ADDIS ABABA UNIVERSITY SCHOOL OF COMMERCE

DEPARTMENT OF PROJECT MANAGEMENT

POST GRADUATE PROGRAM

Challenges of Implementing DevOps (Development Operations) as a Tool of Project Integration Management: the Case of Temenos Infinity System Implementation at Bank of Abyssinia

In Partial Fulfillment of the Requirements for the Award of Master of Arts Degree in Project Management

By: Amlakie Yitayih

Advisor: Dr. Teklegiorgis Assefa

June, 2024

Addis Ababa, Ethiopia

ADDIS ABABA UNIVERSITY SCHOOL OF COMMERCE

DEPARTMENT OF PROJECT MANAGEMENT

POST GRADUATE PROGRAM

Challenges of Implementing DevOps (Development Operations) as a Tool of Project Integration Management: the Case of Temenos Infinity System Implementation at Bank of Abyssinia

By: Amlakie Yitayih

A Project Work Submitted to Addis Ababa University School of Commerce in Partial Fulfillment of the Requirements for the Award of Master of Arts Degree in Project Management

Advisor: Dr. Teklegiorgis Assefa

June, 2024

Addis Ababa, Ethiopia

Addis Ababa University School of Commerce

Department of Project Management

Post Graduate Program

Challenges of Implementing DevOps (Development Operations) as a Tool of Project Integration Management: the Case of Temenos Infinity System Implementation at Bank of Abyssinia

By: Amlakie Yitayih

Approved by Board of Examiners

	Signature	Date
Dr. Tekelegiorgis Assefa (Advisor)	_____	_____
(Internal Examiner)	_____	_____
(External Examiner)	_____	_____

Declaration

I declare that this research paper entitled **"Challenges of Implementing DevOps (Development Operations) as a Tool of Project Integration Management: the Case of Temenos Infinity System Implementation at Bank of Abyssinia"** is my original work. I've conducted the study individually with the guidance and commentary of my advisor.

This research has not been presented to any other university and is not concurrently submitted in candidature of any other degree. It is conducted for the partial fulfillment of the Master of Arts Degree in Project Management.

Declared by:

Amlakie Yitayih	_____	June 2024
Student	Signature	Date

Certified by:

Dr. Tekelegiorgis Assefa	_____	June 2024
Advisor	Signature	Date

Certification

This is to certify that this research work by Amlakie Yitayih entitled **"Challenges of Implementing DevOps (Development Operations) as a Tool of Project Integration Management: the Case of Temenos Infinity System Implementation at Bank of Abyssinia"** is submitted in partial fulfillment of the requirements for the degree of Master of Arts in Project Management. This work complies with the regulations of the university and meets the accepted standards with respect to originality and quality.

Dr. Tekelegiorgis Assefa

Signature _____

Date_____

Acknowledgment

I am wholeheartedly thankful and grateful to God for providing me with strength and guidance in his grace throughout the research and writing of this project work.

I extend my deepest gratitude to my advisor, Dr. Tekelegiorgis, whose guidance, support, and mentorship have been invaluable throughout this research journey. I am also thankful to the staff members of Bank of Abyssinia for their generous allocation of time and attention, enabling me to access essential resources and data vital for this study. Additionally, I am immensely grateful to my friends, colleagues, and family members whose unwavering support, understanding, and encouragement have sustained me through the challenges this academic pursuit. Their belief in my abilities has been a constant source of motivation, and their presence has enriched my experience immeasurably.

Table of Contents

List of Tables

List of Figures

ACRONYMS

DevOps – Development and Operations

VCS – Version control system

CI – Continuous Integration

CD – Continuous Delivery

IaC – Infrastructure as Code

QA – Quality Assurance

BOA – Bank of Abyssinia

DevSecOps – Development, Security and Operations

IT – Information Technology

XP – Extreme Programing

GDPR – General Data Protection Regulation

PCI DSS – Payment Card Industry Data Security Standard

API – Application Programing Interface

SDLC – Software Development Lifecycle

Abstract

This study investigates the extent of adoption, and perceived barriers and facilitators of implementing DevOps practices within the context of bank of Abyssinia's Temenos Infinity software project. Using a mixed-methods approach, data were collected through questionnaires and interviews from 51 respondents involved in the project. The analysis reveals that strong management support, comprehensive training programs, and effective use of collaboration tools are essential for successful DevOps adoption. However, the study also finds that DevOps practices are minimally and inconsistently adopted across the project, with significant challenges such as lack of training, integration issues, and resource constraints impeding progress. The findings highlight the need for a more structured approach to training and implementation, and emphasize the importance of leveraging management support and collaboration tools to overcome barriers and enhance the adoption of DevOps practices. These insights provide valuable guidance for organizations seeking to implement DevOps in their IT projects, particularly within the banking sector.

Keywords

DevOps, banking sector, IT projects, Temenos Infinity, adoption, barriers, facilitators, management support, training, collaboration tools.

Chapter 1: Introduction

This chapter introduces the importance of DevOps in banking, focusing on its implementation within the Bank's Temenos Infinity software project. The discussion highlights the project's background, the research objectives, and the critical questions that guide this study, aiming to uncover the extent of DevOps adoption, the perceived barriers and facilitators within the bank's project management framework.

1.1 Background

According to the 2021 State of DevOps Report by Puppet Labs, organizations that fully implement DevOps practices are 55% more likely to achieve higher levels of organizational performance. Additionally, these high-performing organizations deploy code 46 times more frequently and have a fivefold lower change failure rate compared to their lower-performing counterparts (Puppet Labs, 2021).

In the rapidly evolving landscape of software development and IT operations, organizations face constant pressure to innovate and deliver high-quality software at an accelerated pace. Traditional development methodologies often result in separate sections between development and operations teams, leading to inefficiencies and delays in software delivery. To address these challenges, the DevOps (Development and Operations) methodology has emerged as a transformative approach that emphasizes collaboration, automation, and continuous improvement.

DevOps integrates software development (Dev) and IT operations (Ops) to shorten the system development life cycle while delivering features, fixes, and updates frequently in close alignment with business objectives. Key practices in DevOps include Continuous Integration (CI), Continuous Delivery (CD), automated testing, and Infrastructure as Code (IaC). These practices aim to enhance the speed and reliability of software releases, thereby improving overall software quality and customer satisfaction.

Bank of Abyssinia, a leading private bank in Ethiopia, has established itself as a cornerstone of the country's banking sector. With a rich history spanning several decades, Bank of Abyssinia has

played a pivotal role in shaping Ethiopia's financial landscape. As one of the largest private banks in the nation, it boasts a widespread network of over 909 branches, strategically positioned across urban centers and rural areas alike (Bank of Abyssinia Website, 2024).

Bank of Abyssinia has earned a reputation for its unwavering commitment to excellence, innovation, and customer-centricity. Since its inception, the bank has been at the forefront of technological advancement, continuously striving to leverage cutting-edge solutions to enhance service delivery and customer experience. In line with its vision of becoming a digital leader in the banking industry, Bank of Abyssinia has embarked on a journey of digital transformation. Recognizing the evolving needs and preferences of its diverse customer base, the bank has made significant investments in technology, infrastructure, and talent to modernize its operations and offerings (Bank of Abyssinia Website, 2024).

The software development industry has undergone significant transformations over the past few decades, driven by the need for faster and more reliable software delivery. Traditional software development methodologies, such as the Waterfall model, often result in extended development cycles, delayed feedback, and a lack of agility to respond to changing business needs (Balaji & Murugaiyan, 2012). In response to these challenges, the Agile methodology emerged, emphasizing iterative development, customer collaboration, and flexibility. However, Agile alone does not fully address the operational aspects necessary for rapid and reliable software deployment.

DevOps, a set of practices that combine software development (Dev) and IT operations (Ops) to enable continuous delivery and integration with high software quality. The term "DevOps" was popularized by Patrick Debois in 2009 during the DevOpsDays conference in Ghent, Belgium, and has since evolved into a comprehensive approach for managing the end-to-end software delivery lifecycle (Huttermann, 2012). DevOps aims to bridge the gap between development and operations teams by promoting a culture of collaboration, shared responsibilities, and continuous improvement.

Key practices within DevOps include Continuous Integration (CI), Continuous Delivery (CD), automated testing, and Infrastructure as Code (IaC). CI involves the frequent integration of code changes into a shared repository, followed by automated builds and tests to detect integration issues early (Fowler & Foemmel, 2006). CD extends CI by automating the deployment process,

allowing code to be deployed to production environments quickly and safely (Humble & Farley, 2010). Automated testing ensures that code changes do not introduce defects, while IaC allows for the automated provisioning and management of infrastructure, ensuring consistency and repeatability across environments (Kim, Humble, Debois, & Willis, 2016).

Despite the advantages of DevOps, its implementation is not without challenges. Organizations often face difficulties in adopting DevOps practices due to cultural resistance, legacy systems, and the complexity of integrating new tools and processes with existing infrastructure (Forsgren, Humble, & Kim, 2018). These challenges are particularly pronounced in highly regulated industries such as banking, where compliance and security are paramount.

1.2 Problem Statement

The rapid evolution of technology in the banking sector necessitates continuous innovation and efficient software delivery processes. Since the bank intends to keep doing innovative technological pioneering(spearheading) in the digitalization realm, DevOps practices are key for success of future IT projects. DevOps, a methodology that emphasizes collaboration between development and operations teams, has emerged as a vital practice for enhancing software delivery speed and reliability. However, the implementation of DevOps in banking institutions presents unique challenges, particularly when dealing with complex systems such as Temenos Infinity. Bank of Abyssinia, like many other financial institutions, operates with complex IT infrastructures that include legacy systems requiring extensive customization and integration. This complexity makes the adoption and integration of DevOps practices within the bank's project management framework a formidable task.

The integration of DevOps within the banking sector is fraught with complexities due to the coexistence of legacy systems, stringent regulatory requirements, and the need for seamless customization and integration of new software solutions. Despite the proven benefits of DevOps in enhancing software delivery processes, banks face significant obstacles in adopting these practices effectively. The primary challenge lies in aligning DevOps practices with the integration management framework, ensuring that all components of the software and IT infrastructure work together harmoniously.

Effective DevOps implementation involves a set of practical steps and adherence to critical success factors. Identifying these steps and factors is crucial for banks to streamline their software development processes and enhance overall operational efficiency. Yet, there is a significant gap in the literature regarding the specific actions and conditions that contribute to successful DevOps adoption in the banking sector. This gap underscores the need for a detailed exploration of the practical steps and critical success factors for implementing DevOps in such complex environments.

Moreover, understanding the extent to which DevOps practices have been adopted in specific projects, such as Bank of Abyssinia's Temenos Infinity software implementation, is essential. This assessment provides a baseline for evaluating the current state of DevOps integration and highlights areas needing improvement. However, many banking institutions, including Bank of Abyssinia, have not systematically documented or evaluated their DevOps adoption levels, leaving a void in both academic research and practical applications. Failure to effectively integrate DevOps can result in numerous issues, including prolonged software development cycles, increased operational risks, and potential non-compliance with regulatory requirements. Moreover, the complexity of legacy systems can hinder the automation and collaboration essential for successful DevOps implementation (Bass, Weber, & Zhu, 2015).

In addition to the level of adoption, it is imperative to explore the perceived barriers to and facilitators of adopting DevOps within the bank's project management framework. Banks operate under stringent regulatory requirements and have complex workflows that can either hinder or support the integration of DevOps practices. Identifying these barriers and facilitators from the perspectives of various stakeholders—including developers, IT operations, project managers, and business executives—can provide invaluable insights into optimizing DevOps practices within the banking sector.

This study aims to address these gaps by investigating the coordination of DevOps integration through the lens of integration management in the context of the Bank of Abyssinia's Temenos Infinity system implementation. By examining success factors, assessing the extent of DevOps adoption, and exploring perceived barriers and facilitators, this research seeks to provide comprehensive insights and practical guidelines for effectively integrating DevOps in banking projects. This approach not only contributes to the academic understanding of DevOps

implementation in the banking sector but also offers actionable recommendations for practitioners aiming to enhance their software delivery processes.

1.3 Research Questions

The study seeks to answer the following research questions:

1. What is the extent to which DevOps practices have been adopted in the bank's Temenos Infinity software implementation project?
2. What are the perceived barriers to and facilitators of adopting DevOps practices within the bank's project management framework?

1.4 Objectives of the Study

1.4.1 General objective

The general objective of this study is to analyze and evaluate the adoption, implementation, and impact of DevOps practices within Bank of Abyssinia's Temenos Infinity software implementation project, with a focus on identifying barriers, and facilitators within the bank's project management framework.

1.4.2 Specific Objectives

The specific objectives of this study are:

1. To assess the extent to which DevOps practices have been adopted in the bank's Temenos Infinity software implementation project.
2. To investigate the perceived barriers to and facilitators of adopting DevOps practices within the bank's project management framework.

1.5 Significance of the Study

The significance of the study lies in its potential to address critical gaps in the adoption and integration of DevOps practices within the banking sector, particularly in complex projects like the Temenos Infinity system implementation at the Bank of Abyssinia. By focusing on the

integration of DevOps practices within this sector, the study addresses industry-specific challenges related to compliance, legacy systems, and the need for seamless customization and integration. The banking sector operates within a highly regulated environment and relies heavily on complex IT systems, making the successful adoption of DevOps practices crucial for operational efficiency and regulatory compliance.

Successful DevOps implementation can significantly enhance operational efficiency by streamlining software delivery processes, reducing development cycles, and minimizing operational risks. By identifying practical steps, critical success factors, and barriers to adoption, the study provides valuable insights for banks aiming to improve their operational efficiency through DevOps. Additionally, the study contributes to the academic understanding of DevOps implementation in the banking sector by exploring theoretical foundations, practical applications, and integration management reflections. This enriches academic discourse and provides a foundation for future research in this area.

The insights and practical guidelines generated by the study have direct implications for practitioners in the banking sector. Project managers, IT professionals, and other stakeholders can use the study's findings to optimize their DevOps practices, mitigate adoption challenges, and align DevOps with integration management frameworks effectively. Furthermore, the study's findings can inform strategic decision-making processes within banking institutions, guiding investments in technology, infrastructure, and human resources. By understanding the extent of DevOps adoption and its impact on project management frameworks, decision-makers can make informed choices to drive organizational growth and competitiveness.

1.6 Scope of the study

The study mainly focuses on coordination of DevOps practices with an emphasis on its influence on integration of project components. It includes stakeholder perspectives from key departments at the head office, namely online systems team, core system team, datacenter and server management team and middleware team to provide a comprehensive analysis.

By concentrating on these focus areas and incorporating insights from relevant stakeholders, this study aims to elucidate the integration management challenges and opportunities associated with

adopting DevOps in the banking sector. The goal is to provide actionable recommendations for improving coordination, change management, and risk management processes that are used in the Temenos Infinity software implementation at the Bank of Abyssinia, thereby contributing to a broader understanding of DevOps practices and their impact on project management within the banking industry.

1.7 Limitation of the study

This study acknowledges several limitations that may impact the comprehensiveness and generalizability of its findings. One significant limitation is related to data access. Given the sensitive nature of banking operations, certain confidential data from the Bank of Abyssinia may not be accessible, potentially restricting the depth of analysis on some aspects of the Temenos Infinity software implementation project. Furthermore, the generalizability of the study's findings is limited to financial institutions with similar organizational structures. This constraint means that while the insights gained may be highly relevant to similar banks, they may not fully apply to institutions with different operational frameworks or regulatory environments.

1.8 Structure of the Thesis

The thesis comprises five chapters. Chapter 1, the Introduction, establishes the foundation by addressing the background, problem statement, objectives, research questions, significance of the study, scope and limitations. Chapter 2, the Literature Review, delves into both conceptual and empirical studies concerning DevOps and integration management. Chapter 3, Methodology, outlines the research design, data collection methods, and analysis techniques utilized in the study. Chapter 4, Findings and Analysis, presents the study's results and analyzes them in relation to the research questions. Finally, Chapter 5, Discussion and Conclusion, synthesizes the key findings, discusses their implications, and offers recommendations for future research and practice.

1.9 Definition of Key Terms

DevOps – is a methodology that integrates development and operations teams to automate and streamline the software development lifecycle for faster and more reliable software delivery.

Code – refers to instructions written in a computer programing language that can be executed by a computer to perform specific tasks or operations.

Continuous Integration (CI) - is a development practice where code changes are frequently integrated into a shared repository, followed by automated testing and validation, enabling early detection and resolution of issues in the development process.

Continuous Delivery (CD) – is a software engineering approach where code changes are automatically built, tested, and prepared for production deployment, ensuring that software can be reliably released at any time with minimal manual intervention.

Infrastructure as Code (IaC) – is a practice where infrastructure provisioning and management are done through code and configuration files, enabling consistent and repeatable deployment, scaling, and management of IT infrastructure.

Software development – is the process of creating computer programs or applications by writing, designing, testing, and maintaining code. It involves various stages, including requirements gathering, planning, coding, testing, deployment, and maintenance, and it can be performed by individuals or teams using different programming languages, tools, and methodologies.

Integration management – involves coordinating various project management processes and activities to achieve the project's objectives (Project Management Institute, 2021).

Temenos Infinity – is a digital banking platform developed by Temenos, a leading provider of banking software solutions.

Chapter 2: Literature Review

This chapter delves into the theoretical foundations and existing research relevant to the implementation of DevOps practices, particularly within the banking sector. This chapter begins by exploring the evolution and core principles of DevOps, emphasizing its significance in enhancing software development and operational efficiencies. It then examines the factors that affect DevOps adoption and also provides an overview of the benefits and potential impacts of DevOps on project integration and management within banking IT projects, setting the stage for the subsequent analysis of the empirical findings.

2.1 Conceptual Literature Review

2.1.1 Introduction to DevOps

2.1.1.1 Defining DevOps

DevOps, a portmanteau of "Development" and "Operations," represents a set of practices aimed at unifying software development (Dev) and IT operations (Ops). This synergy is intended to shorten the system development life cycle while delivering features, fixes, and updates frequently in close alignment with business objectives (Bass et al., 2015).

DevOps is a cultural and technical movement that unifies software development (Dev) and IT operations (Ops) to enhance the speed and quality of software delivery. It encompasses a set of practices, tools, and a philosophy that emphasizes collaboration and communication between software developers and IT operations. The goal of DevOps is to shorten the system development life cycle while delivering features, fixes, and updates frequently and in close alignment with business objectives (Bass et al., 2015).

At its core, DevOps seeks to address the historical separation and friction between development and operations teams. Traditionally, these teams operated in silos, each with its distinct goals and processes. Development teams focused on creating new features and delivering them quickly, often without considering the stability and maintenance of the software. Conversely, operations teams prioritized stability and reliability, sometimes at the expense of rapid development and

deployment. This misalignment often led to delays, inefficiencies, and a lack of responsiveness to market demands (Kim, Humble, Debois, & Willis, 2016).

2.1.1.2 Historical Evolution and Context

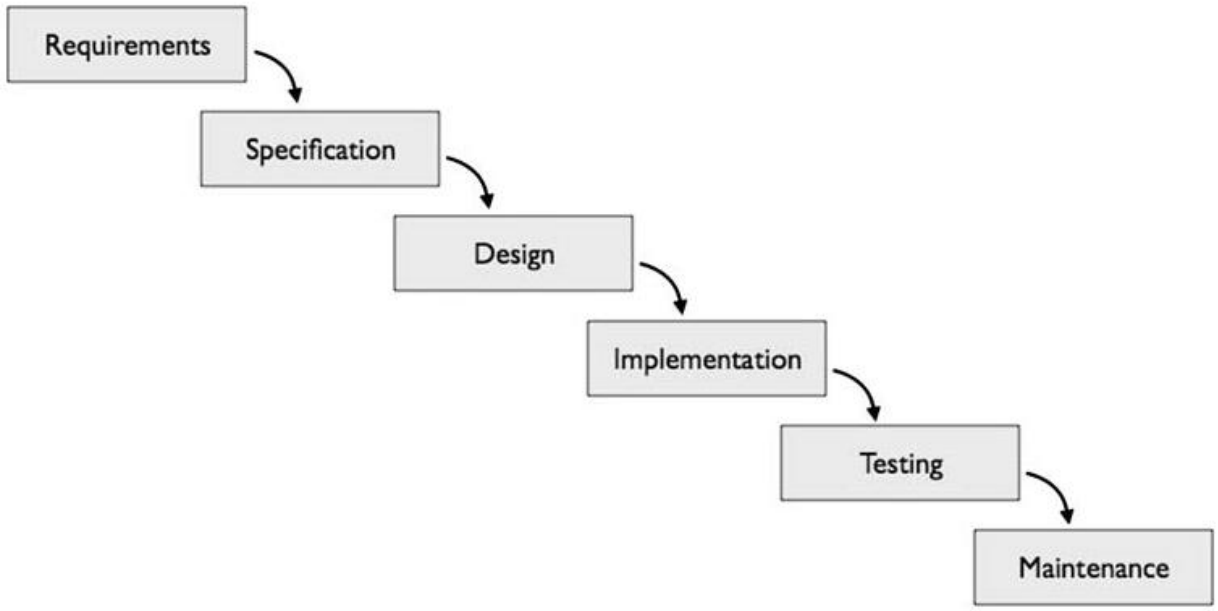
Waterfall Model

The Waterfall model, introduced in the 1970s, was one of the earliest software development methodologies or SDLC, software development lifecycle model. It follows a linear and sequential approach, where each phase of development must be completed before moving on to the next. The phases typically include requirements gathering, design, implementation, verification, and maintenance. This model assumes that all requirements can be clearly defined at the beginning of the project, which rarely reflects the reality of complex software projects (Atlassian, n.d.).

The rigidity of the Waterfall model often resulted in long development cycles and inflexibility in responding to changes. Any change in requirements or design necessitated revisiting and revising previous phases, leading to significant delays and increased costs. Moreover, the Waterfall model's sequential nature meant that integration and testing occurred late in the development process, often resulting in the discovery of critical issues too late to address them efficiently (Royce, 1970).

A typical depiction of waterfall SDLC model is shown in *Figure 2.1*.

Figure 2-1 Common representation of Waterfall



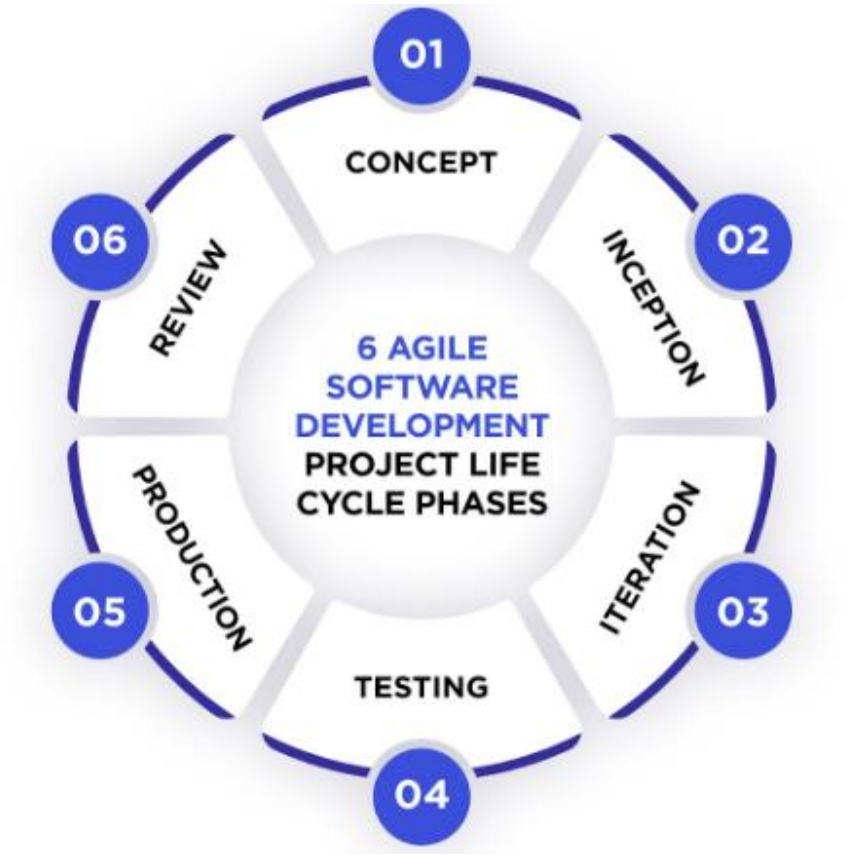
*Source – adopted from a study paper (*Cohn, et al, 2009)

Agile Methodology

In response to the limitations of the Waterfall model, the Agile methodology emerged in the early 2000s, formalized by the publication of the Agile Manifesto in 2001. Agile methodologies, such as Scrum, Kanban, and Extreme Programming (XP), emphasize iterative development, customer collaboration, and responsiveness to change. Agile projects are broken down into small, manageable increments or iterations, typically lasting a few weeks. Each iteration involves a cycle of planning, development, testing, and review, allowing teams to deliver small, functional pieces of software frequently (Beck et al., 2001).

Agile's iterative nature allows for continuous feedback and adaptation, making it more flexible and responsive to changing requirements. However, while Agile improved the development process, it did not fully address the operational challenges associated with deploying and maintaining software. This gap between development and operations persisted, leading to inefficiencies and friction in many organizations (Beck et al., 2001). The depiction of agile methodology is presented in *Figure 2.2*.

Figure 2-2 Diagrammatic depiction of Agile methodology



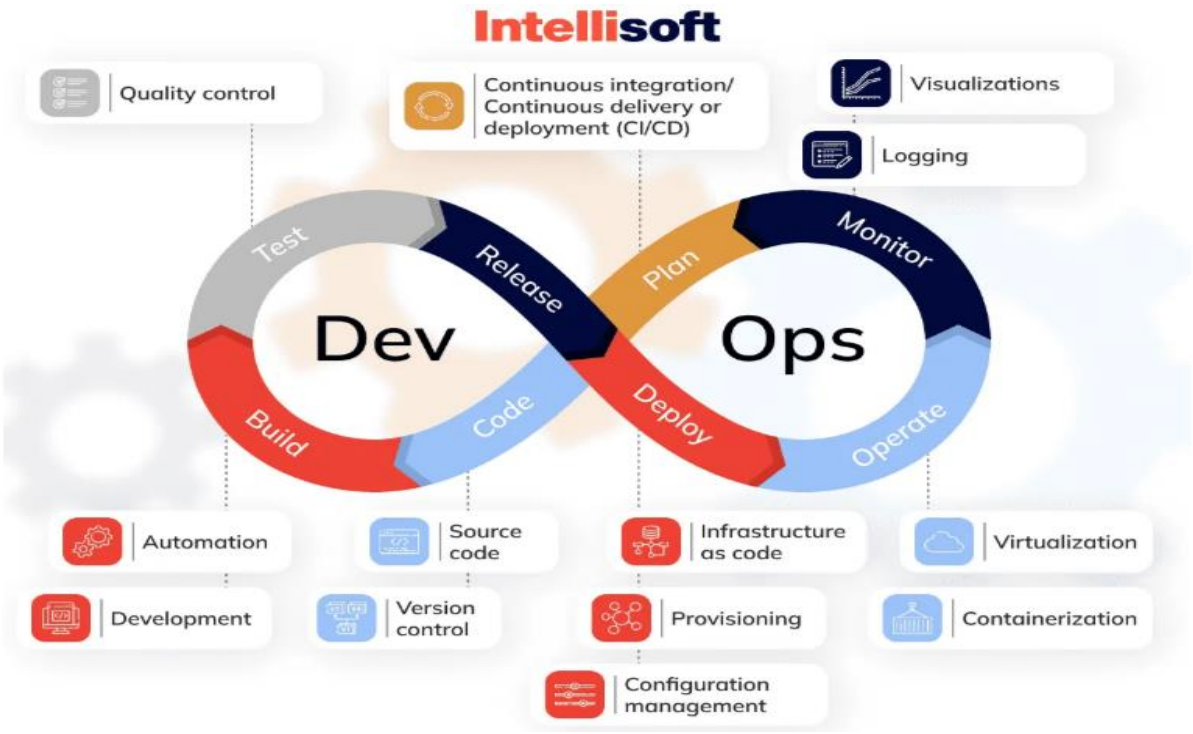
*Source – adopted from relevant.software (*Cohn, et al, 2009)

The Emergence of DevOps

The term "DevOps" was popularized by Patrick Debois in 2009 during the first DevOpsDays event in Ghent, Belgium. The movement emerged from the need to bridge the gap between development and operations, fostering a culture of collaboration, automation, and continuous improvement. DevOps builds on Agile principles, extending them to encompass the entire software delivery lifecycle, from development through to deployment and operations (Debois, 2011).

DevOps emphasizes breaking down silos between development and operations teams, encouraging them to work together towards common goals. By fostering a culture of collaboration and shared responsibility, DevOps aims to improve the overall efficiency, reliability, and quality of software delivery. This cultural shift is supported by a set of practices and tools that automate and streamline various aspects of the software development and deployment process (Mazyar, 2018).

Figure 2-3 Common representation of DevOps



Source – adopted from Intellisoft (2023)

The comparison between agile and DevOps is shown in two tables. The first one, *Table 2.1*, shows their differences from the point of view of different aspects like focus, objective, team structure and so on. The second table, *Table 2.2*, focuses on the problems with Agile practices and the solutions presented by DevOps practices.

Table 2-1 Comparison between Agile and DevOps

Aspect	**Agile**	**DevOps**

Focus	Primarily on software development	Integrates development and operations
Objective	Delivering functional software incrementally	Ensuring continuous delivery and integration
Team Structure	Cross-functional teams (developers, testers)	Combined teams (developers, operations, QA)
Feedback Cycle	Short, iterative feedback from end-users	Continuous feedback from monitoring and operations
Practices	Scrum, Kanban, Sprints	Continuous Integration (CI), Continuous Delivery (CD), Infrastructure as Code (IaC)
Cultural Shift	Collaboration within development team	Collaboration across the entire IT lifecycle (development, operations, QA, security)

Source: own – collected from multiple literatures

The comparison includes only Agile, any other prior frameworks that were historically used before agile are not included. That is because their relation with DevOps is not close enough to compare them with it. Agile, on the other hand, is the latest or even can be referred to as contemporary project management framework available just before DevOps. But still, there are challenges with agile framework that are introduced to development and other relevant teams that work on delivery of applications. DevOps is the contemporary project undertaking practice which tackles these challenges introduced by Agile framework. *Table 2.2* shows the problems in Agile and their respective solution by DevOps from the point of view of different aspects.

Table 2-2 problems in Agile and solutions by DevOps

Aspect	Problems in Agile	Solutions by DevOps
End-to-End Responsibility	Agile lacks focus on operational aspects, leading to "throw it over the wall" mentality (Kim et al., 2016).	DevOps promotes end-to-end responsibility, integrating development and operations (Kim et al., 2016).
Deployment Frequency	Agile's iterative cycles can still lead to infrequent deployments (Erich et al., 2017).	DevOps emphasizes continuous deployment, enabling more frequent releases (Humble & Farley, 2010).
Collaboration	Agile focuses on collaboration within development teams but often excludes operations (Bass et al., 2015).	DevOps fosters cross-functional collaboration, including operations, QA, and security (Kim et al., 2016).
Automation	Agile practices may lack extensive automation, relying more on manual processes (Loukides, 2012).	DevOps prioritizes automation in testing, integration, and deployment (Huttermann, 2012).
Feedback Loop	Agile feedback is primarily from end-users, not from the operational environment (Forsgren et al., 2018).	DevOps integrates continuous feedback from monitoring and operations (Kim et al., 2016).
Scalability	Agile methodologies can struggle with scalability in large, complex environments (Bass et al., 2015).	DevOps practices are designed to scale with automated infrastructure and CI/CD pipelines (Kim et al., 2016).

Source: own – collected from multiple literatures

2.1.1.3 Core Principles and Practices of DevOps

1. **Continuous Integration (CI)**

Continuous Integration (CI) is a cornerstone of DevOps practices, aiming to improve software quality and expedite the software delivery process. At its core, CI is the practice of frequently integrating code changes into a shared repository, multiple times a day. Each integration triggers

an automated build and testing process, ensuring that any integration issues are detected and resolved promptly (Beck, 1999).

The concept of CI emerged as a response to the challenges of integrating code changes in traditional software development methodologies. Historically, development teams worked in isolation on separate features or components, often merging their changes late in the development cycle. This late integration, sometimes referred to as "big bang integration," often resulted in significant integration problems, prolonged debugging sessions, and delays in delivery.

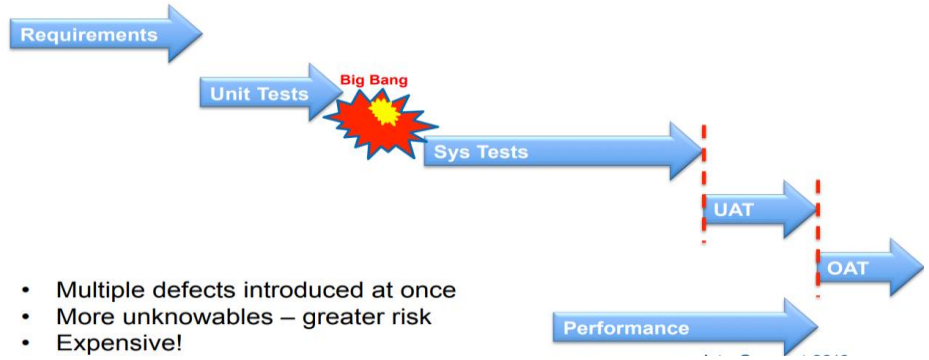
The term "Continuous Integration" was first coined by Grady Booch in the 1990s, and the practice was popularized by the Extreme Programming (XP) methodology, one of the Agile frameworks. XP advocates for integrating code frequently, at least daily, to ensure that the software is always in a releasable state (Beck, 1999).

Key Components of CI:

i. **Automated Builds:**
   - Automated builds are the backbone of CI. Whenever a developer commits code to the version control system (VCS), an automated build process is triggered. This process involves compiling the code, packaging it, and performing initial tests to verify the build's integrity. Automated builds ensure that code changes do not break the application and that the software can be reliably built from source.

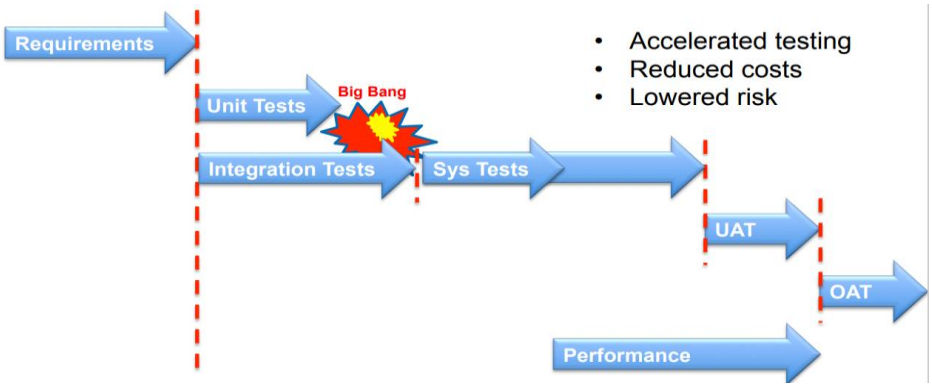
ii. **Automated Testing:**
   - Automated testing is integral to CI, providing immediate feedback on the quality and functionality of the integrated code. Tests are executed automatically as part of the build process, covering various levels such as unit tests, integration tests, and sometimes even end-to-end tests. Automated testing helps in early detection of defects, reducing the cost and effort required for debugging and fixing issues.

Figure 2-4 The old world



Source: from DevOps for Banking by Peter Eeles of IBM, 2016

Figure 2-5 The new world



Source: from DevOps for Banking by Peter Eeles of IBM, 2016

Figures *figure 2.2* and *figure 2.3* depict how automated testing reduces the time taken to identify errors or problems in the development process and consequently the time it takes to solve them.

iii. **Version Control Systems (VCS):**

- VCS such as Git, Subversion, and Mercurial play a critical role in CI by enabling developers to manage code changes, track revisions, and collaborate effectively. VCS facilitates branching and merging strategies, allowing teams to work on multiple features or fixes concurrently while maintaining a stable main branch.

iv. **Continuous Feedback:**

- Continuous feedback is a vital aspect of CI, providing developers with immediate insights into the state of the codebase. CI servers and dashboards display build and test results, alerting teams to any failures or issues. This rapid feedback loop enables

developers to address problems quickly, maintaining the overall health of the project.

The Role of CI in Integration Management

Integration Management, as outlined in the Project Management Body of Knowledge (PMBOK), involves coordinating various project management processes and activities to ensure the seamless integration of all project components. In the context of software development, Integration Management encompasses the processes required to combine and unify the efforts of development and operations teams, ensuring that all aspects of the project are aligned and working harmoniously (Project Management Institute, 2021). More details about integration management are provided in the next section, section 2.1.2.

CI plays a crucial role in Integration Management by providing a structured and automated approach to integrating code changes. Here's how CI aligns with and enhances Integration Management practices:

i. ***Defining Clear Objectives and Requirements*:**

In Integration Management, defining clear objectives and requirements is essential for project success. CI supports this by ensuring that code changes are consistently integrated and tested against defined requirements. Automated tests validate that the integrated code meets the specified functionality and performance criteria, helping teams stay aligned with project objectives.

CI can be used to automate the integration and testing of new features against predefined requirements. This ensures that each code change is validated for compliance with functional and performance standards, reducing the risk of deviations from project goals (Bass, Weber, & Zhu, 2015).

ii. ***Facilitating Collaboration*:**

Effective collaboration between development and operations teams is a fundamental aspect of Integration Management. CI fosters collaboration by providing a shared platform where

code changes are integrated and tested continuously. Developers and operations personnel can work together to resolve integration issues, ensuring that the software is always in a deployable state.

Development and operations teams can use CI tools like Jenkins or Travis CI to collaboratively manage the integration process. Automated builds and tests provide immediate feedback on code changes, enabling teams to identify and address issues collectively, enhancing overall collaboration (Kim et al., 2016).

iii. ***Managing Changes*:**

Managing changes effectively is critical in Integration Management. CI helps manage changes by integrating code frequently and providing automated testing and feedback mechanisms. This ensures that changes are incorporated smoothly and that any potential issues are identified and addressed early in the development cycle.

In the context of a banking software customization project, CI allows the development team to integrate changes continuously. Automated tests validate these changes, ensuring that they do not introduce defects or negatively impact existing functionality. This proactive approach to change management helps maintain the stability and reliability of the software (Humble & Farley, 2010).

iv. ***Monitoring and Controlling Project Work*:**

Monitoring and controlling project work involves tracking the progress of the project and ensuring that it stays on course. CI provides real-time visibility into the state of the codebase, enabling project managers to monitor the progress of development and integration activities. CI dashboards and reports offer insights into build and test results, helping managers identify potential issues and take corrective actions promptly.

Project managers can use CI dashboards to monitor the status of builds and tests. Real-time feedback from CI tools allows managers to track the integration of new features and

identify any issues that need to be addressed, ensuring that the project stays on track (Forsgren, Humble, & Kim, 2018).

v. ***Ensuring Quality and Reliability*:**

Ensuring the quality and reliability of the software is a key aspect of Integration Management. CI contributes to quality assurance by providing a framework for continuous testing and feedback. Automated tests verify that code changes meet quality standards, and continuous integration ensures that the software is always in a deployable state.

CI can be used to implement a comprehensive testing strategy. Automated tests validate the functionality, performance, and security of the software, ensuring that each integration meets the required quality standards. This continuous validation process enhances the reliability of the software (Garousi & Elberzhager, 2017).

Hence, key benefits of CI can be summarized below:

- **Early Detection of Defects:** By integrating and testing code changes frequently, teams can identify and fix defects early in the development process, reducing the cost and effort required to resolve issues.
- **Reduced Integration Problems:** Frequent integration helps to mitigate the risk of integration problems that typically occur when merging large code changes at the end of a development cycle.
- **Improved Collaboration:** CI encourages collaboration among developers, as they frequently share and integrate their work, leading to a more cohesive and synchronized development process (Fowler & Foemmel, 2006).

Continuous Integration is a fundamental practice in DevOps that significantly enhances Integration Management in software projects. By automating the integration and testing of code changes, CI provides a structured and efficient approach to managing the complexities of software development. It fosters collaboration, improves quality, and ensures that the software is always in a releasable state. In the context of banking software implementation and customization, CI plays

a crucial role in ensuring that the software meets defined requirements, integrates smoothly with existing systems, and delivers high-quality, reliable functionality (Fowler & Foemmel, 2006).

2. **Continuous Delivery (CD)**

Continuous Delivery extends the principles of CI by automating the deployment process, ensuring that code changes are automatically tested and prepared for release to production. The goal of CD is to enable organizations to deliver software updates quickly, reliably, and with minimal manual intervention.

CD pipelines typically involve several stages, including:

- **Build:** The code is compiled and built into an executable format.
- **Test:** Automated tests, including unit tests, integration tests, and end-to-end tests, are executed to validate the functionality and quality of the software.
- **Release:** The software is packaged and prepared for deployment.
- **Deploy:** The software is deployed to production environments.

By automating these stages, CD ensures that software can be released at any time, reducing the time-to-market for new features and enabling rapid response to customer feedback (Humble & Farley, 2010).

Once again, **Integration Management** seeks to unify and coordinate various project elements to ensure they work together seamlessly. It involves monitoring and controlling project work to ensure quality and address risks, managing changes to project scope and objectives ensuring they are integrated smoothly into the project plan, promoting collaboration and communication among stakeholders to align project goals and activities. It also involves continuous monitoring and feedback to improve project performance. Ensuring that all aspects of the project are harmonized and streamlined for effective execution is also the goal of integration management.

CD's principles and practices align closely with the goals of integration management, enhancing project coordination and delivery in several ways:

Unified Project Components: CD automates the integration and deployment processes, ensuring that all software components are continuously integrated and ready for deployment. This reduces integration issues and aligns development with operational requirements.

Quality Assurance and Risk Mitigation: CD implements comprehensive automated testing to catch defects early in the development cycle. This proactive approach minimizes the risk of defects reaching production and enhances overall project quality.

Efficient Change Management: CD facilitates frequent and incremental releases, allowing for continuous delivery of updates and new features. This adaptability helps manage changes more efficiently and reduces the complexity of integrating significant changes.

Enhanced Collaboration and Communication: CD encourages collaboration between development, operations, and QA teams, breaking down silos and fostering a culture of shared responsibility. This improves coordination and ensures that all teams are aligned with the project's objectives.

Continuous Improvement and Feedback: CD provides continuous feedback through automated monitoring of applications in production. This feedback loop helps teams to quickly identify issues, gather insights, and make improvements, thus enhancing the overall project performance.

Streamlined Deployment Process: CD streamlines the deployment process through automation, reducing manual intervention, and human error. This efficiency ensures faster time-to-market and more reliable deployments, which is crucial for meeting project deadlines and objectives (Humble & Farley, 2010).

By ensuring that software is always in a deployable state, CD reduces integration risks, improves quality, and facilitates continuous improvement. This alignment supports the goals of Integration Management by promoting seamless coordination, efficient change management, and enhanced collaboration among project stakeholders. Ultimately, CD contributes to the successful delivery of high-quality software that meets business objectives and stakeholder needs.

3. **Automated Testing**

Automated testing is a crucial aspect of CI/CD pipelines. It involves writing tests that run automatically to validate the functionality, performance, and security of the software. Automated testing helps developers discover their mistakes quickly (usually within minutes), which enables faster fixes as well as genuine learning— learning that is impossible when mistakes are discovered six months later during integration testing, when memories and the link between cause and effect have long faded (Kim et al., 2016).

Different levels of testing are employed to ensure comprehensive coverage:

i. **Unit Tests** are tests that verify the functionality of a specific section of code, typically at the function or method level. Its purpose is to ensure that individual components of the software work as expected in isolation.
ii. **Integration Tests:** are tests that validate the interactions between different modules or services. Its purpose is to ensure that integrated components function correctly together. Tools like TestNG, Apache Camel for Java, and Postman for API testing can be used.
iii. **System Tests:** are tests that evaluate the entire system's compliance with specified requirements. They are used to Validate the complete and integrated software product to ensure it meets the specified requirements.
iv. **Acceptance Tests:** are tests that assess the system's behavior against user requirements and business processes, and they are used to verify that the software meets the business needs and is ready for production deployment.
v. **Performance Tests:** are tests that measure the software's performance under load, stress, and varying conditions. The purpose of these kinds of testing is to ensure that the software performs well under expected and peak load conditions.
vi. **Security Tests:** are tests that identify vulnerabilities in the software. They are done to ensure that the software is secure against various threats and attacks.

Automated testing helps to identify defects early, reduce manual testing effort, and ensure that code changes do not introduce regressions (Garousi & Elberzhager, 2017). Automated testing is integral to DevOps as it enhances both the speed and efficiency of the development cycle. By supporting continuous testing, automated tests can be executed automatically whenever code changes occur, providing rapid feedback to developers and enabling faster, more reliable releases.

Automated tests eliminate the likelihood of human error, ensuring consistent and repeatable test results, and can cover more scenarios than manual testing, thus providing comprehensive validation of the software. Moreover, automated tests are seamlessly integrated into Continuous Integration (CI) and Continuous Delivery (CD) pipelines, ensuring continuous testing throughout the development lifecycle and early detection of issues. This integration supports a culture of shared responsibility among development, QA, and operations teams, fostering improved collaboration and communication. Furthermore, automated testing scales efficiently, handling large codebases and complex applications, and supports parallel execution to further speed up the testing process and ensure rapid feedback (Kim et al., 2016).

4. **Infrastructure as Code (IaC):**

Infrastructure as Code (IaC) is a fundamental principle in DevOps that involves managing and provisioning computing infrastructure through machine-readable configuration files rather than through physical hardware configuration or interactive configuration tools. IaC is central to the automation and streamlining of IT infrastructure processes, ensuring that environments are consistent, repeatable, and scalable.

IaC treats infrastructure configuration in a similar manner to software development. By writing code to manage configurations and automate provisioning, IT operations teams can leverage version control, continuous integration, and other software development practices. This approach significantly reduces the time and effort required to deploy infrastructure, minimizes human error, and ensures that infrastructure configurations are consistent across different environments (Hüttermann, 2012).

Common IaC tools include Terraform, Ansible, and Puppet. These tools enable the automation of infrastructure provisioning, configuration management, and deployment, allowing teams to treat infrastructure as a first-class citizen in the software delivery process (Huttermann, 2012).

Core components of IaC can be summarized as the following.

i. *Declarative and Imperative Approaches*:

- Declarative: Specifies *what* the infrastructure should look like (e.g., Terraform, AWS CloudFormation).
- Imperative: Details *how* to achieve the desired infrastructure state (e.g., Ansible, Chef).

ii. *Version Control*: infrastructure configurations are stored in version control systems (e.g., Git), enabling tracking of changes, collaboration, and rollback capabilities.

iii. *Automation*: automation tools execute the infrastructure code, facilitating consistent environment setups and reducing manual interventions.

From its definition and nature as defined above, the benefits of IaC can be categorized and summarized as follow in terms of different attributes.

- *Consistency*: Ensures uniformity across different environments, eliminating configuration drift.
- *Speed and Efficiency*: Automates infrastructure setup, reducing time from days/weeks to minutes/hours.
- *Scalability*: Easily replicates infrastructure for different environments (development, testing, production).
- *Version Control and Collaboration*: Enables better team collaboration and change tracking through version control systems.
- *Reduced Risk*: Minimizes human error and increases reliability and predictability of infrastructure deployments (Kief, 2017).

2.1.1.4 Relevance of IaC in DevOps

Infrastructure as Code (IaC) is integral to DevOps practices, providing the foundation for continuous delivery and integration by automating the provisioning and management of infrastructure. It aligns with the DevOps goal of increasing deployment speed and efficiency while ensuring high quality and reliability. IaC enables the automation of environment setups, ensuring consistency across various stages of the pipeline (development, testing, production), which reduces configuration drift and human error, leading to more reliable deployments. IaC scripts can be reused and modified to quickly scale infrastructure up or down based on demand, which is crucial for continuous delivery and scaling applications seamlessly.

With IaC, infrastructure can be quickly recreated in case of failures, providing robust disaster recovery solutions. Moreover, IaC encourages collaboration among development, operations, and security teams by storing infrastructure configurations in version control systems, making changes visible and trackable. This integration of IaC in DevOps practices supports the overall objective of delivering software more rapidly and reliably, while maintaining high standards of quality and compliance (Hüttermann, 2012; Kief, 2017).

Infrastructure as Code (IaC) is closely aligned with integration management within the project management framework and contributes to it by providing:

i. ***Unified Configuration Management***:
    - IaC ensures that the infrastructure configurations are standardized and controlled, which is critical for integrating different project components. This standardization facilitates smoother integration of development and operational workflows.

ii. ***Efficient Resource Utilization***:
    - By automating infrastructure provisioning, IaC ensures optimal use of resources, reducing costs and increasing efficiency. This aligns with integration management's objective of resource optimization to achieve project goals.

iii. ***Enhanced Collaboration***:
    - Integration management focuses on aligning different teams and processes. IaC, by using version control and shared repositories, enhances collaboration among teams, ensuring that changes in infrastructure are transparent and coordinated.

iv. ***Risk Mitigation***:
    - Integration management involves managing risks that could affect project outcomes. IaC minimizes risks related to infrastructure changes by ensuring that configurations are tested, versioned, and easily reversible, thus maintaining project stability.

v. ***Compliance and Auditing***:
    - Compliance with regulatory requirements is critical in many sectors, including banking. IaC facilitates compliance by providing an auditable trail of infrastructure changes, ensuring that all configurations meet compliance standards.

vi. ***Streamlined Deployments***:

- Integration management seeks to harmonize different project activities for seamless execution. IaC supports this by enabling rapid, consistent, and error-free deployments, ensuring that development and operational changes are integrated smoothly into the production environment.

2.1.2 Integration Management and DevOps

Integration Management in project management refers to the processes and activities required to identify, define, combine, unify, and coordinate various project management processes and activities. This encompasses the development of the project charter, project management plan, direct and manage project work, manage project knowledge, monitor and control project work, and perform integrated change control (Project Management Institute, 2021).

2.1.2.1 Coordination of DevOps in Integration Management

The integration of DevOps within the framework of integration management for software implementation and customization involves several steps:

1. **Defining Clear Objectives and Requirements**:
    - Establishing clear objectives for the software, detailing expected functionality, performance, and user experience.
    - Outlining specific customization requirements and integration points with the bank's existing systems (Bass, Weber, & Zhu, 2015).
2. **Setting Up DevOps Pipelines**:
    - Implementing CI/CD pipelines to automate code integration, testing, and deployment processes.
    - Utilizing IaC tools to manage infrastructure provisioning and configuration, ensuring consistency and repeatability in environments (Huttermann, 2012).
3. **Facilitating Collaboration**:
    - Promoting collaboration among the bank’s development team, operations team, and the vendor’s technical support through agile practices, daily stand-ups, and shared repositories.

- Utilizing collaborative tools and practices to enhance communication and coordination (Loukides, 2012).

4. **Implementing Automated Testing**:
    - Integrating automated testing within the DevOps pipeline to ensure that customizations do not introduce defects.
    - Employing unit tests, integration tests, and end-to-end tests to validate customizations (Garousi & Elberzhager, 2017).
5. **Ensuring Continuous Monitoring and Feedback**:
    - Deploying monitoring tools to track system performance, security, and user experience.
    - Establishing feedback loops to continuously gather and act on user feedback and system performance data (Erich, Amrit, & Daneva, 2017).
6. **Managing Security and Compliance**:

    - Integrating security practices into the DevOps pipeline (DevSecOps) to ensure customizations comply with security standards and regulations. DevOps security, more commonly referred to as DevSecOps, refers to the discipline and practice of safeguarding the entire DevOps environment through strategies, policies, processes, and technology. The DevSecOps philosophy is that security should be built into every part of the DevOps life cycle, including inception, design, build, test, release, support, maintenance, and beyond (Chris, 2020).

    - Performing regular security audits and vulnerability scans (Sharma & Coyne, 2018).
7. **Planning for Change Management**:
    - Preparing for ongoing changes and updates post-deployment, ensuring the team is ready to handle new requirements or modifications efficiently.
    - Establishing a change management process that leverages DevOps practices to implement changes swiftly and safely (Kim, Debois, Willis, & Humble, 2016).

2.2 Empirical Review

2.2.1 Case Studies - DevOps Implementations

2.2.1.1 Amazon: Transforming through DevOps

Amazon stands out as a fundamental example of DevOps implementation success. By integrating DevOps practices, Amazon revolutionized its software development and IT operations, significantly enhancing its service delivery and operational efficiency. The company adopted a microservices architecture, which allowed for the development of small, independent services that could be deployed and managed separately (Carter, 2017).

This approach facilitated continuous integration and continuous delivery (CI/CD), enabling Amazon to deploy code changes frequently and reliably. Automated testing and deployment pipelines reduced manual errors and sped up the release cycles. As a result, Amazon achieved remarkable scalability and robustness in its services, handling millions of transactions seamlessly during peak times like Black Friday and Cyber Monday ("DevOps Case Study: Amazon," n.d.).

2.2.1.2 Netflix: Pioneering Chaos Engineering

Netflix is another notable success story in the realm of DevOps. The company adopted DevOps principles to enhance its streaming service's reliability and resilience. A significant innovation by Netflix was the development of Chaos Monkey, a tool that randomly shuts down instances in their production environment to ensure their system can withstand failures without affecting the user experience. This approach, known as chaos engineering, helped Netflix identify weaknesses and build robust, fault-tolerant systems (Cois, 2015).

Moreover, Netflix's adoption of microservices and CI/CD allowed for rapid development, testing, and deployment of new features, thereby maintaining high availability and performance of its streaming service. This strategy not only improved Netflix's operational efficiency but also supported its exponential growth in subscribers and streaming hours ("How Netflix Became A Master of DevOps," 2023).

2.2.2 DevOps in Banking

In the dynamic and competitive landscape of the banking sector, institutions are continually seeking innovative ways to enhance their operational efficiency, reduce costs, and improve customer satisfaction. The advent of digital transformation has necessitated the rapid adoption of

new technologies and methodologies in software development and IT operations. One such methodology that has gained significant traction is DevOps.

The banking sector is characterized by its stringent regulatory requirements and legacy IT systems. It presents unique challenges for DevOps adoption. Banks must ensure that their software systems are not only efficient and reliable but also compliant with regulatory standards such as the General Data Protection Regulation (GDPR) and the Payment Card Industry Data Security Standard (PCI DSS). These regulations mandate strict controls over data privacy, security, and financial transactions, necessitating rigorous testing and validation processes (Sharma & Coyne, 2018).

Furthermore, banks often rely on legacy systems that were not designed for modern development and deployment practices. Integrating these systems with new DevOps tools and processes requires careful planning and execution. The complexity of these systems can impede the automation and collaboration essential for successful DevOps implementation (Bass, Weber, & Zhu, 2015).

2.2.2.1 Case Study: Capital One's DevOps Transformation

Capital One provides a compelling case study of successful DevOps adoption in the banking sector. Facing the need to enhance its software delivery processes and foster innovation, Capital One embarked on a comprehensive DevOps transformation. This involved adopting CI/CD pipelines, automated testing, and IaC to streamline development and operations. The transformation led to significant improvements in software delivery speed, quality, and reliability, demonstrating the potential benefits of DevOps in the banking sector (Kim, Humble, Debois, & Willis, 2016).

Factors Affecting DevOps Implementation:

1. **Cultural Shift**:
    - Emphasizing the importance of a collaborative culture where development and operations teams work closely together.
    - Encouraging experimentation and learning from failures to continuously improve processes (Forsgren, Humble, & Kim, 2018).

2. **Automation and Tooling**:
    - Implementing automated testing and deployment pipelines to reduce manual errors and increase deployment speed.
    - Utilizing tools like Jenkins for continuous integration and delivery, which facilitated more efficient and reliable software releases (Kim, Debois, Willis, & Humble, 2016).
3. **Continuous Improvement and Learning**:
    - Fostering a culture of continuous improvement by regularly reviewing and refining processes.
    - Encouraging feedback and collaboration to identify and address bottlenecks and inefficiencies in the workflow (Forsgren et al., 2018).

Outcomes and Benefits

Capital One experienced significant improvements in software delivery speed, quality, and reliability. The adoption of DevOps practices led to reduced time-to-market for new features and enhancements, improved customer satisfaction through better-performing applications, and enhanced operational efficiency by minimizing downtime and errors during deployments (Kim et al., 2016).

2.3 Case Study Context: Temenos Infinity System

Temenos Infinity stands as a flagship digital banking platform provided by Temenos, a globally renowned provider of banking software systems. Engineered to empower banks and financial institutions, Temenos Infinity offers a comprehensive suite of cutting-edge solutions aimed at revolutionizing the digital banking experience.

At its core, Temenos Infinity is designed to seamlessly integrate with the existing infrastructure of banks, enabling them to deliver personalized and intuitive banking experiences across various channels. By harnessing the power of advanced technology and innovative design principles, Temenos Infinity empowers banks to meet the evolving needs and preferences of their customers in an increasingly digital world (“Temenos Infinity,” 2020).

2.3.1 Key features of Temenos Infinity

Omni-Channel Banking: Temenos Infinity enables banks to deliver consistent and seamless banking experiences across multiple channels, including web, mobile, and other digital touchpoints. This omni-channel approach ensures that customers can access banking services anytime, anywhere, and on any device ("Temenos Infinity," 2020).

Digital Onboarding: With Temenos Infinity, banks can streamline and digitize the customer onboarding process, allowing for quick and convenient account opening and enrollment. Through intuitive interfaces and automated workflows, banks can enhance customer acquisition and reduce onboarding timeframes ("Temenos Infinity," 2020).

Personalization: Temenos Infinity empowers banks to personalize the banking experience for each customer, delivering targeted product recommendations, tailored promotions, and personalized communications. By leveraging data analytics and machine learning algorithms, banks can better understand customer preferences and anticipate their needs.

Self-Service Capabilities: Through self-service features, Temenos Infinity enables customers to perform a wide range of banking transactions independently, without the need for human intervention. From account management to bill payments and fund transfers, customers can access essential banking services with ease.

Integration with Third-Party Services: Temenos Infinity offers seamless integration with third-party services and fintech solutions, allowing banks to expand their service offerings and deliver additional value to customers. By leveraging APIs and open banking standards, banks can foster collaboration and innovation within the broader financial ecosystem.

Security and Compliance: Security is paramount in digital banking, and Temenos Infinity prioritizes robust security measures to protect customer data and prevent unauthorized access. From encryption and access controls to compliance auditing and fraud detection, Temenos Infinity ensures that banks can maintain trust and confidence in their digital channels.

Agile Development and Deployment: Temenos Infinity embraces agile development methodologies and continuous delivery practices, enabling banks to rapidly iterate and deploy new

features and updates. By fostering a culture of agility and innovation, banks can stay ahead of the curve and respond swiftly to changing market dynamics (“Temenos Infinity,” 2020).

With its comprehensive feature set and industry-leading capabilities, Temenos Infinity empowers banks to differentiate themselves in the digital landscape, drive customer engagement, and unlock new revenue opportunities. As banks continue to prioritize digital transformation initiatives, Temenos Infinity remains a trusted partner in their journey towards digital excellence (“Temenos Infinity,” 2020).

2.3.2 Objectives of the Temenos Infinity Implementation

The implementation of the Temenos Infinity System at Bank of Abyssinia is guided by a set of strategic objectives aimed at transforming the bank's digital banking landscape and delivering superior experiences to its customers. These objectives encompass various facets of digital banking excellence and align closely with the bank's overarching strategic goals. Key objectives of the Temenos Infinity implementation include (“Temenos Infinity,” 2020):

Enhancing Digital Banking Experiences: A primary objective of the Temenos Infinity implementation is to enhance digital banking experiences for customers. By leveraging the advanced capabilities of Temenos Infinity, Bank of Abyssinia aims to deliver seamless, intuitive, and personalized banking experiences across all digital touchpoints. From account opening to transaction management and customer support, the bank seeks to streamline and optimize every aspect of the digital banking journey.

Improving Customer Onboarding Processes: The implementation of Temenos Infinity aims to streamline and digitize the customer onboarding process, making it quicker, more efficient, and more user-friendly. By leveraging Temenos Infinity's digital onboarding capabilities, Bank of Abyssinia aims to reduce the time and effort required for customers to open new accounts, enroll in services, and access banking products. This objective is aligned with the bank's commitment to delivering exceptional customer service and enhancing customer satisfaction.

Facilitating End-to-End Digital Banking Journeys: Temenos Infinity enables banks to offer end-to-end digital banking journeys that span the entire customer lifecycle. From initial account setup to ongoing account management and beyond, Bank of Abyssinia aims to leverage Temenos

Infinity to provide comprehensive and seamless banking experiences. By facilitating end-to-end digital banking journeys, the bank aims to increase customer engagement, loyalty, and retention, ultimately driving long-term value for both customers and the bank (“Temenos Infinity,” 2020).

Laying the Foundation for Future Innovation: The implementation of Temenos Infinity serves as a foundational step in Bank of Abyssinia's journey towards digital innovation. By adopting Temenos Infinity as its digital banking platform, the bank aims to establish a robust and scalable foundation that can support future innovation and growth. This includes the ability to rapidly deploy new features, integrate with emerging technologies, and adapt to evolving customer needs and market trends. By laying the foundation for future innovation, Bank of Abyssinia aims to stay ahead of the curve and remain a leader in the digital banking space (“Temenos Infinity,” 2020).

Driving Business Growth and Competitiveness: Ultimately, the overarching objective of the Temenos Infinity implementation is to drive business growth and competitiveness for Bank of Abyssinia. By delivering exceptional digital banking experiences, streamlining operations, and fostering innovation, the bank aims to attract new customers, retain existing ones, and expand its market share. This objective is aligned with the bank's strategic vision of becoming a digital leader in the banking industry and maintaining its position as a trusted financial institution in Ethiopia and beyond ("Temenos Infinity," 2020).

2.4 Analytical Framework of the Study

The analytical framework of this study is designed to examine the implementation and impact of DevOps practices within banking software development projects. This framework integrates the theoretical foundations of DevOps, agile methodologies, and continuous delivery principles, highlighting their interconnections and how they collectively contribute to enhancing software delivery processes.

2.4.1 Key Components of the Analytical Framework

1. **DevOps Practices**:
    - *Continuous Integration (CI):* Frequent integration of code changes into a shared repository, coupled with automated builds and tests, to identify and resolve integration issues promptly.

- *Continuous Delivery (CD)**:*** Automation of the deployment process, ensuring that code changes are always in a deployable state, facilitating frequent and reliable software releases.
- *Automated Testing**:*** Implementation of automated tests at various levels (unit, integration, system, acceptance) to ensure software quality and detect defects early in the development process.
- *Infrastructure as Code (IaC):* Managing and provisioning infrastructure through code, enhancing consistency and reducing manual errors.

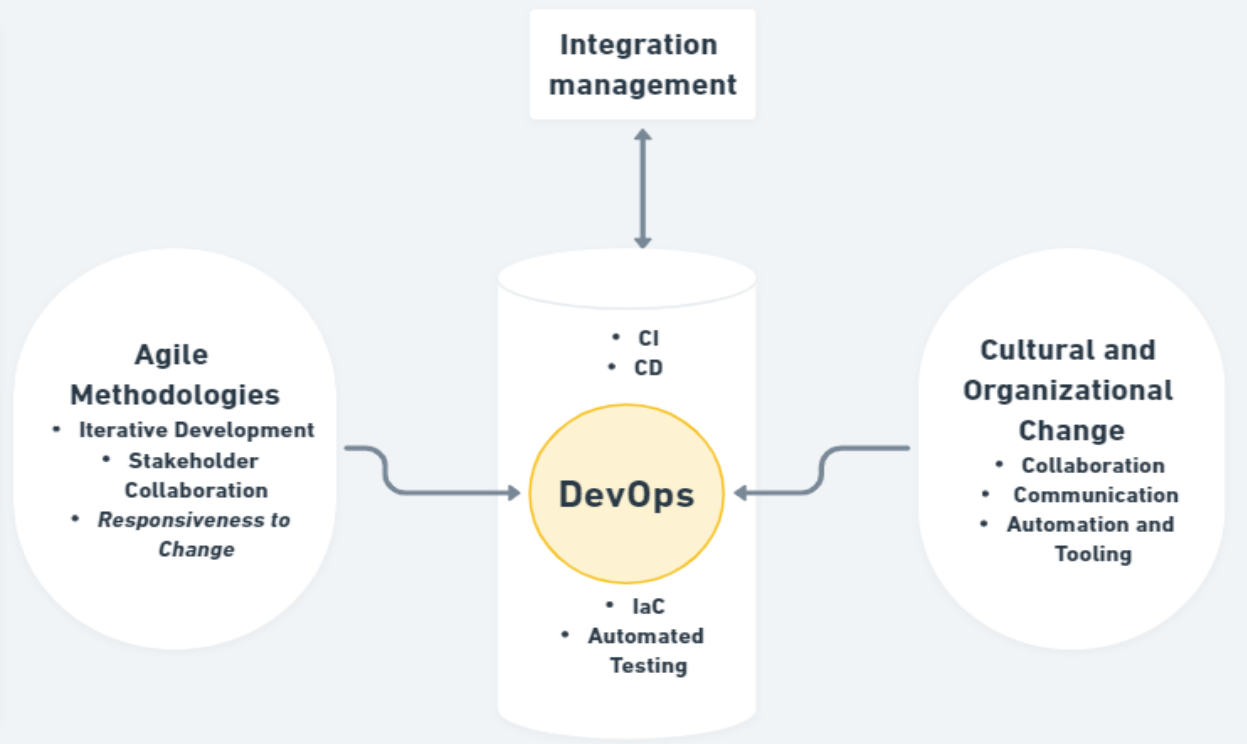
2. **Agile Methodologies**:
    - *Iterative Development**:*** Breaking down projects into smaller iterations, allowing for continuous feedback, adaptation, and delivery of functional software increments.
    - *Customer Collaboration**:*** Involving customers and stakeholders in the development process to ensure that the software meets their needs and expectations.
    - *Responsiveness to Change**:*** Flexibility to adapt to changing requirements and market conditions, ensuring that the software remains relevant and valuable.

3. **Cultural and Organizational Change**:
    - *Collaboration and Communication**:*** Fostering a culture of collaboration between development and operations teams, breaking down separation of teams and promoting shared responsibility.
    - *Automation and Tooling**:*** Utilizing tools and automation to streamline development, testing, and deployment processes, reducing manual effort and increasing efficiency.

2.4.2 Analytical Model Diagram

Figure 2.4 is the analytical model that depicts the framework of the study, highlighting the interrelationships between DevOps practices, agile methodologies, and their impact on software delivery processes.

Figure 2-6 Conceptual Framework Model



Source: own – after reading multiple literature, with the help of tool from https://whimsical.com/

The analytical framework places DevOps at the center, underscoring its role as the integrating force that bridges Agile methodologies, cultural and organizational changes, and project integration management. DevOps practices such as Continuous Integration (CI), Continuous Delivery (CD), Automated Testing, and Infrastructure as Code (IaC) are pivotal in ensuring efficient and reliable software development and deployment. These practices interact closely with one another; for instance, CI and CD facilitate frequent and automated integration and deployment of code changes, while Automated Testing ensures quality by detecting defects early in the development process. IaC supports these processes by providing a consistent and reproducible infrastructure environment, crucial for the seamless operation of CI/CD pipelines. This interconnectedness highlights the importance of DevOps practices in maintaining high standards of software quality and operational efficiency.

Project Integration Management plays a crucial role in this framework by orchestrating and aligning various project components towards achieving the project's objectives. It ensures that Agile methodologies and cultural changes are effectively integrated with DevOps practices to foster an adaptive, responsive, and collaborative environment. Agile practices like Iterative Development and Stakeholder Collaboration promote continuous feedback and adaptability, which are essential for meeting user needs and market demands. Cultural and organizational changes, such as fostering collaboration between development and operations teams and utilizing automation and tooling, enhance efficiency and reduce manual effort. By positioning DevOps at the center, this conceptual framework emphasizes its critical role in integrating these diverse elements to enhance project success and operational excellence.

Chapter 3: Research Methodology

This chapter outlines the research methodology employed in this study to investigate the integration of DevOps practices within the Bank of Abyssinia's Temenos Infinity system implementation. A robust research design is essential to ensure the reliability and validity of the findings. This chapter provides a detailed description of the research design, study variables, study area, and target population, as well as the sampling techniques and sample size. It also outlines the data collection methods, data analysis techniques, and the steps taken to ensure the reliability and validity of the study. Additionally, the ethical considerations adhered to during the research process are discussed.

3.1 Research Approach

The research design for this study is a mixed-methods approach, integrating both quantitative and qualitative research methods to provide a comprehensive understanding of the DevOps integration in the Bank of Abyssinia's Temenos Infinity system implementation. The mixed-methods approach allows for the triangulation of data, ensuring the validity and reliability of the findings.

Quantitative data is collected through structured questionnaires distributed to relevant stakeholders, including IT professionals, developers and business officers. This data will provide measurable insights into the extent of DevOps adoption, critical success factors, and barriers to implementation. Qualitative is gathered from the above-mentioned stakeholders through semi-structured interviews, offering in-depth perspectives on the experiences and perceptions of the stakeholders involved in the project. This approach ensures that both numerical data and rich, contextual information are captured, providing a holistic view of the DevOps integration process.

This study employed a concurrent mixed-methods research approach, mainly focusing on quantitative data collection and analysis, complemented by qualitative insights. The quantitative aspect of the research involves a detailed analysis of survey responses, providing statistical evidence to understand the extent of DevOps implementation, the barriers and facilitators within the bank's Temenos Infinity software project. The qualitative component, even though smaller, enriches the study by offering contextual depth and understanding through interviews and open-

ended questionnaire responses. This blended approach allows for a comprehensive examination of the research questions, ensuring that the findings are robust and reflective of both numerical trends and personal experiences within the organization.

3.2 Research design

For this study on the adoption of DevOps practices in the Temenos Infinity system implementation at Bank of Abyssinia, a descriptive research design has been chosen. This design allows for a comprehensive understanding of the current state of DevOps adoption, the perceived barriers and facilitators.

This research design focuses on describing the extent to which DevOps practices have been adopted in the Temenos Infinity software implementation project, perceived barriers and facilitators. This involves collecting detailed information on the specific practices being used, their frequency, and the departments involved.

3.3 Description of Key Study Factors

The study factors measured by the efficiency, speed, and quality of software delivery, as well as overall project success are discussed as follow.

- **Adoption of DevOps Practices:** The extent to which DevOps practices have been implemented in the project.
- **Facilitators:** Factors that facilitate the successful implementation of DevOps, such as management support, team collaboration, and automation tools.
- **Barriers to Adoption:** Challenges faced in implementing DevOps, including resistance to change, lack of skills, and legacy systems.
- **Integration Management Practices:** The alignment of DevOps with integration management processes, such as coordination of project activities and compliance with regulatory standards.

3.4 Description of Study Area and Target Population

The study area is the head office of Bank of Abyssinia, focusing on specific departments involved in the Temenos Infinity system implementation: the digital products (business requirements), online systems team (development and operations), datacenter and server management team (to handle infrastructure), middleware team and network team. The target population consists of employees from these departments who are directly involved in the system's implementation process. This includes IT professionals, project managers, developers, system administrators, and other relevant stakeholders. By targeting these specific groups, the study aims to capture detailed and relevant data on the DevOps integration process within the bank.

3.5 Sampling Techniques and Sample Size

A purposive sampling technique is used to select participants who have significant experience and involvement in the DevOps implementation at Bank of Abyssinia. This non-probability sampling method is appropriate for this study as it ensures that only knowledgeable and relevant participants are included, thereby enhancing the quality and reliability of the data.

The sample size is determined based on the principle of data saturation for qualitative interviews and a sufficient number to ensure statistical significance for quantitative questionnaires. The study targeted 51 participants which were selected and surveyed for quantitative data. 3 participants were interviewed for qualitative insights. The sample consists of all stakeholders that were directly or indirectly involved on the project. This sample size is expected to provide a balanced representation of the perspectives and experiences of the stakeholders involved.

3.6 Data Collection Methods and Tools

Data will be collected from both primary and secondary sources. Primary data will be gathered through:

- **Questionnaires:** Structured questionnaires will be used to collect quantitative data on the extent of DevOps adoption, facilitators and barriers to implementation. The questionnaires will include closed-ended questions with Likert scale responses to ensure consistency and ease of analysis and one open-ended question.

- **Interviews:** Semi-structured interviews will be conducted with key stakeholders to obtain qualitative insights into their experiences and perceptions of DevOps integration. The interviews will follow an interview guide to ensure that all relevant topics are covered while allowing for flexibility in responses.

Secondary data are obtained from organizational reports, project documentation, and relevant literature on DevOps and integration management.

3.7 Data Analysis and Presentation

Quantitative data is analyzed using a statistical technique called descriptive statistics. Software tool, SPSS, is used to perform these analyses, ensuring accuracy and reliability in the results. The analysis focuses on identifying patterns in the context of aforementioned factors.

Qualitative data is analyzed using thematic analysis. Transcripts from interviews are coded and categorized into themes that align with the research questions. The integration of quantitative and qualitative data will provide a comprehensive understanding of the implementation process.

3.8 Reliability and Validity Analysis

3.8.1 Reliability

The reliability of the questionnaires used in this study is crucial to ensure that the results are consistent and reproducible. Internal consistency, measured by Cronbach's alpha, is a primary method to assess the reliability of the questionnaires. A Cronbach's alpha value of 0.7 or higher indicates acceptable reliability (Tavakol & Dennick, 2011). To ensure this, each questionnaire item is designed to measure the same underlying construct, such as the extent of DevOps adoption, perceived barriers, and facilitators.

Using SPSS, Cronbach’s alpha is calculated. As indicated in *Table 3-1*, the Cronbach’s alpha is 0.708 which is above the acceptable alpha value, 0.7.

Table 3-1 Reliability test using Cronbach's alpha

Reliability Statistics		
Cronbach's Alpha	Cronbach's Alpha Based on Standardized Items	N of Items
.708	.701	51

Source: Reliability analysis from SPSS

3.8.2 Validity

Validity ensures that the questionnaires accurately measure the constructs they are intended to measure. Content validity is ensured by involving experts' recommendations in DevOps and integration management during the development of the questionnaires, ensuring that as much relevant aspects as possible of these concepts are covered comprehensively.

3.9 Ethical Consideration

The study adhered to strict ethical guidelines to ensure the integrity and ethical conduct of the research. Key ethical considerations include:

- **Informed Consent:** Participants are informed about the purpose of the study, their role, and their rights, including the right to withdraw at any time. Written consent is obtained from all participants.
- **Confidentiality:** Participants' identities and responses will be kept confidential. Data will be anonymized to protect participants' privacy.
- **Data Security:** All data collected is securely stored and only accessible to the researcher. Digital data is password-protected, and physical data (if any) will be stored in a locked cabinet.

Chapter 4: Data Analysis and Interpretation

This chapter presents the data analysis and interpretation of the research findings on the adoption and integration of DevOps practices in the Temenos Infinity system implementation at the Bank of Abyssinia. The analysis is based on data collected through questionnaires and semi-structured interviews with key stakeholders across various divisions of the bank. The chapter begins with a demographic overview of the respondents, followed by descriptive statistics

The analysis includes both descriptive and statistical analysis. Data analysis was done by using a software by IBM, ***IBM SPSS Statistics version 29.0.2.0(20)***.

4.1 Demographic Information of the Respondents

The demographic information of the respondents provides a context for understanding the data collected. A total of 51 respondents participated in the study, representing various roles and departments within the bank. The demographic variables include age, gender, job role/position, years of experience, department/division, and educational background.

4.1.1 Gender and Age

Table 4-1 Frequency analysis of age

Age					
		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	25 years to 30 years	44	86.3	86.3	86.3
	30 years or above	5	9.8	9.8	96.1
	less than 25 years	2	3.9	3.9	100.0
	Total	51	100.0	100.0	

Source: own survey – generated from SPSS

The majority of respondents were males, accounting for 68.6% (8) of all responders and the rest 31.4% (4) were females. As shown in *Table 4-1*, the respondents' age distribution reveals a significant concentration in the 25 to 30 years bracket, with 44 out of 51 respondents (86.3%) falling into this category. Additionally, there are 2 respondents under 25 years and 5 respondents

(9.8%) aged 30 and above. This distribution suggests that the bank's Temenos Infinity software implementation project is predominantly staffed by young professionals who are typically more adaptable to new technologies like DevOps. However, they may require additional training and support to effectively implement these practices due to potential gaps in extensive professional experience. The younger demographic's high energy and innovative thinking can drive the adoption of modern practices if supported by appropriate resources and management. Moreover, the presence of more experienced individuals provides opportunities for mentorship and knowledge transfer, balancing youthful enthusiasm with seasoned expertise.

4.1.2 Educational Background

Table 4-2 Frequency analysis of education level

Education Level					
		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	Bachelor	47	92.2	92.2	92.2
	Masters	4	7.8	7.8	100.0
	Total	51	100.0	100.0	

Source: own survey – generated from SPSS

Among the 51 respondents, as shown in *Table 4-2,* the respondents' educational backgrounds reveals that the vast majority hold a bachelor's degree, with 47 respondents (92%) having achieved this level of education. In contrast, only 4 respondents (8%) have a master's degree. This distribution indicates a predominantly undergraduate-educated team, suggesting a solid foundational knowledge suitable for implementing and understanding new practices such as DevOps. The presence of a few team members with master's degrees adds a higher level of expertise and potentially more advanced skills, which can be advantageous in strategic decision-making and complex problem-solving.

4.1.3 Work Experience

In the analysis of the collected data, *Table 4-3*, the respondents' work experience reveals that a significant majority (86.3%) have 2 to 5 years of experience, indicating that the team consists largely of relatively early-career professionals. In contrast, 5 respondents (16%) have over 10 years of experience, and only 2 respondents (4%) have less than 2 years of experience. This distribution

suggests that while the team has a strong foundation of mid-level experience, there is also a valuable presence of highly experienced professionals who can provide mentorship and strategic insights. The early-career professionals likely bring fresh perspectives and a willingness to adopt new practices like DevOps, though they may require more structured guidance and training. The blend of mid-level and senior experience could be leveraged to create a balanced approach to implementing DevOps, combining innovative practices with the stability and depth of knowledge provided by more seasoned team members.

Table 4-3 Frequency analysis of education level

Work experience					
		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	2 to 5 years	44	86.3	86.3	86.3
	Above 10 years	5	9.8	9.8	96.1
	Less than 2 years	2	3.9	3.9	100.0
	Total	51	100.0	100.0	

Source: own survey – generated from SPSS

4.1.4 Job Position

Table 4-4 Frequency analysis of job position

Respondent's Job Position					
		Frequency	Percent	Valid Percent	Cumulative Percent
Valid		2	3.9	3.9	3.9
	Administrator	10	19.6	19.6	23.5
	Analyst and Developer	14	27.5	27.5	51.0
	Division Manager	2	3.9	3.9	54.9
	Officer (Junior/Associate/business)	23	45.1	45.1	100.0
	Total	51	100.0	100.0	

Source: own survey – generated from SPSS

The analysis of the respondents' job positions, as shown in *Table 4-4*, indicates a diverse distribution within the organization. The largest proportion of respondents, comprising 23 individuals (45%), hold the position of Officer. Following closely behind are Analysts and Developers, representing 14 respondents (27%). Administrators account for 10 individuals (20%), while Division Managers make up the smallest group, with 2 respondents (4%). This distribution

reflects a varied mix of roles and responsibilities within the organization, suggesting a multifaceted approach to DevOps implementation. The significant representation of Officers and Analysts/Developers suggests that these roles may play pivotal roles in the adoption and execution of DevOps practices.

4.1.5 Response Rate

The data collection for this study achieved a high response rate. Out of 53 targeted respondents, 51 provided complete responses, resulting in a response rate of approximately 96.2%. Only 2 respondents did not participate. This high response rate underscores the engagement and relevance of the study to the participants, which in turn enhances the reliability and validity of the findings.

4.2 Descriptive Statistics

The descriptive statistics section provides an overview of the key variables measured in the study, including the extent of DevOps adoption, perceived barriers, facilitators of DevOps practices, and the observed benefits.

Table 4-5 response: extra relevant information about respondents

What team member do you consider yourself as					
		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	Development team	27	52.9	52.9	54.9
	Both Dev and Operations	10	19.6	19.6	72.5
	Operations	14	27.5	27.5	100.0
	Total	51	100.0	100.0	

Source: own survey – generated from SPSS

Among the respondents, 27 individuals (52.9%) consider themselves as a development team member and 14 of them (27.5%) as operations. 10 of the respondents also said that they consider themselves as both Dev and Ops team member indicating that there is no a clear line set by the organization in its structure.

In another question presented for respondents about their role in the project, 60.7% of them were responsible for integration highlighting the critical task of ensuring seamless connections between various system components. Both testing and development were the responsibility of 16

respondents (31.3%), reflecting the importance of verifying software functionality and quality. Additionally, 4 respondents (7.8%) were engaged in requirement gathering, crucial for understanding and documenting project needs, while another 4 respondents (7.8%) focused on resource provisioning and support, vital for maintaining the necessary infrastructure and operational support.

4.2.1 Extent of DevOps Adoption

4.2.1.1 Familiarity and training

Table 4-6 familiarity with DevOps practices

I am familiar with DevOps practices					
		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	Neutral	23	45.1	45.1	45.1
	Agree	26	51.0	51.0	96.1
	Strongly Agree	2	3.9	3.9	100.0
	Total	51	100.0	100.0	

Source: own survey – generated from SPSS

Table 4-7 response – if there was a formal training

Have you received any formal training in DevOps practices?					
		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	No	43	84.3	84.3	84.3
	Yes	8	15.7	15.7	100.0
	Total	51	100.0	100.0	

Source: own survey – generated from SPSS

The analysis of respondents' familiarity with DevOps practices and their training backgrounds reveals mixed levels of awareness and significant gaps in formal training. For the question "I am familiar with DevOps practices" indicated in *Table 4-6*, 26 respondents (51%) agreed, indicating a fair level of awareness among a majority. However, 23 respondents (45.1%) were neutral, suggesting some uncertainty or limited knowledge, and 2 respondents (3.9%) strongly disagreed, highlighting a complete lack of familiarity.

When asked about formal training in DevOps as shown in *Table 4-7*, the majority of respondents, 43 out of 51 (84.3%), reported that they had not received any formal training. Only 8 respondents (15.7%) confirmed having undergone formal training in DevOps. This significant disparity

between familiarity and formal training underscores a critical issue: while there is some basic awareness of DevOps practices, the lack of structured training programs may be hindering the effective adoption and implementation of DevOps within the organization.

4.2.1.2 DevOps Implementation and Continuous Integration (CI) Processes

Table 4-8 frequency of code commits

How frequently do you commit code to the shared repository?					
		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	Multiple times a day	4	7.8	7.8	7.8
	Once a week or less	8	15.7	15.7	23.5
	Several times a week	13	25.5	25.5	49.0
	There is no such practice	26	51.0	51.0	100.0
	Total	51	100.0	100.0	

Source: own survey – generated from SPSS

Table 4-9 rating the effectiveness of project's CI process

Would you rate the CI processes in your project as effective?					
		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	Disagree	5	9.8	9.8	9.8
	Neutral	31	60.8	60.8	70.6
	Agree	14	27.5	27.5	98.0
	Strongly Agree	1	2.0	2.0	100.0
	Total	51	100.0	100.0	

Source: own survey – generated from SPSS

The analysis of responses regarding DevOps implementation and the Continuous Integration (CI) process indicates significant variability in practices and perceptions of effectiveness. For the first question about frequency of code commit as shown in *Table 4-8*, responses were varied. Only 4 respondents (7.8%) commit code multiple times a day, showing a high level of CI activity. Conversely, 26 respondents (51%) indicated there is no such practice, highlighting a major gap in

CI adoption. The remaining respondents commit code either several times a week (13 respondents, 25.5%) or once a week or less (8 respondents, 15.7%), suggesting occasional use of CI practices.

Regarding the effectiveness of CI processes in the project, the responses were mixed as shown in *Table 4-9*. Fourteen respondents (27.5%) agreed that the CI processes are effective, while more than double that number of respondents (31, 60.8%) were neutral. Notably, 5 respondents (9.8%) disagreed, indicating significant dissatisfaction or issues with the current CI processes.

Table 4-10 challenges faced in the CI process

What challenges do you face with the CI process?			
		Frequency	Percent
Valid		10	19.6
	Insufficient automated tests	29	56.8
	Integration issues	21	41
	Lack of resources	14	27.5
	Slow build times	12	23.5
	Neutral	2	3.9
	Total	51	100.0

Source: own survey – generated from SPSS and amended

When asked about the challenges faced with the CI process as indicated in *Table 4-10*, the most commonly cited issues were insufficient automated tests (29 respondents, 56.8%) and integration issues (21 respondents, 41%). Other notable challenges included a lack of resources (14 respondents, 27.5%) and slow build times (12 respondents, 23.5%). These findings underscore the critical need for improving automated testing, addressing integration issues, and enhancing resource allocation to improve the effectiveness and efficiency of the CI process.

4.2.1.3 Continuous Delivery (CD) and Infrastructure as Code (IaC)

Table 4-11 Temenos provided tool is used for CI/CD tasks

We used App Deployment Manager provided by Temenos for CI/CD tasks.					
		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	I don't know	33	64.7	64.7	64.7
	No	16	31.4	31.4	96.1
	Yes	2	3.9	3.9	100.0
	Total	51	100.0	100.0	

Source: own survey – generated from SPSS

Table 4-12 Temenos provided CI/CD tool improved the process

The App Deployment Manager provided by Temenos improved the process of CI/CD					
		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	Disagree	8	15.7	15.7	15.7
	Neutral	42	82.4	82.4	98.0
	Agree	1	2.0	2.0	100.0
	Total	51	100.0	100.0	
Our deployment process is fully automated					
		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	Disagree	13	25.5	25.5	25.5
	Neutral	37	72.5	72.5	98.0
	Agree	1	2.0	2.0	100.0
	Total	51	100.0	100.0	

Source: own survey – generated from SPSS

The analysis of responses regarding Continuous Delivery (CD) and Infrastructure as Code (IaC) reveals notable insights into their adoption and perceived effectiveness. For the use of the App Deployment Manager provided by Temenos for CI/CD tasks as shown in *Table 4-11*, the responses indicate a mixed level of awareness and usage. 33 respondents (64.7%) indicated that they do not know about its use, suggesting a lack of awareness or communication regarding the tools employed. 2 respondents (3.9%) confirmed its use, while 16 respondent (31.4%) said they didn't use it.

In *Table 4-12, i*n terms of the App Deployment Manager's impact on improving the CI/CD process, the majority of respondents (42, 82.4%) were neutral, indicating uncertainty or a lack of noticeable

impact. Eight respondents (15.7%) disagreed, and only one (2%) agreed, suggesting that those who have formed an opinion view the tool as ineffective.

Regarding the automation of the deployment process, responses indicate limited automation. 13 respondents (25.5%) disagreed that their deployment process is fully automated, while majority of the respondents (37, 72.5%) were neutral, indicating partial automation or uncertainty about the extent of automation.

Table 4-13 IaC is used to manage infrastructure

We use Infrastructure as Code (IaC) to manage our infrastructure					
		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	Disagree	14	27.5	27.5	27.5
	Neutral	36	70.6	70.6	98.0
	Agree	1	2.0	2.0	100.0
	Total	51	100.0	100.0	
Using IaC has improved our infrastructure management					
		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	Disagree	3	5.9	5.9	5.9
	Neutral	44	86.3	86.3	92.2
	Agree	4	7.8	7.8	100.0
	Total	51	100.0	100.0	

Source: own survey – generated from SPSS

As indicated in *Table 4-14,* For the use of IaC to manage infrastructure, responses show low adoption. Only one respondent (2%) agreed that IaC is used, while a combined total of fifty respondents (98%) either disagreed (14, 27.5%) or were neutral (36, 70.5%), reflecting limited implementation and understanding of IaC practices.

Finally, the perceived improvement in infrastructure management due to IaC was minimal. Four respondents (7.8%) agreed that IaC has improved infrastructure management, while the majority (44 respondents, 86.3%) remained neutral, indicating no perceived difference or a lack of sufficient experience with IaC. Three respondents (5.9%) disagreed, reflecting dissatisfaction or perceived

ineffectiveness of IaC. These findings highlight a significant opportunity for enhancing the adoption and effectiveness of CD and IaC practices within the organization.

4.2.1.4 Automated Testing

Table 4-14 implementation of automated tests

Automated tests are implemented at various levels (unit, integration, system, acceptance) frequently					
		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	Disagree	10	19.6	19.6	19.6
	Neutral	24	47.1	47.1	66.7
	Agree	15	29.4	29.4	96.1
	Strongly Agree	2	3.9	3.9	100.0
	Total	51	100.0	100.0	
Our automated tests in ensuring software quality and detecting defects early are effective					
		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	Disagree	4	7.8	7.8	7.8
	Neutral	28	54.9	54.9	62.7
	Agree	17	33.3	33.3	96.1
	Strongly Agree	2	3.9	3.9	100.0
	Total	51	100.0	100.0	

Source: own survey – generated from SPSS

The analysis of responses related to automated testing indicates varied levels of implementation and perceived effectiveness within the organization. For the implementation of automated tests at various levels (unit, integration, system, acceptance) as can be seen in *Table 4*-14, responses are somewhat balanced. 15 respondents (29.4%) agreed, while two (3.9%) strongly agreed that automated tests are frequently implemented at various levels. However, 10 respondents (19.6%) disagreed, and 2 (8.3%) strongly disagreed. In general, this reflects challenges or inconsistencies in implementing automated tests across different levels. The remaining 24 respondents (47.1%) were neutral, indicating either partial implementation or uncertainty regarding the extent of automated testing.

Regarding the effectiveness of automated tests in ensuring software quality and detecting defects early as indicated in *Table 4-14*, responses were generally positive but also showed room for improvement. 17 respondents (33.3%) agreed, and 2 (3.9%) strongly agreed that automated tests are effective in these areas. However, 4 respondents (7.8%) disagreed, and 2 (3.9%) strongly agreed, indicating a little bit of both some dissatisfaction and perceived ineffectiveness. The majority of respondents (28, 54.9%) were neutral, suggesting that the effectiveness of automated tests might be inconsistent or not fully realized across the divisions.

These findings highlight a mixed level of adoption and perceived effectiveness of automated testing within the organization. While some respondents recognize the benefits and effectiveness of automated tests, others point out significant gaps and challenges that need to be addressed to achieve more consistent and reliable software quality assurance.

4.2.1.5 Collaboration and Communication

The analysis of responses regarding collaboration and communication between development and operations teams shows a generally positive perception. For the level of collaboration between these teams as shown in *Table 4-15*, 21 respondents (41.2%) agreed that the collaboration is very good. However, 6 respondents (11.8%) disagreed, indicating some areas for improvement or variability in experiences. 23 respondents (45.1%) were neutral, suggesting that while the collaboration might be adequate, it is not exceptional for everyone involved.

Table 4-15 level of collaboration

The level of collaboration between development and operations teams is very good					
		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	Disagree	6	11.8	11.8	11.8
	Neutral	23	45.1	45.1	56.9
	Agree	21	41.2	41.2	98.0
	Strongly Agree	1	2.0	2.0	100.0
	Total	51	100.0	100.0	

Source: own survey – generated from SPSS

Table 4-16 tools for collaboration

What tools do you use to facilitate collaboration?					
		Frequency	Percent	Valid Percent	Cumulative Percent
Valid		12	23.5	23.5	23.5
	Jira and Confluence	12	23.5	23.5	47.1
	Microsoft Teams	8	15.7	15.7	62.7
	Skype	6	11.7	11.7	73.4
	None	13	25.5	25.5	100.0
	Total	51	100.0	100.0	

Source: own survey – generated from SPSS

In terms of the tools used to facilitate collaboration, as can be seen in *Table 4-16*, the responses reveal a diverse set of tools being employed. Jira and Confluence is reported to be used by 12 respondents (23.5%), indicating these tools' popularity for project management and documentation within the organization. 6 respondents (11.7%) reported using Skype for communication, highlighting its role in facilitating real-time discussions and meetings. Notably, 13 respondents (25.5%) indicated they do not use any specific tools to facilitate collaboration, which could suggest reliance on informal communication methods or a potential area for introducing more structured collaboration tools.

Overall, the data indicates that while there is a strong level of collaboration between development and operations teams, there is an opportunity to standardize and enhance the use of collaboration tools to ensure consistent and efficient communication across all teams. This could help address the neutral and negative responses by providing a more structured approach to collaboration.

4.2.2 Perceived Barriers and Facilitators

4.2.2.1 Barriers

The analysis of responses regarding perceived barriers to adopting DevOps practices reveals significant challenges, particularly with legacy systems and regulatory compliance requirements. When asked if legacy systems are a significant barrier to adopting DevOps practices, as shown in *Table 4-17*, 22 respondents (43.1%) agreed, and 2 (3.9%) strongly agreed, indicating a notable concern. The remaining 26 respondents (51%) were neutral while the other 1 disagreed, suggesting that while legacy systems are a recognized barrier, they might not be universally perceived as

unconquerable. This division implies that while legacy systems pose a challenge, the extent of this impact may vary depending on individual experiences and perspectives within the organization.

Table 4-17 Barriers

Legacy systems are a significant barrier to adopting DevOps practices					
		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	Disagree	1	2.0	2.0	2.0
	Neutral	26	51.0	51.0	52.9
	Agree	22	43.1	43.1	96.1
	Strongly Agree	2	3.9	3.9	100.0
	Total	51	100.0	100.0	
Regulatory compliance requirements make it difficult to implement DevOps					
		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	Disagree	2	3.9	3.9	3.9
	Neutral	42	82.4	82.4	86.3
	Agree	7	13.7	13.7	100.0
	Total	51	100.0	100.0	

Source: own survey – generated from SPSS

As indicated in *Table 4-17 r*egarding the difficulty of implementing DevOps due to regulatory compliance requirements, 7 respondents (13.7%) agreed that this is a significant barrier, highlighting a clear but less widespread concern. The majority, 42 respondents (82.4%), were neutral, and the rest 2 (3.9%) disagreed, indicating that while compliance is acknowledged as a potential barrier, it may not be as pressing or uniformly experienced across all respondents. This could suggest that regulatory obstacles are more context-specific and might affect certain aspects of the DevOps implementation more than others.

Overall, the data suggests that while there are significant perceived barriers to adopting DevOps practices, particularly with legacy systems, these barriers do not universally inhibit progress. The varied responses indicate that while some areas face significant challenges, others may have found ways to mitigate these barriers, or these barriers may not be as impactful in certain contexts. Addressing these barriers effectively will require targeted strategies that consider the specific challenges posed by legacy systems and regulatory compliance, potentially through phased implementation plans and compliance-aligned DevOps processes.

4.2.2.2 Facilitators

Table 4-18 Facilitators

Upper management support facilitates the adoption of DevOps practices					
		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	Disagree	5	9.8	9.8	9.8
	Neutral	30	58.8	58.8	68.6
	Agree	14	27.5	27.5	96.1
	Strongly Agree	2	3.9	3.9	100.0
	Total	51	100.0	100.0	
The availability of resources (e.g., tools, budget) supports DevOps implementation					
		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	Disagree	11	21.6	21.6	21.6
	Neutral	23	45.1	45.1	66.7
	Agree	14	27.5	27.5	94.1
	Strongly Agree	3	5.9	5.9	100.0
	Total	51	100.0	100.0	

Source: own survey – generated from SPSS

The analysis of responses regarding facilitators of DevOps implementation highlights key factors that support the adoption and success of DevOps practices within the bank.

Regarding upper management support as shown in *Table 4-18*, the data indicates a mixed perception among respondents. While 14 respondents (27.5%) agreed and 2 respondent (3.9%) strongly agreed that upper management support facilitates the adoption of DevOps practices, the majority, 30 respondents (58.8%), remained neutral. 5 respondents (9.8%) disagreed. This suggests that while some individuals perceive support from upper management as beneficial, there may be ambiguity or variability in the extent of support provided or its effectiveness in driving DevOps initiatives forward.

In terms of resource availability, significant portion of respondents, 23 individuals (45.1%), remain neutral on whether resources support DevOps implementation. This suggests a general uncertainty or lack of strong opinion regarding resource sufficiency. Meanwhile, 14 respondents

(27.5%) agree that resources are available, indicating a moderate level of satisfaction with the support provided for DevOps initiatives. Conversely, 11 respondents (21.6%) disagree, reflecting concerns or dissatisfaction with resource allocation. A smaller group, 3 respondents (5.9%), strongly agree that resources are adequate. This mixed response highlights the need for more consistent resource allocation and support to ensure effective DevOps implementation across the organization. Addressing these resource concerns can enhance the adoption and efficiency of DevOps practices.

Overall, the analysis suggests that while upper management support and resource availability are perceived as facilitators of DevOps implementation by some respondents, there is room for improvement or clarification in these areas to ensure consistent and effective support for DevOps initiatives throughout the bank.

4.2.3 Project Integration Management and DevOps

The analysis of responses regarding the integration of DevOps practices and Project Integration Management indicates a generally positive perception among the respondents. For the impact of DevOps practices on project timelines and deliverables as shown in *Table 4-24*, the majority of respondents agreed or strongly agreed, with 23 (45.1) and 10 (19.6), respectively, indicating agreement. 14 of them (27.5%) were neutral and only 4 respondents (7.8%) disagreed, suggesting a minor discrepancy in perceptions, possibly due to varied experiences or interpretations.

Table 4-19 DevOps and PIM

Integration of DevOps practices impact the management of project timelines and deliverables

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	Disagree	4	7.8	7.8	7.8
	Neutral	14	27.5	27.5	35.3
	Agree	23	45.1	45.1	80.4
	Strongly Agree	10	19.6	19.6	100.0
	Total	51	100.0	100.0	

DevOps practices align very well with the objectives of Project Integration Management

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	Neutral	19	37.3	37.3	37.3
	Agree	22	43.1	43.1	80.4
	Strongly Agree	10	19.6	19.6	100.0
	Total	51	100.0	100.0	

Source: own survey – generated from SPSS

As can be seen from *Table 4-19, i*n terms of alignment with the objectives of Project Integration Management, the responses were predominantly positive. 32 respondents (62.7%) agreed or strongly agreed that DevOps practices align well with these objectives. 19 respondents (37.3%) were neutral, which could indicate a lack of strong opinion or awareness regarding the specific alignment between DevOps practices and Project Integration Management objectives. There was no disagreement in this regard.

Overall, the data suggests that there is a perceived synergy between DevOps practices and Project Integration Management objectives, particularly in terms of their impact on project timelines and deliverables. The relatively high agreement rates indicate that DevOps practices are viewed favorably in the context of project integration, highlighting their potential to enhance project management processes and outcomes.

Table 4-20 improvement after DevOps practices

I have noticed an improvement in software quality since implementing DevOps practices

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	Disagree	1	2.0	2.0	2.0
	Neutral	30	58.8	58.8	60.8
	Agree	18	35.3	35.3	96.1
	Strongly Agree	2	3.9	3.9	100.0
	Total	51	100.0	100.0	

The extent to which DevOps practices are adopted had significant effect on the integration management aspect of the project

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	Disagree	3	5.9	5.9	5.9
	Neutral	25	49.0	49.0	54.9
	Agree	21	41.2	41.2	96.1
	Strongly Agree	2	3.9	3.9	100.0
	Total	51	100.0	100.0	

Source: own survey – generated from SPSS

The analysis of responses regarding outcomes and feedback from the implementation of DevOps practices reveals mixed perceptions among respondents. For the improvement in software quality, as shown in *Table 4-20*, 18 respondents (35.3%) agreed and 2 respondents (3.9%) strongly agreed that they have noticed an improvement. However, one respondent (8.3%) disagreed, indicating some skepticism or limited observation of improvements. 30 respondents (58.8%) were neutral, suggesting a lack of clear consensus on the extent of improvement in software quality.

As indicated in *Table 4-20*, *i*n terms of the effect of DevOps practices on integration management, four respondents (41.2%) agreed and 2 respondents (3.9%) strongly agreed that the adoption of DevOps practices could have a significant effect. However, 3 respondents (5.9%) disagreed, indicating skepticism or limited perceived impact. 25 respondents (49%) were neutral, suggesting uncertainty or lack of consensus on the potential effect of DevOps practices on integration management.

Overall, the data reveals a range of perceptions regarding the outcomes and feedback from the implementation of DevOps practices, highlighting both positive observations and areas for improvement or further evaluation. These mixed perceptions underscore the complexity of

DevOps implementation and the importance of ongoing monitoring and adjustment to ensure desired outcomes are achieved.

4.2.4 Summary of Descriptive Statistics

4.2.4.1 The extent of DevOps Implementation

The analysis of the mean values from the questionnaire responses provides insight into the extent to which DevOps practices have been implemented within the bank's Temenos Infinity software implementation project. The thematic analysis reveals several key themes related to infrastructure management, automation, continuous integration and continuous delivery (CI/CD), testing, and team collaboration.

Table 4-21 Summary of the extent of DevOps practices implementation

Descriptive Statistics about extent of DevOps practices	
	Mean
We use Infrastructure as Code IaC to manage our infrastructure	2.75
Our deployment process is fully automated	2.76
The App Deployment Manager provided by Temenos improved the process of CI/CD	2.86
Using IaC has improved our infrastructure management	3.02
Automated tests are implemented at various levels unit integration	3.18
Would you rate the CI processes in your project as effective	3.22
Our automated tests in ensuring software quality and detecting defects are effective	3.33
The level of collaboration between development and operations team is very good	3.33
I am familiar with DevOps practices	3.59

Source: own survey – generated from SPSS

Infrastructure Management and IaC Utilization

As shown in *Table 4.21,* the use of Infrastructure as Code (IaC) to manage infrastructure received a mean score of 2.75, indicating a general disagreement to neutrality among respondents about the adoption of IaC practices. This is slightly corroborated by the score of 3.02 for the statement "Using IaC has improved our infrastructure management," which leans more towards neutrality. These scores suggest that while IaC concepts are recognized, their practical implementation and perceived benefits are not fully realized within the project.

Automation in Deployment Processes

The deployment process's automation is another critical aspect, as shown in *Table 4.21,* with a mean score of 2.76, indicating that the respondents are not convinced that their deployment processes are fully automated. The score of 2.86 for the App Deployment Manager provided by Temenos improving the CI/CD process further supports this perception, showing a general lack of strong agreement on the effectiveness and utility of the provided tools for automating deployment.

Effectiveness of Continuous Integration and Testing

In terms of continuous integration, as indicated in *Table 4.21,* respondents rated the effectiveness of CI processes at 3.22, suggesting a neutral to slightly positive view. This moderate rating is echoed in the scores for automated testing, with automated tests implemented at various levels (mean score of 3.18) and the effectiveness of these tests in ensuring software quality and detecting defects (mean score of 3.33). These scores indicate recognition of the presence of automated testing practices but with room for improvement in their perceived effectiveness and integration within the overall CI framework.

Collaboration and Familiarity with DevOps Practices

The level of collaboration between development and operations teams scored a mean of 3.33, suggesting a moderately positive perception of team collaboration, which is shown in *Table 4.21*. Similarly, familiarity with DevOps practices scored the highest at 3.59, indicating that while respondents are generally familiar with DevOps concepts, this familiarity has not entirely translated into consistent and widespread implementation.

4.2.4.2 Perceived Barriers and Facilitators

Table 4-22 Summary of barriers and facilitators

Descriptive Statistics about barriers and facilitators	
	Mean
Regulatory compliance requirements make it difficult to implement	3.10
The availability of resources (e.g. tools, budget) supports DevOps implementation	3.18
Upper management support facilitates the adoption of DevOps practices	3.25
Legacy systems are a significant barrier to adopting DevOps practices	3.49

Source: own survey – generated from SPSS

Barriers

Regulatory compliance requirements, with a mean score of 3.10, are perceived as a moderate obstacle. These compliance demands can complicate the integration of DevOps practices. More notably, legacy systems emerged as the most significant barrier, scoring 3.49. This indicates that outdated or incompatible systems are a major hindrance, making it difficult to implement modern DevOps methodologies effectively.

Facilitators

Conversely, the availability of resources, including tools and budget, and upper management support are perceived as key facilitators of DevOps implementation. Resource availability scored a mean of 3.18, suggesting that while some teams may face resource constraints, overall, the provision of necessary tools and budget is viewed positively. Upper management support, with a mean score of 3.25, underscores the crucial role of leadership in promoting DevOps practices.

4.3 Qualitative analysis

The qualitative analysis of the data collected from interviews and open-ended questionnaire responses provides relatively deeper insights into the state of DevOps practices within the bank's Temenos Infinity software implementation project. One respondent mentioned the limited use of infrastructure as code tools like Terraform, explaining that it's primarily used for cloud provisioning, which is not a significant focus within their current setup. However, automation tools such as Ansible are widely used, indicating a partial adoption of DevOps practices, particularly in automation. This reveals a selective implementation of DevOps tools tailored to the organization's specific needs and existing infrastructure, rather than a comprehensive adoption across all available tools.

Further insights from the interviews highlight a distributed approach to DevOps responsibilities within the organization. Although there are no explicit DevOps roles, the tasks and principles of DevOps are embedded across various teams. This distributed model suggests an informal adoption of DevOps practices, where the integration of these practices into daily activities occurs organically rather than through a structured framework. This fragmented adoption might limit the full potential of DevOps benefits but shows an intrinsic understanding of its principles among the teams. The qualitative data suggests that while there is an awareness and partial implementation of DevOps tools and practices, there is a significant opportunity to formalize and enhance these practices for greater efficiency and operational improvement.

Another respondent said that the quality of team interactions was reported to be good, and there was substantial skill acquisition among team members. This suggests a well-structured project environment that supported the foundational elements of DevOps, even if not fully realized in practice.

Respondents also highlighted effective handling of changing requirements, rigorous testing practices, and strong collaboration and communication within the team. There were no significant gaps in resources, and the team showed high capability in integrating the system with existing workflows. When taken collectively, these elements depict an organization that, while not fully embracing DevOps in a formal sense, is incorporating many of its elements effectively into their project management and operational processes. This provides a robust foundation for

further formalization and enhancement of DevOps practices to drive future improvements in efficiency and project outcomes.

Additional feedback from open-ended questionnaire suggests that while DevOps roles are not formally recognized at BOA, the responsibilities inherent to DevOps are distributed across various teams. By integrating these tasks more cohesively, the organization can enhance operational efficiency and foster a culture of continuous improvement. Some respondents noted the absence of an organized DevOps implementation during the project, describing it as nearly non-existent. However, there is also a recognition of the potential of DevOps practices to meet future business needs if taken seriously. The concept of observability, highlighted by one respondent, underscores the importance of monitoring and alerting components within DevOps, pointing to a foundational element that could be better utilized. Overall, these insights reflect a mixed but hopeful perspective on the potential for DevOps to improve project outcomes and operational efficiency.

Chapter 5: Summary, Conclusions, and Recommendations

This chapter provides a comprehensive synthesis of the research findings, draws conclusions, and offers recommendations based on the study of DevOps adoption in BOA's Temenos Infinity software implementation project. The chapter aims to address the extent of its adoption, the perceived barriers and facilitators within the bank's project management framework. It also outlines the limitations of the study and suggests areas for future research.

5.1 Summary of Findings

The research aimed to address two key questions: assessing the extent of DevOps adoption and understanding the perceived barriers and facilitators within the bank's project management framework. The findings are summarized as follow.

5.1.1 Extent of DevOps Adoption

The analysis based on the questionnaire responses reveals a mixed level of DevOps implementation within the bank's Temenos Infinity software project. There is a reasonable awareness and understanding of DevOps principles among the team members, indicating a foundational knowledge of these concepts. However, the practical application of these principles, particularly in areas such as Infrastructure as Code (IaC) and fully automated deployment processes, appears limited. These limited practical applications are due to the current setup of the organization as can be inferred from the interview questions. This suggests that while the team is familiar with DevOps concepts, their application is not yet fully realized within the project, possibly due to complexities in the existing infrastructure and insufficient resources or expertise.

Continuous Integration (CI) processes and automated testing were found to be moderately effective, suggesting ongoing efforts in these areas but highlighting significant potential for further enhancement. The current implementation of these processes shows that the team recognizes their importance but still faces challenges in achieving optimal efficiency and reliability in software delivery.

Team collaboration received a moderately positive rating, underscoring the necessity for better integration and cooperation between development and operations teams. While there is a good

level of teamwork, improvements are needed to fully embrace robust DevOps practices. Overall, the findings indicate that the bank's Temenos Infinity software project has made some progress in adopting DevOps practices, but there is substantial room for improvement in applying these practices more consistently and comprehensively. Addressing the gaps in IaC, automated deployment, CI processes, and team collaboration will be essential for achieving a more effective and efficient DevOps implementation.

The extent of DevOps implementation within the bank's Temenos Infinity software project reveals a landscape of partial adoption and significant opportunities for enhancement. Despite a reasonable level of awareness and understanding of DevOps principles among team members, the practical application of these principles remains limited, particularly in the areas of Infrastructure as Code (IaC) and fully automated deployment processes. Continuous Integration (CI) and automated testing processes are moderately effective, indicating that while some efforts have been made, there is considerable potential for improvement. The moderately positive view on team collaboration highlights the need for better integration and cooperation between development and operations teams. Overall, while the project has made progresses in adopting DevOps practices, substantial work remains to achieve comprehensive and effective implementation. Addressing the current gaps will be crucial for the bank to fully realize the benefits of DevOps and enhance their software development and operational processes.

5.1.2 Perceived Barriers and Facilitators

Key barriers identified include a lack of formal training, insufficient automated tests, integration issues, and resource constraints. Legacy systems and regulatory compliance requirements also posed significant challenges reflecting the challenges posed by standards like PCI DSS and central bank regulations. Upper management support and the availability of specific tools were seen as facilitators. However, their impact was limited by lack of awareness and inconsistent use.

Management support emerged as a crucial factor for the successful adoption of DevOps practices. Strong upper management support includes endorsing DevOps initiatives, providing the necessary resources, and fostering a culture that embraces change. This top-down support is essential for overcoming resistance and ensuring that DevOps practices are integrated into the organization’s workflows effectively.

The research identified a significant gap in formal training programs, which hinders the effective implementation of DevOps practices. Without proper training, employees may lack the knowledge and skills required to adopt and implement DevOps methodologies successfully. Therefore, enhanced training and continuous learning opportunities are essential to bridge this gap and ensure that all team members are equipped to contribute to DevOps initiatives.

Tools such as Jira and Confluence were used to facilitate collaboration among teams. However, their utilization was inconsistent across the organization. Effective use of these collaboration tools is vital for fostering communication and coordination between development and operations teams. Ensuring that all teams consistently use these tools can enhance project integration and overall efficiency.

5.2 Conclusion

The study concludes that while there is a foundational awareness of DevOps practices at Bank of Abyssinia, significant gaps exist in formal training, resource allocation, and consistent implementation. The findings from this study highlight several critical aspects necessary for the successful implementation of DevOps in the banking sector's IT projects. First and foremost, strong management support emerged as a vital factor. This includes not only endorsing DevOps practices but also providing the necessary resources and fostering a culture that embraces continuous improvement and change. Without robust support from upper management, initiatives are likely to face significant resistance and resource limitations, undermining their potential success.

The study also underscores the importance of comprehensive training programs. A significant gap in formal training on DevOps practices was identified, which directly impacts the effectiveness of implementation. Employees familiar with DevOps principles are more likely to successfully integrate these practices into their daily workflows, leading to improved project outcomes. Thus, investing in continuous learning and development opportunities is crucial to equip the workforce with the necessary skills and knowledge.

Furthermore, the extent of DevOps adoption in the bank’s Temenos Infinity software implementation project was found to be minimal and inconsistent. While some elements of

DevOps, such as Continuous Integration (CI), Continuous Delivery (CD), and Infrastructure as Code (IaC), were present, their application varied significantly across the project. This inconsistency suggests a need for a more structured and cohesive approach to implementing DevOps practices to ensure they are uniformly adopted and effectively utilized across all teams.

Lastly, the study identified key barriers and facilitators to DevOps adoption. The primary barriers include integration issues, resource constraints, and a lack of formal training. These challenges must be addressed to facilitate smoother implementation. On the other hand, facilitators such as strong management support and the availability of specific tools like Jira and Confluence were identified as critical enablers. Addressing the barriers while leveraging these facilitators can significantly enhance the adoption and effectiveness of DevOps practices, ultimately leading to better project outcomes and higher operational efficiency. DevOps implementation requires coordinated efforts across various levels of the organization. A strategic approach that addresses training needs, ensures consistent application of practices, and leverages strong managerial support will be essential for successful DevOps adoption in the banking sector.

5.3 Recommendations

Based on the findings, the following recommendations are proposed to enhance DevOps adoption and effectiveness at the Bank of Abyssinia.

Enhanced Training Programs

To bridge the knowledge gap and ensure effective implementation of DevOps practices, the bank should implement comprehensive DevOps training programs. These programs should include regular workshops, certification courses, and continuous learning opportunities to equip all employees with the necessary skills and understanding of DevOps methodologies. By investing in training, the bank can foster a more knowledgeable and capable workforce, better prepared to implement and sustain DevOps practices.

Increase Automation

Automation should be a key focus to improve efficiency and reduce manual errors in the development and deployment processes. The bank should invest in robust Continuous Integration/Continuous Delivery (CI/CD) tools and Infrastructure as Code (IaC) practices. By

automating more aspects of these processes, the bank can streamline operations, reduce the risk of errors, and accelerate the delivery of software updates and new features, ultimately enhancing project outcomes and customer satisfaction.

Resource Allocation

Sufficient resources, including tools and budget, are crucial for the success of DevOps initiatives. The bank should address challenges such as insufficient automated tests and integration issues by ensuring that all necessary resources are readily available. This may involve investing in advanced testing tools, enhancing the infrastructure to support DevOps activities, and allocating budgets specifically for DevOps projects. Proper resource allocation will enable smoother and more efficient implementation of DevOps practices.

Management Support

Strong upper management support is essential for the adoption and success of DevOps practices. Management should actively promote a DevOps culture by endorsing DevOps initiatives, providing clear policies and support structures, and regularly monitoring progress. By fostering a culture that embraces change and innovation, management can help overcome resistance and ensure that DevOps practices are integrated into the organization's workflows effectively.

Continuous Improvement

Establishing a feedback loop to continuously assess and improve DevOps practices is crucial for long-term success. The bank should encourage a culture of continuous improvement and shared responsibility, where teams regularly review their processes and outcomes, identify areas for enhancement, and implement changes as needed. This adaptive approach will help the bank stay responsive to emerging challenges and advancements in technology, ensuring that DevOps practices remain effective and aligned with organizational goals.

5.4 Research Limitation and Areas of Further Research

5.4.1 Limitation of the Study

The study's limitations include a small sample size, potential response biases, and the specific focus on a single bank, which may limit the generalizability of the findings. The existence of predominantly young professionals in the bank's IT work units is another issue that may limit the generalizability of the study. Additionally, the lack of regression analysis limits the depth of understanding of the relationships between variables.

5.4.2 Suggestion for Future Research

Future research could expand the sample size and include multiple financial institutions to enhance the generalizability of the findings. Further studies could explore the impact of specific DevOps tools and techniques on project outcomes and conduct a more detailed analysis of the relationships between various barriers and facilitators. Incorporating regression analysis in future studies would provide a deeper understanding of these relationships. Exploratory researches could help identify more variables, factors and key players in implementing DevOps in the banking sector.

References

Anna D. (2023). *Navigating the Agile Software Development Life Cycle: Phases, Tools, Roadmap.* Retrieved from https://relevant.software/blog/agile-software-development-lifecycle-phases-explained/

Atlassian. (n.d.). Waterfall methodology for project management. Retrieved from https://www.atlassian.com/software-development/software-development-methodologies/waterfall

Balaji, S., & Murugaiyan, M. S. (2012). Waterfall vs. V-Model vs. Agile: A comparative study on SDLC. *International Journal of Information Technology and Business Management*, 2(1), 26-30.

Bass, L., Weber, I., & Zhu, L. (2015). *DevOps: A Software Architect's Perspective*. Addison-Wesley Professional.

Beck, K. (1999). *Extreme Programming Explained: Embrace Change*. Addison-Wesley.

Campbell, D. T., & Fiske, D. W. (1959). *Convergent and discriminant validation by the multitrait-multimethod matrix*. Psychological Bulletin, 56(2), 81-105.

Carter, P. (2017). *DevOps Success Stories: How Amazon, Netflix, and Other Companies are Winning with DevOps.* Retrieved from https://www.qualitylogic.com/devops-success-stories/

Chris, C. (2020). *What is DevOps.* Retrieved from https://blog.mobcoder.com/what-is-devops/

Cohn, Marisa & Sim, Susan & Lee, Charlotte. (2009). *What Counts as Software Process? Negotiating the Boundary of Software Work Through Artifacts and Conversation. Computer Supported Cooperative Work.* 18. 401-443. 10.1007/s10606-009-9100-4.

Cois, C. (2015). *DevOps Case Study: Netflix and the Chaos Monkey. Retrieved from* https://insights.sei.cmu.edu/blog/devops-case-study-netflix-and-the-chaos-monkey/

"*DevOps Case Study: Amazon." (n.d.). Retrieved from* https://www.atlassian.com/devops/case-studies/amazon

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.

Fowler, M., & Foemmel, M. (2006). Continuous Integration. Retrieved from https://martinfowler.com/articles/continuousIntegration.html

Garousi, V., & Elberzhager, F. (2017). Test automation in software product line engineering. In Proceedings of the 13th International Conference on Software Testing, Verification, and Validation (ICST), 439-444.

"*How Netflix Became A Master of DevOps." (2023). Retrieved from* https://www.simform.com/netflix-devops-case-study/

Humble, J., & Farley, D. (2010). *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation*. Addison-Wesley Professional.

Huttermann, M. (2012). *DevOps for Developers*

IntelliSoft, (2023). *DevOps Automation: A Complete Guide to Efficient Development*. Retrieved from https://medium.com/@IntelliSoft/devops-automation-a-complete-guide-to-efficient-development-8a79ad209909

Kim, G., Humble, J., Debois, P., & Willis, J. (2016). *The DevOps Handbook: How to Create World-Class Agility, Reliability, & Security in Technology Organizations*. IT Revolution Press.

Mazyar, M. (2018, December 18). *DevOps: The Ultimate Way to Break Down Silos*. DevOps.com. Retrieved from https://devops.com/devops-the-ultimate-way-to-break-down-silos/

Project Management Institute. (2008). *A Guide to the Project Management Body of Knowledge (PMBOK Guide)* (4th ed.). Project Management Institute.

Puppet labs. (2021). *State of DevOps Report 2021*

Rubin, K. S. (2013). *Essential Scrum: A Practical Guide to the Most Popular Agile Process*. Addison-Wesley Professional.

Sharma, S., & Coyne, B. (2018). *DevSecOps: A leader's guide to producing secure software without compromising flow, feedback, and continuous improvement*. IT Revolution Press.

Tavakol, M., & Dennick, R. (2011). *Making sense of Cronbach's alpha.* International Journal of Medical Education, 2, 53-55.

"Temenos Infinity." (2020). Retrieved from https://www.temenos.com/wp-content/uploads/2019/08/products-infinity-brochure-2019-Aug-20.pdf

APPENDIX A: QUESTIONNAIRE

Addis Ababa University College of Business and Economics School of Commerce

Master of Art in Project Management

Dear Respondent,

My name is Amlakie Yitayih and I am conducting a study titled " *Challenges of Implementing DevOps (Development Operations) as a Tool of Project Integration Management: the Case of Temenos Infinity System Implementation at Bank of Abyssinia*" as part of my research efforts to understand the adoption and integration of DevOps practices within the banking sector. Your participation in this survey is invaluable in helping me gather crucial data to address the following research questions: To what extent have DevOps practices been adopted in the bank's Temenos Infinity software implementation project? And what are the perceived barriers to and facilitators of adopting DevOps practices within the bank's project management framework and key success factors?

I kindly request your participation in this questionnaire, which aims to collect insights and feedback from key stakeholders involved in the project. Your responses will provide essential information on the implementation, challenges, and outcomes associated with DevOps practices at the Bank of Abyssinia.

The questionnaire is designed to be straightforward and should take approximately 5-10 minutes to complete. ***Your responses will be kept confidential and used solely for academic purposes***. I assure you that no personal information will be disclosed, and the data will be reported in aggregate form.

Please find the questionnaire attached. Your cooperation and honest feedback are highly appreciated and will significantly contribute to the success of this study.

Thank you in advance for your time and participation!

Sincerely,

Amlakie Yitayih

Note:

- You don’t need to write your name.
- If you need further explanation, you may contact me with:

Amlakie Yitayih: - amlakie.yitayih@gmail.com or 0953223529

Part 1: Personal information for the respondents

1. Sex: Male ☐ Female ☐
2. Age: less than 25 years ☐ 25 years to 30 years ☐
   30 years or above ☐

3. Educational level: Diploma or Less ☐ Bachelor ☐
   Masters ☐ Ph.D. ☐
4. Work experience: Less than 2 years ☐ 2 to 5 years ☐
   5 to 10 years ☐ Above 10 years ☐
5. Respondent's Position: Division Manager ☐ Project Manager ☐
   Analyst and Developer ☐ Administrator ☐
   Officer (Junior/Associate/business) ☐ IT Trainee ☐
6. location: Head Office (Addis Ababa) ☐ Project site ☐

Part 2: DevOps Practices during the undertaking of **Infinity/apollo** Project

Note:

- Choose the option that expresses what you know best depending on the nature of the choices**.** The Likert Scale is presented below as an example.
  - If the item strongly matches with your response, choose **Strongly Agree.**
  - If you moderately agree on the idea, choose **Agree.**
  - If you don't have any idea or information on the point, choose **Neutral.**
  - If you moderately disagree with the point, choose **Disagree.**
  - If you completely disagree with the point, choose **Strongly Disagree**
  - Please answer by marking in the box that corresponds to your choice by a tick mark [√].

<u>Closed ended questionnaire</u>

Table 1: Familiarity and Training in DevOps

Question	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
1. I am familiar with DevOps practices (e.g., Continuous Integration, Continuous Delivery, IaC, Automated Testing)?					
2. Have you received any formal training in DevOps practices?	**Yes**		**No**		

Table 2: DevOps Implementation and Continuous Integration (CI) Processes

Question	Multiple times a day	Once a day	Several times a week	Once a week or less	There is no such practice
1. How frequently do you commit code to the shared repository?					

2. Would you rate the CI processes in your project as effective?	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree

3, What challenges do you face with the CI process? (select all that apply)	Integration issues	Slow build times	Insufficient automated tests	Lack of resources	Other (Please specify)

Table 3: Continuous Delivery (CD) and Infrastructure as Code (IaC)

Question	Yes	No	I don't know
1. We used App Deployment Manager provided by Temenos for CI/CD tasks.			

Question	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
1. The App Deployment Manager provided by Temenos improved CI/CD					
2. Our deployment process is fully automated.					
3. We use Infrastructure as Code (IaC) to manage our infrastructure.					
4. Using IaC has improved our infrastructure management.					

Table 4: Automated Testing

Questions	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
1. Automated tests are implemented at various levels (unit, integration, system, acceptance) frequently?					
2. Our automated tests in ensuring software quality and detecting defects are effective.					

Table 5: Collaboration and Communication

Question	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
1. The level of collaboration between development and operations team is very good.					

2. What tools do you use to facilitate collaboration?	Slack	Microsoft Teams	Jira	Confluence	Other (Please specify)

Table 6: Perceived Barriers and Facilitators

Question	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
1. Legacy systems are a significant barrier to adopting DevOps practices.					
2. Regulatory compliance requirements make it difficult to implement DevOps.					
3. Upper management support facilitates the adoption of DevOps practices.					
4. The availability of resources (e.g., tools, budget) supports DevOps implementation.					

Table 7: Project Integration Management and DevOps

Questions	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
1. Integration of DevOps practices impact the management of project timelines and deliverables					
2. DevOps practices align very well with the objectives of Project Integration Management					

Table 8: Outcomes and Feedback

Question	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
1. I have noticed an improvement in software quality since implementing DevOps practices.					
2. I am satisfied with the overall performance of DevOps practices in the project.					
3. The extent to which DevOps practices are adopted has/could have had significant effect on the integration management aspect of the project.					

Open-ended questionnaire

1. Please write your suggestion about any part of the DevOps practices/principles that are related to the infinity/apollo project.