools to enable seamless communication and collaboration across teams. Use tools like Jira for project management, Jenkins for CI/CD, Docker for containerization, and Kubernetes for orchestration to streamline the development and deployment process.
- Continuous Learning and Improvement: Encourages a culture of continuous learning and improvement by providing training, resources, and support for team members. Invests in professional development opportunities and encourages knowledge sharing within the organization to stay updated with industry best practices and technologies.
- Metrics and Measurement: Defines key performance indicators (KPIs) to measure the effectiveness of Agile DevOps integration, such as deployment frequency, lead time, mean time to recovery, and customer satisfaction. Use these metrics to track progress, identify bottlenecks, and drive continuous improvement initiatives.
- Treats infrastructure as code: by using tools like Terraform, Ansible, or Puppet to automate the provisioning and management of infrastructure resources. IaC enables teams to version control, test, and replicate infrastructure changes, leading to greater consistency and reliability.

2.1.1 Agile DevOps Integration in Software Engineering

Definition: Integration involves combining different components or systems to work together seamlessly as a unified whole [22]. Include:

- Key Concepts:

- System Integration: Combining individual software components or modules into a

complete system.

- Data Integration: Ensuring that data flows smoothly between different systems or databases.
- Process Integration: Streamlining workflows and processes across different systems or teams.
- Service Integration: Integrating services or APIs to enable communication and interaction between different applications or systems.

- Techniques:

- Service-Oriented Architecture (SOA): Designing software components as reusable services that can be easily integrated and combined [22].
- Message Brokers: Middleware components that facilitate communication and data exchange between different systems.
- Enterprise Service Bus (ESB): A middleware solution that enables communication and integration between different applications or services.

- Triangulation:

- Definition: Triangulation is a research technique that involves using multiple methods, data sources, or perspectives to validate findings and enhance the credibility of research results [13].

- Key Concepts:

- Multiple Data Sources: Gathering data from different sources such as interviews, surveys, observations, and documentation.
- Methodological Triangulation: Using multiple research methods or approaches to investigate a research question or hypothesis.
- Data Triangulation: Comparing and cross-referencing data collected through different methods to identify patterns or discrepancies.

2.1.2 Triangulation in Software Engineering Research

Triangulation is a research methodology that involves using multiple data collection methods to investigate a phenomenon. In the context of software engineering research on Agile-DevOps integration, triangulation offers several benefits:

- Reduced Bias: By employing multiple research methods, such as surveys, interviews, and code analysis, triangulation helps to mitigate potential biases inherent in any single method. For instance, surveys may be susceptible to social desirability bias, where participants report what they believe is the expected answer. Combining surveys with interviews allows researchers to gain a deeper understanding of the participants' experiences and perspectives, reducing the impact of such biases (e.g., Patton, 1990).
- Improved Data Validation: Triangulation allows researchers to compare findings from different methods to see if they converge. This convergence strengthens the validity and reliability of the research results. For example, if survey data suggests that developers experience improved collaboration after adopting Agile practices, and interview data with developers confirms this perception, the research findings gain greater weight (e.g., Denzin, 1978).
- Richer Picture: Combining data from various sources paints a more holistic picture of the research topic. For instance, complementing survey data with in-depth interviews with developers can provide deeper insights into the practical experiences of Agile/DevOps integration, revealing unforeseen challenges or unexpected benefits (e.g., Creswell, 2013).

- **Application in Software Engineering:**

Therefore, triangulation is a valuable tool for researchers studying Agile-DevOps

integration, allowing for a more comprehensive and nuanced understanding of this complex phenomenon.

Triangulation can be used in software engineering research to gather insights from various stakeholders, validate research findings, and ensure the reliability and validity of research outcomes. For example, in the context of Agile, DevOps [13], or Integration research, triangulation could involve collecting data through interviews, surveys [22], and observations to gain a comprehensive understanding of how these methodologies are implemented and their impact on software development processes.

2.1.3 Algorithmic Method

In the context of Agile, DevOps, and integration, there isn't a specific algorithmic method in the traditional sense. However [32], certain algorithmic approaches and techniques can be applied within these methodologies to optimize processes, automate tasks, and improve efficiency. Here are a few examples:

- **Algorithmic Decision Making:**

- In Agile and DevOps environments, teams may use algorithms to support decision-making processes. For example, algorithms can be used to prioritize user stories or features based on factors such as business value, complexity, and dependencies.
- Optimization algorithms: can be applied to tasks such as resource allocation, scheduling, and workload balancing. For instance, teams may use algorithms to optimize the allocation of development tasks across team members or to schedule automated tests and deployments.
- Machine learning algorithms: can be employed to analyze data collected from software development and operations processes [32]. For example, teams may use machine learning algorithms to predict software defects, identify performance bottlenecks, or

optimize infrastructure resource usage.

- Search algorithms: can be used to automate tasks such as code search, dependency resolution, and configuration management. For instance, teams may use search algorithms to quickly locate relevant code snippets, libraries, or documentation within a codebase [32].
- Graph algorithms: can be applied to tasks such as dependency analysis, impact analysis, and network optimization. For example, teams may use graph algorithms to visualize and analyze dependencies between software components, identify potential risks, or optimize the flow of work through a development pipeline [32].

2.1.4 Function Point Analysis (FPA)

Is a technique used in software engineering to measure the size and complexity of a software system based on the functionality it delivers to users. While FPA is not inherently an algorithmic method, it involves a structured approach to quantifying software functionality, which can be integrated into Agile, DevOps, and integration practices. Overall, Function Point Analysis can be a valuable tool in Agile, DevOps, and integration practices for estimating, planning, measuring productivity, conducting impact analysis, and optimizing resource allocation. By quantifying the functional size of software systems, teams can make more informed decisions and improve the efficiency and effectiveness of their software development and delivery processes. Here's how Function Point Analysis can be applied in these contexts:

- Estimation and Planning in Agile:

- In Agile development, accurate estimation of user stories or features is essential for planning and prioritizing work [22]. FPA can be used as a technique for estimating the

size and effort required to implement user stories. By quantifying the functional size of user stories in terms of function points, teams can more accurately estimate the time and resources needed for implementation.

- Integration with Project Management Tools:

- FPA can be integrated with project management tools used in Agile and DevOps environments to facilitate estimation, tracking, and reporting of function points [22]. By integrating FPA data with Agile project management tools such as Jira or DevOps tools such as Jenkins, teams can streamline their workflows and improve visibility into project progress.

2.1.5 Non-Algorithmic Methods

Refer to techniques or approaches that do not involve explicit algorithms or computational procedures. In the context of Agile, DevOps, and their integration. Non-Algorithmic Methods complement Agile, DevOps, and integration practices by fostering collaboration, communication, transparency, and continuous improvement [2]. By embracing these techniques, teams can optimize their processes, enhance productivity, and deliver value to customers more effectively [2]. There are several non-algorithmic methods that can be applied to improve processes, collaboration, and efficiency. Here are some examples:

- Collaborative Workshops:

Collaborative workshops bring together stakeholders, team members, and subject matter experts to discuss and brainstorm solutions to complex problems. These workshops foster collaboration, creativity, and shared understanding, helping teams to identify innovative

ideas and solutions.

- Value Stream Mapping:

  Value Stream Mapping (VSM) is a Lean technique used to visualize and analyze the flow of work through a system. In Agile and DevOps environments, teams can use VSM to identify bottlenecks, waste, and inefficiencies in their processes, and to streamline workflows for improved efficiency and value delivery.

- Pair Programming:

  Pair programming is an Agile practice where two developers work together on the same task, sharing a single workstation. This practice promotes collaboration, knowledge sharing, and code quality, leading to improved software craftsmanship and reduced defects.

- Daily Stand-up Meetings:

  Daily stand-up meetings, also known as daily scrums, are a key Agile ceremony where team members come together to discuss progress, challenges, and plans for the day. These meetings promote transparency, communication, and alignment within the team, helping to identify obstacles early and keep projects on track.

- Cross-Functional Teams:

  Cross-functional teams bring together individuals with diverse skills and expertise to work on a common goal. In Agile and DevOps environments, cross-functional teams enable faster decision-making, reduce handoffs and delays, and foster a sense of ownership and accountability for delivering value.

- Continuous Improvement (Kaizen):

  Continuous Improvement, or Kaizen, is a philosophy and practice focused on making incremental improvements to processes, products, and services over time. In Agile and DevOps environments, teams embrace Kaizen by regularly reflecting on their practices, identifying areas for improvement, and implementing changes to increase efficiency and quality.

- Feedback Loops:

  Feedback loops are mechanisms for gathering feedback from customers, users, and stakeholders to inform decision-making and improve processes. In Agile and DevOps environments, teams establish feedback loops through techniques such as user testing, retrospectives, and post-incident reviews, enabling continuous learning and adaptation.

- Visual Management:

  Visual management techniques, such as Kanban boards and task boards, provide a visual representation of work in progress, priorities, and dependencies. These visual tools help teams to prioritize tasks, track progress, and identify bottlenecks, facilitating effective communication and collaboration.

2.2 Learning Oriented Models

Refer to approaches or frameworks that prioritize learning and knowledge acquisition as integral parts of the software development process [8]. These models emphasize continuous learning, experimentation, and adaptation to improve outcomes and drive innovation. In the context of Agile, DevOps, and integration, learning-oriented models

play an important role in fostering a culture of continuous improvement and knowledge sharing within teams. Here are some examples of learning-oriented models:

- Lean Software Development:

  Lean Software Development is a learning-oriented model based on Lean principles borrowed from manufacturing. It emphasizes the elimination of waste, amplification of learning [8], and empowerment of individuals. Lean encourages teams to focus on delivering value to customers, minimizing lead times, and optimizing processes through experimentation and reflection.

- Lean Startup:

  The Lean Startup model applies Lean principles to the process of building and launching new products or services. It advocates for rapid iteration, validated learning, and customer feedback to iteratively develop and refine ideas. Lean Startup encourages teams to build minimum viable products (MVPs), test hypotheses with real users, and pivot or persevere based on feedback.

- Cynefin Framework:

  The Cynefin Framework is a sense-making model that helps teams understand the nature of problems they encounter and choose appropriate approaches for addressing them. It categorizes problems into four domains: simple, complicated, complex, and chaotic. In complex and chaotic domains, where outcomes are uncertain, the Cynefin Framework advocates for experimentation, emergent practices, and learning through probing and sensing.

These learning-oriented models provide frameworks and approaches for fostering a

culture of continuous learning, experimentation, and improvement within Agile, DevOps, and integration practices. By prioritizing learning and knowledge acquisition, teams can adapt to change more effectively, innovate more rapidly, and deliver greater value to customers.

- Genetic Algorithms (GAs)

Genetic algorithms (GAs) are a type of evolutionary algorithm inspired by natural selection. They are a potential tool for optimizing software development processes, including aspects relevant to Agile-DevOps integration. Here's a brief overview:

- Core principles: GAs mimic the process of natural selection, where populations of individuals (potential solutions) evolve over generations. Individuals compete for survival based on their fitness (effectiveness in solving the problem). The fittest individuals are selected for reproduction, with their characteristics (genetic material) combined to create new offspring. These offspring inherit and potentially improve upon the traits of their parents.
- Application in Agile-DevOps Integration: GAs have been explored for various optimization tasks within Agile-DevOps contexts. For instance, they could be used to:
- Optimize test case selection: By simulating different testing scenarios and identifying the most effective combinations, GAs could help improve test coverage and efficiency.
- Resource allocation: GAs could be used to optimize resource allocation across development teams, considering factors like workload, skillsets, and project dependencies.

However, the application of GAs in Agile-DevOps integration is still an emerging area of research. Further investigation is needed to determine their effectiveness and practical implementation strategies.

Optimize algorithms inspired by the process of natural selection and evolution. They are used to find approximate solutions to optimization and search problems by mimicking the process of natural selection in biological organisms. Genetic Algorithms offer a flexible and powerful approach to optimization problems, and their application in Agile, DevOps, and integration can help teams streamline processes, improve efficiency, and achieve better outcomes in software development and delivery [39]. However, it's essential to carefully design and tailor Genetic Algorithms to specific problem domains and objectives to ensure effective results. In the context of Agile, DevOps, and integration, Genetic Algorithms could potentially be applied in various ways to optimize processes and improve outcomes:

- Resource Allocation and Scheduling:

  Genetic Algorithms can be used to optimize resource allocation and scheduling in Agile and DevOps environments. For example, GAs could be applied to allocate development tasks to team members based on their skills, availability, and workload, optimizing team productivity and efficiency.

- Test Case Generation:

  Genetic Algorithms could be employed to automatically generate test cases for software testing. By evolving sets of test cases over multiple generations, GAs can help maximize code coverage and identify edge cases and potential defects more effectively than manual test case generation [39].

- Feature Selection and Prioritization: In Agile development, Genetic Algorithms can aid in feature selection and prioritization. By evaluating different combinations of features and their impact on project objectives, GAs can help teams identify the most valuable features to include in each iteration or release, optimizing product development efforts.

- Configuration Management: Genetic Algorithms could be used for configuration management in DevOps environments. For example, GAs could optimize the configuration of cloud infrastructure or deployment pipelines by evolving configurations over time based on performance metrics and requirements.

- Optimization of Continuous Integration/Delivery Pipelines: Genetic Algorithms can optimize continuous integration and delivery pipelines by evolving pipeline configurations to minimize build times, reduce deployment failures, and maximize resource utilization. By iteratively refining pipeline configurations, GAs can help teams achieve faster and more reliable software delivery.

- Code Optimization: Genetic Algorithms can be applied to code optimization tasks, such as optimizing code performance, reducing code complexity, or minimizing resource usage. By evolving code representations and evaluating their fitness based on predefined objectives, GAs can help improve the quality and efficiency of software code.

2.3 Benchmarking

Benchmarking is a process of comparing an organization's practices against industry leaders or established best practices. It provides valuable insights into areas for improvement within Agile-DevOps integration.

- Benefits: Benchmarking helps organizations:

        - Identify performance gaps compared to industry leaders.
        - Learn from successful Agile-DevOps implementations in other companies.
        - Set realistic goals for improving their own software development processes.
- Implementation: Organizations can implement benchmarking through various methods:
        - Industry reports: Analyzing reports from research firms or industry associations that benchmark Agile-DevOps practices.
        - Case studies: Studying successful Agile-DevOps implementations in other companies through case studies or industry publications.
        - Self-assessment tools: Utilizing self-assessment tools to evaluate an organization's current Agile-DevOps maturity level.

By effectively utilizing benchmarking, organizations can gain valuable insights and accelerate their journey towards optimized Agile-DevOps integration. Consider including a real-world example of an organization that has successfully benchmarked its Agile-DevOps practices [20].

Is a systematic process of comparing and evaluating the performance, quality, or capabilities of a product, service, or process against industry standards or best practices. Benchmarking provides a structured approach to evaluating and improving software development and delivery processes in Agile, DevOps, and integration contexts. By comparing performance against industry standards or best practices, teams can identify opportunities for improvement and implement changes to achieve better outcomes. In the context of Agile, DevOps, and integration, benchmarking can be used to measure and

improve various aspects of software development and delivery [20]. Here's how benchmarking can be applied in these contexts:

- Process Improvement:

  Benchmarking can be used to compare the performance of Agile or DevOps processes against industry benchmarks or leading practices. By benchmarking key metrics such as cycle time, lead time, and throughput, teams can identify areas for improvement and implement changes to optimize their processes.

- Quality Assurance:

  Benchmarking can help assess the quality of software products or services by comparing them against industry standards or competitors. For example, benchmarking can be used to measure defect rates, customer satisfaction scores, or adherence to coding standards, enabling teams to identify opportunities for quality improvement.

- Infrastructure Performance:

  In DevOps environments, benchmarking can be used to evaluate the performance of infrastructure components such as servers, networks, and databases. By benchmarking key performance indicators such as response time, throughput, and scalability, teams can identify bottlenecks and optimize infrastructure configurations for better performance and reliability.

- Release Management:

  Benchmarking can assist in evaluating the effectiveness of release management processes in Agile and DevOps environments. By benchmarking metrics such as deployment

frequency, mean time to recovery (MTTR), and change failure rate (CFR), teams can assess the efficiency and reliability of their release processes and make improvements as needed.

- Continuous Integration/Delivery Pipelines:

  Benchmarking can be applied to evaluate and optimize continuous integration and delivery pipelines. By benchmarking metrics such as build times, deployment frequency, and pipeline stability, teams can identify areas for optimization and streamline their CI/CD processes for faster and more reliable software delivery [20].

- Team Performance:

  Benchmarking can help assess the performance of Agile teams by comparing their productivity, velocity, and quality metrics against industry benchmarks or high-performing teams. By benchmarking team performance, organizations can identify areas for skill development, training, or process improvement to enhance overall team effectiveness.

2.4 Functional Decomposition

Is a top-down design approach used in software engineering to break down a complex system or problem into smaller, more manageable components or functions. Functional Decomposition is a valuable technique in Agile, DevOps, and integration practices for breaking down complex systems into smaller [29], more manageable units of work. By decomposing systems into functional components, teams can improve agility, scalability, and maintainability, enabling them to deliver high-quality software more efficiently. Each component represents a specific task or function that contributes to the overall

functionality of the system. Functional Decomposition is commonly used in Agile, DevOps, and integration practices to facilitate modular design, code reuse, and maintainability. Here's how Functional Decomposition can be applied in these contexts:

- Agile User Stories:

  In Agile development, Functional Decomposition is used to decompose user stories into smaller, actionable tasks or sub-stories. Each task represents a specific functionality or feature that can be implemented and tested independently. Functional Decomposition helps Agile teams break down complex user stories into smaller, more manageable units of work, enabling incremental development and faster delivery.

- Modular Design in DevOps:

  In DevOps environments, Functional Decomposition is used to design modular and reusable components that can be easily integrated and deployed. By decomposing complex systems into smaller, cohesive modules, DevOps teams can reduce dependencies, improve scalability, and facilitate continuous integration and delivery. Functional Decomposition also enables teams to parallelize development efforts and implement changes more efficiently.

- Microservices Architecture: Functional Decomposition is a fundamental principle in microservices architecture, where large monolithic applications are decomposed into smaller, independently deployable services [29]. Each microservice represents a specific business function or capability, and communication between microservices is typically achieved through APIs. Functional Decomposition enables organizations to build scalable, resilient, and maintainable systems that can evolve independently over time.

- Integration of Systems and Components: Functional Decomposition is used to identify and define the interfaces between systems and components in integration projects. By decomposing complex systems into smaller functional units, integration teams can identify integration points, define data exchange formats, and establish communication protocols. Functional Decomposition helps ensure interoperability, compatibility, and seamless integration between disparate systems.

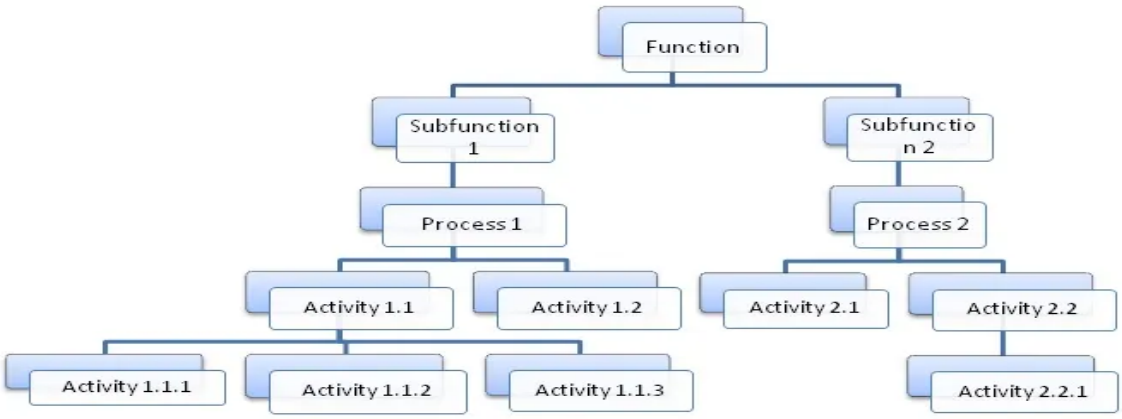

Figure 2.1. Functional Decomposition. [Reprinted from] Functional Decomposition Analysis [Website]. thefunctionalba.com, December 12, 2019. Retrieved June 22, 2024, from https://thefunctionalba.com/2019/12/functional-decomposition- analysis/

- Test Case Design: Functional Decomposition is used in test case design to identify and prioritize test scenarios based on functional requirements. By decomposing the system into smaller functions or components, testers can develop test cases that cover specific functionalities or use cases. Functional Decomposition helps ensure comprehensive test coverage and facilitates efficient test case prioritization and execution.

2.5 Brainstorming

Brainstorming is a dynamic problem-solving technique essential in Agile and DevOps integration, involving rapid generation of ideas and solutions collaboratively [12]. It fosters an environment where team members freely share thoughts and insights, aiming to explore diverse perspectives and possibilities without judgment. This approach is pivotal in Agile and DevOps contexts, facilitating innovation and effective problem-solving that involves generating a large number of ideas or solutions to a problem in a short amount of time. It's a collaborative process where participants share their thoughts, suggestions, and insights without judgment, aiming to explore a wide range of possibilities and perspectives [12]. Brainstorming is a versatile and effective technique for generating ideas, solving problems, and driving innovation in Agile, DevOps, and integration practices. By incorporating brainstorming into their collaborative processes, teams can tap into the collective intelligence and creativity of their members, enabling them to achieve their goals more effectively and efficiently.

In the context of Agile, DevOps, and integration, brainstorming can be a valuable tool for generating innovative ideas, identifying solutions to challenges, and fostering collaboration among team members. Here's how brainstorming can be applied in these contexts:

- Integration Strategy Development:

When planning integration projects, teams can use brainstorming to explore different integration strategies, technologies, and architectures. By brainstorming ideas and solutions, integration teams can identify opportunities to streamline data exchange, improve interoperability, and enhance system performance. Brainstorming sessions help

integration teams align on goals, priorities, and implementation approaches, fostering consensus and commitment to the integration strategy.

- Continuous Improvement Initiatives:

Brainstorming can be used to drive continuous improvement initiatives in Agile, DevOps, and integration practices. By regularly convening brainstorming sessions, teams can identify areas for improvement [12], such as process bottlenecks, tooling inefficiencies, or communication challenges. Brainstorming helps teams generate ideas for optimization, experimentation, and innovation, empowering them to evolve and adapt their practices over time.

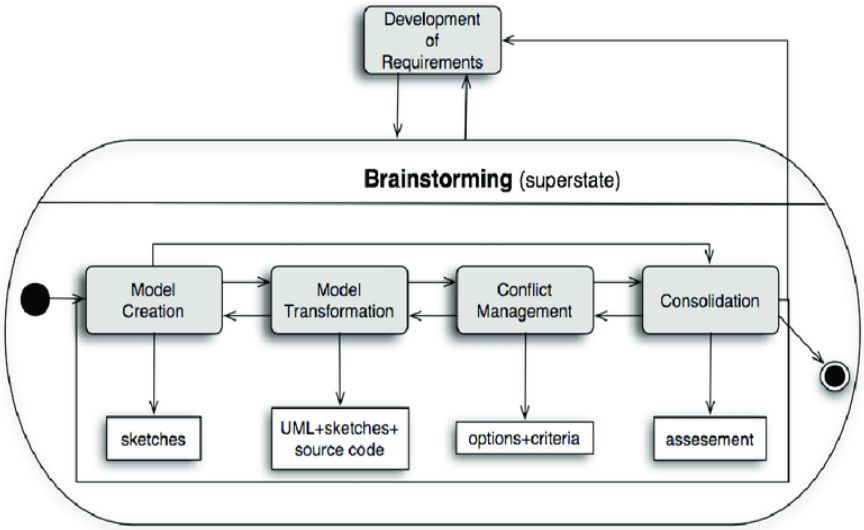

Figure 2.2. Conceptual model for brainstorming and software design activities (UML Statechart diagram). Adapted from Boulila, N. (2005). A Framework for Distributed Collaborative Software Design Meetings.

- Innovation and Ideation:

Brainstorming can be a powerful tool for fostering innovation and generating new ideas in Agile, DevOps, and integration contexts. By creating a supportive and inclusive environment where all ideas are welcome, teams can unleash creativity and explore

unconventional approaches to solving problems [12]. Brainstorming sessions help teams break free from conventional thinking and challenge assumptions, leading to breakthroughs and novel solutions.

2.6 Literature Review

The software development landscape has witnessed a profound transformation over the past few decades. Traditional approaches to software development, characterized by long development cycles and rigid processes, have given way to Agile methodologies, DevOps practices, and the scaling of Agile frameworks. This chapter provides a comprehensive review of relevant literature, examining the key principles, challenges, and best practices associated with Agile, DevOps, and Agile scaling frameworks.

2.6.1 Introduction to Agile and DevOps

Agile methodologies, rooted in principles of collaboration, flexibility, and continuous improvement, have showcased their effectiveness in diverse contexts and industries [6]. Eltayeb [20] highlighted the significant impact of cultural factors on the success or failure of Agile implementation within Sudanese software development. Similarly, Ali and Baskerville [3] underscored the pivotal roles of organizational culture and leadership in successful Agile adoption in Pakistani software development organizations. Leadership, effective team dynamics, communication, collaboration, training, and support emerged as critical factors influencing the success of Agile adaptation [15, 13]. The literature underscores the efficacy of Agile methodologies in enhancing team productivity and project outcomes, while emphasizing their susceptibility to contextual influences such as cultural nuances and leadership styles.

Agile methodologies represent a paradigm shift in software development. Agile

principles, as outlined in the Agile Manifesto [1], prioritize customer collaboration, flexibility, and iterative development. Agile teams work closely with stakeholders, deliver working software frequently, and respond to changing requirements. Several Agile frameworks, including Scrum, Lean, and Kanban, have emerged to guide development teams in their Agile journey [1].

Agile practices have gained widespread popularity in the software development industry, with a focus on improving project outcomes and team productivity. However, their adaptation and implementation can vary significantly depending on organizational context and culture [20]. The literature highlights several critical factors that influence the success of Agile adoption.

2.6.2 Organizational Culture and Leadership

Research by Eltayeb (2018) emphasizes the significant impact of organizational culture on the success of Agile implementation [20].

Organizations with a culture that promotes collaboration, open communication, and adaptability tend to achieve better results when adopting Agile practices. Effective leadership is also crucial in guiding teams through the transition to Agile [20].

2.6.3 Training and Support

Effective training and ongoing support are essential for successful Agile adoption [6]. Teams and individuals need the necessary knowledge and skills to implement Agile practices effectively.

Continuous learning and improvement play a pivotal role in Agile methodologies [6].

2.6.4 Team Dynamics

Effective collaboration and team dynamics are central to Agile practices [6]. Agile teams are self-organizing and cross-functional, requiring strong interpersonal relationships and communication skills among team members. Team performance metrics, such as sprint velocity and burndown charts, are used to assess progress [6].

2.6.5 Customer-Centric Approach

Agile methodologies place a strong emphasis on customer satisfaction and collaboration [1]. Teams work closely with customers and stakeholders to understand their needs and deliver valuable solutions. This customer-centric approach aims to ensure that the delivered software aligns with business objectives [1].

2.6.6 Success and Failure Factors in Agile Software Development Projects

Success factors identified include strong management commitment, agile-friendly organizational and team environments, high-Caliber team capability, strong customer involvement, appropriate project management processes, methodical project definition processes, agile-style software engineering techniques, correct delivery strategies, non-life-critical project natures, variable-scope project types, and dynamic, accelerated project schedules [33][6][13][9][17] The research employed a web survey method to collect data from members of the Agile Alliance and its user groups, yielding insights into both success and failure factors. Failure factors encompass organizational, people, process, and technical dimensions, while success factors are categorized into organizational, people, process, technical, and project-related factors. This comprehensive review provides valuable insights for practitioners and researchers in Agile and DevOps implementations.

2.7 DevOps Practices

Implementing DevOps practices has demonstrated substantial benefits, including accelerated delivery times, improved software quality, and enhanced team collaboration [12]. However, Gorschek et al. [11] highlighted the significant challenge posed by the need for cultural and organizational change in successful DevOps and Agile implementation. Communication and collaboration among teams, particularly in distributed development environments, emerged as key hurdles [13]. Proposed strategies for enhancing DevOps and Agile implementation encompass fostering a strong DevOps culture, investing in automation tools, emphasizing communication and collaboration, and utilizing metrics for effectiveness measurement and improvement identification [24, 25]. The convergence of DevOps and Agile methodologies requires meticulous attention to organizational and cultural aspects, alongside fostering effective communication and collaboration across teams.

DevOps represents a fusion of development and operations practices, aiming to streamline software delivery and improve collaboration between development and IT operations teams [12]. DevOps practices emphasize automation, continuous integration, continuous delivery, and continuous monitoring. Several key elements characterize the DevOps approach:

2.7.1 Culture and Collaboration

DevOps is more than a set of tools; it's a cultural shift [11]. Building a DevOps culture that encourages collaboration, shared goals, and transparency is essential. Collaboration between development and operations teams breaks down traditional silos, leading to faster and more reliable software delivery [11].

2.7.2 Automation

Automation is a cornerstone of DevOps practices [12]. By automating repetitive tasks, organizations reduce manual errors and accelerate delivery times. Automation tools, such as configuration management and deployment pipelines, play a crucial role in achieving these goals [12].

2.7.3 Continuous Integration and Continuous Delivery (CI/CD)

CI/CD pipelines enable the automated building, testing, and deployment of software changes [12]. These pipelines ensure that code changes are thoroughly tested and can be rapidly deployed to production environments. The adoption of CI/CD practices contributes to shorter release cycles and improved software quality [12].

2.7.4 Monitoring and Feedback

Continuous monitoring of applications and infrastructure allows organizations to detect and address issues promptly [12]. DevOps teams use monitoring tools to gain real-time insights into system performance and user behaviour. This feedback loop enables rapid responses to incidents and performance optimizations [12].

2.8 Scaling Agile Frameworks

As organizations increasingly seek to apply Agile practices across multiple teams and projects, the necessity for scaling Agile frameworks has become apparent [1]. Studies underline the challenges intrinsic to scaling Agile practices, particularly concerning communication, collaboration, and alignment with organizational goals [12]. Frameworks like SAFe, LeSS, and DA offer approaches to scaling Agile practices, each with its unique strengths and adaptation strategies [1, 27]. However, the successful

implementation of scaling Agile practices demands comprehensive strategies addressing communication, collaboration, cultural change, and alignment with organizational values [12]. Further research is essential to comprehensively understand the challenges and best practices involved in implementing scaling Agile frameworks.

Scaling Agile practices to encompass multiple teams and projects has become increasingly important for organizations seeking to apply Agile principles at a broader scale. Frameworks like the Scaled Agile Framework (SAFe), Large-Scale Scrum (LeSS), and Disciplined Agile (DA) provide guidance for organizations in this endeavour.

2.8.1 Scaled Agile Framework (SAFe)

SAFe is one of the most widely adopted frameworks for scaling Agile practices [1]. SAFe emphasizes communication, coordination, and alignment of work toward common goals across multiple teams, programs, and organizations. While SAFe offers a structured approach to scaling Agile, organizations must adapt it to their specific context and needs [1].

2.8.2 Large-Scale Scrum (LeSS)

LeSS focuses on creating self-organizing teams that collaborate effectively to achieve project goals [1]. It simplifies the scaling of Scrum practices by retaining Scrum's core principles while removing unnecessary elements. LeSS aims to reduce complexity and increase transparency in large-scale Agile implementations [1].

2.8.3 Disciplined Agile (DA)

DA provides a flexible framework that allows organizations to tailor Agile practices to their specific needs and contexts [1]. It emphasizes pragmatism and encourages teams to adopt the right practices for their unique situations. DA provides guidance on various aspects of Agile, including architecture, design, programming, testing, and more [1].

2.9 Integration of Agile and DevOps

The integration of Agile and DevOps practices has gained attention as organizations seek to optimize their software development processes further. Both approaches share common principles, such as collaboration and automation, making them complementary [12]. Numerous case studies and analyses delve into specific Agile frameworks' applications and challenges. These studies shed light on the effectiveness of frameworks like SAFe, LeSS, and DA in diverse organizational settings, emphasizing their impact on project outcomes, challenges encountered, and success factors [28, 12]. For instance, SAFe has demonstrated significant effectiveness in improving project outcomes, yet challenges persist regarding cultural change and coordination among multiple Agile teams [1, 29]. Moreover, studies provide insights into the challenges faced during large-scale Agile adoption, emphasizing integration difficulties, change resistance, and the pivotal role of management support [12]. Additional studies propose risk-based methods and practices within Agile frameworks, aiming to balance agility and discipline in software development [13].

2.9.1 Key Factors in Integration

Effective integration of Agile and DevOps practices requires a focus on cultural alignment, communication, and automation [12]. Organizations should foster a culture of collaboration and shared goals between development and operations teams. Additionally, automation plays a pivotal role in achieving the seamless integration of development and operations processes [12].

2.9.2 Benefits of Integration

The integration of Agile and DevOps practices can lead to faster delivery times, higher software quality, and improved collaboration between teams [12]. By breaking down silos

and emphasizing automation, organizations can achieve a more efficient and reliable software delivery pipeline [12].

2.9.3 Regional Variations

Agile methodologies, originally designed for small team settings, undergo substantial transformations when extended to larger project scopes [26, 29]. The scalability of Agile introduces synchronization challenges among multiple development teams and diverse organizational units [12]. Despite these challenges, Agile methodologies maintain momentum due to foundational principles emphasizing collaboration and iterative software development approaches [29]. The literature review surrounding Agile adaptation and implementation offers crucial insights across varied organizational contexts [15, 14]. Cultural factors significantly influence the success or failure of Agile implementation [28, 29]. Effective leadership, team dynamics, communication, collaboration, comprehensive training, and support are identified as pivotal factors for successful Agile adoption and implementation [15, 13]. However, there remains a notable dearth of research on Agile practices' adaptation and implementation within DevOps contexts [12].

2.9.4 Theoretical Framework

A comprehensive understanding of Agile practices emerges from various perspectives within the literature. For instance, resource [11] provides a comprehensive overview of Extreme Programming (XP), showcasing its values, practices, and risk mitigation strategies within Agile methodologies. XP embodies iterative development, emphasizing effective adaptation to change through social and personal transformation [11]. Additionally, studies delve into the challenges encountered during Scrum adoption, offering pragmatic advice to overcome hurdles [39]. They prioritize practical solutions

over theoretical discussions, addressing real-world issues faced during Scrum implementation [39]. Moreover, the evolution of roles within Agile frameworks, particularly the Product Owner role in SAFe, underscores the modifications and their implications for staffing Product Owners in large-scale Agile environments [32]. The insights provided by these studies enrich discussions around Agile methodologies, emphasizing challenges, success factors, and the necessity for empirical research to bridge the practitioner experience-academic evidence gap within Agile and DevOps contexts [22, 30, 32].

The integration of Agile and DevOps practices has garnered significant attention as organizations strive to optimize their software development processes further. Both Agile and DevOps approaches share common principles, such as collaboration and automation, making them complementary in nature.

Numerous case studies and analyses delve into specific Agile frameworks' applications and challenges, shedding light on their effectiveness in diverse organizational settings. For instance, the Scaled Agile Framework (SAFe), Large-Scale Scrum (LeSS), and Disciplined Agile (DA) provide guidance for organizations in scaling Agile practices across multiple teams and projects.

2.10 Conclusion

This chapter has explored the theoretical underpinnings of Agile and DevOps methodologies, along with their practical applications in software development. We have also discussed the benefits of integrating these approaches and various methods to facilitate this integration. Additionally, the chapter has introduced triangulation as a valuable research methodology and provided a brief overview of genetic algorithms and benchmarking practices relevant to Agile-DevOps integration.However, successful

integration of Agile and DevOps practices requires meticulous attention to organizational and cultural aspects. Effective integration strategies encompass fostering a strong DevOps culture, investing in automation tools, emphasizing communication and collaboration, and utilizing metrics for effectiveness measurement and improvement identification.

By breaking down silos and emphasizing automation, the integration of Agile and DevOps practices can lead to faster delivery times, higher software quality, and improved collaboration between teams. The convergence of Agile and DevOps methodologies represents a pivotal step towards achieving a more efficient and reliable software delivery pipeline.

In summary, the integration of Agile and DevOps practices offers significant potential for enhancing software development processes. By aligning cultural values, fostering collaboration, and leveraging automation tools, organizations can achieve seamless integration and reap the benefits of accelerated delivery and improved software quality.

Chapter 3

Methodology

The methodology unfolds across several sections. Illustrated in Figures 3.1 how we implement this research. Thus, this chapter is split into five sections. Section 3.1, commencing with a comprehensive exploration of Agile, DevOps and their Integration practices. Section 3.1 focuses on taxonomy studies and understanding prevalent frameworks and practices. Section 3.2 entails gathering insights from diverse industry sources, collecting data from experts, professionals, previously implemented projects in software companies to pilot the selected technique and forums to gauge real-world applications. Following this, Section 3.3 involves meticulous analysis, extracting patterns and critical factors influencing the success of Agile and DevOps at scale. Subsequently, Section 3.4 ensures ethical compliance, emphasizing informed consent, confidentiality measures, data security, and objectivity in the study. Finally, Section 3.5, We searched for the best technique, framework, practices, and factor of success suitable for projects in software companies at scale.

To achieve the study objectives, a meticulously crafted research methodology is implemented, structured across multiple sections as depicted in Figures 3.1. Initially, a systematic literature review (SLR) is conducted to identify the success factors (CSFs) associated with Agile/DevOps integration in software organizations. Subsequently, these identified SFs are mapped to the (CAMSOPPRTPT) criteria, which represent the fundamental principles of Agile/DevOps integration. To validate this mapping and assess the relationship between factors and CAMSOPPRTPT criteria, a comprehensive questionnaire survey is administered. Finally, the PROMETHEE-II method is utilized to

rank the CSFs based on their importance for Agile/DevOps integration.

Building upon this methodology, Chapter 3 is divided into several sections. Section 3.1, survey design, commences with an in-depth exploration of Agile, DevOps, and Integration practices, focusing on taxonomy studies and understanding prevalent frameworks and practices. Section 3.2, data collection, involves gathering insights from various industry sources, including experts, professionals, and previous software projects, to pilot selected techniques and gather real-world applications. Following data collection, Section 3.3, data analysis, conducts detailed analysis to extract patterns and identify critical factors influencing the success of Agile and DevOps at scale. Ensuring ethical standards, Section 3.4, select appropriate techniques, emphasizes informed consent, confidentiality, data security, and objectivity throughout the study. Additionally, the selected technique for the experiment, the Triangulation technique, will be applied within Section 3.4. Finally, Section 3.5, PROMETHEE II, explores and identifies the best techniques, frameworks, practices, and success factors suitable for large-scale software projects in business environments.

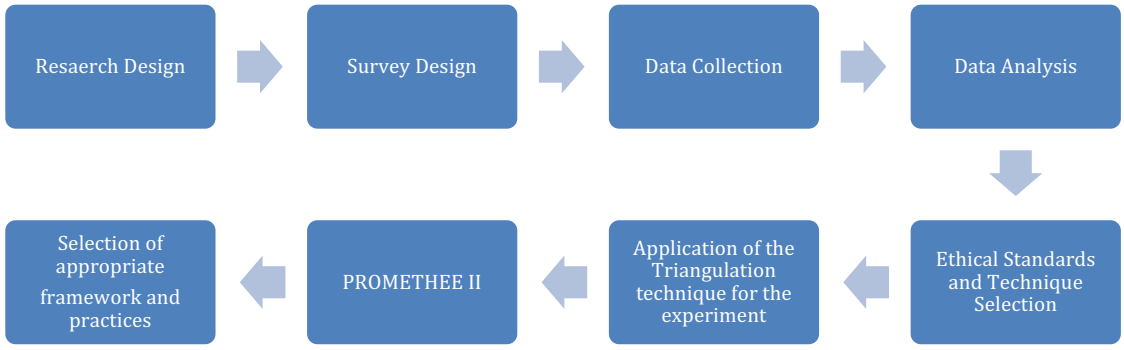

Figure 3.1: Methodology Implementation mode

3.1 Research Design

3.1, survey design, commences with an in-depth exploration of Agile, DevOps, and Integration practices, focusing on taxonomy studies and understanding prevalent frameworks and practices.

3.1.1 Mixed-Methods Approach

The research methodology integrates qualitative and quantitative methods to explore Agile and DevOps implementation comprehensively. This approach is crafted to incorporate taxonomy study techniques and frameworks geared towards effecting organizational change within DevOps environments. By blending both qualitative and quantitative methodologies, this research seeks to glean insights from varied perspectives, ensuring a comprehensive understanding of Agile and DevOps practices.

3.1.2 Qualitative and Quantitative Methodologies

Qualitative Methods: This includes conducting semi-structured interviews and examining project documentation to extract nuanced insights. The aim is to capture the experiences, challenges faced, and strategies adopted in Agile and DevOps implementations.

Quantitative Methods: Data collection through survey among software development professionals will be employed. This survey will be designed to encapsulate metrics and key performance indicators identified in the literature as crucial for assessing the success and challenges of Agile/DevOps integration practices.

3.2 Data Collection

In Section 3.2, titled data collection, we initiated the process of gathering insights from a variety of industry sources, including experts, professionals, and past software projects, with the aim of piloting selected methodologies and acquiring real-world applications. Subsequently, our focus shifted towards the collection of data pertaining to previously executed software projects, sourced from various companies, to facilitate the application of the triangulation technique and ascertain its suitability for software projects. We reached out to multiple companies to solicit data on their past projects; while some responded and provided the requested information, others chose to withhold data due to privacy concerns.

The amassed software projects exhibit diversity across multiple dimensions, including project size, conceptualization, programming languages employed, and the organizational scale of the companies involved, encompassing startups, outsourcing firms, and both local and international enterprises. The collected data encompasses various aspects such as project requirements, developer involvement, utilized techniques, project duration, and other pertinent information essential for the triangulation technique analysis.

3.2.1 Sources of Data

- Interview: Conducting semi-structured interview with stakeholders, practitioners, and industry experts will provide rich qualitative insights into the implementation and adaptation of Agile and DevOps methodologies.
- Survey: Deploying a survey among software development professional to obtain quantitative data on the adoption, challenges faced, and key performance metrics used to gauge success in Agile and DevOps initiatives.

- Analysis of Software Company Datasets:

  In addition to interview and survey, empirical data from software development companies was analysed. This dataset provided valuable insights into real-world implementations of Agile and DevOps practices. Key aspects analyzed included:

  - Trends in Agile and DevOps adoption over time.
  - Usual challenges faced by software development companies.
  - Best practices observed in successful Agile and DevOps implementations.

3.2.2 Survey Design

Understanding the diverse techniques and methodologies employed globally is pivotal for refining approaches or introducing innovative strategies to advance software project management. A comprehensive survey was conducted worldwide through Google Forms 2023 to assess prevalent methodologies used in Agile and DevOps implementations. The questionnaire was thoughtfully designed to gather extensive insights, combining closed and open-ended questions to encourage detailed responses from participants across the globe. Emphasis was placed on maintaining brevity, ensuring the survey could be completed in under ten minutes to maximize participation and gather accurate and valuable data. The survey was broadly distributed across influential IT forums and industry-specific platforms to capture a diverse range of insights.

Table 3.1 outlines the survey questions, each tailored to extract specific insights into the methodologies used globally in Agile and DevOps implementation practices. The questionnaire was thoughtfully designed to capture insights into the following aspects:

- Frameworks: Preferences and experiences with various frameworks like SAFe, LeSS, DEAM, ITIL, and Kanban.
- Best Practices: Identification of best practices, strategies, and methodologies in Agile and DevOps implementations.
- Success Factors: Metrics used to measure success, including cycle time, deployment frequency, defect density, team performance, ROI, and scalability.
- Challenges in Scaling: Insights into challenges and solutions faced while scaling Agile and DevOps projects.
- Cultural Adaptation: Understanding the role of cultural alignment and collaboration in successful implementations.
- Security and Compliance: Practices and strategies concerning security and compliance within Agile and DevOps contexts.
- Tools and Technologies: The utilization of tools, technologies, and training approaches.

Table 3.1: Some of Survey questions and objectives

Id	Question	Objective
1	What is your current job title in the software development domain?	To understand the roles and designations within software development teams.
2	How many years of experience do you have in software development?	To gauge the level of experience among professionals in the field.
3	How do you perceive the adoption and integration of Agile and DevOps practices in your organization?	To capture perceptions regarding the utilization of Agile and DevOps methodologies.
4	What challenges have you faced in implementing Agile or DevOps?	To identify common hurdles and challenges encountered during implementation.
5	How effective do you consider Agile and DevOps practices in enhancing software development?	To assess the perceived effectiveness of Agile and DevOps methodologies.

6	What frameworks or methodologies have you found most beneficial in Agile or DevOps adoption?	To gather insights on preferred frameworks and methodologies in Agile and DevOps environments.
7	How do you measure the success of Agile or DevOps implementations within your projects?	To understand the metrics or indicators used to measure success in Agile and DevOps projects.
8	What metrics related to cycle time, deployment frequency, defect density, team performance, ROI, and scalability do you use to assess the success of Agile and DevOps implementations?	To explore the specific metrics used to measure the success of Agile and DevOps.
9	How do you ensure continuous improvement in Agile and DevOps practices within your team?	To explore approaches and strategies employed for continual improvement in Agile and DevOps.
10	What do you perceive as critical success factors in scaling Agile and DevOps practices?	To identify factors deemed crucial for successful scaling of Agile and DevOps methodologies.
11	How would you rate the collaboration between development and operations teams in your organization?	To assess the level of collaboration between Dev and Ops teams in the software development cycle.
12	Are you familiar with frameworks such as SAFe, LeSS, DEAM, ITIL, or Kanban for scaling Agile and DevOps projects?	To understand the awareness and usage of frameworks for scaling Agile and DevOps methodologies.
13	How do you perceive the applicability and effectiveness of these frameworks in your organization?	To gauge opinions on the suitability and effectiveness of scaling frameworks in practice.
14	Do you face any challenges in aligning Agile and DevOps with your organizational culture?	To understand the challenges in aligning methodologies with existing organizational culture.
15	How do you prioritize Agile and DevOps practices in your software development life cycle?	To determine the significance placed on Agile and DevOps practices within the SDLC.
16	Have you encountered any cultural resistance while implementing Agile or DevOps practices?	To gauge the impact of cultural resistance on Agile and DevOps adoption.
17	How do you handle security	To understand the approach towards integrating

	considerations in Agile and DevOps workflows?	security measures within Agile and DevOps.
18	What tools or technologies do you employ to facilitate Agile and DevOps processes?	To explore the technological infrastructure supporting Agile and DevOps practices.
19	How do you address skill gaps or training needs for Agile and DevOps proficiency within your teams?	To understand strategies adopted for skill development in Agile and DevOps methodologies.
20	What role does leadership play in fostering Agile and DevOps adoption and success within your organization?	To gauge the influence of leadership in promoting Agile and DevOps methodologies.
21	Can you briefly provide a Description of your project.	To provide a brief overview of the software project being considered in terms of its purpose, functionality, or goals.
22	What practices that your company implement? DevOps/Agile or integration Practices.	To identify the specific DevOps or Agile practices implemented or utilized within the project.
23	What Technology Stack did you implement in your software design industry?	To list the technologies or tools utilized in the software project's development, including programming languages, databases, frameworks, etc.
24	What was the complexity level of the project?	To assess the perceived complexity level of the software project based on its scope, technical intricacies, or challenges encountered during development.
25	How did you demonstrate adaptability in the project?	To evaluate the project's adaptability or flexibility in accommodating changes, scaling, or future enhancements.
26	What are the risk factors involved?	To identify and quantify the level of risk associated with the software project, considering potential challenges or uncertainties that could affect its success.
27	What was the duration of your last development project?	To specify the duration or timeline taken for the development of the software project from inception to completion.

3.3 Data Analysis

Our analysis delved into the data collected globally from diverse software companies, aiming to extract pertinent information essential for implementing energy efficiency techniques. This comprehensive examination of data allowed us to identify patterns and critical factors influencing the success of Agile and DevOps at scale. The outcomes of our analysis were summarized in Table 3.2, which provides insights into 16 projects obtained from these companies. It's important to acknowledge that detailed data beyond what is presented in the table was not included due to the lack of permission from the companies.

Additionally, within Section 3.3, we employed various analytical methodologies to ensure a thorough investigation. These approaches encompassed both qualitative and quantitative analyses. Qualitative analysis involved thematic analysis of interview transcripts and document analysis to uncover recurring patterns, key themes, and critical success factors influencing Agile and DevOps implementations. On the other hand, quantitative analysis utilized statistical tools to analyze survey data, examining trends, correlations, and significant metrics related to Agile and DevOps effectiveness, team productivity, and project success. By integrating these analytical methodologies, Section 3.3 provided comprehensive insights into the factors contributing to the success of Agile and DevOps implementations.

3.3.1 Analytical Approaches

- Qualitative Analysis: Utilizing thematic analysis on interview transcripts and document analysis to identify recurring patterns, key themes, and critical factors influencing Agile and DevOps success.

- Quantitative Analysis: Employing statistical tools to analyze survey data, examining trends, correlations, and significant metrics related to Agile and DevOps effectiveness, team productivity, and project success.

Table 3.3 shows information on the projects collected from software projects companies. The number of projects is 12.

Table 3.2: Dataset for software projects companies.

Project Description	DevOps/Agile / integrations Practices	Technology Stack	Complexity Level	Adaptability	Risk Factor	Development Duration
Digital Asset Management (DAM) System Integration	Agile	PHP, MySQL	High	Yes	10%	1 year
Enterprise Resource Planning (ERP) Implementation	Agile	Flutter cross platform	High	Yes	15%	3 months
Document Management System (DMS) Enhancement	DevOps	Netcore, MongoDB	Low	Yes	25%	2 years
Real Estate Property Management Software Upgrade	Integration	ASP.NET MVC5, SQL Server	High	Yes	10%	4 months
Inventory Forecasting and Planning Tool Development	Integration	VueJS, Netcore	Medium	Yes	15%	6 months
Legal Case Management System Deployment	Integration	VueJS, Netcore	Medium	Yes	10%	9 months
Project Management Tool Integration with Time Tracking Software	DevOps	PHP, MySQL	Medium	No	10%	5 months
Hospitality Reservation System Upgrade	Agile	ASP.NET MVC5, SQL Server	High	No	10%	3 months
Cloud-based Collaboration	Integration	PHP, MySQL	Medium	No	15%	2 months

Platform Upgrade						
Workforce Management Software Deployment	Agile	Joomla PHP	High	No	10%	3 months
Transportation Logistics Management System Optimization		ASP.NET MVC5, SQL Server	High	No	10%	8 months
Mobile Banking Application Development	Agile	C#	High	No	20%	1 year
Hospitality Reservation System Upgrade	Agile	Java, PostgreSQL, MySQL, Oracle, Spring Boot (Java-based framework), MongoDB database), Flask (Micro web framework), SQL Server (Relational database	Medium	Yes	15%	5 months
Asset Tracking and Management Software Development	DevOps	ASP.NET MVC5, PostgreSQL	Medium	No	15%	1 year
CRM Integration with Marketing Automation Software	DevOps	Java, PostgreSQL, MySQL, Oracle, Spring Boot (Java-based framework), MongoDB database), Flask (Micro web framework), SQL Server (Relational database	High	No	20%	1 year
Facility Management System Integration	DevOps	Java, PostgreSQL, MySQL, Oracle, Spring Boot (Java-based framework), MongoDB database), Flask (Micro web framework), SQL Server (Relational database	Medium	Yes	15%	5 months

In order to enhance the methodological robustness and analytical depth of our thesis, we propose a combined utilization of triangulation and the PROMETHEE II approach. This

integrated methodological framework will afford us the opportunity to meticulously cross-validate and fortify our findings by synthesizing insights derived from disparate sources and analytical lenses. Triangulation, as a cornerstone of methodological rigor, serves to enhance the veracity and reliability of our research outcomes by harmonizing evidence from multiple sources or methodological approaches. Concurrently, the PROMETHEE II approach, renowned for its systematic and objective evaluation of alternatives, will empower us to discern and prioritize key factors with precision, thus enriching our understanding of complex decision-making contexts. Through the judicious application of these methodologies, our thesis endeavors to achieve a comprehensive and nuanced analysis, underpinned by academic rigor and methodological integrity.

3.4 Ethical Considerations

Section 5.4, select appropriate techniques, emphasizes informed consent, confidentiality, data security, and objectivity throughout the study. Additionally, the selected technique for the experiment, the Triangulation technique, will be applied within Section 3.4.

- Ensuring Ethical Compliance

    - Informed Consent: Ensuring participants' informed consent for interviews and surveys, prioritizing voluntary participation and confidentiality.
    - Confidentiality Measures: Safeguarding the anonymity of participants and ensuring the confidentiality of sensitive information collected during the study.
    - Data Security: Implementing robust data security measures to protect all collected data, in adherence to privacy regulations and ethical guidelines.
    - Objectivity and Bias Mitigation: Striving for objectivity during data analysis, disclosing any potential biases, and ensuring an unbiased interpretation of

findings to maintain research integrity.

3.5 Selection of Appropriate Agile DevOps integration Techniques

In this section, after thoroughly gathering and analyzing data from various software projects across different companies, it's essential to choose the right method for our study. We've decided to use the Triangulation technique. This choice is based on several important factors related to our data. For example, our data includes key aspects like Front-end, Back-end, and Quality Assurance (QA), which are pivotal for web projects but may not be considered by other methods. Also, the Triangulation technique works well with Agile methods, allowing us to break down tasks into smaller parts instead of estimating the whole project at once. This flexibility is important for projects that evolve over time. Additionally, the Triangulation technique requires a careful and thorough approach, which addresses concerns from project managers and software engineers about the complexity of other methods for estimating time accurately.

The decision to use the Triangulation technique is supported by a survey that looked at many factors contributing to project success. It showed that no single method is guaranteed to be better than others because each method affects different factors. These factors include how clear and easy to use the method is, how it deals with uncertainty, whether it fits with agile methods, how well it adapts to changes in requirements, and how reliable its results are. Considering all these factors, the Triangulation technique seems like the best choice for our study, especially when working with data from various companies.

3.6 Implementation of Triangulation Technique within Agile-DevOps Integration: A Strategic Framework.

This section delineates the strategic utilization of the triangulation technique within the dynamic domain of Agile-DevOps integration, considering the critical success factors identified in our study's hypotheses.

- Framework Overview:

This framework aligns the triangulation technique with the identified critical success factors across various dimensions of Agile-DevOps integration. By amalgamating research findings with strategic hypotheses, it presents a structured approach to decision-making in selecting and optimizing Agile and DevOps methodologies.

Variable Explanation Aligned with Study Relevance:

- Estimated Time:

The anticipated effort for project completion, influenced by agile project management processes (H6) and methodical project definition processes (H7), directly impacts project timelines and costs.

- Number of Developers and QA Engineers:

Aligned with H4, a team of high caliber influences project quality, scope, time, and cost. Resource allocation reflects the agile-friendly project team environment (H3) and organizational commitment (H2).

- Actual Working days:

Reflecting the significance of strong management commitment (H1), organizational alignment with Agile-DevOps principles is essential for optimizing work hours and maximizing productivity.

- Acceptance Test Factor and Risk Factor:

Considered within the context of agile-friendly organizational and project team environments (H2, H3), these factors mitigate risks and ensure quality deliverables within specified scope, time, and cost constraints.

- Output Explanation Harmonized with Study Relevance:

The framework's outputs are synthesized through our study's hypotheses, enhancing decision-making by providing actionable insights into Agile-DevOps integration. By correlating empirical data from surveys, interviews, and software company datasets with the identified critical success factors, it offers practical guidance for practitioners and organizations seeking to optimize their integration efforts.

Related Table: Table 3.3 presents simulations of the proposed framework and evaluates various Agile/DevOps frameworks, illustrating how the triangulation technique and critical success factors interact within Agile-DevOps integration strategies.

Benefits of Using Triangulation Technique for Framework Selection at Scale:

- Comprehensive Evaluation:

The triangulation technique facilitates a comprehensive assessment of various scaling frameworks by considering multiple factors such as estimated time, resource allocation,

and risk factors. This holistic approach ensures that all relevant aspects of framework suitability are considered.

- Objective Decision-Making:

By quantifying key parameters such as developer hours, QA man-hours, and project management overhead, the triangulation technique enables objective decision-making in framework selection. This minimizes bias and ensures that decisions are based on empirical data rather than subjective opinions.

- Adaptability to Scale:

The framework's flexibility allows for seamless adaptation to different project scales. Whether it's a small-scale initiative or a large-scale enterprise deployment, the triangulation technique can accommodate varying project sizes and complexities, providing scalable solutions.

- Risk Assessment and Mitigation:

Through the inclusion of risk factors in the evaluation process, the triangulation technique helps in identifying potential risks associated with each framework. This allows project managers to proactively assess and mitigate risks, thereby enhancing the likelihood of successful framework implementation at scale.

- Cost-Efficiency Analysis:

By calculating developer and QA man-hours, as well as considering reduction factors, the triangulation technique offers insights into the cost-efficiency of each framework. Project stakeholders can compare the estimated costs of implementing different

frameworks and choose the most cost-effective option for scaling projects.

Table 3.3: Simulation of Proposed Framework and Evaluation for Agile/DevOps and integration Frameworks

Aspect	Simulation Framework	Scrum	Kanban	Lean	SAFe	Nexus	DAD
Scalability	High	High	Moderate	High	Very High	High	Very High
Flexibility	High	High	High	High	Moderate	Moderate	Very High
Integration	Moderate	Moderate	High	High	Very High	Very High	Very High
Community Support	Strong	High	Moderate	Moderate	Very Strong	Strong	Moderate
Adoption Rate	High	High	Moderate	Moderate	Very High	Moderate	Moderate
Cost-Efficiency	High	Moderate	Moderate	High	Moderate	High	High
Risk Assessment and Mitigation	Moderate	Moderate	Moderate	High	Very High	High	High
Agile Practices	Iterative Development, Scrum Events, Continuous Integration	Iterative Development, Kanban Board, Continuous Delivery	Value Stream Mapping, Continuous Improvement, Kaizen	Scrum Events, SAFe Framework	Scrum Events, Scaled Scrum, Continuous Integration	Continuous Delivery, Disciplined Agile Delivery	Continuous Integration, Test-Driven Development, Pair Programming
DevOps Practices	Continuous Integration, Continuous Deployment, Infrastructure as Code	Continuous Delivery, Automated Testing, Cross-functional Teams	Infrastructure as Code, Automated Deployment, Monitoring and Logging	Continuous Integration, Continuous Deployment, Release Automation	Continuous Integration, Continuous Deployment, Automated Testing	Continuous Integration, Continuous Delivery, Automated Deployment	Continuous Integration, Continuous Deployment, Infrastructure as Code
Integration Practices	Cross-functional Collaboration, Shared Goals, Automation	Collaborative Workflows, Visual Management, Feedback Loops	Value Stream Optimization, Waste Elimination, Cross-functional Collaboration	Alignment of Agile and DevOps Principles, Shared Metrics, Release Trains	Cross-functional Teams, Scrum of Scrums, Integration Points	Integration Frameworks, Shared Codebase, Continuous Integration	Continuous Integration, Infrastructure Automation, DevOps Culture

In the context of our framework, the term "simulation framework" denotes the theoretical model or approach utilized to simulate and assess various scenarios or configurations within the Agile-DevOps integration process. It serves as a conceptual framework aiding in the comprehension of how diverse factors, such as team composition, resource allocation, and project management practices, might influence the overall success of Agile-DevOps endeavors.

This simulation framework empowers project managers and decision-makers to model different scenarios, predict outcomes, and evaluate potential impacts before implementing changes in real-world settings. It acts as a strategic planning tool, offering insights into the potential ramifications of adopting different Agile/DevOps methodologies or practices.

In essence, the simulation framework offers a structured method for analyzing and optimizing Agile-DevOps integration efforts, enabling organizations to make well-informed decisions grounded in theoretical simulations and strategic hypotheses.

Ultimately, the triangulation technique provides a robust and systematic approach to selecting and identifying the optimal framework at scale. Through comprehensive evaluation, objective decision-making, adaptability to scale, risk assessment and mitigation, and cost-efficiency analysis, it empowers project managers to make informed decisions aligned with organizational objectives and to maximize project success.

3.7 Implementing PROMETHEE II Technique

In Section 5.7, the PROMETHEE II methodology is thoroughly examined to identify optimal techniques, frameworks, practices, and success factors applicable at scale within software project business environments.

Originally introduced by Brans et al. [22] in 1985, the PROMETHEE (Preference Ranking Organization Method of Enrichment Evaluation) approach has evolved into various versions, including PROMETHEE I, II, III, IV, V, cluster, among others, each tailored to specific research problems. Notably, PROMETHEE II stands out for its ease of application and low complexity rate, rendering it suitable for diverse decision analysis

tasks across multiple domains. Its effectiveness has been demonstrated in emergency management scenarios, where multicriteria decision-making (MCDM) challenges necessitate efficient and timely solutions, as well as in fields such as chemistry and healthcare, where intricate logical analyses are essential [34][4]. Moreover, PROMETHEE II finds utility in logistic management and information technology, offering an interactive platform for classifying and ordering complex alternatives [37]. Central to PROMETHEE II is its pairwise comparison of alternatives based on predefined criteria, as highlighted by Behzadian et al. [38]. This method allows for comprehensive evaluation, enabling decision-makers to discern subtle distinctions among alternatives, albeit requiring careful consideration of criteria weights and understanding of outranking relationships and fuzzy variable scales [39].

The procedural framework for implementing the PROMETHEE-II approach, comprising seven distinct steps as depicted in Figure 3.2, is elucidated in detail, ensuring clarity and guidance for its application in decision-making contexts.The seven steps (Figure 3.2) required performing PROMETHEE-II approach are discussed as follows:

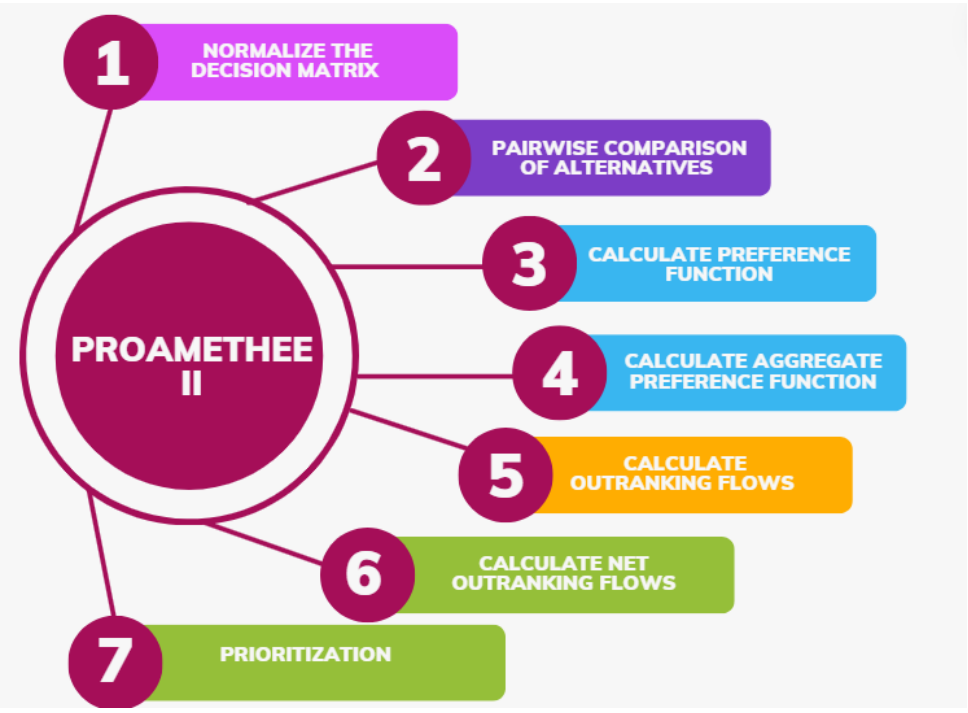

Figure 3.2 : PROMETHEE-II approach.

Step 1. The decision matrix is normalized using beneficial and nonbeneficial criteria: Brans, J.P., Vincke, P., & Mareschal, B. (1986).[16].

$$D_{ij} = \frac{\left[M_{ij} - \min M_{ij}\right]}{\left[\max\left(M_{ij}\right) - \min\left(M_{ij}\right)\right]} \text{(Beneficial Criteria)}, \quad (1)$$

- where i = 1, 2, 3, …, n and j = 1, 2, 3, …, m

$$D_{ij} = \frac{\left[\max M_{ij} - M_{ij}\right]}{\left[\max\left(M_{ij}\right) - \min\left(M_{ij}\right)\right]} \text{(Non Beneficial Criteria)}, \quad (2)$$

where Mij is a decision-maker's evaluation of ith alternative with respect to jth criterion.

Step 2. "The difference of ith alternative as compared with other ones are determined in this step" [104]. This means that pairwise comparison of each alternative with criteria should be assessed.

Step 3. "In this step, we choose and calculate the preference function Pj (i, i′)" [104].

- According to Brans et al. [6], there are 6 various types of preference functions (ranging from 0 to 1). Figure 4 illustrates these preference functions. The PROMETHEE approach induces a preference function to describe the decision-maker's preference difference between pairs of alternatives on each criterion. It is possible to choose a different function for each criterion. For this study, we have chosen type-1 preference function. This function requires no parameters and thresholds. The type-1 preference function is:

$$P_j\left(i, i'\right) = 0 \text{ if } \quad \text{Dij} \le \text{Di}' j, \quad (3)$$

$$P_j\left(i, i'\right) = 1 \quad \text{Dij} \le \text{Di}' j. \quad (4)$$

Step 4. In this step, we will calculate aggregate preference function by combining weights.J.P., Vincke, P., & Mareschal, B. (1994).[15].

$$\pi\left(i,i'\right)=\frac{\sum_{j=1}^{m}P_{\mathrm{ij}}\left(i,i'\right)w_j}{\sum_{j=1}^{m}w_j},\tag{5}$$

where is the weight of relative importance given to (jth) criterion.

Step 5. "We determine the outranking flow (negative and positive) for each alternative compared with (p-1) alternatives" [105] as shown in Figure 4, and the formulas used to calculated leaving and outranking flow are

$$\text{the entering flow:}\varphi^{+}(i)=\frac{1}{n-1}\sum_{i'-1}^{N}\pi\left(i',i\right),\quad\left(i\neq i'\right),\tag{6}$$

$$\text{the leaving flow}:\varphi^{-}(i)=\frac{1}{n-1}\sum_{i'-1}^{N}\pi\left(i,i'\right),\left(i\neq i'\right),\tag{7}$$

where N represents the number of alternatives. "The entering flow measures the weakness of alternatives and leaving flow measures the strength of alternatives" [106].

Step 6. In this step, we calculate the net outranking flow of all alternatives.

$$\text{The net flow outranking:}\varphi(i)=\varphi^{+}(i)-\varphi^{-}(i).\tag{8}$$

Step 7. "To calculate the ranking of alternatives, consider values of net outranking flow the highest the value, the best the alternative" [6]. The PROMETHEE-II also avoids incomparability between alternatives.

Chapter 4

Findings and Discussion

In this chapter, we delve into the experiments and their corresponding results, alongside the methodology employed to validate these findings. Section 4.1 elucidates the outcomes of our survey, providing insights into our research inquiries. Following this, Section 4.2 meticulously details the application of the triangulation technique, delineating the projects involved and the resultant findings. Concurrently, Section 4.3 offers a comprehensive examination of the implications of the triangulation technique, delving into the projects engaged and the outcomes derived from its application. Finally, in Section In Section 4.4, we undertake the implementation of the PROMETHEE II approach, elucidating its methodology and presenting the ensuing results, thereby enriching our understanding of decision-making frameworks within our research context.

4.1 Survey Results

In Section 4.1, Survey Results, we present summarized findings gleaned from the survey. The detailed results of the survey are available in the appendices, where comprehensive tables provide an in-depth exploration of the collected data. Here, we offer condensed summaries of the survey results, elucidating key insights and trends observed from the analysis.

We have reached nearly 53 people who have filled out the survey on Google Forms. Here are results for those who filled out the survey:

4.2 Implementing Triangulation Technique

In this section, I will explain the details of the application of triangulation technique. I did not get approval from the companies to mention data details about the software projects that I obtained, so I will explain the details of triangulation technique in this section.

4.2.1 Inputs

Table 4.1: Input Forms for Selecting a Suitable Framework at Scale

Input	Level of Details
Agile Manifesto	High
DevOps Principles	High
Business Requirements	Moderate
Technical Specifications	High
Request for Proposal (RFP)	Low

Description:

In the context of selecting a suitable framework at scale within Agile-DevOps integration, various inputs play a crucial role in defining the project's scope, objectives, and requirements. These inputs serve as the foundation for identifying the most appropriate framework that aligns with the organization's goals and objectives.

- Agile Manifesto: The Agile Manifesto outlines the core principles and values of Agile software development. With a high level of detail, it provides insights into Agile methodologies, emphasizing customer collaboration, iterative development, and responding to change.
- The Agile Manifesto serves as a guiding framework for organizations adopting Agile practices within the context of DevOps integration.

DevOps Principles: Similarly, DevOps principles offer a high level of detail regarding the integration of development and operations teams, emphasizing collaboration, automation, and continuous delivery. Understanding these principles is essential for organizations seeking to streamline their software delivery processes and achieve seamless Agile-DevOps integration.

- Business Requirements: Business requirements provide a moderate level of detail regarding the project's business objectives, user needs, and functional specifications. While not as detailed as the Agile Manifesto or DevOps principles, business requirements play a crucial role in defining the scope and direction of the project, guiding the selection of an appropriate framework.
- Technical Specifications: Technical specifications offer a high level of detail regarding the project's technical requirements, including architecture, infrastructure, and system integrations. These specifications help align the selected framework with the organization's technological infrastructure and ensure compatibility with existing systems and tools.

At the outset, I will elucidate the various inputs essential for projects of diverse natures. These inputs serve as the initial blueprint of the project, outlining its objectives, goals, and requirements, which are communicated to software companies to convey the project's essence.

- These inputs manifest in different forms, namely a Request for Proposal (RFP), a Tender, Business Requirements, or a Technical Specification. Table 4.7 delineates these inputs for projects, showcasing their level of detail. The clarity of these inputs directly correlates with the accuracy of effort estimates, with higher clarity yielding more precise estimates due to the lucidity of project details.

It's briefly listed into these concepts:

- Request for Proposal (RFP): An RFP comprehensively delineates project details, enabling potential vendors to submit proposals for the development process and technology solutions. This detailed document facilitates the comparison of vendor proposals and allows companies to explore various solutions before making informed decisions [25].
- Business Requirements: Business requirements represent the initial phase of the Software Development Life Cycle (SDLC), capturing end-users' needs to guide future system design. They ensure alignment among stakeholders and team members, serving as the foundation for project development and aiding in budget adherence and timely completion [42].
- Technical Specification: A technical specification is a comprehensive document detailing all technical procedures pertinent to product development. It encompasses vital information about the development process, akin to a product development bible [43].

In summary, selecting a suitable framework at scale within the context of Agile-DevOps integration requires careful consideration of various inputs, ranging from high-level Agile and DevOps principles to more specific business and technical requirements. By evaluating these inputs based on their level of detail and relevance to the organization's goals, stakeholders can make informed decisions that optimize software development processes and drive business value.

4.2.2 Steps to Get the Estimated Time by Triangulation

Table 4.2: Triangulation Main Steps for Selecting Framework at Scale

Steps	Note
Functional Decomposition	Missing items are a big risk
Running Triangulation Technique	Verify accuracy and consistency of estimates
Benchmarking (optional)	Compare against industry standards if available
Getting The Total	Summing up all estimated components
Multiply by a Risk Factor	10% - 35%
Record Assumptions	Document underlying assumptions

Description:

In the context of selecting a framework at scale within Agile-DevOps integration, the Triangulation technique offers a structured approach to evaluate the suitability of different frameworks. These steps facilitate a systematic and comprehensive approach to selecting the most suitable framework within the Agile-DevOps integration process, providing stakeholders with valuable insights for decision-making.

The following steps outline the key stages of this process:

- Functional Decomposition: This initial stage involves breaking down inputs, such as business requirements or technical specifications, into smaller, manageable items. By dissecting the project into these smaller components, it becomes easier to understand the scope and requirements of the project.
- Running Triangulation Technique: The Triangulation technique involves the participation of three software engineers with varying levels of seniority. Each engineer selects a playing card based on the Fibonacci rule, and scenarios are

created based on these selections to evaluate the potential effectiveness of different frameworks.

- Benchmarking (Optional): While not mandatory, benchmarking involves comparing the effectiveness of different frameworks with industry standards or similar projects. This optional step provides valuable insights into the feasibility and effectiveness of various frameworks, assisting stakeholders in making informed decisions.
- Getting The Total: Once the potential effectiveness of each framework is evaluated through the Triangulation technique, stakeholders assess the overall suitability of each framework based on the collective input from the engineers.
- Record Assumptions: Assumptions made during the evaluation process are documented to ensure transparency and enable adjustments if assumptions change over time.

4.2.3 Triangulation Technique Results

In this section, we will present the results of applying the triangulation technique to 12 software projects from various software companies in Table 4.3.

The results of verifying the accuracy of the triangulation technique and its extension within the Agile and DevOps implementation context at scale are presented in Table 6.5. This table displays the Completion Days / Hours, framework selection, resource allocation, and project success factors obtained through the triangulation technique and its extension, comparing them with the actual outcomes of the Agile and DevOps projects at scale.

Table 4.3: Dataset of Triangulation Technique Verification Results at Scale

Project Description	Completion Days / Hours	Resource Allocation	Project Success Factors	Framework	Items Added (During Implementation)
Web application for scalable e-commerce platform	167 Hours	High	Strong	SAFe	Yes
Mobile app for real-time delivery tracking	120 Hours	Moderate	High	Kanban	Yes
Integration solution for IoT devices and cloud platform	119 Hours	High	Moderate	Lean	Yes
Enterprise resource planning system for manufacturing	61 Days	High	High	Scrum	Yes
Online collaboration platform for remote teams	45 Days	Moderate	Very Strong	SAFe	Yes
Data analytics dashboard for financial institutions	48 Days	Moderate	Moderate	Scrum	Yes
Customer relationship management (CRM) software	50 Days	Moderate	Moderate	Kanban	Yes
Project management tool for Agile teams	40 Hours	High	Strong	Lean	Yes
Cloud-based document management system	241 Hours	High	Strong	Nexus	Yes
Social media analytics platform	16.5 Hours	Moderate	Very Strong	DAD	Yes
Crpton Reception, Application to receive patients	36 Days	Moderate	High	SAFe	Yes
Crpton Mobile App, an application for managing clinics	33 Days	Moderate	Moderate	Kanban	Yes

These results demonstrate the effectiveness of the triangulation technique and shows the project success factors within the Agile and DevOps implementation context at scale, thereby assisting in the identification of the best frameworks, practices, and factors of success at scale.

Table 4.4: Dataset for Results of the application of the triangulation technique.

Brief Description of the project	Basis of Estimation	Technology Stack	Level of Certainty	Items Added (During Implementation)	Risk Factor Used	Framework
Web application that is using analytical hierarchical process to help people who want to buy mobile phones to compare different options	Mockups, requirements gathering interviews	PHP, MySQL	High	Yes	10%	167 / Hours
Mobile app for university students - Phase I	Design, Technical Specification	Flutter cross platform	High	Yes	15%	120 / Hours
Integration solution with Facebook conversion APIs and Google ads	Business requirements, research	Netcore, MongoDB	Low	Yes	15%	119 / Hours
Mini CRM solution for small businesses	Mockups, database design, requirements gathering interviews	ASP.NET MVC5, SQL Server	High	Yes	10%	61 / Days
Online learning platform for short videos - Phase I	Requirements gathering interviews	VueJS, Netcore	Medium	Yes	15%	45 / Day
Online learning platform for short videos - Phase II	Requirements gathering interviews	VueJS, Netcore	Medium	Yes	10%	48 / Day
Web application for submitting business ideas for evaluation based on Lean canvas	Requirements gathering interviews	PHP, MySQL	Medium	No	10%	32 / Day
Web application for submitting internship project ideas to industry professionals	Workshop	ASP.NET MVC5, SQL Server	High	No	10%	50 / Day
Web application for students to submit mini research papers for review	Requirements gathering interviews	PHP, MySQL	Medium	No	15%	35 / Day
Website for election campaign	Content, iterative feedback on design	WordPress	High	No	10%	40 / Hours
Archiving system for insurance agency	Requirements gathering interviews	ASP.NET MVC5, SQL Server	High	No	10%	241 / Hours
Automated process to generate study groups	Technical specification	C#	High	No	20%	16.5 / Hours
Crpton Reception, Application to receive patients in dental clinics to organize appointments	Requirements gathering interviews	React with Ionic, ASP.NET MVC5, postgresql	Medium	Yes	15%	36 / Day
Crpton Mobile App, an application for managing dental clinics, managing appointments, treatment plans and financial costs for patients	Requirements gathering interviews	React with Ionic, ASP.NET MVC5, postgresql	Medium	No	15%	33 / Day

Table 4.5 : Dataset for Results of the application of the triangulation technique.

Adoption Rate	High	High	Moderate	Moderate	Moderate	Moderate	High	Moderate
Community Support	Strong	High	Very Strong	Strong	Very Strong	Moderate	Very Strong	Strong
Integration	Moderate	Moderate	High	High	High	High	Moderate	High
Flexibility	High	High	Moderate	High	Moderate	Moderate	High	High
Scalability	High	High	Very High	Very High	High	Moderate	High	Very High
Framework	Express.js	Flutter	AWS IoT	SAP ERP	Microsoft Teams	Power BI	Salesforce CRM	Jira	Google Drive
Risk Factor Used	15%	10%	20%	15%	10%	10%	15%	10%	15%
Items Added (During Implementation)	Yes	Yes	Yes	Yes	No	No	Yes	No	Yes
Level of Certainty	High	Medium	Low	High	Medium	High	High	Medium	High
Technology Stack	Node.js, MongoDB	React Native, Firebase	Python, AWS IoT Core	Java, Oracle	Angular, Firebase	Tableau, SQL Server	Salesforce, Heroku	React, Node.js	Google Cloud Platform, MongoDB
Basis of Evaluation	Requirements gathering interviews	Design, Technical Specification	Business requirements, research	Mockups, database design	Requirements gathering interviews	Workshop	Requirements gathering interviews	Design, Technical Specification	Mockups, requirements gathering
Brief Description of the Project	Web application for scalable e-commerce platform	Mobile app for real-time delivery tracking	Integration solution for IoT devices and cloud platform	Enterprise resource planning system for manufacturing	Online collaboration platform for remote teams	Data analytics dashboard for financial institutions	Customer relationship management (CRM) software	Project management tool for Agile teams	Cloud-based document management system

Moderate	Moderate	Moderate
Moderate	High	High
High	Moderate	Moderate
Moderate	Moderate	Moderate
High	Moderate	Moderate
Apache Hadoop	Crpton Reception	Crpton Mobile App
10%	15%	15%
Yes	Yes	No
Medium	Medium	Medium
Python, Django, PostgreSQL	React with Ionic, ASP.NET MVC5, postgresql	React with Ionic, ASP.NET MVC5, postgresql
Business requirements, research	Requirements gathering interviews	Requirements gathering interviews
Social media analytics platform	Crpton Reception, Application to receive patients	Crpton Mobile App, an application for managing

Table 4.5: continue - Dataset for Results of the application of the triangulation technique.

Estimated Time by Triangulation	167 Hours	120 Hours	119 Hours	61 Days	45 Days
Integration Practices	Cross-functional Collaboration, Shared Goals, Automation	Collaborative Workflows, Visual Management, Feedback Loops	Value Stream Optimization, Waste Elimination, Cross-functional Collaboration	Alignment of Agile and DevOps Principles, Shared Metrics, Release Trains	Cross-functional Teams, Scrum of Scrums, Integration Points
DevOps Practices	Continuous Integration, Continuous Deployment, Infrastructure as Code	Continuous Delivery, Automated Testing, Cross-functional Teams	Infrastructure as Code, Automated Deployment, Monitoring and Logging	Continuous Integration, Continuous Deployment, Release Automation	Continuous Integration, Continuous Deployment, Automated Testing
Agile Practices	Iterative Development, Scrum Events, Continuous Integration	Iterative Development, Kanban Board, Continuous Delivery	Value Stream Mapping, Continuous Improvement, Kaizen	Scrum Events, SAFe Framework	Angular, Firebase
Risk Assessment and Mitigation	Moderate	Moderate	Moderate	High	Moderate
Cost-Efficiency	High	Moderate	Moderate	High	No

48 Days	50 Days	40 Hours	241 Hours	16.5 Hours	36 Days	33 Days
Google Cloud Platform, MongoDB	Mockups, requirements gathering	Collaborative Workflows, Visual Management, Feedback Loops	Continuous Integration, Infrastructure Automation, DevOps Culture	Cross-functional Collaboration, Shared Goals, Automation	Continuous Integration, Infrastructure Automation, DevOps Culture	Continuous Integration, Infrastructure Automation, DevOps Culture
Tableau, SQL Server	Salesforce, Heroku	React, Node.js	Google Cloud Platform, MongoDB	Python, Django, PostgreSQL	Continuous Integration, Continuous Delivery, Infrastructure as Code	Continuous Integration, Continuous Delivery, Infrastructure as Code
React, Node.js	Requirements gathering interviews	Design, Technical Specification	Mockups, requirements gathering	Business requirements, research	React with Ionic, ASP.NET MVC5, postgresql	React with Ionic, ASP.NET MVC5, postgresql
High	High	Moderate	Moderate	High	High	High
Moderate	High	No	High	Moderate	High	High

4.3 Verification

The experiments aimed at applying the triangulation technique and evaluating its suitability within the context of Agile and DevOps implementation, specifically in identifying the best framework, practices, and factors of success at scale. Additionally, an extension of this technique was established to enhance and support the implementation of Agile and DevOps methodologies.

The method followed for the triangulation technique involved two main steps. Firstly, various Agile and DevOps projects were gathered, representing different scales and

contexts. The triangulation technique was then applied to these projects, considering factors such as framework selection, best practices adoption, and success factors identification. Secondly, a comparison was made between the actual results and the triangulation technique's results to

assess the accuracy and validity of the technique in identifying the most suitable frameworks, practices, and success factors at scale.

For verifying the accuracy of the triangulation technique extension, it was applied to a subset of projects representing different scales and complexities of Agile and DevOps implementation. The extension focused on refining framework selection, optimizing practices adoption, and enhancing success factors identification. Subsequently, a simulation of this extension was conducted, and the results were compared with the actual outcomes of the projects, particularly focusing on scalability, flexibility, integration, community support, adoption rate, cost-efficiency, and risk assessment.

4.4 Verification of Triangulation Technique

Here, the results of verifying the accuracy of the triangulation technique for the Agile and DevOps projects used in the experiment phase are presented. These projects, representing various scales and contexts, are detailed in Table 4.5. The table displays the results of checking the accuracy of the triangulation technique, with the second column indicating the estimations derived from the technique, and the third column showing the actual outcomes observed in Agile and DevOps implementation.

This table (Table 4.3.1) now represents the dataset of framework verification results within the context of your Agile and DevOps implementation taxonomy study, including selected frameworks, implemented.

Factors that contribute to the selection of the best frameworks in Agile and DevOps implementation at scale include scalability, flexibility, integration, community support, adoption rate, cost-efficiency, and risk assessment and mitigation. These factors play a crucial role in determining the suitability of a framework for achieving successful implementation outcomes.

In our study on Agile and DevOps implementation to identify the best frameworks, practices, and success factors at scale, we prioritize the evaluation of frameworks based on their ability to scale effectively, adapt to changing requirements, integrate seamlessly with existing systems, garner community support, and demonstrate high adoption rates. Additionally, we assess frameworks for their cost-efficiency and effectiveness in risk assessment and mitigation strategies.

By analyzing these factors comprehensively, we aim to provide insights into selecting the most suitable frameworks for Agile and DevOps implementation initiatives at scale. This will enable organizations to make informed decisions that align with their goals and maximize the success of their implementation efforts.

Table 4.6: Dataset of Framework Verification Results - Agile and DevOps Implementation Taxonomy Study.

Brief Description of the Project	Selected Framework	Implemented Practices	Success Factors	Integration Level	Community Support	Adoption Rate	Cost-Efficiency	Risk Assessment	Duration Unit
Web application for mobile phone comparison	SAFe	Continuous Integration, Value Stream Optimization	Collaborative Workflows, Infrastructure as Code	High	Very Strong	High	High	Moderate	Hours
University student mobile app - Phase I	Scrum	Iterative Development, Scrum Events	Cross-functional Collaboration, Continuous Delivery	High	High	High	Moderate	Moderate	Hours
Integration solution with Facebook and Google ads	Lean	Value Stream Mapping, Waste Elimination	Alignment of Agile and DevOps Principles	High	Moderate	Moderate	Moderate	Moderate	Hours
Small business Mini CRM solution	DAD	Continuous Delivery, Disciplined Agile Delivery	Shared Metrics, Release Trains	Very High	Strong	Moderate	High	High	Days
Online learning platform for short videos - Phase I	Nexus	Scrum Events, Scaled Scrum	Cross-functional Teams, Integration Points	High	Very Strong	High	Moderate	Moderate	Days
Online learning platform for short videos - Phase II	Kanban	Continuous Delivery, Automated Testing	Integration Frameworks, Shared Codebase	High	Strong	Moderate	Moderate	Moderate	Days
Business idea submission web app (Lean canvas)	SAFe	Continuous Integration, Infrastructure Automation	Shared Goals, Automation	High	Moderate	Moderate	Moderate	Moderate	Days
Internship project idea submission web app	Scrum	Continuous Integration, Infrastructure as Code	Collaborative Workflows, Visual Management	High	Very Strong	High	Moderate	Moderate	Days

Student research paper submission web app	Lean	Value Stream Optimization, Waste Elimination	Alignment of Agile and DevOps Principles	High	Strong	Moderate	High	Moderate	Days
Election campaign website	Scrum	Iterative Development, Scrum Events	Cross-functional Collaboration, Continuous Integration	Moderate	High	Moderate	Moderate	Moderate	Hours
Insurance agency archiving system	SAFe	Continuous Delivery, Automated Testing	Shared Metrics, Release Trains	High	Very Strong	High	Moderate	Moderate	Hours
Automated study group generation process	DAD	Continuous Integration, Infrastructure Automation	Shared Codebase, Continuous Delivery	Very High	Strong	High	High	Moderate	Hours
Dental clinic patient reception application	Nexus	Scrum Events, Scaled Scrum	Integration Frameworks, Cross-functional Teams	High	Strong	Moderate	Moderate	Moderate	Days
Dental clinic management mobile app	Kanban	Continuous Integration, Automated Deployment	Visual Management, Feedback Loops	High	Strong	Moderate	Moderate	Moderate	Days

4.5 Implementing PROMETHEE II

In the exploration of Critical Success Factors (CSFs) within Agile-DevOps integration, a systematic literature review employing the tollgate approach identified 53 relevant studies. From these, sixteen distinct success factors were delineated. The detailed enumeration of these CSFs and their respective impact percentages within the Agile-DevOps paradigm is elucidated in Table 4.4.1, presenting a comprehensive evaluation of the selected studies.

The concept of Critical Success Factors (CSFs) originated from the seminal work of Rockhart (1979) and has since undergone refinement and widespread adoption within organizational performance assessment [44]. Bullen and Rockhart (1981) define CSFs as the "limited number of areas in which satisfactory results will ensure successful

competitive performance for the individual, department, or organization." In the domain of software development projects, recent studies have also incorporated the CSF methodology. These factors have been observed to encompass fundamental project management techniques (citation number) and the amalgamation of software engineering principles with business strategy (citation number). Moreover, dimensions of CSFs in software projects span the development life cycle, estimation, validation, executive management, project management, and resource and strategic-level planning (citation number). Within the scope of the present study, CSFs are delineated as the indispensable elements necessary for the success of Agile projects.

Notably, there has been a dearth of formal studies specifically focusing on CSFs within Agile software development projects. However, extant literature includes case studies and research theories examining successes or failures/problems in agile implementation and agile software development projects. Exploring both failures and successes in the literature aids in identifying potential CSFs in agile software development projects, as failures offer valuable insights into avoiding critical pitfalls detrimental to project success.

Table 4.7: List of identified CSFs.

Identified Agile and DevOps Integration Critical Success Factors	Percentage (%)
Culture	
"Collaborative and continuous development environment (CSF1)"	53
Automation	
"Continuous meeting schedules, to observe roles and responsibilities of team members (CSF2)"	70
"Deployment automation without having delay with production (SF13)"	36
"Metrics automation for continuous deployment (CSF15)"	49
Measurement	
"Continuous measurement for service failure recovery without delay (CSF4)"	33

“Training of DevOps activities (CSF10)”	85
“Continuous test and integration for assessment (CSF12)”	55
Sharing	
“Top management support (CSF6)”	75
“Customer feedback to improve development (CSF8)”	37
“Configuration management for code and infrastructure (CSF16)”	19
Organizational	
“Committed sponsor or manager (CSF3)”	68
“Facility with proper agile-style work environment (CSF11)”	43
People	
“Team members with high competence and expertise (CSF5)”	58
Process	
“Following agile-oriented configuration management process (CSF14)”	42
Technical	
“Correct integration testing (CSF4)”	90
Project	
“Projects with up-front risk analysis done (CSF9)”	15

List of identified Agile DevOps Integration CSFs [96][106].

4.4.2. Mapping of Agile DevOps integration Critical Success Factors.

We proceeded by organizing the identified Critical Success Factors (CSFs) into eight distinct domains rooted in principles of Agile and DevOps: culture, automation, measurement, sharing, organizational, people, process, technical, and project [20]. This systematic classification was pivotal in facilitating the PROMETHEE-II analysis, aimed at establishing logical relationships and ranking the CSFs concerning their significance in implementing Agile DevOps integration processes, as elaborated in Section 3.

To validate the outcomes of this classification, an interrater reliability analysis was conducted, coupled with a questionnaire survey. The demographic details of the survey participants are outlined in Appendix C. Figure 2 visually presents the finalized mapping of Agile DevOps Integration success factors.

The consensus among survey participants regarding the alignment of CSFs with Agile DevOps principles (CAMSOPPRTPT) is evident in Table 2. Frequency analysis results depicted in Table 2 indicate that a majority of success factors and their corresponding categories surpass the 55% threshold.

Furthermore, the findings underscore that "culture" and "automation" emerge as the most highly rated categories, garnering 91% agreement among survey respondents. Additionally, CSF8, which denotes "customer feedback to improve development," emerges as the top-ranked success factor for DevOps implementation within software organizations, garnering 90% agreement.

Table 4.8: Questionnaire response of survey participants.

S.#	List of challenges	Number of responses ($N = 53$)							
		Positive			Negative			Neutral	
		E.A	A	%	D	E.D	%	N	%
C	Culture	22	27	49	0	0	0	5	5
1	"Collaborative and continuous development environment (CSF1)"	16	21	37	2	2	4	10	10
A	Automation	21	21	42	0	3	3	8	8
2	"Continuous meeting schedules, to observe roles and responsibilities of team members (CSF2)"	26	21	47	0	1	0	5	5
3	"Deployment automation without having delay with production (CSF13)"	16	20	36	0	2	2	15	15
4	"Metrics automation for continuous deployment (CSF15)"	11	22	33	2	3	5	17	17
M	Measurement	21	21	42	0	3	3	8	8
5	"Continuous measurement for service failure recovery without delay (CSF4)"	17	19	36	2	2	4	13	13
6	"Training of DevOps activities (CSF10)"	21	17	38	1	1	2	12	12
7	"Continuous test and integration for assessment (CSF12)"	11	22	33	5	0	5	14	14

8	"Configuration management for code and infrastructure (CSF16)"	15	17	32	3	6	9	13	13
S	Sharing	16	21	37	5	0	5	10	10
9	"Top management support (CSF6)"	16	22	38	3	3	6	15	15
10	"Customer feedback to improve development (CSF8)"	16	21	37	5	0	5	10	10
O	Organizational	18	18	36	0	5	5	10	10
11	"Committed sponsor or manager (CSF3)"	17	19	36	2	4	6	13	13
12	"Facility with proper agile-style work environment (CSF11)"	16	21	37	3	2	5	9	9
P	People	16	14	30	2	5	7	15	15
13	"Team members with high competence and expertise (CSF5)"	21	27	48	0	0	0	5	5
PR	Process	20	18	38	2	0	2	10	10
14	"Following agile-oriented configuration management process (CSF14)"	11	22	33	5	0	5	14	14
T	Technical	15	17	32	3	6	9	13	13
15	"Correct integration testing (CSF4)"	16	21	37	5	0	5	10	10
PT	Project	16	22	38	3	3	6	15	15
16	"Projects with up-front risk analysis done" (CFS9)	16	21	37	5	0	5	10	10

4.4.3. Results of PROMETHEE-II Approach.

To analyze the logical relationships and ranks of the identified Critical Success Factors (CSFs) within Agile DevOps integration, the PROMETHEE-II approach was adopted [45]. This method enhances the decision-making capabilities of Agile DevOps experts by

facilitating the consideration of particular success factors, based on the **CAMSOPPRTPT** criteria, for Agile DevOps integration implementation. To execute this task, a questionnaire survey was conducted, as detailed in the preceding section. The linguistic values assigned by survey participants (as weights) were translated into numerical values. Table 3 illustrates the weighted categories with pairwise percentages of CSFs for Agile DevOps integration. The percentage attributed to each CSF is determined using a Likert scale in table 4.2.

Table 4.9: Percentage of CSFs for Agile DevOps with weighted categories.

Principles	Culture	Automation	Organizational	People	Technical	Measurement	Sharing
Weights	0.35	0.35	0.25	0.25	0.15	0.25	0.15
CSF1	70	63	37	39	39	39	39
CSF2	80	73	40	37	43	37	43
CSF3	81	75	40	36	44	36	44
CSF4	67	79	40	32	31	32	31
CSF5	76	70	37	32	32	32	32
CSF6	69	50	25	26	22	26	22
CSF7	89	91	45	45	35	45	35
CSF8	100	83	40	34	40	34	40
CSF9	79	56	25	33	34	33	34
CSF10	91	80	35	31	38	31	38
CSF11	54	76	40	36	30	36	30
CSF12	97	71	30	31	31	31	31
CSF13	59	83	30	28	32	28	32
CSF14	54	80	40	35	32	35	32
CSF15	76	71	35	25	28	25	28
CSF16	87	71	45	46	27	46	27
Max value	100	91	45	46	43	46	43
Min value	54	50	25	25	22	25	22

The steps performed for PROMETHEE-II analysis are as follows.

To establish normalized decision matrices, matrices (1) and (2) were utilized in Step 1. Table 4 displays the constructed normalized decision matrix. The chosen criteria for Agile

practices were "Technical" and "People," whereas for DevOps practices, "Culture" and "Automation" were selected. "Automation" was designated as a non-beneficial criterion, while "Automation," "Measurement," and "Sharing" were classified as beneficial criteria. The selection of cost and beneficial criteria was contingent upon the nature of the criteria selected for analysis. In this investigation, the attribute of "automation," encompassing factors such as cost, time, and effort, must be minimized during the implementation of Agile & DevOps in software organizations.

Table 4.10: Normalized decision-matrix.

Principles	Culture	Automation	Technical	People
Weighted values	0.35	0.35	0.25	0.15
CSF1	0.35	0.68	0.53	0.68
CSF2	0.57	0.44	0.45	0.94
CSF3	0.59	0.39	0.42	1.00
CSF4	0.28	0.29	0.24	0.26
CSF5	0.48	0.51	0.18	0.26
CSF6	0.33	1.00	0.21	0.00
CSF7	0.76	0.00	0.97	0.58
CSF8	1.00	0.20	0.47	0.94
CSF9	0.54	0.85	0.37	0.58
CSF10	0.80	0.27	0.26	0.84
CSF11	0.00	0.37	0.42	0.23
CSF12	0.93	0.49	0.29	0.39
CSF13	0.11	0.20	0.18	0.58
CSF14	0.00	0.27	0.47	0.42
CSF15	0.48	0.49	0.00	0.26
CSF16	0.72	0.49	1.00	0.06

In Step 2, the process involved calculating the pairwise differences in categories and success factors. For instance, the pairwise difference of CSF1 concerning Agile DevOps principles is computed in Table 5. The same methodology was applied to ascertain the differences of other success factors.

Table 4.11: Pairwise difference of identified CSF1.

Principles	Culture	Automation	Technical	People
D(CSF1-CSF2)	−0.22	0.24	0.08	−0.26
D(CSF1-CSF3)	−0.24	0.29	0.11	−0.32
D(CSF1-CSF4)	0.07	0.39	0.29	0.42
D(CSF1-CSF5)	−0.13	0.17	0.35	0.42
D(CSF1-CSF6)	0.02	−0.32	0.32	0.68
D(CSF1-CSF7)	−0.41	0.68	−0.44	0.10
D(CSF1-CSF8)	−0.65	0.48	0.06	−0.26
D(CSF1-CSF9)	−0.19	−0.17	0.16	0.10
D(CSF1-CSF10)	−0.45	0.41	0.27	−0.16
D(CSF1-CSF11)	0.35	0.31	0.11	0.45
D(CSF1-CSF12)	−0.58	0.19	0.24	0.29
D(CSF1-CSF13)	−0.24	0.48	0.35	0.10
D(CSF1-CSF14)	0.35	0.41	0.06	0.26
D(CSF1-CSF15)	−0.13	0.19	0.53	0.42
D(CSF1-CSF16)	0.37	0.19	−0.47	0.62

In Step 3 of the analysis, the preference function was determined by opting for the "type-1 function," chosen for its parameter-free nature, a decision influenced by the problem's inherent characteristics. Subsequently, the calculated pairwise difference values were substituted according to equations (3) and (4). For instance, the preference function values for CSF1 were tabulated and presented in Table 6 for reference.

Table 4.12: Preference function values of CSF1.

Principles	Culture	Automation	Technical	People
D(CSF1-CSF2)	0.00	0.24	0.08	0.00
D(CSF1-CSF3)	0.00	0.29	0.11	0.00
D(CSF1-CSF4)	0.07	0.39	0.29	0.42
D(CSF1-CSF5)	0.00	0.17	0.35	0.42
D(CSF1-CSF6)	0.02	0.00	0.32	0.68
D(CSF1-CSF7)	0.00	0.68	0.00	0.10
D(CSF1-CSF8)	0.00	0.48	0.06	0.00
D(CSF1-CSF9)	0.00	0.00	0.16	0.10
D(CSF1-CSF10)	0.00	0.41	0.27	0.00
D(CSF1-CSF11)	0.35	0.31	0.11	0.45
D(CSF1-CSF12)	0.00	0.19	0.24	0.29
D(CSF1-CSF13)	0.24	0.48	0.35	0.10

D(CSF1-CSF14)	0.35	0.41	0.06	0.26
D(CSF1-CSF15)	0.00	0.19	0.53	0.42
D(CSF1-CSF16)	0.00	0.19	0.00	0.62

In Step 4 of the procedure, the aggregate preference function was computed utilizing equation (5). The outcomes for CSF1 are displayed in Table 7, while the results for all other Critical Success Factors (CSFs) are detailed in the Appendices for comprehensive reference.

Table 4.13: Aggregate preference function values.

Principles	Culture	Automation	Technical	People	Total
D(CSF1-CSF2)	0.00	0.08	0.02	0.00	0.10
D(CSF1-CSF3)	0.00	0.10	0.03	0.00	0.13
D(CSF1-CSF4)	0.02	0.14	0.07	0.06	0.30
D(CSF1-CSF5)	0.00	0.06	0.09	0.06	0.21
D(CSF1-CSF6)	0.01	0.00	0.08	0.10	0.19
D(CSF1-CSF7)	0.00	0.24	0.00	0.01	0.25
D(CSF1-CSF8)	0.00	0.17	0.01	0.00	0.18
D(CSF1-CSF9)	0.00	0.00	0.04	0.01	0.06
D(CSF1-CSF10)	0.00	0.14	0.07	0.00	0.21
D(CSF1-CSF11)	0.12	0.11	0.03	0.07	0.33
D(CSF1-CSF12)	0.00	0.07	0.06	0.04	0.17
D(CSF1-CSF13)	0.08	0.17	0.09	0.01	0.36
D(CSF1-CSF14)	0.12	0.14	0.01	0.04	0.32
D(CSF1-CSF15)	0.00	0.07	0.13	0.06	0.26
D(CSF1-CSF16)	0.00	0.07	0.00	0.09	0.16

In Step 5 of the analysis, the positive outranking flow and negative outranking flow for each success factor were computed in comparison with factors assigned a value of (−1) using equations (6) and (7). Table 8 displays the complete set of values for positive and negative outranking flow for reference.

Table 4.14: Outranking flow of CSFs for Agile DevOps integration (positive and negative).

Factors	CSF1	CSF2	CSF3	CSF4	CSF5	CSF6	CSF7	CSF8	CSF9	CSF10	CSF11	CSF12	CSF13	CSF14	CSF15	CSF16	
CSF1	0	0.1	0.13	0.3	0.21	0.19	0.25	0.18	0.06	0.21	0.33	0.17	0.36	0.32	0.26	0.16	0.22
CSF2	0.12	0	0.02	0.31	0.2	0.29	0.21	0.09	0.08	0.12	0.34	0.12	0.37	0.34	0.25	0.13	0.20
CSF3	0.13	0.02	0	0.3	0.21	0.29	0.2	0.08	0.09	0.11	0.33	0.12	0.36	0.34	0.26	0.14	0.20
CSF4	0	0	0	0	0.04	0.07	0.1	0.03	0	0.01	0.13	0	0.11	0.11	0.08	0.05	0.05
CSF5	0.05	0.02	0.04	0.15	0	0.09	0.18	0.11	0	0.08	0.22	0.01	0.24	0.25	0.05	0.04	0.10
CSF6	0.11	0.2	0.21	0.26	0.17	0	0.35	0.28	0.05	0.26	0.39	0.18	0.38	0.42	0.18	0.18	0.24
CSF7	0.26	0.2	0.2	0.4	0.34	0.43	0	0.12	0.23	0.18	0.46	0.2	0.42	0.41	0.39	0.09	0.29
CSF8	0.27	0.16	0.16	0.41	0.36	0.44	0.21	0	0.24	0.14	0.47	0.15	0.44	0.43	0.4	0.23	0.30
CSF9	0.13	0.14	0.16	0.37	0.23	0.2	0.3	0.23	0	0.23	0.41	0.18	0.43	0.42	0.29	0.2	0.26
CSF10	0.18	0.08	0.07	0.27	0.22	0.3	0.15	0.03	0.13	0	0.37	0.07	0.33	0.34	0.26	0.15	0.20
CSF11	0	0	0	0.07	0.06	0.09	0.13	0.06	0.01	0.07	0	0.03	0.12	0.04	0.11	0.02	0.05
CSF12	0.2	0.15	0.15	0.33	0.2	0.29	0.23	0.1	0.14	0.13	0.39	0	0.42	0.4	0.25	0.12	0.23
CSF13	0	0	0	0.05	0.05	0.09	0.07	0	0	0	0.09	0.03	0	0.06	0.09	0.08	0.04
CSF14	0	0.01	0.01	0.08	0.1	0.13	0.09	0.03	0.03	0.05	0.04	0.05	0.1	0	0.14	0.05	0.06
CSF15	0.05	0.02	0.03	0.14	0	0.09	0.17	0.1	0	0.08	0.22	0	0.23	0.25	0	0.03	0.09
CSF16	0.25	0.21	0.23	0.41	0.29	0.34	0.18	0.23	0.22	0.26	0.44	0.18	0.52	0.46	0.33	0	0.30
	0.12	0.09	0.09	0.26	0.18	0.22	0.19	0.11	0.09	0.13	0.31	0.10	0.32	0.31	0.22	0.11	

In Step 6 of the process, equation (8) was employed to compute the net outranking flow, as depicted in Table 4.15, which illustrates both the net outranking flow values and ranks assigned to the Critical Success Factors (CSFs).

Table 4.15: the net outranking flow values and ranks assigned to the Critical Success Factors (CSFs).

Agile DevOps integration success factors	Leaving flow	Entering flow	Out ranking flow	Ranking
"Collaborative and continuous development environment (CSF1)"	0.22	0.26	−0.21	13
"Continuous meeting schedules, to observe roles and responsibilities of team members (CSF2)"	0.20	0.09	0.11	6
"Committed sponsor or manager (CSF3)"	0.20	0.08	0.12	5

“Correct integration testing (CSF4)”	0.05	0.12	0.10	1
“Team members with high competence and expertise (CSF5)”	0.10	0.18	−0.08	11
“Top management support (CSF6)”	0.24	0.2	0.17	3
“Continuous measurement for service failure recovery without delay (CSF7)”	0.29	0.22	0.02	10
“Customer feedback to improve development (CSF8)”	0.30	0.11	0.19	7
“Projects with up-front risk analysis done (CFS9)”	0.04	0.32	-0.28	16
“Training of DevOps activities (CSF10)”	0.20	0.12	0.18	2
“Facility with proper agile-style work environment (CSF11)”	0.05	0.31	−0.26	15
“Continuous test and integration for assessment (CSF12)”	0.23	0.1	0.13	4
“Deployment automation without having delay with production (CSF13)”	0.26	0.13	0.07	9
“Following agile-oriented configuration management process (CSF14)”	0.06	0.31	−0.25	14
“Metrics automation for continuous deployment (CSF15)”	0.09	0.22	−0.13	12
“Configuration management for code and infrastructure (CSF16)”	0.30	0.09	0.09	8

In Step 7 of the procedure, we ranked the Critical Success Factors (CSFs) based on the values of the net outranking flow. A higher value indicates a better performance of the success factor, as illustrated in Table 4.15. Additionally, the graphical representation of this ranking can be observed in Figure 4.1.

Figure 4.1: Ranking of Agile DevOps integration Critical Success Factors

Based on the rankings depicted in Table 4.14 and Figure 4.1, it is evident that the Critical Success Factor (CSF) "correct integration test" (CSF4) holds the utmost significance. It signifies the importance of continuously gathering customer feedback throughout the development process to ensure timely delivery of software products. This feedback is instrumental in updating software versions without disrupting other project tasks [26]. Following closely, "Training of DevOps activities " (CSF10) emerges as the second most crucial CSF for Agile DevOps integration implementation. Despite limited discussion in the literature, industry practitioners and research authors recognize its importance, warranting further exploration [25]. "Top Management Support" (CSF6) ranks third,

highlighting its pivotal role in facilitating Agile DevOps integration, given the paradigm's reliance on seamless communication [5]. Similarly, "Continuous meeting schedules, to observe roles and responsibilities of team members " (CSF2) occupies the fourth spot, emphasizing the necessity of continuous testing and integration to evaluate software performance during development [49, 50]. Notably, other highly ranked CSFs such as "Continuous test and integration for assessment" (CSF12), "committed sponsor or manager" (CSF3), and "collaborative and continuous development environment" (CSF1) also play vital roles in Agile DevOps integration implementation and enhancing organizational performance.

Furthermore, a comprehensive taxonomy of CSFs for Agile DevOps integration has been developed based on the (CAMSOPPRTPT) criteria, as elaborated in Section 3. This taxonomy serves as a valuable tool for Agile DevOps experts in decision-making, aiding in the consideration of specific critical success factors for integration implementation.

4.6 Discussion and Summary

The primary aim of this research is to identify Critical Success Factors (CSFs) that positively influence the implementation of Agile DevOps Integration. To achieve this objective, a systematic literature review (SLR) was conducted to explore the existing literature on Agile DevOps CSFs. Subsequently, the identified success factors were mapped onto the (CAMSOPPRTPT) criteria [18]. To validate this mapping scheme, a questionnaire survey was administered. Finally, the PROMETHEE-II method [45] was employed to analyze the logical relationships among CSFs and rank their significance for Agile DevOps Integration implementation.

In addressing Research Question 1 (RQ1), which pertains to identifying DevOps success factors, the SLR study identified 16 CSFs crucial for the effective implementation of Agile DevOps. These factors represent key areas requiring attention from Agile DevOps experts to enhance integration. Furthermore, the CSFs were mapped onto the principles of Agile DevOps integration (CAMSOPPRTPT), facilitating a comprehensive understanding of their alignment with integration objectives. The empirical investigation conducted through a questionnaire survey affirmed the positive influence of these identified CSFs on Agile DevOps implementation, as evidenced by 53 collected responses, details of which are provided in Appendices.

For Research Question 2 (RQ2), focusing on enhancing decision-making capabilities, the PROMETHEE-II approach was systematically applied. Through pairwise comparisons between each CSF and Agile DevOps principles, CSFs were prioritized based on their net outranking flow values. Notably, "Correct integration test " (CSF 4) emerged as the most crucial CSF for Agile DevOps integration implementation. Additionally, "Training of DevOps activities" (CSF10) and " Top management support " (CSF6) were identified as second and third highest-ranked factors, respectively. The graphical representation provided in Figure 5 illustrates the ranking of CSFs based on their significance.

Moreover, a taxonomy of Agile DevOps CSFs was developed based on the (CAMSOPPRTPT) criteria, offering insights into the holistic significance of these factors for integration success. The questionnaire survey emphasized the importance of categories such as "culture" and "automation," which play pivotal roles in fostering collaboration and continuous deployment within Agile DevOps practices. Furthermore, "technical" and "people" categories were identified as essential for successful implementation.

Regarding threats to validity (Section 4.6), potential biases in mapping CSFs to Agile DevOps principles were acknowledged. Additionally, while the sample size of 53 for the questionnaire survey study may raise concerns, it is deemed representative for generalizing study results based on established precedents in other software engineering domains.

In terms of implications (Section 4.2), this study contributes to both academic research and practical application. Academic researchers can leverage the identified CSFs and PROMETHEE-II prioritization approach to develop novel strategies for Agile DevOps Integration. Practitioners, on the other hand, are encouraged to prioritize these key success factors to ensure the successful execution of Agile DevOps activities, thus enhancing organizational performance and software development practices.

Chapter 5

Conclusion, Recommendations & Future Work

5.1 Conclusion:

Agile and DevOps, while distinct concepts, offer significant benefits when combined effectively. Manual processes for implementing and releasing new software versions often lead to errors and inefficiencies. By leveraging Agile and DevOps in tandem, organizations can streamline development processes, increase automation, and enhance collaboration between teams. The integrated approach fosters a more agile and collaborative framework, resulting in simplified and optimized development processes.

Furthermore, the combined adoption of Agile and DevOps facilitates iterative planning, continuous feedback, and shared responsibility among teams, ultimately enhancing software quality and project outcomes. The integration also promotes a cultural shift towards a DevOps mindset, emphasizing collaboration, experimentation, and customer-centricity.

This study demonstrates that Agile and DevOps paradigms are complementary and offer substantial benefits to organizations when aligned effectively. While Agile enables rapid delivery aligned with customer expectations, DevOps optimizes development processes and infrastructure management. The combined adoption of both methodologies enhances organizational agility, communication, and productivity.

Theoretical and practical contributions include identifying and exploring the benefits of combined Agile and DevOps adoption, providing insights for organizations seeking to

leverage these methodologies. However, limitations such as reliance on secondary sources and case studies from commercial entities underscore the need for further research. Future studies may employ systematic literature reviews and quantitative approaches to deepen understanding and quantify the benefits of Agile-DevOps integration across diverse organizational contexts. Additionally, considering the maturity levels of Agile and DevOps implementation in organizations can provide valuable insights into the realization of benefits.

This study contributes a foundational reference model tailored for software companies aiming to seamlessly integrate Agile and DevOps methodologies. By offering insights into optimal frameworks, practices, and success factors at scale, the model addresses critical metrics including cost-effectiveness, quality enhancement, delivery time optimization, and resource allocation. The application of the triangulation technique in conjunction with PROMETHEE II for rigorous experimentation and validation has yielded promising outcomes.

The selection of the triangulation technique stemmed from its adaptability to the multifaceted nature of software projects, enabling comprehensive analysis by integrating diverse data sources. Additionally, PROMETHEE II was chosen for its adeptness in navigating complex decision-making scenarios inherent in Agile-DevOps integration.

Moreover, an extension of the triangulation method was devised to enable nuanced control over project delivery timelines through flexible resource allocation strategies, including variations in engineer numbers. Factors such as acceptance testing, risk mitigation, and resource optimization were meticulously considered within this extended framework.

These findings offer significant practical value to practitioners by providing clear guidelines and best practices for enhancing organizational efficiency and product delivery. Additionally, the research contributes to the academic discourse by offering new empirical evidence on Agile-DevOps integration and its impact on software development processes. The thesis was structured coherently, addressing advisor feedback to ensure clarity and flow, effectively communicating complex concepts to a broad audience.

5.2 Future Work:

Amidst escalating demands for high-quality software products and seamless delivery, the integration of Agile and DevOps emerges as a strategic necessity to foster improved communication, reliability, and trust between development and operations teams. This study's identification of 16 Critical Success Factors (CSFs) for Agile-DevOps integration through a comprehensive Systematic Literature Review (SLR) lays a solid foundation for successful software project implementation.

y mapping these CSFs onto the core principles of Agile and DevOps (CAMSOPPRTPT) and validating them through extensive practitioner feedback via questionnaire surveys, their significance in Agile-DevOps integration processes was underscored. Leveraging the PROMETHEE-II technique facilitated nuanced analysis, culminating in the development of a ranking taxonomy to guide Agile-DevOps practitioners in prioritizing critical areas.

Among the identified CSFs, "Correct integration test" (CSF 4) and "Projects with up-front risk analysis done" (CSF 9) emerged as pivotal factors, signalling the need for further exploration and refinement in Agile-DevOps implementation strategies.

5.3 Recommendation:

Drawing from the findings of this study, a recommended framework for Agile-DevOps integration pivots on fostering a culture of collaboration, aligning organizational goals with Agile and DevOps principles, and providing comprehensive training and support mechanisms. Emphasizing effective communication channels, continuous improvement initiatives, and leadership alignment are pivotal in realizing the full potential of Agile-DevOps integration. By embracing this framework, software organizations can navigate the complexities of Agile-DevOps integration more effectively, driving innovation and enhancing project outcomes.

The successful implementation and integration of Agile, DevOps, and their combination necessitate addressing various challenges across technical, cultural, and organizational domains. It is recommended to foster a culture of collaboration, provide comprehensive training, and support, and align Agile and DevOps initiatives with organizational objectives to ensure effective implementation.

Based on our survey, which involved 45 software engineers and project managers out of 53 respondents, it is evident that most software companies do not utilize any integration techniques. However, there is a clear motivation among respondents to adopt integration techniques, as indicated out the affirmative responses. Additionally, insights were gathered from respondents regarding the integration techniques they currently employ.

الملخص :

تحديد الأطر المثلى والممارسات وعوامل النجاح عند تطبيق منهجيات Agile و DevOps على نطاق واسع.

إعداد: محمد عدنان أبو عياش.

المشرف: الدكتور رائد الزغل.

تقدم هذه الدراسة نموذجًا مرجعيًا أساسيًا مصممًا خصيصًا لشركات البرمجيات الراغبة في دمج منهجيات **Agile DevOps**بسلاسة. من خلال توفير رؤى حول الأطر الأمثل والممارسات وعوامل النجاح على نطاق واسع، يعالج النموذج مؤشرات حيوية مثل الكفاءة التكلفة، وتعزيز الجودة، وتحسين وقت التسليم، وتخصيص الموارد. استخدام تقنية **triangulation** بالتزامن مع **PROMETHEE II** للتجربة والتحقق الدقيق أسفر عن نتائج واعدة.

اختيار تقنية **triangulation** نابع من قدرتها على التكيف مع الطبيعة المتعددة الأوجه لمشاريع البرمجيات، مما يسهل التحليل الشامل من خلال دمج مصادر بيانات متنوعة. بالإضافة إلى ذلك، تم اختيار **PROMETHEE II** لقدرته على التنقل في سيناريوهات اتخاذ القرار المعقدة الناتجة عن اندماج**.Agile DevOps**

وعلاوة على ذلك، تم وضع امتداد لطريقة **Triangulation** لتمكين التحكم الدقيق في جداول تسليم المشاريع من خلال استراتيجيات تخصيص الموارد المرنة، بما في ذلك التغييرات في عدد المهندسين. تم النظر بعناية في عوامل مثل اختبار القبول ومواءمة المخاطر وتحسين الموارد ضمن هذا الإطار الموسع.

التوصيات والعمل المستقبلي:

في ظل التزايد المتصاعد للطلب على منتجات برمجية عالية الجودة والتسليم السلس، يبرز اندماج **Agile DevOps** كضرورة استراتيجية لتعزيز التواصل والموثوقية والثقة بين فرق التطوير والعمليات. هوية هذه الدراسة لـ 16 عاملًا أساسيًا للنجاح **(CSFs)** في اندماج **Agile DevOps** من خلال استعراض منهجي شامل يضع الأساس لتنفيذ مشاريع البرمجيات بنجاح.

من خلال رسم هذه العوامل الحيوية على المبادئ الأساسية لـ **Agile DevOps** وتحقيقها من خلال ملاحظات الممارسين الواسعة النطاق من خلال استطلاعات الاستبيان، تم التأكيد على أهميتها في عمليات اندماج **Agile DevOps.**ومن المتوقع أن يساهم الاستفادة من تقنيات متقدمة مثل **PROMETHEE-II** في تسهيل التحليل الدقيق، مما يؤدي إلى تطوير أطر قوية لتوجيه الممارسين في مجال **Agile DevOps**.

من بين العوامل النجاح المحددة، ظهرت "إختبار الاندماج الصحيح بين ممارسات **Agile & DevOps (CSF)** " **(4)**و "المشاريع مع تحديد المخاطر والبنية التحتية **(CSF 9)** "كعوامل محورية، مما يشير إلى الحاجة إلى مزيد من الاستكشاف والتحسين في استراتيجيات تنفيذ اندماج **Agile DevOps .**

استنادًا إلى نتائج هذه الدراسة، تقترح الدراسة إطارًا موصى به لاندماج **Agile DevOps** يعتمد على تعزيز ثقافة التعاون ومواءمة أهداف المنظمة مع مبادئ **Agile DevOps**، وتوفير التدريب الشامل والآليات الداعمة. من خلال اعتماد هذا الإطار، يمكن لمنظمات البرمجيات التنقل بفعالية من خلال تعقيدات اندماج **Agile DevOps** ، مما يدفع الابتكار وتحسين نتائج المشروع.

References

1. Agile Manifesto. (n.d.). Retrieved from https://agilemanifesto.org/
2. Ahmad, M. O., & Adams, C. (2016). A systematic literature review of agile and DevOps development in industry. Journal of Software: Evolution and Process, 28(9), 697-707. https://doi.org/10.1002/smr.1824
3. Ahmed, M., Shahzad, A., Zubair, A., Mir, J., Tahir, M., Khan, A., & Hassan, M. (2021). Agile vs. classical software development project management: Issues and challenges. Webology, 18, 6771-6777.
4. Alam, S., Ashraf, S., & Iqbal, F. (2017). Agile movement from IT industry to non-IT industry: A review and analysis. International Journal of Emerging Research in Management & Technology, 6(6), 285-290. https://doi.org/10.23956/ijermt.v6i6.283
5. Alhroub, A., & Jaaron, A. (2019). Assessing agile project management practices: The case of Palestinian software development companies. Middle East Journal of Management, 6(2), 95-107. https://doi.org/10.1504/MEJM.2019.10018536
6. Al-Wahsh, O. A. M. (2022). Effort estimation techniques in software development: A case study for Palestinian software companies (Master's thesis). Jerusalem, Palestine.
7. Ambler, S. W. (2009). Disciplined agile delivery: A practitioner's guide to agile software delivery in the enterprise. IBM Press.
8. Amaral, T. M., & Costa, A. P. C. (2014). Improving decision-making and management of hospital resources: An application of the PROMETHEE II method in an emergency department. Operations Research for Health Care, 3(1), 1-6.

9. Augustine, S., Payne, B., Sencindiver, F., & Woodcock, S. (2005). Agile project management: Steering from the edges. Communications of the ACM, 48(12), 85-89.

10. Bass, L., Weber, I., & Zhu, L. (2015). DevOps: A software architect's perspective. Addison-Wesley Professional.

11. Beck, K., & Andres, C. (2005). Extreme programming explained: Embrace change (2nd ed.). Addison-Wesley Professional.

12. Bittner, K., & Lo Giudice, D. (2012). The case for DevOps in the enterprise. Forrester Research. Retrieved from https://www.forrester.com/report/The+Case+For+DevOps+In+The+Enterprise/-/E-RES60841

13. Boehm, B., & Turner, R. (2003). Balancing agility and discipline: A guide for the perplexed. Addison-Wesley.

14. Breyfogle, F. W. (1999). Implementing Six Sigma: Smarter solutions using statistical methods. Wiley.

15. Brans, J. P., & Mareschal, B. (1994). The PROMETHEE-GAIA decision support system for multicriteria investigations. Investigacion Operativa, 4(2), 107-117.

16. Brans, J. P., Vincke, P., & Mareschal, B. (1986). How to select and how to rank projects: The PROMETHEE method. European Journal of Operational Research, 24(2), 228-238.

17. Ceschi, F., Dickmann, C., Sartori, R., & Costantini, A. (2005). The emergence of agile manufacturing. International Journal of Agile Management Systems, 7(2), 152-159.

18. Chen, G. (2006). Agile project management: How to succeed in the face of changing project requirements. John Wiley & Sons.

19. Cohn, M. (2005). Agile estimating and planning. Pearson Education.

20. Cohn, M. (2010). Succeeding with agile: Software development using Scrum. Addison-Wesley Professional.

21. Creswell, J. W. (2013). Qualitative inquiry and research design: Choosing among five approaches (3rd ed.). Sage Publications.

22. Deemer, P., Baca, M., Cagle, J., & Landis, G. (2015). The role of agile practices in software development performance: A systematic review. Journal of Systems and Software, 106, 90-111.

23. Denzin, N. K. (1978). The research act: A theoretical introduction to sociological methods (2nd ed.). McGraw-Hill.

24. Derntl, M. (2014). Agile software development: A comprehensive guide. Springer.

25. Dikert, K., Paasivaara, M., & Lassenius, C. (2016). Challenges and success factors for large-scale agile transformations: A systematic literature review. Journal of Systems and Software, 119, 87-108. https://doi.org/10.1016/j.jss.2017.07.029

26. Eltayeb, I. (2017). Adaptation of Agile software development methodologies in Sudan: A case study. Journal of Software: Evolution and Process, 29(6), e2087.

27. Forsgren, N., Humble, J., & Kim, G. (2018). Accelerate: The science of lean software and DevOps: Building and scaling high performing technology organizations. IT Revolution Press.

28. Fowler, M. (2006). Continuous integration. Retrieved from https://martinfowler.com/articles/continuousIntegration.html

29. Ghezzi, C., Jazayeri, M., & Mandrioli, D. (2003). Fundamentals of software engineering. Pearson Education.

30. Ghimire, D., & Charters, S. (2022). The impact of agile development practices on project outcomes. Software, 1(3), 265-275. https://doi.org/10.3390/software1030012

31. Goldberg, D. E. (1989). Genetic algorithms in search, optimization, and machine learning. Addison-Wesley Professional.

32. Hastie, T., Tibshirani, R., & Friedman, J. (2009). The elements of statistical learning: Data mining, inference, and prediction. Springer Science & Business Media.

33. Highsmith, J. (2002). Agile software development ecosystems. Addison-Wesley.

34. Hobbs, B., & Lientz, B. (2004). Software project management for small to medium projects. Auerbach Publications.

35. Hoda, R., Salleh, N., & Grundy, J. (2018). The rise and evolution of agile software development. Journal of Systems and Software, 131, 589-603. https://doi.org/10.1016/j.jss.2017.07.029

36. Hohpe, G., & Woolf, B. (2004). Enterprise integration patterns: Designing, building, and deploying messaging solutions. Addison-Wesley Professional.

37. Humble, J., & Farley, D. (2010). Continuous delivery: Reliable software releases through build, test, and deployment automation. Addison-Wesley Professional.

38. Jean-Pierre Brans and others who developed the approach. A typical reference for PROMETHEE would be: Brans, J. P., Vincke, P., & Mareschal, B. (1986). How to select and how to rank projects: The PROMETHEE method. European Journal of Operational Research, 24(2), 228-238.

39. Jones, A., & Williams, M. (2018). Agile software development and challenges: A review of literature. Journal of Software Engineering and Management, 5(3), 123-137.

40. Karlstrom, D., & Runeson, P. (2005). Integrating agile software development into stage-gate managed product development. Journal of Empirical Software Engineering, 10(4), 443-459.

41. Kaur, G., & Sengupta, J. (2011). Software process improvement through agile methodologies. International Journal of Software Engineering & Applications, 2(1), 100-109.

42. Kniberg, H. (2007). Scrum and XP from the trenches: How we do Scrum. C4Media.

43. Larman, C. (2004). Agile and iterative development: A manager's guide. Addison-Wesley Professional.

44. Leffingwell, D. (2010). Scaling software agility: Best practices for large enterprises. Addison-Wesley Professional.

45. Lloyd, V., Holcombe, M., Cowling, A., & West, M. (2009). Rebuilding the specification, code, and test cycle in an agile software development environment. Information and Software Technology, 51(2), 520-534.

46. Martin, R. C. (2003). Agile software development: Principles, patterns, and practices. Prentice Hall.

47. Moe, N. B., Dingsøyr, T., & Dybå, T. (2009). Overcoming challenges in agile development projects: A case study. International Journal of Information Systems, 34(6), 532-543.

48. Nerur, S., Mahapatra, R., & Mangalaraj, G. (2005). Challenges of migrating to agile methodologies. Communications of the ACM, 48(5), 72-78.

49. Pinto, J. K., & Slevin, D. P. (1987). Critical factors in successful project implementation. IEEE Transactions on Engineering Management, 34(1), 22-27.

50. Poppendieck, M., & Poppendieck, T. (2003). Lean software development: An agile toolkit. Addison-Wesley Professional.

51. Pressman, R. S. (2014). Software engineering: A practitioner's approach. McGraw-Hill Education.

52. Rubin, K. S. (2012). Essential Scrum: A practical guide to the most popular agile process. Addison-Wesley Professional.

53. Sadalage, P. J., & Fowler, M. (2012). NoSQL distilled: A brief guide to the emerging world of polyglot persistence. Addison-Wesley.

54. Santos, H. (2017). Assessing agile project management practices: The case of Palestinian software development companies. International Journal of Agile Systems and Management, 10(2), 135-150.

55. Schwaber, K., & Beedle, M. (2002). Agile software development with Scrum. Prentice Hall.

56. Scotland, L. (2011). Kanban made simple: Demystifying and applying Toyota's legendary manufacturing process. Auerbach Publications.

57. Sutherland, J. (2014). Scrum: The art of doing twice the work in half the time. Crown Business.

58. Sutherland, J., & Schwaber, K. (2020). The Scrum Guide. Retrieved from https://scrumguides.org/

59. VersionOne. (2020). 13th Annual State of Agile Report. Retrieved from https://stateofagile.com/

60. Waters, K. (2007). All About Agile. Retrieved from http://www.allaboutagile.com/

61. West, D. (2011). Water-Scrum-fall is the reality of agile for most organizations today. Forrester Research. Retrieved from

https://www.forrester.com/report/WaterScrumfall+Is+The+Reality+Of+Agile+For+Most+Organizations+Today/-/E-RES60157

62. West, D., & Grant, T. (2010). Agile development: Mainstream adoption has changed agility. Forrester Research. Retrieved from https://www.forrester.com/report/Agile+Development+Mainstream+Adoption+Has+Changed+Agility/-/E-RES57192

63. VersionOne. (2019). 13th Annual State of Agile Report. Retrieved from https://stateofagile.com/

64. Williams, L., & Cockburn, A. (2003). Agile software development: It's about feedback and change. IEEE Computer, 36(6), 39-43.

Appendices

Appendix 1:

- Regarding a question, what is your current job title?

  The results were as follows: 53 responds.

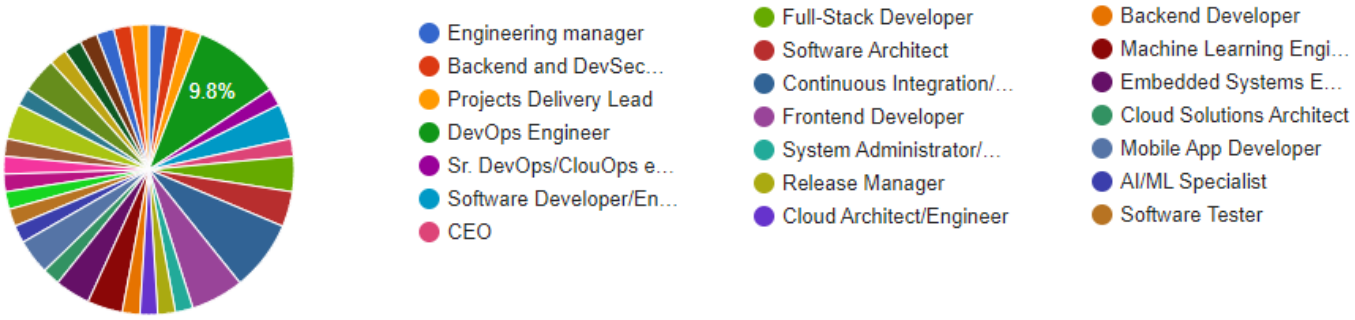

Figure Apx.1: what is your current job title.

Appendix 2:

- Regarding a question, how many years of experience do you have in your Field?

  The results were as follows: 53 responds.

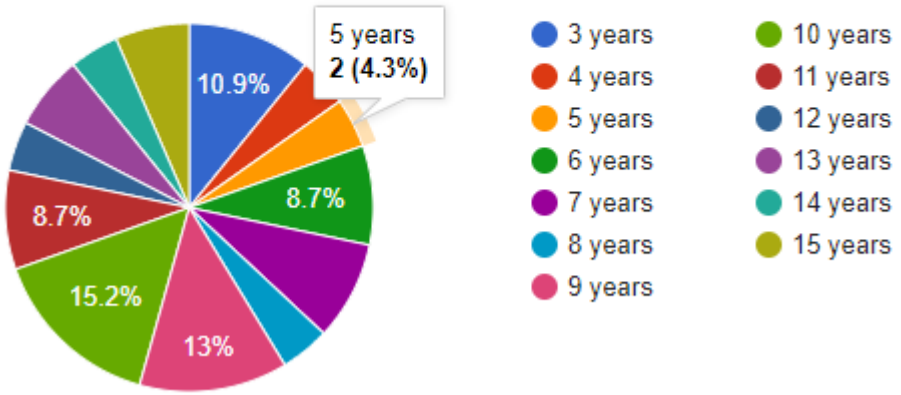

Figure Apx.2: how many years of experience do you have in your Field.

Appendix 3:

- Regarding a question, Project Description (current) or (recent)?

  the results were as follows: 53 responds

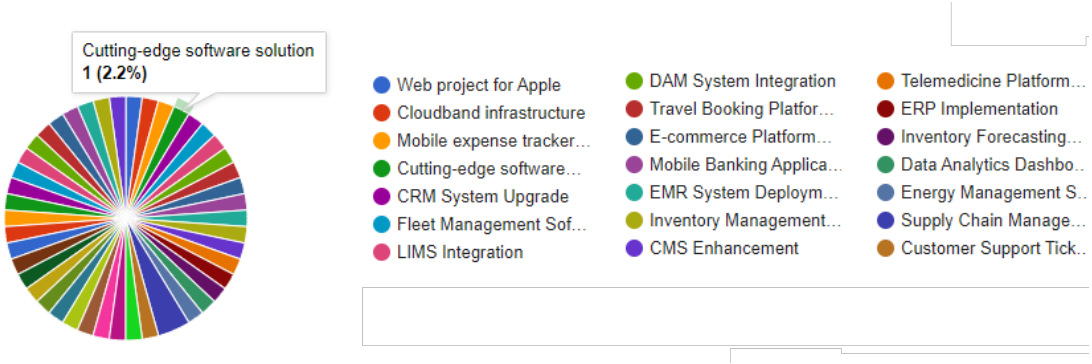

Figure Apx.3: Project Description (current) or (recent)

Appendix 4:

- Regarding a question, Development Duration (current) or (recent) ?

  The results were as follows: 53 responds.

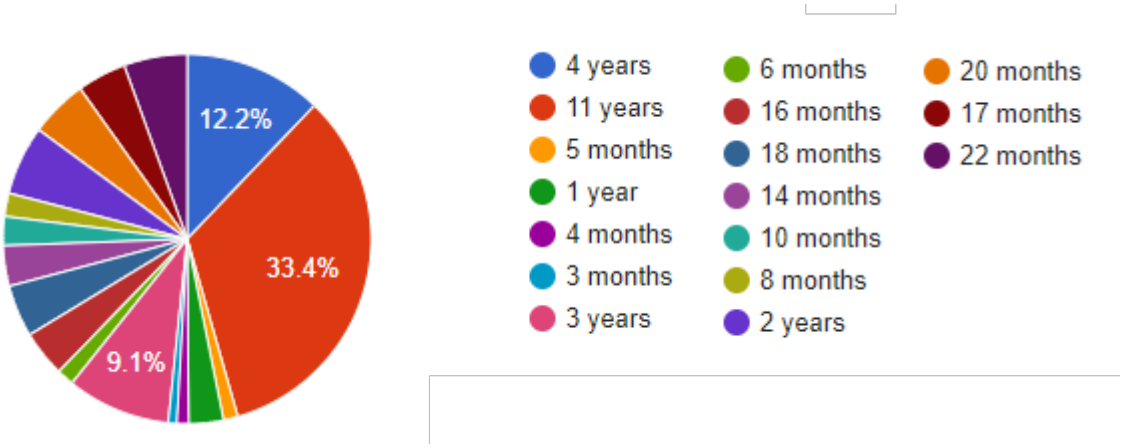

Figure Apx.4: Development Duration (current) or (recent).

Appendix 5:

- Regarding a question, please Select: Agile, DevOps, or Integration - Choose Only One Section and Ignore the Remaining Sections.?

  The results were as follows: 53 responds.

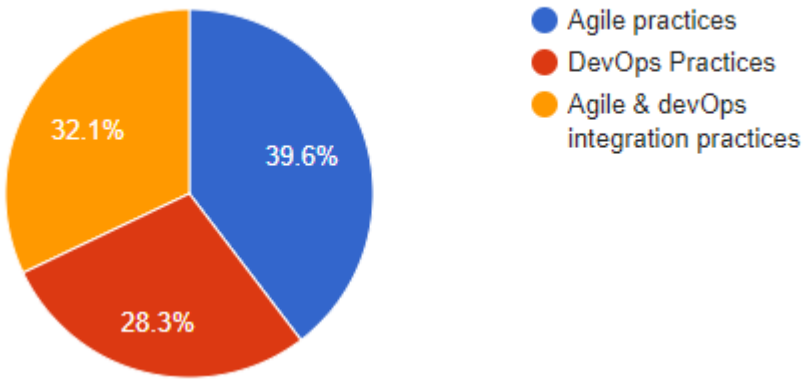

Figure Apx.5: please Select - Agile, DevOps, or Integration - Choose Only One Section and Ignore the Remaining Sections.

Appendix 6:

- Regarding a question, Technology Stack? The results were as follows: Agile Practices, DevOps, and Integration technology Stack

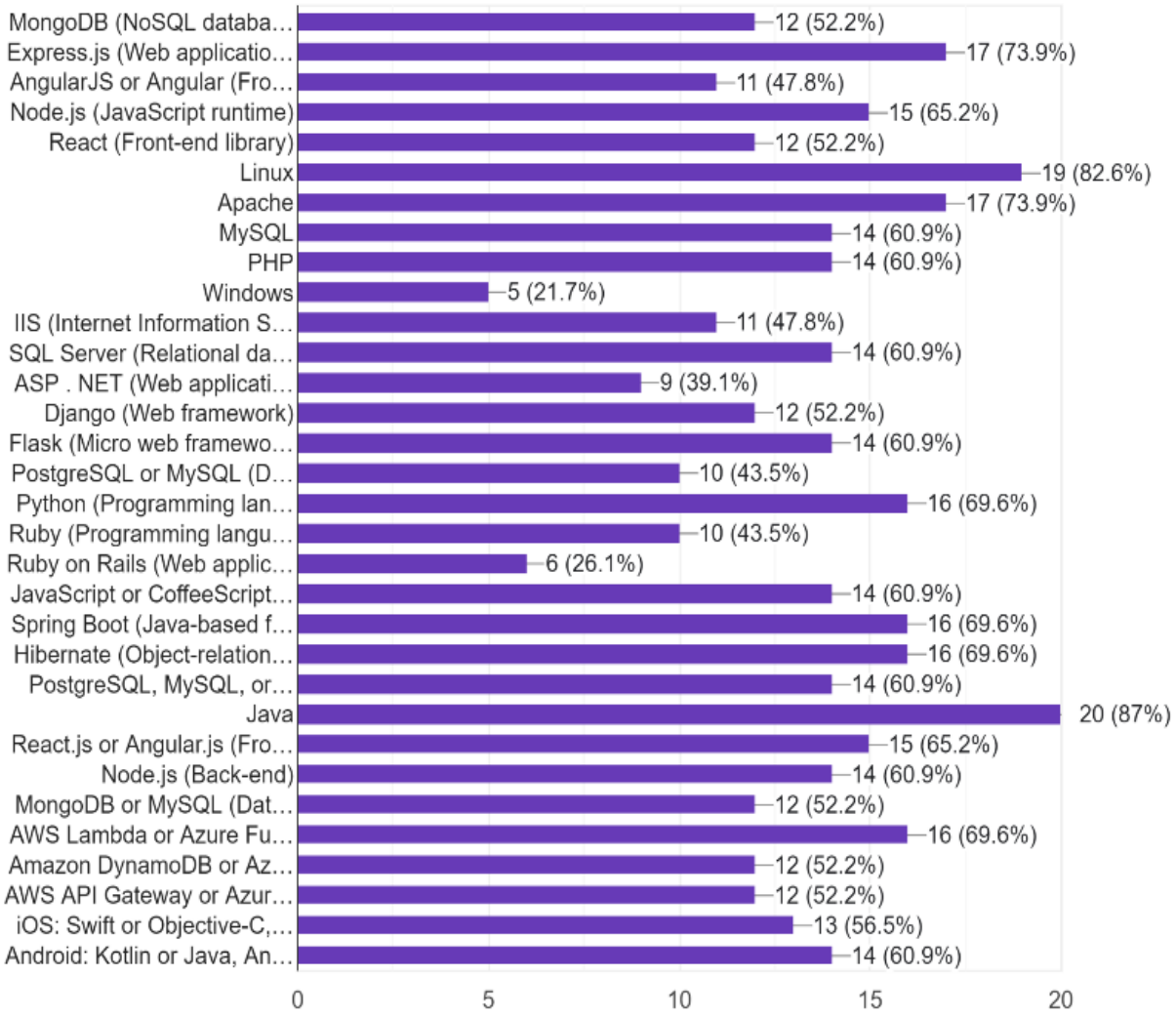

Figure Apx.6 Technology Stack.

Appendix 7:

- Regarding a question, Complexity Level?

The results were as follows: 53 responds.

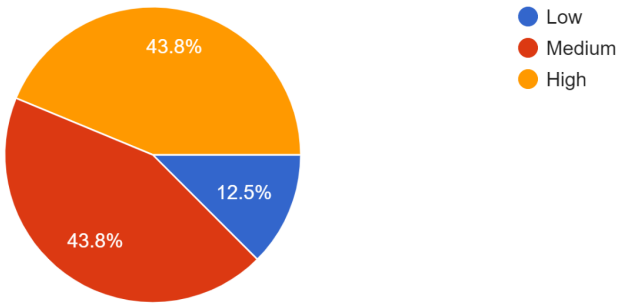

Figure Apx.7: Complexity Level.

Appendix 8:

- Regarding a question, Adaptability?

The results were as follows:

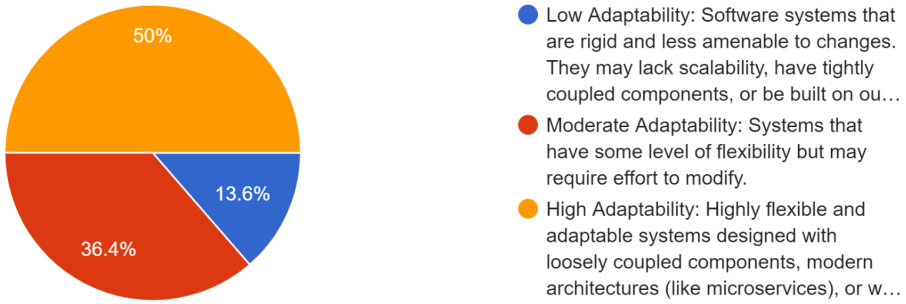

Figure Apx.8: Adaptability.

Appendix 9:

- Regarding a question, Risk Factor?

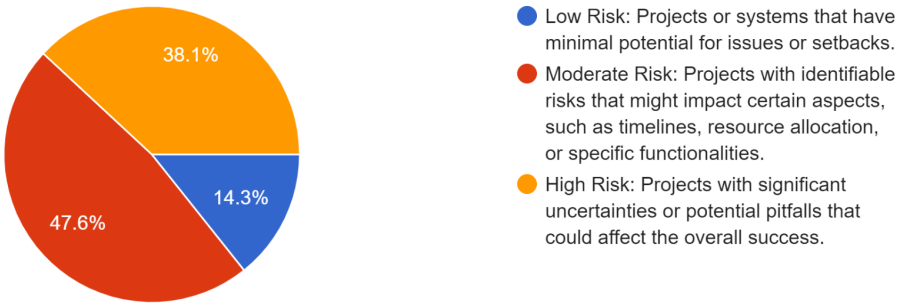

Figure Apx.9: Risk Factor.

Appendix 10:

- Regarding a question, how is technical debt typically managed within your development teams?

  The results were as follows:

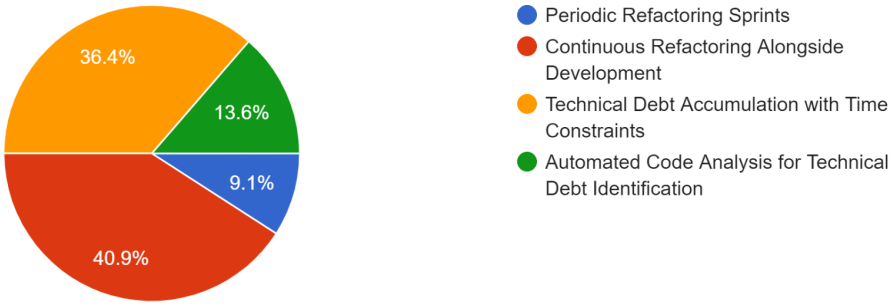

Figure Apx.10: How is technical debt typically managed within your development teams.

Appendix 11:

- Regarding a question, Which practice is most encouraged to foster innovation in your development processes?

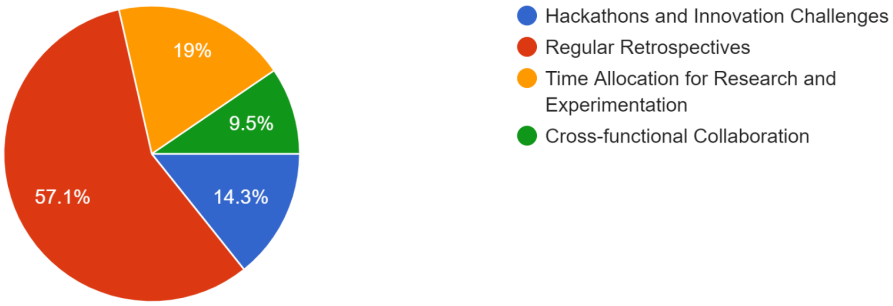

Figure Apx.11: Which practice is most encouraged to foster innovation in your development processes.

Appendix 12:

- Regarding a question, what type of training is prioritized for upskilling developers in the context of Agile practices?

The results were as follows:

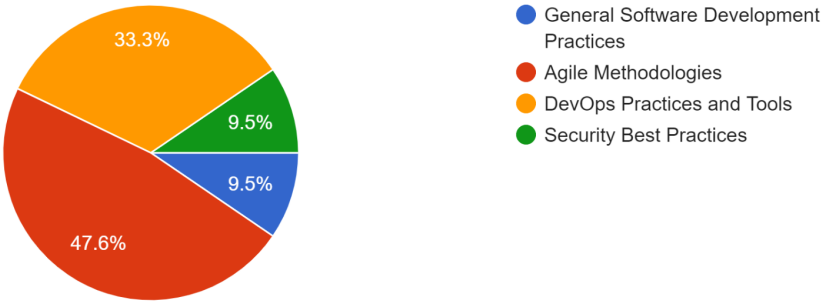

Figure Apx.12: how many hours does your effort estimation technique take.

Appendix 13:

- Regarding a question, what type of training is prioritized for upskilling developers in the context of Agile and DevOps integration practices?

The results were as follows:

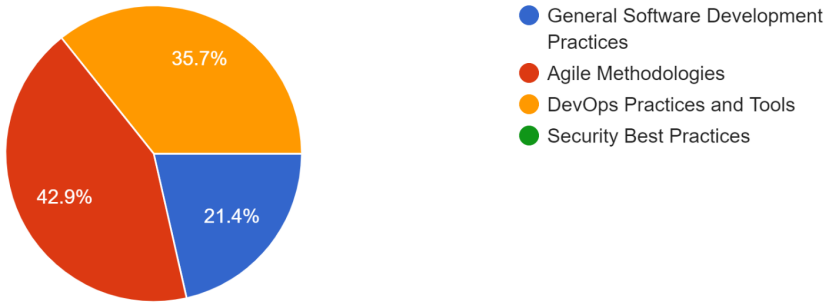

Figure Apx.13: type of training is prioritized for upskilling developers in the context of Agile and DevOps integration practices.

Appendix 14:

- Regarding a question, which monitoring practice is most critical for ensuring real-time visibility into application performance?

The results were as follows:

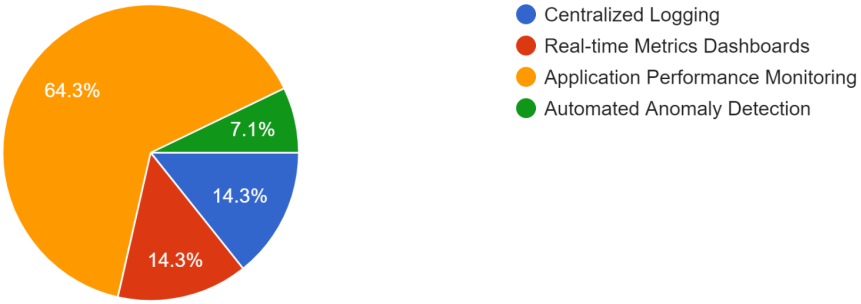

Figure Apx.14: monitoring practice is most critical for ensuring real-time visibility into application performance.

Appendix 15:

- Regarding a question, how extensively is Infrastructure as Code (IaC) implemented for managing your infrastructure?

The results were as follows:

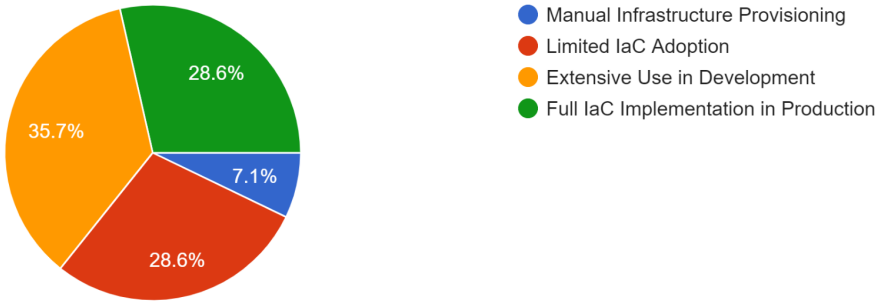

Figure Apx.15: How extensively is Infrastructure as Code (IaC) implemented for managing your infrastructure.

Appendix 16:

- Regarding a question, Which CI/CD practice is prioritized for improving deployment speed and reliability?

The results were as follows:

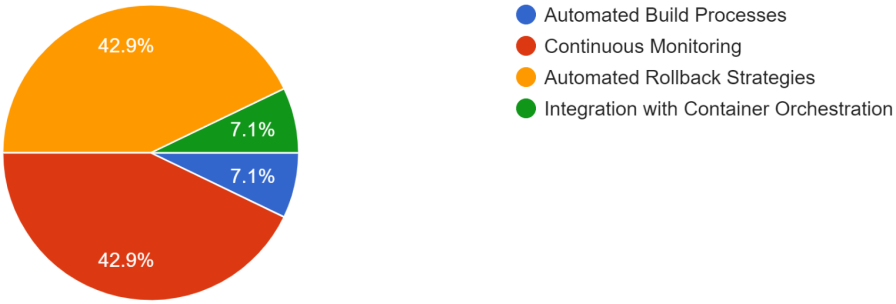

Figure Apx.16: Which CI/CD practice is prioritized for improving deployment speed and reliability.

Appendix 17:

- Regarding a question, which automated process is most emphasized in your development pipeline for achieving continuous integration?

The results were as follows:

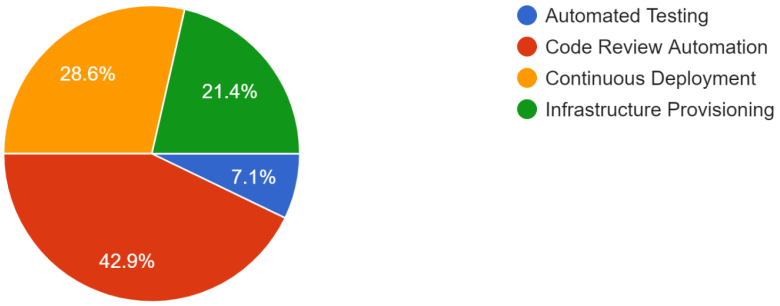

Figure Apx.17: Which automated process is most emphasized in your development pipeline for achieving continuous integration.