Tampereen yliopisto

Vesa Järvimäki

CONTINUOUS INTEGRATION PIPELINES AS AN ENABLER FOR BIG DATA -ANALYTICS AND MACHINE LEARNING

Master's Thesis
Faculty of Management and Business
July 2025

TIIVISTELMÄ

Vesa Järvimäki: Jatkuvan integraation putket big data analytiikan ja koneoppimisen mahdollistajana
Diplomityö
Tampereen yliopisto
Tietojohtaminen
Kesäkuu 2025

Jatkuvan integraation (CI) putkistot ovat keskeinen osa nykyaikaista ohjelmistokehitystä, mahdollistaen tiheät koodimuutosten integroinnit, automaattiset testaukset ja sujuvat julkaisut. Vaikka CI-putkistot ovat levinneet laajaan käyttöön, organisaatiot eivät usein pysty hyödyntämään niiden tuottamaa suurta datamäärää tehokkaasti. CI putkien tuottaman datan avulla organisaatiot voisivat paremmin seurata projektien etenemistä, etsiä tehostettavia kohteita ja tuoda nopeammin analytiikkaa virheistä ohjelmistokehittäjille. Heikkolaatuinen ja epäjohdonmukainen data kuitenkin rajoittaa CI-putkistojen mahdollisuuksia Big Data -analytiikassa (BDA) ja koneoppimisessa (ML), mikä estää organisaatioita hyödyntämästä täysimääräisesti datavetoista päätöksentekoa.

Tämän tutkimuksen tarkoituksena oli selvittää, kuinka CI-putkistoja voidaan parantaa tuottamaan korkealaatuista dataa, joka tukee BDA- ja ML-sovelluksia. Tutkimus hyödyntää Action Design Research (ADR) -lähestymistapaa kehitettämään ja testataamaan iteratiivisesti malli CI-putkistojen optimointiin, keskittyen datan keräyksen, rakenteen ja hallinnan parantamiseen. Tutkimus toteutettiin todellisessa organisaatioympäristössä, jossa alustavaa mallia arvioitiin ja kehitettiin jatkuvien käytännön interventioiden avulla.

Tutkimuksen tulokset korostavat tarkkuuden, täydellisyyden ja validiuden merkitystä tärkeimpinä datan laatuominaisuuksina BDA:lle ja ML:lle. Keskeisiksi haasteiksi CI-putkistojen datan hyödyntämisessä tunnistetaan epäjohdonmukaiset lokimuodot, standardoitujen tiedonkeruumenetelmien puute sekä vaikeudet integroida CI-dataa laajempiin analytiikkajärjestelmiin. Näiden ongelmien ratkaiseminen voi muuttaa CI-putkistot arvokkaiksi tietolähteiksi, jotka tukevat ennakoivaa analytiikkaa, automaatiota ja suorituskyvyn optimointia.

Tämän tutkimuksen tuloksena ehdotetaan rakenteellista kehystä CI-putkistojen datan laadun parantamiseen, varmistaen, että kerätty data on sekä luotettavaa että käyttökelpoista. Mallin käyttöönotto voi parantaa projektinhallintaa, tehostaa virheenjäljitystä ja mahdollistaa analytiikan ja koneoppimisen tehokkaamman hyödyntämisen ohjelmistokehityksessä. Vaikka tutkimus perustuu yhteen organisaatiotapaukseen, ehdotettu malli tarjoaa yleistettäviä näkemyksiä, joita voidaan soveltaa eri toimialoilla ja kehitysympäristöissä.

Avainsanat**:** Jatkuva integraatio, datan laatu, big data -analytiikka, koneoppiminen, Action Design Research, ohjelmistokehitys

ABSTRACT

Vesa Järvimäki: Continuous integration pipelines as an enabler for big data -analytics and machine learning.
Master's Thesis
Tampere University
Master's Degree Program in Information and Knowledge Management
June 2025

Continuous Integration (CI) pipelines have become a fundamental component of modern software development, enabling frequent code integrations, automated testing, and streamlined deployments. However, despite their widespread adoption, organizations often fail to leverage the vast amount of data generated by these pipelines effectively. Data generated by CI pipelines could be utilized to manage projects, increase CI effiency and provide more detailed analytics to developers. But poor data quality and inconsistencies limit the potential of CI-generated data for Big Data Analytics (BDA) and Machine Learning (ML), preventing organizations from fully benefiting from data-driven decision-making.

This study investigates how CI pipelines can be improved to generate high-quality data that supports BDA and ML applications. Using an Action Design Research (ADR) approach, the study iteratively develops and refines a model for optimizing CI pipelines, focusing on enhancing data extraction, structuring, and management. The research is conducted within a real organizational setting, where a preliminary framework is tested, evaluated, and continuously improved through practical interventions.

Findings from the study highlight accuracy, completeness, and validity as the most critical data quality dimensions for BDA and ML. The research identifies common challenges in CI pipeline data extraction, including inconsistent logging formats, lack of standardization in data collection methods, and difficulties in integrating CI data with broader analytics infrastructures. By addressing these issues, organizations can transform their CI pipelines into valuable data sources that drive predictive analytics, automation, and performance optimization.

As a result, this study proposes a structured framework for improving data quality within CI pipelines, ensuring that extracted data is both reliable and actionable. Implementing this model can enhance project management, improve debugging efficiency, and enable more effective use of analytics and machine learning in software development. While the study is based on a specific organizational case, the proposed framework offers generalized insights that can be adapted across different industries and development environments.

Keywords**:** Continuous Integration, Data Quality, Big Data Analytics, Machine Learning, Action Design Research, Software Development

USE OF AI IN THESIS

I have utilised AI tools in my thesis:

☐ No
☒ Yes

The AI tools utilised in my thesis and their purposes are described below:

Names and versions of AI tools: ChatGPT (Gpt4.0 including Scholar AI plugin)

Purpose of using AI tools: In this thesis, AI tools were employed for two primary purposes. Firstly, AI was used to assist in identifying relevant sources. Specifically, the ChatGPT Scholar AI plugin was utilized to locate academic articles and generate summaries of their content. Based on these summaries, I selected promising studies for further manual review and analysis. AI was also used for language refinement. I employed ChatGPT to detect potential grammatical errors and to suggest more eloquent or academically appropriate wording where needed. ChatGPT was not used for generative content creation.

Sections where AI tools were used: AI was utilized in all sections of this thesis, besides the Abstract

I acknowledge that I am fully responsible for the entire content of my thesis, including the parts generated by AI, and accept accountability for any violations of ethical standards in publications.

PREFACE

This master's thesis was carried out with the support of both Tampere University and a partner organization. While the company has chosen not to be named, I would like to express my sincere gratitude to the organization and its representatives for giving me the opportunity and support to work on this thesis.

I would also like to extend my sincere thanks to my examiners, Jukka Huhtamäki and Jussi Myllärniemi, for their invaluable guidance in shaping this thesis from the ground up. Especially Jukka, as my primary examiner, helped a lot in forming this from unorganized idea into formalized thesis.

Tampere, 18.7.2025

Vesa Järvimäki

CONTENTS

GLOSSARY

ADR	Action Design Research
BDA	Big Data Analytics
CI	Continuous Integration
CDE	Continuous Delivery
CD	Continuous Deployment
DevOps	Development and Operations
DIKW	Data–Information–Knowledge–Wisdom hierarchy
IT	Information Technology
ML	Machine Learning
SDLC	Software Development Life Cycle

1. INTRODUCTION

This idea for this research was sparked by a perceived challenge in a target organization involved in software development. The organization had complex continuous integration pipelines that were tracked through generic metrics, but it was perceived that the existing infrastructure could not provide data with necessary quality and consistency to enable more sophisticated data management practices. This sparked an interest in researching and developing ways to improve data collection from these pipelines to provide higher quality data.

1.1 Research background

Continuous Integration (CI) has become an essential practice in modern software development, aimed at improving the quality and efficiency of software projects (Soares et al., 2022). CI involves the frequent integration of code changes into a shared repository, which is then automatically built and tested helping to identify and address integration issues early, reducing the risk of defects and enabling a more streamlined development process (Soares et al., 2022). In recent years, the importance of CI has been amplified by the growing complexity of software systems and the need for rapid and reliable delivery (Yiran et al., 2018). CI provides a framework that supports continuous delivery and deployment, allowing teams to deliver new features and updates faster and more efficiently (Yiran et al., 2018).

The motivation for this research stems from the challenges associated with extracting valuable data from CI pipelines. Despite the widespread adoption of CI, there isn't much talk about organizations' ability to leverage the data generated during the integration process effectively. This data is believed to hold significant potential for improving various aspects of software development, including project management, quality assurance, and machine learning applications. In study by Hilton et al. (2017) it is said that most CI engineers that participated in the study used the CI output logs for debugging, and because these output logs can be hundreds of thousands of lines long, this process can be very time consuming.

Big Data Analytics (BDA) and Machine Learning (ML) are two fields that could greatly benefit from high-quality data extracted from CI pipelines. BDA involves analyzing large

volumes of data to uncover patterns, trends, and insights that can drive decision-making and innovation (Dai et al., 2019). ML, on the other hand, relies on data to train models that can make predictions, automate tasks, and enhance software quality (Rana & Staron, 2015).

However, the effectiveness of BDA and ML is heavily dependent on the quality of the data. It's possible for poor data quality to lead to inaccurate insights and suboptimal models, underscoring the need for effective data management practices within CI environments. This research aims to address this gap by exploring methods to improve the data extraction process from CI pipelines, ensuring that the data is accurate, relevant, and useful for various analytical purposes.

The focus of this study is on developing a model that describes how to enhance CI pipelines to produce higher quality data. By addressing the issues faced by a specific organization, the research seeks to create solutions that can be generalized and applied in other contexts as well. This pragmatic approach aligns with the research philosophy adopted in this study, which emphasizes the practical application of concepts to support action and solve real-world problems.

1.2 Research objective and questions

The primary objective of this thesis is to explore the connection between continuous integration pipelines and software project data management. With the goal being to improve the extraction of quality data, with a particular focus on project management and data management purposes. The goal also includes addressing challenges related to obtaining useful information from CI pipelines within the target organization and to propose a generalizable model that can be applied across various organizations facing similar issues.

To achieve this objective, the study will answer the main research question: **How can companies improve their continuous integration pipelines to produce quality data for data management purposes?** This main question is supported by the following secondary research questions

1. What defines quality data?
2. What are the most important data quality traits of machine learning and big data analytics?
3. How can we improve continuous integration pipelines to provide us with more useful data?

4. Does undertaking the improvement process provide an organization with tangible benefits that are worth the effort?

To put it simply, this research aims to solve a problem related to the difficulty of extracting useful information from CI pipelines in the target organization. This study is hopefully able to solve the issues reported in the target organization and the achieved situationally specific results can be extrapolated into a more general model that can be utilized by solving problems in other organizations as well.

1.3 Research scope

This study focuses on extracting valuable data from continuous integration (CI) pipelines for purposes related to data management, machine learning, and project management. It is important to note that the scope of this study does not extend to the actual utilization of the extracted data. While data analytics, machine learning, and other technologies define the criteria for what constitutes valuable data, this research primarily addresses these criteria from a data requirements perspective. Consequently, the study does not delve into the practical applications of the data. Additionally, this research does not serve as a practical guide for implementing CI improvements. Instead, its goal is to generate findings that are broadly applicable and not tied to specific technologies, thereby providing general insights and recommendations.

It's also important to note that even though this study was conducted for a specific organization and the practical implementation of improvements was tailored to fit it, the study itself is interested in generic solutions to generic problems. Thus, while rooted in a particular organizational context, the findings aim to contribute to the broader field of CI pipeline optimization, offering valuable guidance for a wide range of settings.

1.4 Research structure

This thesis consists of seven chapters and is structured in the following manner. After this introduction to the research topic, chapters two and three discuss the existing theory about the subject and define the terms used in the research.

The second chapter delves into the theoretical aspects of software development projects and explores the application of continuous integration within this context. It discusses various software development life cycle models and the role of CI in them, laying the foundation for understanding how to extract useful data from CI pipelines. In the third chapter, key concepts such as data, information, knowledge, and wisdom are de-

fined. It also explores data quality from a broader perspective, focusing on the requirements set by big data analytics and machine learning for high-quality data. The chapter concludes with a discussion on extracting quality data from CI pipelines.

After we have explored the existing theory, the fourth chapter explains and justifies the chosen research design, utilizing the research onion framework. It covers research philosophy, approach, methodology, and strategy.

The fifth chapter explores the empirical part of the study and its results, and the implications of the findings are discussed in chapter six, reflecting on the research questions and the limitations and implications of the study. Finally, the seventh and final chapter sums up the research and lists the study's contributions and potential for future research.

2. CONTINUOUS INTEGRATION IN SOFTWARE DEVELOPMENT PROJECTS

The objective of this chapter is to delve into the theoretical aspects of software development projects and explore the application of continuous integration (CI) within this context. While the focus of this study does not specifically revolve around best practices in software development or CI, it is essential to establish a clear understanding of the terms employed in it. The chapter first discusses software development projects on a larger scale, and then the role of CI systems as its support. Laying foundation to further analysis on how to extract useful data from them. As an additional note, this chapter doesn't aim to discuss how software development or CI is built in this study's target organization, but to discuss the surrounding theory in general.

2.1 Software development life cycle process

Software development can be viewed as a sequence of stages, structured to develop an intended software product. Software development life cycle (SDLC) process is a methodology used in software development to create high-quality software products (Shetty et al., 2023). This life cycle aims to describe the development process from analysis to maintenance. The steps as described by Garg et al. (2022) are described below and they are show in the Figure 1.

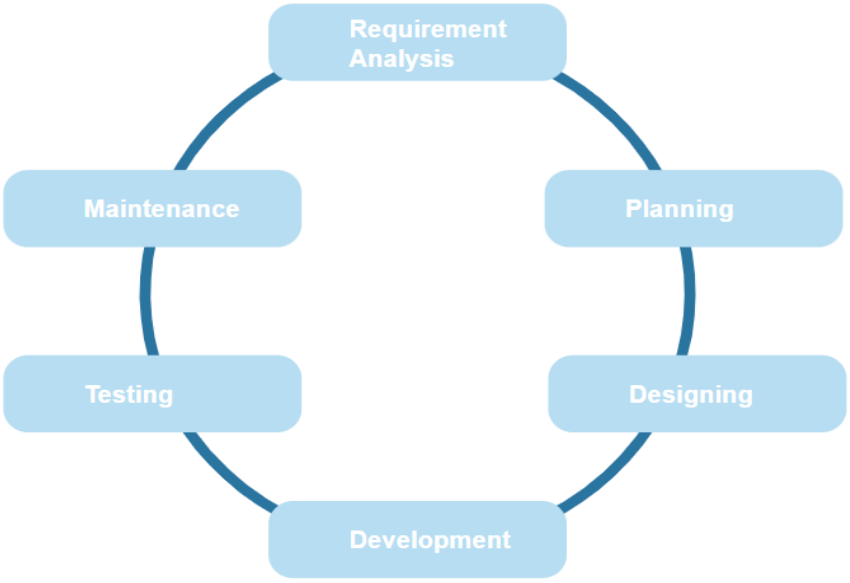

Figure 1. *SDLC as described by Garg et al., (2022)*

1. Planning: This initial phase involves outlining the project’s goals, scope, purpose, and procedures. It sets the stage for effective software development and is crucial for project management.
2. Requirements: This step involves collecting detailed requirements from stakeholders. These requirements define what the software should do and how it should perform, serving as a guideline for the entire project.
3. Designing: This phase includes both architectural and detailed design. The architectural design defines the software's overall structure, while detailed design focuses on how each component of the software functions internally.
4. Development: During this phase, actual coding of the software takes place. Programmers write code according to the requirements and design documents prepared in earlier phases.
5. Testing: Once the software is developed, it is tested to ensure it is bug-free and meets all requirements. This phase involves various types of testing like unit testing, integration testing, system testing, and acceptance testing.
6. Maintenance: After successful testing, the software is delivered to the customer or made available to the public. This could involve initial installation and configuration of the software in its intended environment.

By following this sequential process teams can ensure that the final product aligns with stakeholder expectations and functions reliably in its intended environment. This disciplined approach not only reduces risks but also enhances collaboration and long-term maintainability of software systems (Shetty et al., 2023).

2.2 SDLC Models

SDLC process offers us an overview of how software development projects are usually organized, but in practice this process is implemented and structured in different ways in different projects. This is where SDLC models come into the picture, giving us a more detailed view of SDLC process in a more practical environment, with each one of these models, coming with their own set of characteristics, advantages, and disadvantages. These models provide a framework for planning, executing, and managing software development projects. This section aims to quickly describe the overall characteristics of some of the most popular SDLC models. We will start with the oldest waterfall model and then describe its derivates and other newer models.

The **Waterfall Model** is a linear and sequential approach to software development and it is considered to be the oldest SDLC model (Garg et al., 2022). In it each SDLC phase must be completed before the next phase begins, and there is no overlapping in the phases (Kazim, 2017). The Waterfall model is straightforward and easy to understand and manage, however, it's less flexible and doesn't easily accommodate changes during the development process (Garg et al., 2022; Jain, 2011). Even though Waterfall model is old and sometimes considered outdated it still works well in smaller projects with more static definitions (Kazim, 2017),

The **Iterative Model** is a derivative of waterfall model and in the project is divided into smaller distinct and cascading phases (Jain, 2011). Instead of one big cycle like in the waterfall model, the iterative model runs the same steps several times for smaller requirement subsets (Kazim, 2017). This allows software developers to review, learn and take advantage of what was learned during the development of earlier parts or versions of the system (Garg et al., 2022). This model is more flexible than basic waterfall and allows for changes based on feedback from earlier iterations, but iterations can accumulate costs and add confusion to the project (Garg et al., 2022).

The **V-Model**, also known as the Validation and Verification model, is an extension of the Waterfall model and is characterized by its V-shape (Kazim, 2017). The V-Model emphasizes the importance of testing and involves several phases where the system specifications and testing is planned with the development phase (Jain, 2011). It shares many of the advantages and disadvantages with the waterfall model, but the increased focus on testing can improve development outcomes (Garg et al., 2022).

The **Spiral Model** combines elements of waterfall and iterative models to a spiral like loop (Garg et al., 2022). But unlike in iterative model, both planning and deployment are also included in the loop, which means that spiral model produces a working prototype in each one of its loops (Jain, 2011). The model allows for incremental releases of the product, or incremental refinement through each iteration around the spiral (Jain, 2011). The Spiral Model is particularly useful for large, complex, and high-risk projects, where requirements are not well defined, but these type of projects are difficult to keep inside a budged and in schedule, in part due to required expertise and documentation (Jain, 2011; Kazim, 2017).

The **Rapid Application Development model** is a significant departure from the other models in the way it is time driven rather than requirements driven (Jain, 2011). The Rapid Application Development Model emphasizes quick development and delivery of

high-quality products with low investment costs (Garg et al., 2022). It is a type of incremental model that focuses on the development of prototypes, user feedback, and the reuse of software components (Kazim, 2017). The model allows for adjustments and changes to be made during the development process, making it flexible and adaptable to changes, thus making it best suited for projects that need to be completed in a short time span (Garg et al., 2022).

The **Agile model** merges iterative and incremental approaches, focusing on flexibility and ensuring client satisfaction through the quick delivery of functional software (Kazim, 2017). In this method, the software is developed in small, manageable increments, each completed by cycles called iterations. During an iteration, the development team engages in several of the steps in SDLC process and with the conclusion of each iteration, a usable version of the software is presented to the customer and stakeholders (Kazim, 2017). The Agile model is distinctive because it treats each software project as unique, applying various methodologies tailored to the specific demands of the project (Kazim, 2017). There are several different methods for agile development, but all of employ an iterative process that leads to the delivery of a working product at the end of each cycle, until the until the final version of the software, incorporating all requested features, is completed and delivered (Kazim, 2017). The agile methods are flexible solutions that offer a realistic approach to software development with opportunities for early delivery without fixed requirements, but it requires a lot of communication from customer and tight deadlines can cause problems for example with lack of documentation and maintainability (Kazim, 2017).

2.3 Continuous Integration as part of SDLC

In the previous section we saw how the SDLC process and defined how different software development projects are structured. In this section we will move on the role of continuous integration as part of this process. Even though continuous integration doesn’t show up as a separate step in most SDLC models it has become an integral part of the practical implementation of the development, testing and maintenance steps and almost a necessity in Agile models that aim for high frequency of changes (Virmani, 2015)

2.3.1 Continuous Integration

Idea of Continuous Integration was first introduced in 1991 by Grady Booch and was later adopted as one of the core practices of Extreme Programming, an Agile develop-

ment model (Hilton et al., 2016). CI aims to reduce the cost and complexity of code integration among developers or teams by making integration a frequent activity (Soares et al., 2022). The practice requires no more than a few hours between integrations to encourage shared code and knowledge, thus minimizing the risks associated with large-scale integrations after extended periods (Soares et al., 2022).

According to Soares et al. (2022) successful implementation of CI hinges on four essential mechanisms:

1. Version control system.
2. Build script.
3. Feedback mechanism.
4. Process for integrating source code changes.

In CI the mechanism forms a loop where a change in the version control system triggers a build script that sends feedback about the test results and after success the original change is integrated into the source code managed by the version control system.

The adoption of CI has been increasing over the years and some sources advocate that CI is the centerpiece of software development (Duvall, 2007; Hilton et al., 2016). But even though CI increased adoption and attention of CI there is no existing consensus on CI processes, leading to variations in its implementation across different projects (Soares et al., 2022). Related to this there also exists discussion around the definitions and relationships between CI, Continuous Delivery (CDE), Continuous Deployment (CD), and their integration with DevOps, highlighting the lack of a unified definition of CI and its practices (Soares et al., 2022).

According to Istifarulah and Tiaharyadini (2023) these terms can be defined in following manner, they are also visualized in figure 2:

- CI refers to the practice of frequently integrating code changes into a shared repository. Developers often merge their changes several times a day, and each integration is verified by an automated build and test process. The primary goals of CI are to detect and resolve conflicts early, reduce integration problems, and ensure that the software is always in a release-ready state.
- CDE then extends CI by ensuring that the software can be released to production at any time. It involves automating the release process to make predictable deployments, routine affairs that can be performed on demand.

- CD takes this even further by automatically deploying every change that passes the test phase into production. There is no human intervention, and only a failed test will prevent a new change from being deployed to production.
- DevOps is a set of practices that combines software development (Dev) and IT operations (Ops), aiming to shorten the systems development life cycle while delivering features, fixes, and updates frequently in close alignment with business objectives. DevOps is not only about automation but also about integrating development and operations teams culturally and technically. CI, CDE and CD are usually included in the Dev part of DevOps.

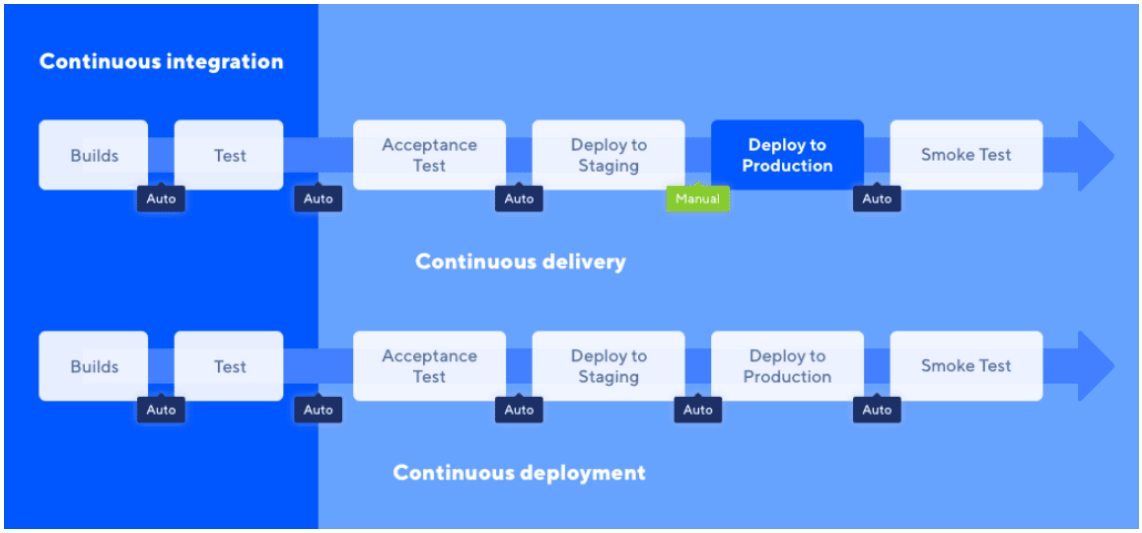

Figure 2. *CI, CDE and CD as described by Istifarulah & Tiaharyadini (2023)*

But other perspectives define these practices as less sequential and see CI in a more holistic view, in which CI incorporates both CDE and CD (Fitzgerald & Stol, 2017). These differences in the definitions lead to countless different implementations and CI practices (Soares et al., 2022). This paper uses, when talking about CI, the definition offered by Istifarulah and Tiaharyadini (2023) in which CI is seen as simpler and sequential part of automated delivery pipeline.

2.3.2 Benefits of CI

The adoption of CI is driven by the perceived benefits provided by it. According to Elazhary et al. (2022) CI has many claimed benefits, but that these are project contextual and differ between implementations. But in general, some overall benefits can be seen across different implementations. One of the primary benefits being the provision of immediate feedback, allowing developers to quickly identify and resolve issues in their code. This immediate feedback loop fosters a collaborative environment, as team members are encouraged to integrate their code frequently, ensuring that everyone is

working with the latest version and reducing the likelihood of integration conflicts (Rossi et al., 2016). CI also contributes to improved code quality through the integration of automated testing frameworks, by running tests with every integration, potential bugs are caught early, ensuring a high standard of code quality is maintained and this early bug detection not only reduces the cost and effort required to fix issues but also enhances the overall stability and reliability of the software (Vasilescu et al., 2015). Furthermore, CI reduces integration problems by minimizing the risk of complex merge conflicts and frequent integration allows for smaller, incremental changes to be reviewed and integrated seamlessly, improving the efficiency of code review processes (Yangyang Zhao et al., 2017). This streamlined approach to development facilitates faster and more reliable release cycles, as automated build processes ensure consistent and reproducible builds across different environments (Yangyang Zhao et al., 2017). In addition to these benefits, CI can also enhance security with regular integration and testing helping to identify and mitigate security vulnerabilities early in the development process, contributing to a more secure codebase (Elazhary et al., 2022).

2.3.3 Challenges and limitations of CI

Despite the aforementioned clear advantages of CI in modern software development, its implementation and maintenance can still come with numerous challenges and limitations. Understanding these obstacles is essential for organizations looking to fully leverage CI's potential. This section outlines the technical, organizational, and inherent constraints that may impact the successful adoption of CI in various settings.

Implementing and working withing CI necessitates a cultural shift within development teams, as the approach demands new workflows, close collaboration, and a commitment to rapid feedback. And it has been identified that challenges related to the developer mindset play an important role when transitioning to CI (Debbiche et al., 2014). Developers often get used to working in a certain way and in environments where developers are unaccustomed to frequent integrations or where established workflows are highly individualized, CI can encounter resistance (Debbiche et al., 2014). Shahin et al. (2017) also note that many studies have identified change resistance as a barrier against continuous practices, but this can be reduced by presenting convincing reasons for changes. Teams may be reluctant to adopt practices that enforce more stringent coding standards or automated testing, particularly when these practices are perceived as disruptive to established workflows. The adoption of CI can also increase stress as the result of requiring more continuous updates and the continuous reviewing of changes by others (Debbiche et al., 2014; Shahin et al., 2017). This is why adopting CI requires organizations to invest in training and ongoing support to help developers

and other team members understand the benefits of CI and adapt to new processes, but this transition period can be lengthy, especially for teams with entrenched workflows or for organizations with limited resources to devote to training.

One of the primary challenges organizations face with CI implementation lies in the complexity of infrastructure, technologies and setup requirements. According to Shahin et al. (2017) several studies have identified the limitations of existing tools and technologies to be an inhibitor of establishing CI practices. For example, existing tools can be inefficient in providing feedback from the tests run in CI. Debbiche et al. (2014) report that for example most code review tools are not equipped to provide user a view of the bigger picture, but instead only provide feedback about small singular changes. Another potential problem is related to cloud applications that use external resources, as any delays or downtime in the service will immediately affect the continuous pipelines (Shahin et al., 2017). And similar issues in on-site solutions will cause similar problems, which means that implementation of CI puts more pressure on maintaining IT infrastructure.

3. QUALITY AND VALUE OF DATA FROM THE POINT OF VIEW OF BIG DATA -ANALYTICS AND MACHINE LEARNING

One potential benefit, one that doesn't get much attention in many of these studies, and the subject of this study is the utilization of CI generated data for data analytics and machine learning. This could potentially provide additional benefits, for example in the measurability of software development projects and further improve detection of problems. In the next section we look into the requirements for data to enable utilization of these technologies.

In order to answer this study's main research question of how to improve CI pipelines in order to extract useful, high-quality data from them we will first need to define what qualifies as quality data. In this chapter we will start with the basics and define data, information, and knowledge, then move on to evaluating data quality from a wider point of view and then finally move on to evaluating requirements created by machine learning and data-analytics.

3.1 Data, information, knowledge, and wisdom

In the software development context, terms like information and data, are often utilized to mean different things, so it's important for us to define what these terms mean in the context of this study. One of the most fundamental models in information and knowledge literature is the data–information–knowledge–wisdom hierarchy (DIKW) often referred as the 'Knowledge Pyramid' (Rowley, 2007). In this model data, information, knowledge, and wisdom are not synonymous, but instead have separate, well defined meanings, that form a hierarchy from data to wisdom, with understanding, meaning, and value increasing with every step on the way. An example of a simple data-information-knowledge pyramid is shown in figure 3.

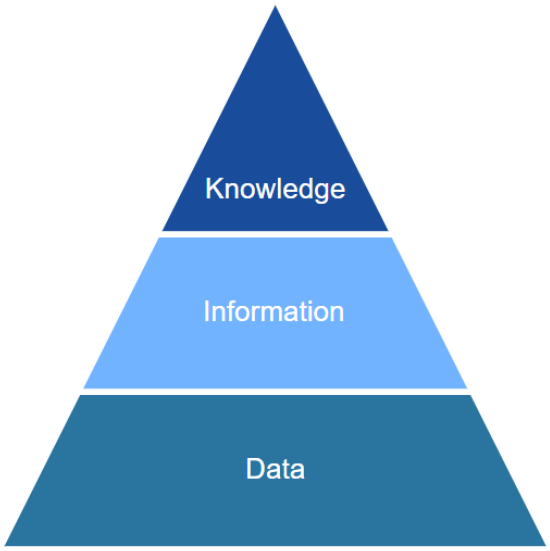

Figure 3. *Data-Information-Knowledge-Pyramid adapted from Hoppe et al., (2011)*

According to Dickerson (2022) the bottom step of the pyramid, data, can be defined as discrete, unprocessed raw set of values. Information, although frequently used interchangeably, is data endowed with additional meaning provided by its context (Dickerson, 2022). So, for example a simple set of numbers like 56, 70, 89 would be a set of data, but knowing that they would be the weights of certain three people, measured in kilograms, would elevate the data into information. Knowledge is harder to define as it doesn't exist in isolation, but it can be defined in terms of information, as the ability to transform information into instructions (Dickerson, 2022). Knowledge can exist in explicit and tacit forms, but always at least partially tied to context and understanding (Dickerson, 2022).

You might notice that this version of the Knowledge Pyramid is missing the often included concepts of wisdom and understanding. Rowley (2007) notes that many books on the subject tend to skip defining wisdom, despite its usual position at the top of wisdom hierarchy, which might be due to its elusive nature and how separated it is from information systems. And because this study is only concerned about extracting information and knowledge from CI system data, we are also going to skip defining wisdom as it falls outside the scope of the study.

3.2 Data life cycle

Data life cycle is a model describing common phases of data in a project in a comparable way to how SDLC models the phases of software development. These models can

help organizations plan and follow the data utilized by them. Figure 4 shows a data life cycle model as described by Rüegg et al. (2014).

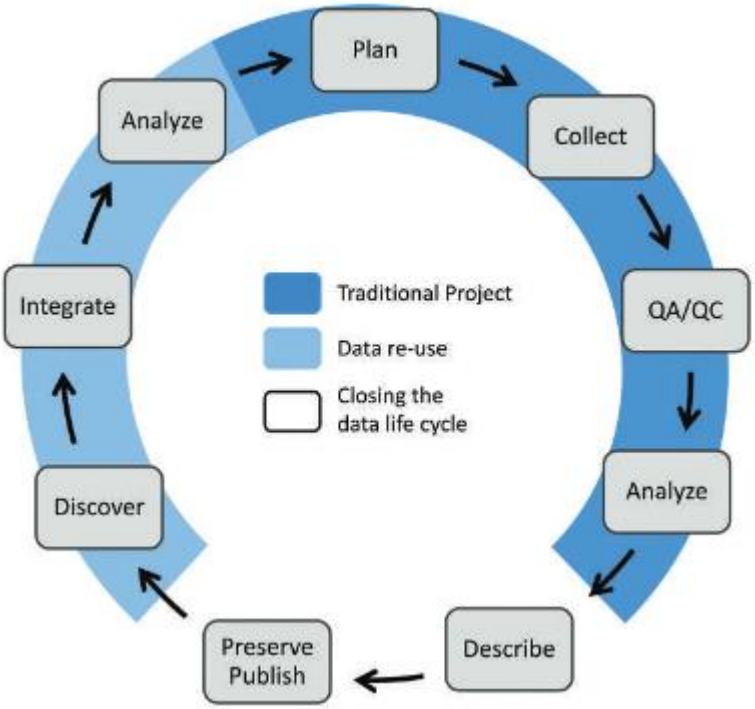

Figure 4. *Data life cycle as described by Rüegg et al. (2014)*

This model consists of two sections, a traditional project phase and data re-use phase. The traditional project phase includes planning and executing data collection, ensuring data quality, and analyzing data (Rüegg et al., 2014). Here you define objectives and requirements, collect data according to the plan, use the methodologies described in the above sections to evaluate data quality and finally use the data in analysis. According to Rüegg et al. (2014) many lifecycles end after this phase, after the analysis results have been successfully utilized, the data is often forgotten in some storage and sometimes even completely lost. But full data lifecycle should also contain data re-use phase. To enable data reuse the data should be described and preserved after it has been utilized (Rüegg et al., 2014). This means associating the data with metadata summarizing the means and purpose of the data, after which it's preserved by publishing it to some accessible location, like cloud storage, from which it can be accessed later on. Completing these steps will enable the second half of the data lifecycle process, including discovery, integration and analysis of the re-used data, which allows the data to retain value even after the associated project is over (Rüegg et al., 2014). With the complete data life cycle, data does not have a clear lifespan, but instead its constantly integrated with new data. This way the old data can be used as a basis for planning requirements for new data collection and analysis.

Hassenstein and Vanella (2022) argue that even though quality assessment, as described in section 3.2, is an individual step in the data life cycle, quality investigations and considerations are advisable at every step of the life cycle. Actions ensuring data quality should be taken during every data collection and usage step, and for example its often easier to make sure that the collected data has high quality instead of fixing issues later on after the quality assurance. This is why Hassenstein and Vanella (2022) recommend quality ensuring activities to be part of comprehensive data quality management process that promotes constant monitoring and assessment during the entire data life cycle.

The model offered by Rüegg et al. (2014) is not the only way to model data life cycles. Alaimo and Kallinikos (2024) have created data life cycle model that emphasizes different parts of data usage in organizations, which consists of data making, which leads to knowledge objects of made of data and finally data use. Figure 5 shows and a simplified example of this model.

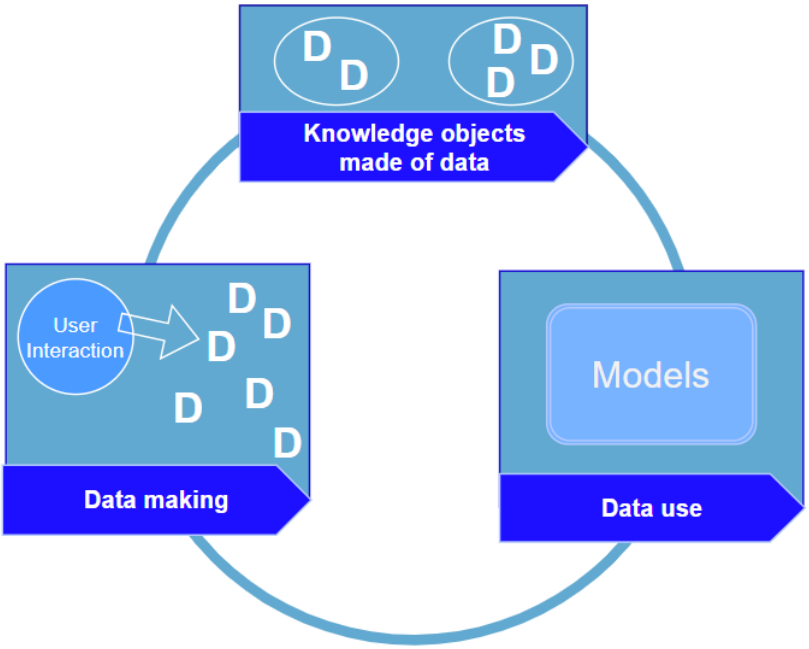

Figure 5. *Data life cycle adapted from Alaimo & Kallinikos (2024)*

According to Alaimo & Kallinikos (2024) data is often made instead of just being simply collected from some source, which is highlighted in their models initial stage which involves the creation of data, instead data collection. According to them data is often crafted through complex and distributed efforts. Alaimo and Kallinikos (2024) highlight that many digital data are native to online environments and result from user interactions designed to align with the operational needs of platforms. They emphasize that

data reuse and repurposing are integral to data making, requiring organizations to reimagine data roles and develop capabilities to integrate these operations with existing practices. In the second phase of data life cycle consists of, aggregated data being transformed into structured entities known as "data objects" (Alaimo & Kallinikos, 2024). These objects, such as customer profiles or credit scores, are formed by organizing data under specific schemas or structures, enabling new forms of knowledge creation, and facilitating novel social interactions and organizational practices. The final stage involves the application of data objects in diverse contexts, including their commodification in the market (Alaimo & Kallinikos, 2024). Data objects function as coordinative devices that structure organizational operations and shape relationships between economic actors. Alaimo and Kallinikos (2024) note that data commodities differ from traditional commodities in that they are perpetually unfinished, continuously updated with new data, and constantly performing specific functions. This dynamic nature necessitates recontextualizing and tailoring data objects in real-time to meet specific commercial needs.

3.3 Data quality

In a literature review conducted by Haug (2021) it is noted that data quality has no clear definition, but it’s generally accepted that it’s a multidimensional concept consisting from different data quality dimension, and Haug was able to identify 110 unique dimensions from the studies evaluated in the review. There doesn’t exist a common standard for which data quality dimension to use which is why researchers and practitioners need to pick a set of classifications they feel match their use case for evaluating data quality. Some of the most commonly used dimensions as identified by Hassenstein and Vanella (2022) are presented below:

1. Accuracy: Accuracy measures the degree of conformity of data to the facts or to the correct values. It is calculated by comparing the number of accurate records to the total number of records. For instance, the accuracy of phone numbers in a population register can be calculated by the percentage of confirmed numbers versus the total listed.
2. Accessibility: Accessibility or availability is related to the ability of the user to access data with the skills and technologies available. The more accessible the data, the less blocks there are between the user and the data.
3. Completeness: This refers to the presence of all expected and required data. Completeness is quantified by comparing the number of available records to the

number expected, generally resulting in a percentage. This dimension also considers whether the data resolution is sufficient for its intended use.

4. Consistency: Consistency looks at whether data are free from contradiction and follows established rules or formats. It can be quantified by examining the proportion of data that adheres to these standards. For example, ensuring dates are formatted consistently in a dataset.
5. Currentness refers to how up-to-date data are in relation to the needs of a specific task. It can often be assessed qualitatively. For example: a dataset of bird observations from the summer of 1969 is likely not suited for estimating bird populations in the year 2022.
6. Relevancy: Relevance determines how well data meets the expectations and requirements of the user. This is typically assessed qualitatively, sometimes using a scorecard to determine how well the data serves the user's needs.
7. Timeliness: This dimension measures the appropriateness of the data for the intended use, as well as the lag between an event occurring and the data being recorded. For example, the timeliness of address entries in a register could be quantified by the time lag between data collection and data entry.
8. Validity: Validity assesses whether the data conform to the required formats and rules established for them. It's similar to accuracy but focuses on adherence to specific data standards and rules.

Alternative studies like the one conducted by Cai & Zhu (2015) further divide these dimensions into smaller data quality elements, for example the data quality dimension of reliability consists of the following elements: accuracy, consistency, integrity and completeness. These elements are then assigned their own indicators, for example the indicators for accuracy include the reflection of data to source information and lack of ambiguity.

Whatever data quality dimensions are chosen their values need to be assessed for them to be actually usable. According to Batini & Scannapieco (2016) data quality assessment can be split into objective and subjective methodologies, with objective methodologies using math like the percentage of null values and subjective methodologies using human assessment. They argue that the usual process followed in assessment methodologies has 3 main activities:

1. Relevant dimensions and metrics are chosen and then measured
2. Subjective assessment is conducted by experts

3. Objective and subjective results are compared

Depending on the type of data and the dimension that is measured objective and subjective measurements are preferred, for example accuracy is typically easy to evaluate mathematically whereas relevancy is best analyzed by a human, which is why assessment should mix both subjective and objective measurements. Figure 6 presents one way to conduct data quality assessment as presented by Hassenstein and Vanella (2022).

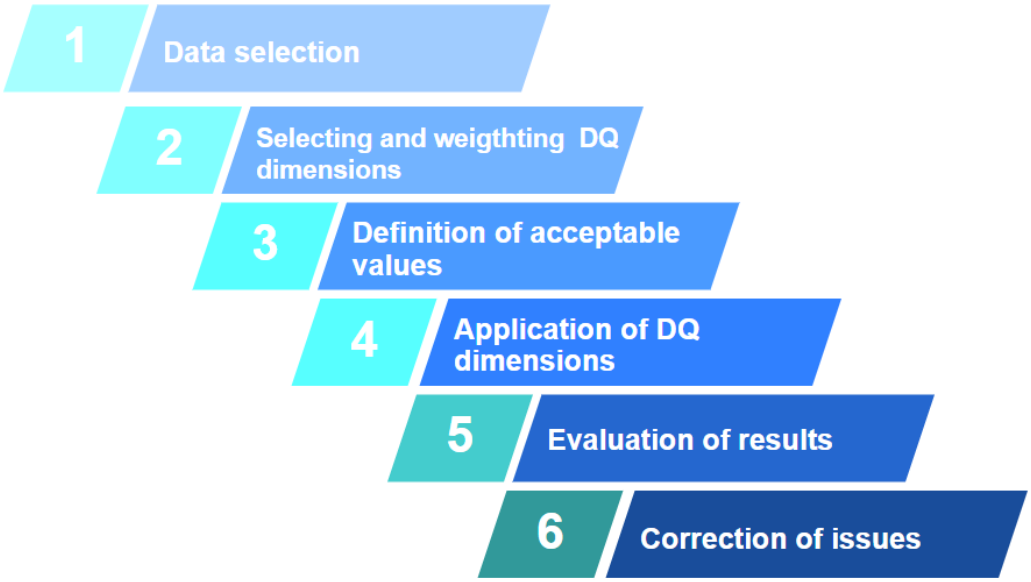

Figure 6. *Data quality assessment steps reproduced from Hassenstein and Vanella (2022)*

According to Hassenstein and Vanella (2022) data quality assessment starts from data selection, where you select the variables or sections of the data you want to evaluate. After selection you want to select the dimensions that you evaluate and weigh their importance according to the use case. In the third step you define the acceptable values for data, so any outliers can be easily spotted during the analysis. After the dimensions and values are defined, they are applied to the data and the assessors give values for the dimensions. After the results are available, they are evaluated, and potential corrections and counteractions can be implemented to fix data quality issues.

3.4 Data management

At its simplest, data management includes the management of data, information represented in a formalized manner suitable for communication, interpretation and processing (Gordon, 2007). This broadest available definition basically includes everything an organization or a person can do with data, but as a formal term data management is normally associated with the provision of organizational service (Gordon, 2007). In this paper data management is used to mean a comprehensive discipline encompassing a range of practices, policies, and procedures designed to manage data as a critical resource inside an organization. At its core, data management involves the systematic collection, organization, storage, maintenance, and use of data to support informed decision-making, operational efficiency, and strategic planning (Dhudasia et al., 2023). This means that one way to look at data management in organizations is to see it as the force making sure that the data life cycle process keeps on cycling. Data management also aims to uphold data quality by providing guidelines and standards for data handling data management seeks to create a reliable information foundation that underpins organizational processes and decisions (Gordon, 2007).

Effective data management requires a clear and structured approach, generally supported by policies that dictate data governance. Data governance policies establish who owns and manages the data, as well as how data should be categorized, protected, and archived (Khatri & Brown, 2010). An example by Khatri and Brown (2010) say that data governance includes establishing who in the organization holds decision rights for determining standards for data quality, whereas data management involves determining the actual metrics employed for data quality.

One of the foundational elements of data management is data modeling, which entails defining the structure of data and its relationships within an organization (Gordon, 2007). Data models serve as blueprints that represent data structures and are essential in organizing information within databases, enabling clear data relationships and efficient data retrieval (Lemahieu et al., 2018). Through data modeling, organizations can design data storage solutions that reflect their unique data requirements and operational workflows, ensuring data accessibility and compatibility across different systems (Lemahieu et al., 2018).

3.5 Data quality in Big Data analytics

In the previous sections, we discussed the general definition of data quality. However, as this study focuses on data as an enabler for big data analytics and machine learning, we will now examine the specific data quality requirements for these technologies.

In this section we will first define these technologies and then move on to analyzing existing theory on data requirements and see if they have additional requirements differing from generic quality dimensions.

3.5.1 Big data and big data analytics

Big data, lacks a clear universal definition, but most common way of defining big data is the so called three V's. (Berman, 2013) define them as:

4. Volume: Large amounts of data
5. Variety: The data comes in different forms, including traditional databases, images, documents, and complex records
6. Velocity: The content of the data is constantly changing, through the absorption of complementary data collections, through the introduction of previously archived data or legacy collections, and from streamed data arriving from multiple sources

By this definition big data can be seen as a dataset that is so large and varied that handling it by traditional means and using simple data processing software. For example, the output from an organization's CI pipeline could be big data by this definition, as they generate tons of changing data that cannot really be analyzed without using sophisticated tools meant for that purpose. But many recent studies on the subject argue that these themselves aren't enough to define big data, and many identify additional "V's" or big data traits. Ali-ud-din Khan et al. (2014) for example use four additional V's to define big data:

7. Veracity: Veracity is the issue of uncertainty about truthfulness of captured data due to lack of trust in data sources.
8. Validity: Somewhat similar to veracity, but veracity of data means the correctness of data with regard to the intended usage. Even if the data is correct, it might not be useful if misunderstood or used for the wrong purpose.
9. Volatility: Volatility refers to the lifetime of data. Many forms of data can have a limited lifespan, and this refers to the problem of handling outdated data in the large datasets.
10. Value: This can be seen as the desired outcome of Big Data processing. This highlights the importance of Big Data bringing value to the organization utilizing it.

These additional V's highlight the role of Big Data as a useful resource. Instead of Big Data just being a large amount of data, it is defined as a large amount of data that is actually useful for organizational value generation. Some studies have identified even more defining features for Big Data, and for example Bansal et al. (2021) have identified 13 different V's in their survey, but it this paper Big Data is defined using these seven V's with the focus on using it to generate value for organizations,

Big data doesn't provide any value to organizations in a vacuum, which is why the term is often utilized in the context of big data analytics (BDA), a process which turns big data into meaningful insights (Gandomi & Haider, 2015). An example of BDA life cycle, as described by Dai et al., (2019) is shown in figure 7.

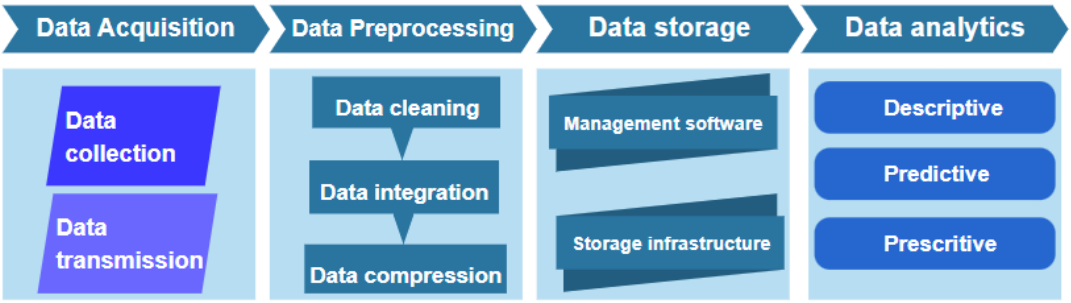

Figure 7. *BDA life cycle adapted from Dai et al. (2019)*

According to Dai et al., (2019) a typical BDA life cycle consists of:

1. Data acquisition. The first step consists of data collection and data transmission. It involves collecting raw data from various data sources with dedicated data collection technologies and then transmitting it to the data storage system via wired or wireless networks.
2. Data preprocessing. After collecting raw data, the raw data needs to be preprocessed before keeping it in data storage systems because of the big volume, duplication, and uncertainty features of the raw data. This includes data cleaning, data integration, and data compression.
3. Data Storage: Data storage involves the processes and technologies used to store and manage large volumes of data. This system is composed of two primary layers: the storage infrastructure and the data management software. The storage infrastructure includes not only the storage devices themselves but also the network devices that interconnect with them. Alongside these networked storage devices, data management software plays a crucial role in the overall data storage system.

4. Data Analytics: In this phase, a variety of data analytics techniques are employed to derive meaningful insights from extensive data sets. These techniques can be broadly classified into three categories: descriptive analytics, predictive analytics, and prescriptive analytics. With descriptive analytics describing what has happened, predictive analytics try to figure out what is going to happen in future and prescriptive analytics extending the results to make the right decisions.

The BDA process can help organizations to generate value and competitive advantage, by generating valuable knowledge resources by utilizing customized BDA applications with proprietary data resources (Dahiya et al., 2022). According to Dahiya et al. (2022) this access to high quality proprietary data, for example one acquired from CI pipeline, is an important requirement for sustainable competitive advantage.

3.6 Machine learning

Machine learning (ML) is a subset of artificial intelligence (AI) that involves the development of algorithms and statistical models enabling computers to perform tasks by learning from data rather than being explicitly programmed (Chopra, 2023). This field allows machines to improve their performance on tasks based on historical, present, and future data, drawing from various disciplines such as computer science, statistics, cognitive science, engineering, and mathematics (Chopra, 2023). ML algorithms are often split into three different categories: supervised, unsupervised and learning algorithms. Figure 8 shows classification of different machine learning methods.

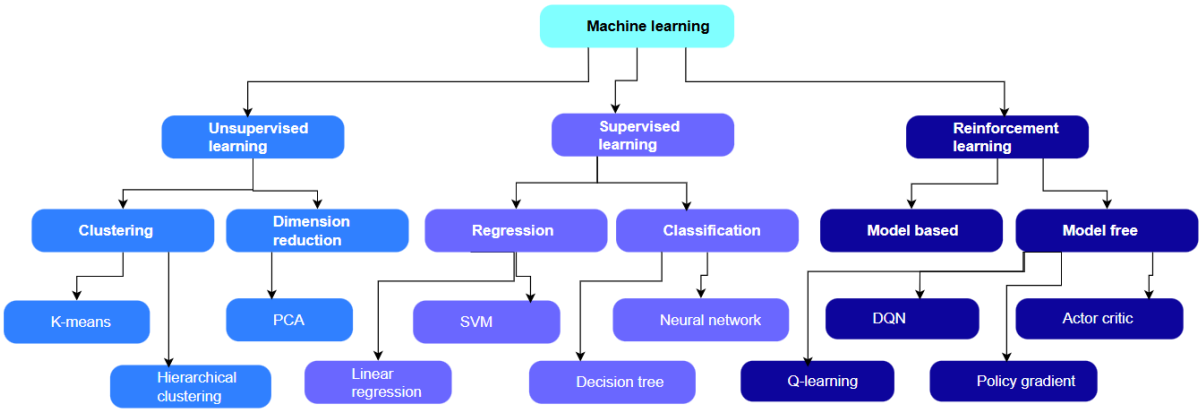

Figure 8. *Machine learning methods adapted from Badillo et al. (2020)*

3.6.1 Supervised learning

Supervised learning is a foundational paradigm in ML, where the goal is to learn a mapping function from input variables, known as features, to output variables, known

as labels, using a labeled dataset (Alloghani et al., 2020).
This technique is widely utilized in tasks where historical data with known outcomes is available, enabling predictive models to generalize patterns for unseen instances (Badillo et al., 2020). According to Badillo et al. (2020), there are two main categories of supervised learning:

1. **Regression**: Used for predicting continuous-valued outputs (e.g., house prices, stock market trends). Uses numerical labels.
2. **Classification**: Used for predicting discrete categories (e.g., spam email detection, disease diagnosis). Uses categorial labels.

These categories can be further broken down into different learning methods. Some examples of the most common supervised learning methods are explained below.

Linear Regression: The simplest regression model uses the relationship between independent variables and a dependent variable using a linear equation. Linear regression predicts outcomes based on the weighted sum of input features, where weights are optimized to minimize the difference between predicted and actual values.

Support Vector Machines (SVM): Similar to all regression methods, the objective of SVR is to postulate a function on the inputs that can help estimate for the observed output (Badillo et al., 2020). SVM is designed to minimize classification error by using supporting vectors that create margin between the different classes, with the idea that this margin is maximized during training (Kolosova, 2021). One downside to this process is the fact that this requires calculation of these additional vectors. This highlights the generic problem in ML, that using methods to minimize errors usually increases the requirements for computational resources.

Decision Trees and Random Forests: Decision trees are an essential building block for many ML algorithms (Badillo et al., 2020). Decision trees operate by recursively partitioning the data based on feature values, creating a tree structure where each internal node represents a decision based on a feature threshold, and each leaf node represents a predicted outcome (Badillo et al., 2020). While straightforward and interpretable, decision trees are prone to overfitting, particularly with complex datasets (Badillo et al., 2020). To mitigate this, ensemble methods like Random Forests combine multiple decision trees, often trained on different subsets of the data and features, to produce more robust predictions (Badillo et al., 2020). The final output is typically determined by averaging (for regression) or voting (for classification) across all trees. Gradient

Boosted Decision Trees represent another advancement, where trees are built sequentially, and each new tree focuses on correcting errors made by previous ones (Badillo et al., 2020).

Neural Networks: The neural network method was created as an attempt to mimic the way humans process and store information (Kolosova, 2021). Neural networks consist of layers of interconnected nodes, or "neurons," that process data iteratively through weighted connections (Badillo et al., 2020). Each neuron applies an activation function, introducing non-linearity into the model. A neural network learns by determining the number of neurons and adjusting the weights for the connections (Kolosova, 2021). This process uses a training dataset composed of input– output pairs and tries to find a function that produces the outputs based on the given inputs. This training typically involves large datasets and requires significant computational power (Badillo et al., 2020). Neural networks can be either fixed or dynamic. Fixed networks become set after the initial learning phase is completed, but dynamic networks keep on adjusting the values based on the input data (Kolosova, 2021).

Supervised learning is integral to various domains, including healthcare, finance, and natural language processing (Alloghani et al., 2020). But despite its versatility, supervised learning requires large amounts of labeled data, which can be expensive and time-consuming to obtain (Kolosova, 2021). Additionally, models are prone to overfitting if not properly regularized, limiting their ability to generalize to unseen data (Kolosova, 2021). Addressing these challenges necessitates robust data preprocessing, feature engineering, and model validation techniques.

3.6.2 Unsupervised learning

Whereas supervised learning is taught with labeled data, unsupervised learning deals with unlabeled data instead. The model tries to learn the underlying structure of the data without explicit instructions on what to predict (Alloghani et al., 2020). According to Alloghani et al. (2020) the unsupervised learning methods can be roughly split into two groups: clustering and dimension reduction.

1. **Clustering**: Used to group similar datapoints together
2. **Dimension reduction:** Aims to reduce the number of variables under consideration

These categories can be further broken down into different learning methods. Some examples of the most common unsupervised learning methods are explained below.

K-means: This simple clustering method uses the distance of each datapoint from the cluster center to assign them to the nearest cluster, and then updates the cluster center based on the newly assigned datapoint (Naeem et al., 2023). The advantage of this method is mainly its speed which makes it good for dealing with large datasets (Naeem et al., 2023). But this comes with the disadvantage that for to be able to execute the analysis you need to know the amount of cluster beforehand, also the input order of data affects the initial cluster formation, which introduces randomness to the method (Naeem et al., 2023).

Hierarchical clustering: This clustering method builds a hierarchy of clusters, for example in neighbor joining the pairwise distances of all data points are calculated and then closest distances are grouped together (Badillo et al., 2020). But this same clustering can be taken further to reduce the number of clusters produced. Because this method calculates distances between all datapoints its much slower than K-means clustering for example which limits in useability with large datasets (Naeem et al., 2023).

PCA: Principal component analysis is a dimension reduction method that reduces the dimensions of big data sets while still retaining most of the information, it uses simple linear algebra that is fast to compute even for large amounts of data (Naeem et al., 2023). Reducing dimensions helps computers by making calculations faster and humans by making the data easier to understand, but like with any dimension reduction method the loss of information is unavoidable (Naeem et al., 2023).

Whereas supervised learning is often used to make predictions, unsupervised learning is used to explore data and find insights from it. Applications of unsupervised learning include customer segmentation in marketing, anomaly detection in network security, and gene expression analysis in bioinformatics (Alloghani et al., 2020).

3.6.3 Reinforcement learning

Reinforcement learning involves training an agent to make sequences of decisions by rewarding it for good actions and punishing it for bad ones. This type of learning is particularly useful in scenarios where an agent interacts with an environment, such as robotics, game playing, and autonomous driving (Krichen, 2023). The agent learns to maximize cumulative rewards over time, effectively improving its performance through trial and error. Reinforcement learning has been successfully applied to optimize supply chain logistics, develop sophisticated game-playing algorithms and create adaptive control systems in various industries (Krichen, 2023).

According to Sutton and Barto (2018), reinforcement learning methods can generally be categorized into two main groups: model-free and model-based methods.

1. **Model-free methods**: These do not rely on a model of the environment and learn directly from interaction.
2. **Model-based methods**: These involve creating a model of the environment to simulate potential outcomes and plan actions accordingly.

Q-learning: a model-free reinforcement learning algorithm that learns a value function to estimate the expected utility of taking a given action in a particular state. The agent iteratively updates the Q-values based on the Bellman equation, allowing it to learn optimal policies over time (Watkins & Dayan, 1992). The advantage of Q-Learning is its simplicity and ability to handle environments with discrete states and actions. However, it struggles with scalability for environments with high-dimensional state or action spaces (Mnih et al., 2015).

Deep Q-Networks (DQN): DQN extends Q-Learning by incorporating deep neural networks to approximate Q-values for complex environments with continuous or high-dimensional states (Mnih et al., 2015). This method leverages techniques such as experience replay and target networks to stabilize learning. While DQN has proven highly effective in tasks like playing video games, it requires significant computational resources and careful tuning of hyperparameters (Mnih et al., 2015).

Policy Gradient Methods: Policy gradient methods directly optimize the policy, a mapping from states to actions, by maximizing the expected cumulative reward (Sutton et al., 1999). Unlike value-based methods, policy gradient approaches can handle stochastic policies and continuous action spaces. The advantage of this method is its flexibility in solving a wide range of tasks, but it can suffer from high variance and slower convergence (Sutton et al., 1999).

Actor-Critic Methods: Actor-critic methods combine value-based and policy-based approaches by maintaining two separate components: an actor, which selects actions, and a critic, which evaluates the quality of the actor's actions (Konda & Tsitsiklis, 2003). This hybrid approach helps to reduce variance in policy gradient methods while retaining their advantages. However, these methods can be more complex to implement and require balancing between the actor and critic components.

While reinforcement learning has demonstrated remarkable success in certain domains, it also comes with challenges such as sample inefficiency, difficulty in designing reward functions, and lack of interpretability in learned policies (Sutton & Barto, 2018).

Despite these hurdles, ongoing advancements in algorithms and computational power continue to push the boundaries of what reinforcement learning can achieve.

3.7 BDA and ML data quality requirements

In this chapter, we have explored the basics of ML and BDA on data quality; this section aims to dig deeper into the data quality requirements of these two technologies. Table 1 highlights the most relevant dimensions of data quality necessary for enabling BDA and ML, along with the corresponding studies that have identified their significance. The sections 3.7.1 and 3.7.2 present the findings of these studies in more detail. This overview on to the data quality requirements shows us that from the data quality dimensions identified by Hassenstein and Vanella (2022), accuracy and completeness are seen as the most important dimensions for ML and BDA. Currentness, accessibility and validity show up in some studies, but not as frequently. Finally, consistency, relevancy and timeliness dimensions were not mentioned by any studies. While this review wasn't exhaustive, it prods us towards focusing on accuracy and completeness of data when trying to improve the data quality for ML and BDA.

Table 1. *Most relevant data quality dimensions for enabling BDA and ML*

Data quality dimension	**Source and related technology**
Accuracy	Côrte-Real et al. (2020) – BDA Chen et al. (2021) – ML Mohammed et al. (2024) – ML Fosso Wamba et al. (2019) - BDA
Accessibility	Wook et al. (2021) - BDA
Completeness	Côrte-Real et al. (2020) – BDA Chen et al. (2021) – ML Mohammed et al. (2024) – ML Fosso Wamba et al. (2019) - BDA
Consistency	
Currentness	Côrte-Real et al. (2020) – BDA Fosso Wamba et al. (2019) - BDA
Relevancy	
Timeliness	
Validity	Chen et al. (2021) - ML

3.7.1 BDA data quality requirements

If organizations want to utilize big data analytics to generate competitive advantage, they will first need to make sure that any proprietary or open data actually meets data quality requirements for meaningful analysis. The inherent qualities of big data provide

challenges in data quality analysis, as identified by Cai and Zhu (2015) the big proportions of unstructured data cause problems for maintaining high data quality. In this section we will discuss the existing theory on which data quality dimensions have the most direct impact on BDA process. The results are also displayed in table 1 with corresponding studies mapped to the most important data quality dimensions identified in it.

Study by Côrte-Real et al. (2020) studied the effects of data quality on organizations BDA capabilities and identified that data quality has significant positive impact on BDA. They studied the effects of completeness, accuracy, and currency with each of these DQDs having a strong correlation with successful BDA. Wook et al. (2021) mapped big data traits and their relations to data quality dimensions and found out that accessibility has a significant influence on BDA application, but didn't find any correlations with other quality dimensions. This result could however have been caused by the study being based on a survey conducted to university students. Fosso Wamba et al. (2019) found out in their study that BDA information quality is reflected by the DQDs of completeness, currency, and accuracy. Their analysis concluded that these data quality dimensions have a positive impact on company performance and business value.

3.7.2 ML data quality requirements

According to Gong et al. (2023) data quality is a crucial aspect of ML and the success ML models heavily relies on the quality of the dataset used to train and evaluate the them, and the saying ''garbage in, garbage out'' emphasizes the importance of using a high-quality dataset in machine learning. In this section we go through existing theory about the requirements for data to be able to be used effectively for machine learning and which data quality dimensions are most important for ML. The results are also shown in table 1 along with BDA results, with corresponding source mapped to the most important dimensions identified in it.

Chen et al. (2021) identified three data quality attributes which were the most important to deep learning: comprehensiveness, correctness, and variety. According to them comprehensiveness means that a dataset contains all representative samples from the population, which is important as especially deep learning models require millions of parameters and need a large amount of data to train them. Their definition can be seen synonymous with the completeness as defined by Hassenstein and Vanella (2022). According to Chen et al. (2021) correctness of data, meaning the fact that a record in a dataset is accurate and valid, is important for ML algorithms to reduce data noise. Mistakes in training data a repeated in the final mode. The final attribute identified by Chen

et al. (2021) is variety, meaning that any datasets used in training should be a representative of the entire population. This attribute can be seen as being part of completeness dimension.

Mohammed et al. (2024) tested the effects of completeness, accuracy and uniqueness on ML algorithms. They learned that compromising completeness of training data affects the models, but that this can be mitigated by training the model with data that is also missing data, but non the less completeness has a noticeable effect on model performance. Accuracy of the data also had an impact on model performance, but performance loss starts to only have a noticeable negative effect after more than 20% of entries are inaccurate. Data uniqueness, meaning data that doesn’t have duplicate values and data consistency, had negligible effect on performance.

4. RESEARCH DESIGN

This chapter aims to explain and justify the chosen research design for this thesis. The thesis was designed with the help of the research onion as described by Saunders et al. (2019). In this chapter we will go through the research onion and explain my choices for the design. The research onion, with the choices made in this study, is shown in figure 9 below.

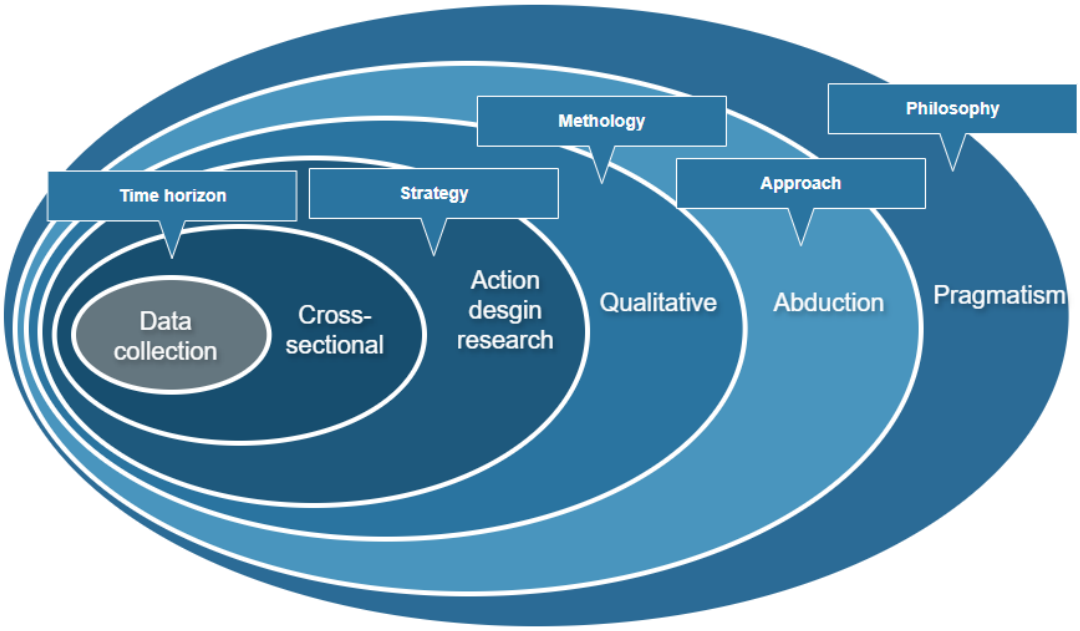

Figure 9. *Research onion adapted from Saunders et al. (2019)*

The research onion consists of layers which help you to formulate your research by going through the layers starting from the outermost layer of choosing the research philosophy and moving on through choosing the research approach, methodology, strategy, and time horizon, with the final layer being the data collection and data analysis. The following sections explain each of the choices made for this research.

4.1 Research philosophy

According to Saunders et al. (2019) research philosophy as a concept refers to a system of beliefs and assumptions about the development of knowledge. Whether you are consciously aware of your assumptions and beliefs or not, they shape the way you un-

derstand your research questions, methods you use and how you interpret your findings (Saunders et al., 2019). Which is why it's preferred to make a conscious effort to understand and document these beliefs and assumptions that affect your research.

This study adopts the research philosophy of pragmatism, as described by Saunders et al. (2019), which asserts that concepts are only relevant where they support action. Saunders et al. (2019) say that researchers that have adopted pragmatic research philosophy are more inclined to get on with their research and trying to make a difference to organizational practice, instead of battling with different philosophical assumptions. Pragmatists start with a problem, and aim to create a practical solution for it, even if it would mean utilizing several different types of knowledge or methods (Saunders et al.,2019). This research philosophy was adopted in this study, because at the center of this research is a practical problem in a specific organization that this research aims to solve and then expand the findings into a more general model.

4.2 Research approach

Next layer of the "research onion" as described by Saunders et al. (2019) is the research approach, for which they list three possible approaches: Deduction, Induction and Abduction. In deductive research you start with theory, with the goal of the study being verifying or falsifying the theory, which is the more classical way of conducting research especially in the natural sciences (Saunders et al., 2019). On the other hand, in inductive research, you start with data collection and exploration with the goal of generating a theory based on the data (Saunders et al., 2019). The third approach, Abduction, is described as the combination of the two previously mentioned approaches, where you don't try to move from data to theory or from theory to data, but to move back and front between the methods while formalizing the theory (Saunders et al., 2019).

This research was decided to be conducted by utilizing the abductive approach, as it complements the pragmatic research philosophy by enabling flexibility in the theory generation. The study will gravitate towards a more inductive approach as the primary goal of the study is to generate theoretical model based on the collected data, but the theory will also be tested iteratively, and existing theory is incorporated in the process.

4.3 Research methodology

According to Saunders et al. (2019) the primary way of differentiating different methodological choices is by their usage of numeric and non-numeric data, in quantitative research data is collected in numeric format and in qualitative research, in words, images, recordings et cetera. In this research the choice has been made to use mixed methods that integrate the use of both quantitative and qualitative data collection and analysis. Saunders et al. (2019) mention that pragmatists usually assert that there are many ways of interpreting the world and different methods are often appropriate within one study, and this one is not an exception. As the goal of this study is to research ways of improving data collection and analysis in continuous integration pipelines, quantitative methods can be used for measuring the data we are able to get from them. But as this study primarily focuses on evaluating the value and potential this improvement gives to the organization, qualitative methods are going to be preferred for project evaluation.

4.4 Research strategy

The research strategy can be defined as a plan of how the research will try to answer the research questions (Sanders et al., 2019). From the possible strategies highlighted by Saunders et al. (2019) the action research was the most interesting, which is said to be an emergent and iterative process of inquire designed to develop solutions to real organizational problems by collaborative approaches. This strategy would function well in this research, as the central purpose is to study how continuous integration pipelines can be improved to provide us with higher quality data in a real organizational setting. But as this study also aims to design a model describing the process of improving the pipelines, from a pragmatist point of view also make sense to combine parts of design science research, which is defined by Collatto et al. (2018) as a research strategy that aims to generate knowledge on designing an artifact or prescribing a satisfactory solution to a problem. As Colatto et al. (2018) note, design science research and action research share a lot of similarities, which is why these different strategies can be painlessly combined in a single study. And this is also why this study will utilize the combination of these two strategies known as Action Design Research (ADR).

According to Sein et al. (2011) ADR is a research strategy for generating prescriptive design knowledge through building and evaluating ensemble IT artifacts in an organizational setting. ADR fits this research perfectly as its fit to deal with two disparate challenges: addressing a problem in a specific organizational setting and by constructing an IT artifact that is fit to address the class of problems in a wider scope (Sein et al.,

2011). The ADR research strategy stages and principles, as described by Sein et al. (2011) are presented in Figure 10 and the following sub-sections explain the steps of the chosen research strategy in more detail.

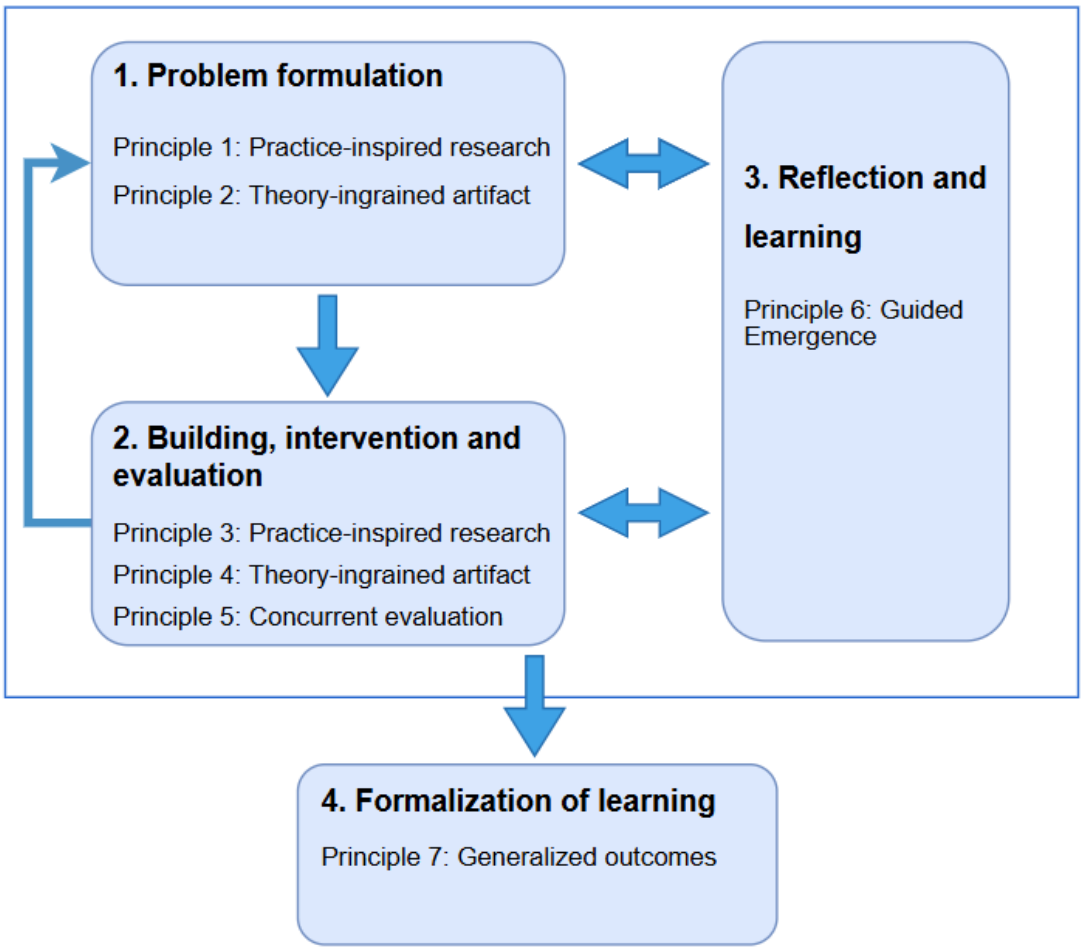

Figure 10. *Action Design Research process reproduced from Sein et al. (2011)*

The ADR process starts with problem formulation that is triggered for the first time by a problem perceived in practice or anticipated by researchers (Sein et al., 2011). In this study the trigger was a perceived problem in the target organization. According to Sein et al. (2011) this stage consists of determining the initial scope, deciding the roles and scope for practitioner participation, and formulating the initial research questions. They also note that this stage draws on two principles: practice-inspired research and theory-ingrained artifact. Practice-inspired research means using the problem as a knowledge creation opportunity and the theory-ingrained artifact that the artifact to be created is being informed by existing theories.

The second stage, building, intervention, and evaluation (BIE) builds on the theoretical premises and formulated problems from stage one, by building an IT artifact, intervention in organization and evaluation (Sein et al. 2011). This whole stage is a process where the two domains, IT-artifact, and the organizational context, are iteratively and being developed and evaluated in parallel, with both helping to understand each other (Sein et al. 2011). This study is going to take a more IT-Dominant approach to this stage, which is identified by Sein et al. (2011) to suit ADR efforts that emphasize creating an innovative technological design at the outset, in which early designs of an artifact are continuously tested in limited scope and then later brought into wider organizational use and evaluated in the use setting. In this study the IT-artifact is the model that describes how to improve CI pipelines, and the organizational context is doing the improvement in practice and evaluating its effects.

Running parallel to the first two stages is the reflection and learning stage, as the two aforementioned stages are more concerned with evaluating and solving the problem in the specific context, reflection and learning stage is concerned with applying the learning to a broader class of problems (Sein et al. 2011). This stage is concerned with the evolution of preliminary design, by conscious reflection on the problem framing, the theories chosen to ensure that contributions to knowledge are identified (Sein et al. 2011). In this study this means when working on the CI pipeline improvement the artifact describing this model is continuously developed based on the experience gained during implementation.

Formalization of learning is the final stage of ADR research, in which the highly specific organizational context of the study is generalized into a general solution for a class of problems (Sein et al., 2011). This means that this stage takes the things learned previously and formalizes it into a concise and general model. This can be done with three steps: first generalizing the problem, secondly generalizing the solution, and then deriving design principles form outcomes (Sein et al., 2011). This stage ensures that the answers generated by this study are not only useful for a single organization but actually have wider scientific value. In this study the results of the formalization of learning are described in section.

4.5 Time Horizon

According to Saunders et al. (2019) study time horizons can be roughly split into two distinct categories: cross-sectional and longitudinal studies. Cross-sectional studies are concerned with particular phenomenon at a particular time, whereas longitudinal research accounts change and development over some time (Saunders, 2019). This

study is going to be more cross-sectional as the study examines the situation in a single target organization during a single project. Where possible, a more longitudinal view of the subject is offered by including applicable data points from other studies, but overall, the scope of this study does not enable more longitudinal analysis of the subject.

4.6 Data collection

At the core of the “research onion” by Saunders et al. (2019) is data collection, arguably the most crucial step, in which the data actually need to conduct the research is collected. As discussed in section 5.4, this study utilizes ADR research strategy, which means that the data collection will be conducted iteratively over the whole research process instead of being separated as its own stage. This research uses three different data collection methods, literature review and implementation process.

This study uses literature review to gain insight into existing information about this topic. To see if similar models also exist and to potentially compare this study’s results to previous research. The research used in this study is explained in chapters two and three. The literature review was not conducted systematically due to the continuous nature of ADR process, but instead the literature was searched to answer questions that were raised during the ADR process. The goal of the literature review was to create an initial theory-based artifact that can be improved through the implementation process.

But at the center of an IT-Dominant ADR process is the practical development of the IT artifact in an organizational context. This section describes the general steps taken during the implementation process. As the iterative implementation process generates tacit knowledge instead of explicit data, the results cannot be explicitly documented, but the resulting modifications to the artifact are documented in section 5.2.

The goal of the implementation process was to improve the quality of the data collected from the CI pipelines to allow better utilization of big data analytics and lay the foundation for possible utilization of machine learning. The project would also help to refactor the existing CI pipelines to improve their maintainability.

The five different steps that were identified during the CI pipeline improvement project are:

1. Requirement analysis
2. Planning
3. Construction of CI job template

4. Implementing common CI template to CI
5. Improving data collection scripts

The process started from the requirement analysis, where the goals and requirements for the implementation process were set, and we came up with the goals for this project. After requirements were set, we entered a planning phase with the goal of creating a structured approach that would guide the project's execution effectively. We, withing the organizations CI team, concluded that the best course of action would be to develop a standardized CI pipeline template. This template would harmonize all CI pipelines to ensure uniformity in the data output. This common template should streamline CI processes, making them more efficient and easier to manage. It was designed to incorporate the best practices and standardized configurations to ensure consistency across different projects. Following the planning phase, the project transitioned to the design phase, focusing on creating a common CI template. This process involved a thorough examination of the existing CI pipelines to identify individual components that were common across different jobs. These common elements were then integrated into a single, unified pipeline template. Once the template was constructed, we proceeded to modify the existing CI pipelines to adhere to the new standardized template. Finally, we updated the scripts responsible for collecting data from the pipelines to utilize the new template, enabling the collection of more detailed data.

5. RESEARCH IMPLEMENTATION

In this chapter we will discuss the ADR process in practice. The chapter has been split into four sections, each highlighting one stage in the process as described in chapter 4.

5.1 Problem Formulation

As mentioned in section 4.4.1 ADR process starts with problem formulation that is set in motion by perceived existing or potential problem. In this research, a problem was perceived in the target organization related to the requirements of data management for CI pipelines. The objective of the improvement conducted during this study was to improve the organization's capability to effectively gather valuable data from continuous integration of CI pipelines. Additionally, this initiative aimed to lay the groundwork for the potential future integration of machine learning techniques to further optimize the process. The primary challenge with the existing CI pipelines was the lack of standardization, as they were developed by various individuals employing diverse methodologies. This resulted in inconsistent data outputs, in many cases just long log files, making it difficult to achieve data consistency for analysis and measurement.

The principle of Practice-Inspired Research emphasizes using these field problems as knowledge creation opportunities and in essence, the implementation process was driven by three practical objectives:

1. Standardized data outputs to facilitate the use of big data analytics.
2. Ensure compatibility with potential future machine learning applications
3. Unify CI pipelines and improve documentation to enhance maintainability.

This study focuses on the first two objectives, as maintenance of CI pipelines falls outside the scope of this study.

The other principle of problem formulation, the Theory-Ingrained Artifact, emphasizes that artifacts are backed by theories. In this study the prior theories mostly guide the design of the artifact. The practical organizational goals require the data to fulfill requirements for both data analytics and machine learning applications. Prior theory was utilized to formalize these requirements which then guided the process. Table 1 shows the most important data quality dimensions for enabling data analytics and machine learning, and the data quality assessment process shown in figure 4, were some of the most impactful theories in formalizing the initial artifact.

After the first problem formulation stage we ended up with the following initial design principles:

1. We need to be able to standardize CI pipelines
2. We need to be to categorize different CI pipeline steps
3. The output should fulfill data quality dimension requirements
4. We need to be able to assess data quality during the process

With these initial principles we could move on to the building, intervention and evaluation stage of ADR process. At this point, it's important to note that ADR is an iterative process as illustrated in figure 8 and the problem formulation is a live process that is revisited after each BIE stage is complete. However, for the sake of clarity in this paper, any insights generated during the subsequent problem formulation stages are consolidated and presented in Section 5.2.

5.2 Building, Intervention and Evaluation

The BIE stage builds on the problem farming and theoretical premises adopted in stage one and is the stage that will create the initial artifact (Sein et al., 2011). As mentioned, this study was conducted at an IT-dominant BIE where the creation of a technological design is emphasized. According to Seint et al. (2011) BIE stage consists of 4 tasks starting from initial discovery of knowledge creation target. In this case the initial discussion within the team resulted in the decision to focus on targeting the creation and evaluation of common CI job template. The rationale behind this approach was that the design principles could be realized through a template that standardizes and categorizes all common CI steps, thereby improving data quality. This template is described in more detail later in this section.

The second task in the BIE stage involves the selection of the BIE form, which provides the framework for the actions taken during each BIE cycle. The form used in this study is presented in figure 11. The key stakeholders in this process include I, as the researcher, the CI team, and the organizational end-users. The CI team is responsible for facilitating the broader adoption of CI-related improvements within the organization, while the organizational end-users consist of various teams utilizing the CI services provided by the CI team. In addition to my role as a researcher, I also partially acted as a practitioner within the CI team.

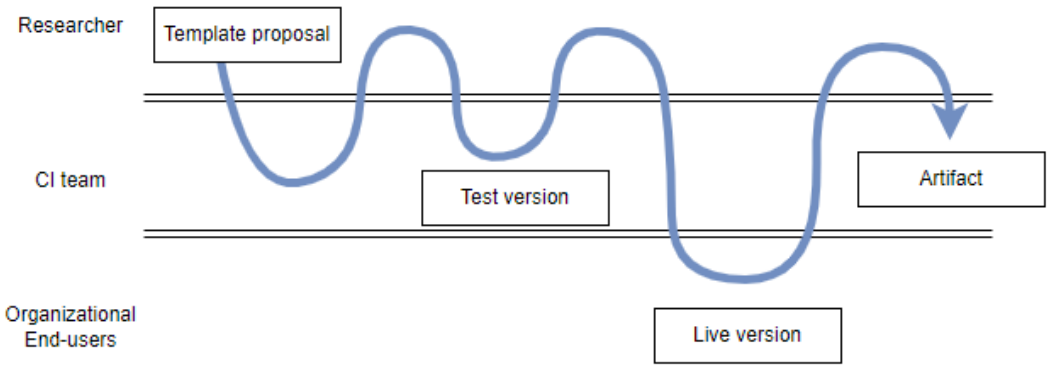

Figure 11. *Study's BIE schema constructed based on Sein et al. (2011)*

This form outlines the practical approach of the study, where the CI template is initially developed by the researcher and then iteratively refined in collaboration with the CI team, incorporating their feedback and modifying the solution accordingly. After the initial iterations with the CI team, the process expands to include input from end users and their feedback is incorporated through additional feedback loops with the process and finally ends to evaluation of the artifact in the live use setting. After the selection of BIE schema, we could move to the third and fourth tasks of BIE stage are execution of the planned cycles and assessment need for any additional cycles.

The initial BIE cycle started from thorough examination of the existing CI pipelines, with the goal being to gain preliminary understanding of the logic behind them and which way they were built. During this process it became immediately clear that it would be impossible to find a single solution that would easily fit into every existing CI job. To address this issue, we would divide existing jobs into modular components that could be standardized and unified. Although each job would comprise different components, this approach would enable the construction of each job from a set of common building blocks, which would then allow us to also control the data outputs. Common blocks identified from the jobs included, for example checkout from version control, compilation, running of different testing tools etc. To enable standardized data output, new shared functions were also created for some tasks. For example, instead of using a unique set of commands in each CI job for file checkout from version control, a common function was implemented to standardize the output across jobs. These common functions would also enable the CI team to make changes to all CI jobs without having to manually update every job.

After identifying the individual CI job components, they were consolidated into a template for different types of CI jobs in use. These templates were designed to guide the

refactoring of existing jobs and to serve as a foundation for creating new jobs in the future. Once the initial template versions were complete, they underwent iterative feedback loops within the CI team and were refined based on this feedback. Subsequently, the templates progressed to the proof-of-concept phase, where they were implemented within a single software team to test the feasibility and functionality of the approach.

This would offer feedback on the practical usage of the templates and highlight any issues missing in the conceptual stage. After the proof-of-concept phase was completed, the refined template was shared with software teams across the organization to obtain additional, practical insights before broader implementation. This step was important, as the developers working directly with software products possess a deeper understanding of the specific requirements for building and testing their components than the CI personnel alone. The feedback from this stage primarily informed refinements to the implementation of common functions, with minimal impact on the template's overall structure. Additionally, this opportunity was used to communicate to organizational end-users about the coming changes and why we consider them to be important to be implemented, to pre-emptively combat any potential resistance that such a substantial modification to the existing system could cause.

Once the initial feedback rounds had been completed the actual implementation of the CI job template was ready to begin, which meant updating all existing jobs to align with the new template. The workload was split between CI team members and software teams, with the idea that CI team members would work together with developers to implement changes to their respective jobs. Notably, the job template remained open to adjustments based on feedback gathered throughout implementation. There were several edge cases, missing during the conceptual phase, that would now be surfacing during the implementation. When any shortcomings were identified, the template was updated to address these cases. The implementation process proved to be labor-intensive, with varying levels of CI knowledge across software development teams regarding their respective jobs. In some cases, no team members had detailed knowledge of the existing CI jobs. Nonetheless, most jobs were eventually converted to align with the new template, and even though upkeeping CI jobs requires constant effort, the converted jobs were sufficiently complete to ensure that the data generated was meaningful and valuable.

Once the CI jobs were modified to follow the template, the data collection phase could commence. The data from the CI pipelines was gathered using a script that leveraged the application programming interface (API) of our CI platform, as well as the logs gen-

erated during script execution. This script collected data based on the components defined in the CI template and utilized the stage API results and logs for preliminary analytics. Preliminary classification was performed using simple logic. For instance, log entries containing phrases such as *"compilation error"* or *"script returned error code"* were categorized as compilation errors. While this approach proved functional, it leaves room for the future integration of more advanced machine learning methods for classification. Following the initial analysis, the data was transferred to a big data analytics platform for further processing. Our first use case involved creating a visualization to map recurring errors in the CI pipelines. This visualization enabled the generation of alerts when a specific error's frequency increased significantly, providing an opportunity to forecast potential new issues in the CI pipelines.

Afterwards we could evaluate the whole improvement process and conclude that the data quality has seemingly improved. It wasn't possible to conduct objective measures of the improvements due to lack of initial data points, but the general consensus among team members was that data quality improved noticeably.

5.3 Reflection and learning

Reflection and learning stage moves from building a solution in a particular instance to generate learning for a broader scale of problems (Sein et al., 2011). As shown in Figure 8, this stage runs parallel to the first two stages. This basically meant that during the implementation process we would also take notes of things that could be generalized to apply in different organizational contexts. The four key takeaways gained from the implementation process are presented in this section

Takeaway 1: Importance of planning. The first key takeaway highlights the importance of the initial planning
Effective planning emerged as a cornerstone for the success of this project. The initial phase of requirement analysis and goal setting provided clarity and direction, ensuring alignment between technical improvements and organizational needs. Without a structured planning phase, the project would have struggled to identify the key dimensions of data quality or to address the diverse requirements of stakeholders. This planning also sets the stage for iterative refinement, allowing flexibility in response to emerging challenges.

Takeaway 2: Categorizing CI job steps. A critical step during this project was the necessity of categorizing CI job steps. By defining and grouping pipeline tasks into clear

categories, we enhanced the artifact's ability to produce standardized outputs. This categorization simplified data extraction processes and improved consistency, aligning with the project's goal of facilitating big data analytics. Moreover, the categories allowed teams to better understand their CI workflows, leading to a more intuitive implementation process.

Takeaway 3: Unifying CI jobs. Unifying CI jobs through a standardized template was one of the most impactful aspects of this study. Early iterations revealed the inefficiencies and inconsistencies caused by diverse, ad hoc CI implementations across teams. By consolidating these into a unified framework, we not only improved maintainability but also ensured that data generated by the pipelines adhered to the required quality dimensions. However, achieving this unification required balancing standardization with the need for flexibility, leading to the adoption of modular templates.

Takeaway 4: Measuring results. The ability to measure results effectively is pivotal in evaluating the artifact and guiding its iterations. This includes both the technical performance of the pipelines, such as data quality metrics, and qualitative feedback from stakeholders. Regular measurements allow us to identify gaps, such as the need for improved accessibility and timeliness in data collection. Additionally, measuring outcomes ensures that the changes made during each iteration contributed tangible improvements to the CI pipelines and aligned with organizational goals. Measurements can be done either manually or automatically, but we concluded that evaluating requires both automated and manual validation mechanisms. With automated validation ensuring consistency and timeliness, while manual audits help maintain contextual accuracy.

5.4 Formalization of learning

The final stage of ADR process is the formalization of learning where the learnings are condensed into a general solution. The solution finalized solution of the full ADR process is presented in this section. Based on the generalized takeaways from the implementation process an CI improvement process model could be formed; this process model is shown in figure 12.

Figure 12. *CI improvement process model*

This model describes the five phases that are viewed crucial when trying to improve CI pipelines for increasing the quality of the output data. These phases are Planning, Dissection and categorization, unifying of common job template, implementation of template to existing jobs and collection, measurement of results and upkeep of improvements.

Planning is the initial and arguably the most important phase in the CI improvement process, where the outlines for the project are being set. The steps in the planning phase can include:

1. Map the organizational scope of improvements
2. Assess the CI jobs within scope
3. Estimate workload
4. Create tasks and assign responsibilities

The planning phase can start from mapping the scope of the improvement within organization, in smaller companies the scope might be the entire CI infrastructure within the organization, but otherwise you will need to map which organizational branches or components are taken into the process. During this step it's also possible to exclude some jobs or products from the scope. Potential reasons for this would include, for example, legacy pipelines that are no longer in active use, and it would be a waste of resources to improve them. The next step is the assessment of CI jobs within the scope. At least this includes counting the number of individual jobs, but ideally this also includes assessing the current quality of existing jobs. Things to look for include the level of current documentation, uniformity of jobs and formation of code. This assessment helps with the third step, that is the estimation of workload. It's important to get a realistic estimate of the workload to make sure that the organization is actually willing and able to commit enough resources to complete the improvement process as it's not something that can be done halfway through. After the workload has been estimated, it needs to be allocated among team members. From our experiences we could recommend having only a few people responsible for categorization and unifying phases with others providing feedback, but having the actual implementation split among several people.

Dissection and categorization is the second phase in the CI improvement process. This is where the actual work of going through the existing CI jobs starts. The steps include:

1. Iterate through existing CI pipeline scripts
2. Identify distinct parts of the CI jobs

3. Categorize the parts based on their functionality

The ones responsible for creating CI job template should start going through the mapped CI pipelines and identify different functional parts. These parts can be likes of “checkout from version control”, “code compilation” or “static analysis”. The person responsible should thrive towards finding as small components as possible. For example, a “testing” part could be split into parts based on which kind of a test are being run. These parts should then be evaluated and categorized in light of other CI jobs. The goal is to identify which components reoccur in different jobs and which ones are unique to their respective use cases, which helps unifying the parts into common templates

Unifying the results is the third phase where the previously dissected components are combined into new templates fitting more than one job. The steps in this phase can include:

1. Identify different job types
2. Identify common components of each job type
3. Create common templates
4. Document templates and individual stages

It’s very likely that not all jobs can follow a similar pattern as pipelines can have wildly different use cases, which is why it can be useful to identify different types of jobs from the previously categorized jobs and create an own template for each of them instead of trying to fit one for all of them. After this, the next step is to identify the common components for each of the job types. This should only include parts that are actually common between the different jobs, trying to fit every potential component into the template runs with the risk of increasing the complexity too much. After the components are identified they should be combined into the initial template solution which should then be thoroughly documented.

Implementation phase can start after the initial templates are ready. The steps in this phase would include:

1. Trial the implementation
2. Document implementation process
3. Inform organization about the coming changes
4. Roll out the updates for all CI pipelines
5. Continuously update templates and documentation

The changes should first be trialed in a few CI jobs to make sure that the templates are complete. During the first trials the templates should be modified based on feedback. Also, the ones doing the trial should document their experiences, creating guidance about the process. Once trials have been successfully completed the organizational shareholders should be made aware of the coming changes, as any changes to CI infrastructure can potentially cause downtime. Afterwards the actual implementation should start, and the templates should be rolled out for all CI jobs. While this process is ongoing, the ones responsible for the improvement process should continuously update templates and the related documentation based on feedback.

Collection and measurement are the phase that starts after implementation is mostly done. The steps in this phase can include:

1. Gather data from CI pipelines
2. Store and process data
3. Utilize newly created data
4. Evaluate data quality and project results

The CI improvement process, until this point, has only been concerned with enabling the generation of higher quality data, but for it to be useful the data also needs to be collected, stored and processed. The practical implementation of this process is highly dependent on project, but it will likely involve a script collecting the data for some data management or business intelligence application. In our case this script would also conduct preliminary data cleaning to make sure that we don't waste data analyzation resources by handling incorrect or unnecessary data. After this the organization can use the data for their needs including for example data analytics and machine learning. Once the data is available for use it would be useful to analyze the data quality of the data generated and also overall the success of the improvement project. With the results, it helps organizations to decide if similar improvement projects are worth it.

Upkeep is the final phase of CI improvement model, unlike other phases this doesn't have concrete steps or set end date, but instead this phase highlights the importance of maintaining the improved CI jobs. This means making sure that any new CI jobs are created using the templates and that any updates to the older ones are also in line with the template. This phase also includes upkeeping the data collection, storage and analysis and making sure that the data generates value for the organization.

6. DISCUSSION

6.1 Interpretation of the results

This study examined the enhancement of continuous integration (CI) pipelines to improve data quality for data management and analytics. During the application of an iterative Action Design Research approach, a structured CI improvement model was developed and evaluated with the goal of systematically enhancing data generation within CI environments. The study's findings reveal that a structured methodology comprising planning, categorization, unification, implementation, measurement, and upkeep can be utilized by organizations seeking to create a more efficient and reliable data processing framework. In chapter 3 we saw how data utilization can be utilized in Figure 13 presents the effects of successful implementation of this process for the data lifecycle, based on Alaimo & Kallinikos (2024) and illustrated in Figure 5.

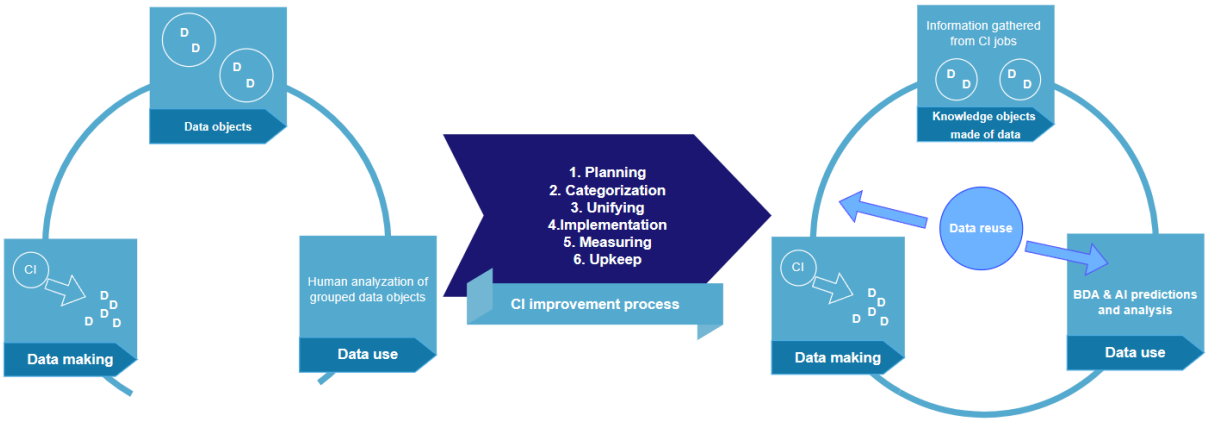

Figure 13. *The effects of successful CI improvement process implementation based on Alaimo & Kallinikos (2024)*

The figure highlights the initial situation encountered in the target organization, as the data generated by CI pipelines was of low quality, so it could only be grouped into simple data objects and then be analyzed by humans, also due to the low data quality the data reuse was difficult, and thus the data didn't have a full lifecycle, but instead ended up being discarded after use. The implementation of the CI improvement model, by integrating structured workflows and standardized practices, helped to close this cycle.

The study's practical focus was on the improvements made to the initial step of data life cycle; data making. The first four steps of this implementation process, described in chapter 5, focused on standardizing CI jobs through a unified template. Prior to this improvement, CI jobs were highly heterogeneous, leading to discrepancies in data output.

By consolidating common CI job structures, the study demonstrated that unifying CI configurations significantly enhances maintainability and ensures a more structured approach to data generation. This standardization also facilitated automation in data collection, reducing the need for manual validation. Step 5 of this improvement process highlighted the need for measuring the quality of resulting data, this study utilized the data quality assessment steps presented by Hassenstein & Vanella (2022) to ensure that the data making step creates usable and valuable data. The improvement process models final stage upkeep, which highlights the fact that CI improvement process doesn't have a clear end but instead should be a continuous process that ties into organizational data management. Drawing on the general principles of data management outlined by Gordon (2007), the following areas can be identified as particularly relevant to maintaining the quality of CI data in future: Firstly, the knowledge objects produced, along with the data they contain, should be systematically stored and managed within the organization to facilitate both usage and reuse, this also means including them inside the organizational data models. Secondly, tasks must be regularly maintained to ensure consistent and high-quality data output. Finally, the tools used for accessing and manipulating CI data must be kept up to date and operational to support reliable data management practices.

A loop in data lifecycle ends to the step of data use. The usage of the CI data generated didn't fit inside this study's practical scope, but this study offers a theoretical overview of some possible technologies. These are discussed in more detail in chapter 3, but for example, supervised classification methods, like Neural Networks as described by Badillo et al. (2020), which use layers of interconnect nodes, could be used to categorize the reasons CI jobs fail. For these technologies to be useful the data needs to maintain high quality. In this study the data quality dimensions by Hassenstein and Vanella (2022) were used to define quality data. From these dimensions accuracy, completeness, and consistency can be considered to be the most crucial for ML and BDA based solutions (Chen et al., 2021; Côrte-Real et al., 2020; Fosso Wamba et al., 2019). These dimensions can be used to evaluate the potential of ML and BDA deployment before practical implementation. In target organization we noticed an improvement in these data quality dimensions, which predicts that this process will improve organizational ability to utilize ML and AI.

In general, this study further confirmed that organizations can achieve measurable benefits by improving CI data quality. Teams that implemented the structured improvement model reported increased efficiency in debugging and troubleshooting. Additionally, the

research emphasized the importance of balancing technical improvements with organizational considerations. Successful adoption of standardized CI practices required active stakeholder engagement and continuous monitoring to ensure sustained improvements. Organizations that prioritized data awareness and transparent data handling practices experienced smoother transitions to improved CI processes.

6.2 Answering research question

This study aimed to answer the primary research question: **How can companies improve their continuous integration pipelines to produce quality data for data management purposes?** To address this question comprehensively, the research was structured around several secondary research questions, each contributing to the overall understanding of CI pipeline optimization. In this section we are going to go through these questions and summarize the generated answer and try to answer the primary question based on them.

The first secondary research question was: ***What defines quality data, and how can we measure data quality?***

Existing literature suggests that data quality is a multidimensional concept rather than a single metric, and that common way to try to measure this, are the so-called data quality dimensions. The most commonly cited dimensions of data quality include accuracy, completeness, consistency, timeliness, and validity. This study confirmed that these dimensions remain highly relevant in the context of CI pipelines, where large volumes of software development data must be managed effectively. We also presented the data quality assessment steps as shown in figure 4, which can be used to integrate the evaluation of these quality dimensions as part of CI improvement process model. These theoretical findings gave us a good understanding of what constitutes as quality data in the first place. Even though data quality is a very studied subject, the existing research doesn’t cover measuring data quality in CI pipelines. During the research process we identified that measuring CI data quality requires both automated and manual validation mechanisms. Automated validation ensures consistency and timeliness, while manual audits help maintain contextual accuracy.

The second question examined the specific data quality traits critical for BDA and ML: ***What are the most important data quality traits for machine learning and big data analytics?***

Prior research highlights that poor data quality can severely impact both BDA and ML model performance (Côrte-Real et al., 2020; Gong et al., 2023). This thesis also studied the most important data quality dimensions for BDA and ML. The most influential dimensions were identified to be accuracy, completeness, and consistency. Accuracy is essential as data must reflect real-world scenarios without errors; incorrect data can propagate flaws in both Big Data analytics and ML models (Chen et al., 2021; Côrte-Real et al., 2020). Completeness is also crucial, as the data should capture all necessary attributes and instances required for robust decision-making based on BDA, and ML model training (Fosso Wamba et al., 2019; Mohammed et al., 2024). Consistency ensures that data conform to standardized formats and rules, maintaining coherence across datasets, which is important factor for BDA (Côrte-Real et al., 2020; Fosso Wamba et al., 2019). These findings influence the focus of the proposed model to especially focus on improving these data quality dimensions.

The third research question focused on practical improvements: ***How can we improve continuous integration pipelines to provide more useful data?***

Prior study by Hilton et al. (2017), found that CI output logs are often extensive and difficult to process. Findings of this study support their claim that improving data extraction mechanisms is crucial and this study found out that one key step in this is to categorize CI job steps. We found out that by grouping pipeline tasks into distinct categories, we were able to enhance the data quality of pipelines outputs. This categorization simplified data extraction processes and improved consistency. We identified that the more categorized each pipeline was, the easier the results were to handle by both humans and machines alike. On top of this CI pipelines can be improved with automated data collection scripts and using them to send the data to management tools for further cleaning and organization.

The final question explored the business impact of these improvements: ***Does undertaking the improvement process provide an organization with tangible benefits that are worth the effort?***

Even though this study didn’t include a thorough evaluation of the CI improvement process and its results, it generated some additional insight on the benefits of improving CI data quality in organizations. Based on qualitative review inside the CI team, the improvement process was worth the effort, and it can enable more accurate analytics, improving project management and resource allocation in the future. It was noted that the completed process has benefits when it comes to efficiency in debugging and trouble-

shooting due to cleaner and more organized log data. Additionally, based on the preliminary analysis the improved data quality should be reflected in reduced training errors and improved prediction accuracy. Additionally creating generic guidelines should allow teams to leverage historical CI data for long-term process optimization. These results should be still further investigated, but they would align with broader discussions in data management research, which emphasize that high-quality data translates to better decision-making and predictive modeling capabilities (Sayyadi, 2024). So even though this process might be a long undertaking, we believe that the benefits will likely outweigh the costs.

By synthesizing these findings, we return to the primary research question: ***How can companies improve their continuous integration pipelines to produce quality data for data management purposes?***

Improving the generation, validation, and application of high-quality data within CI pipelines centers on ensuring consistency, reliability, and seamless integration with data management and analytics frameworks. This study's results show that organizations should be able to improve the quality of their CI generated data, through iterative refinement of CI pipeline workflows, focusing on standardization, automated validation, and structured data storage, while creating a path toward enhanced data usability. The key findings emphasize the importance of both technical and organizational improvements in optimizing CI pipelines for effective data-driven decision-making. This study suggests that one way to conduct this standardization is by utilizing CI improvement process model, which goal is to guide the organizations efforts in improving their CI pipelines. It includes standardizing CI pipeline structures, measuring data quality and upkeep of good CI practices as part of it. As part of this model automated validation mechanisms can be integrated directly into CI workflows, ensuring that real-time detection of anomalies, missing values, and inconsistencies could be handled before storage.

In conclusion, optimizing CI pipelines for high-quality data management requires a holistic approach that integrates structured pipeline standardization, real-time validation, efficient data storage, and advanced analytics. These enhancements not only improve data integrity but also empower organizations to leverage CI-generated data for more accurate decision-making, improved debugging efficiency, and scalable analytics capabilities.

7. CONCLUSION

7.1 Contributions

This thesis has contributed to both theory and practice in understanding how CI pipelines support Big Data Analytics and Machine Learning. On the theoretical side, this research expands on the intersection between CI and data quality by analyzing how CI pipelines can serve as a valuable source of high-quality data for BDA and ML applications. The study presents theory about evaluating data quality and the requirement data quality requirements of ML and BDA. Additionally, the study highlights specific data quality dimensions such as accuracy, completeness, and validity that are particularly critical for leveraging CI data effectively.

In terms of practical contributions, this research provides a generalizable model for improving the quality of data generated from CI pipelines, shown in figure 12. The proposed model offers concrete strategies and best practices for organizations to enhance their CI processes, enabling more reliable and accurate data for analytics and ML applications. Organizations seeking to optimize their software development workflows and data-driven decision-making can use these insights to refine their CI strategies and improve their data infrastructure. Overall, this study offers a starting point for software development organizations looking for ways to improve the quality and utilization of CI generated data.

7.2 Limitations

The main limitation of this study aligns with observations made by Sein et al. (2011) regarding the Action Design Research (ADR) method: it is not a highly controlled research strategy. The iterative, looping development style inherent to ADR means that reproducing this study in its entirety is nearly impossible. The simultaneous stages of the ADR process make documenting the research comprehensively and accurately a significant challenge. This limitation introduces an element of variability that can make it difficult to generalize findings or draw definitive conclusions. The ADR method’s iterative nature also means that the research outcomes evolve continuously throughout the process. While this adaptability is a strength in addressing real-world problems, it complicates efforts to achieve controlled comparisons or establish baseline measurements for evaluating the effectiveness of the proposed solutions. As a result, this study lacks

the level of quantitative analysis that might strengthen the generalizability and robustness of its findings.

Another limitation stems from the influence of the practitioners and their environment on the research process. Challenges specific to the target organization were able to shape the study's direction, which may limit the applicability of the findings to other contexts. While the findings provide valuable insights into the CI improvement process, they are inherently tied to the unique characteristics of the organization under study. The results may not fully address the nuances or challenges of other organizations with different structures, cultures, or technical setups.

Another significant challenge lies in balancing the dual goals of solving a practical problem within a specific organizational context while simultaneously contributing to broader theoretical knowledge. This dual focus can lead to trade-offs in rigor and depth in one domain over the other. For example, while the study successfully generated new insights into improving CI pipelines, it did not fully explore downstream applications of these improvements in machine learning or advanced analytics.

In conclusion, while this study generates insights into the continuous integration improvement process, its findings are best viewed as a foundation for further research.

7.3 Future research

Future research could build on the preliminary understanding generated in this study. The two most interesting avenues are the quantitative evaluation of the model, and the utilization of generated data in the downstream.

The model generated by this study should be tested using a more controlled and reproducible method. This would include applying the process model in different organizations or ideally in multiple different organizational settings, this would help to develop more universally applicable models. The model could also be tested against more generic process models to see if this CI specific process model offers advantages over traditional, more tested, process models. These studies should include objective evaluation of data quality; many data quality dimensions can be measured using different algorithms providing quantitative results.

Another direction for future research would be to evaluate the downstream application of generated data. Even if the model general would be proved to increase generated data quality, the effects of this higher quality data on organization should be studied further. In this study it was assumed that higher data quality will benefit the organization and there was extensive exploration into how these improvements could directly

influence outcomes such as software quality, project timelines, and business decision making. Research could be conducted on the effects of CI data quality on machine learning and big data analytics capabilities and the value generated by them.

7.4 Final Thoughts

This study showed that improving CI pipeline data quality can be a worthwhile endeavor, the results validate the effectiveness of a structured CI improvement model in enhancing data quality. The study demonstrates that categorization, standardization, and automation collectively improve the reliability and usability of CI-generated data. These improvements not only facilitate better analytics but also contribute to long-term efficiencies in CI pipeline management.

Hopefully, this study will inspire further exploration of the subject, as this preliminary study suggests that organizations can gain tangible benefits by improving CI pipelines, but there exists a noticeable lack of research focusing on CI from point of view data, and its utilization.

REFERENCES

Alaimo, C., & Kallinikos, J. (2024). *Data rules: Reinventing the market economy*. The MIT Press.

Ali-ud-din Khan, M., Uddin, M. F., & Gupta, N. (2014). *Seven V's of Big Data understanding Big Data to extract value*. 1–5. https://doi.org/10.1109/ASEEZone1.2014.6820689

Alloghani, M., Al-Jumeily, D., Mustafina, J., Hussain, A., & Aljaaf, A. J. (2020). A Systematic Review on Supervised and Unsupervised Machine Learning Algorithms for Data Science. In M. W. Berry, A. Mohamed, & B. W. Yap (Eds.), *Supervised and Unsupervised Learning for Data Science* (pp. 3–21). Springer International Publishing. https://doi.org/10.1007/978-3-030-22475-2_1

Badillo, S., Banfai, B., Birzele, F., Davydov, I. I., Hutchinson, L., Kam-Thong, T., Siebourg-Polster, J., Steiert, B., & Zhang, J. D. (2020). An Introduction to Machine Learning. *Clinical Pharmacology & Therapeutics*, *107*(4), 871–885. https://doi.org/10.1002/cpt.1796

Bansal, M., Chana, I., & Clarke, S. (2021). A Survey on IoT Big Data: Current Status, 13 V's Challenges, and Future Directions. *ACM Computing Surveys*, *53*(6), 1–59. https://doi.org/10.1145/3419634

Batini, C., & Scannapieco, M. (2016). *Data and Information Quality: Dimensions, Principles and Techniques* (1st ed.). Springer Nature. https://doi.org/10.1007/978-3-319-24106-7

Berman, J. J. (2013). *Principles of big data: Preparing, sharing, and analyzing complex information* (1st edition). Elsevier.

Cai, L., & Zhu, Y. (2015). The Challenges of Data Quality and Data Quality Assessment in the Big Data Era. *Data Science Journal*, *14*, 2-. https://doi.org/10.5334/dsj-2015-002

Chen, H., Chen, J., & Ding, J. (2021). Data Evaluation and Enhancement for Quality Improvement of Machine Learning. *IEEE Transactions on Reliability*, *70*(2), 831–847. https://doi.org/10.1109/TR.2021.3070863

Chopra, D. (with Khurana, R.). (2023). *Introduction to Machine Learning with Python* (First edition.). Bentham Science Publishers Pte. Ltd.

Collatto, D. C., Dresch, A., Lacerda, D. P., & Bentz, I. G. (2018). Is Action Design Research Indeed Necessary? Analysis and Synergies Between Action Research and Design Science Research. *Systemic Practice and Action Research*, *31*(3), 239–267. https://doi.org/10.1007/s11213-017-9424-9

Côrte-Real, N., Ruivo, P., & Oliveira, T. (2020). Leveraging internet of things and big data analytics initiatives in European and American firms: Is data quality a way to extract business value? *Information & Management*, *57*(1), 103141-. https://doi.org/10.1016/j.im.2019.01.003

Dahiya, R., Le, S., Ring, J. K., & Watson, K. (2022). Big data analytics and competitive advantage: The strategic role of firm-specific knowledge. *Journal of Strategy and Management*, *15*(2), 175–193. https://doi.org/10.1108/JSMA-08-2020-0203

Dai, H.-N., Wong, R., Wang, H., Zheng, Z., & Vasilakos, A. (2019). Big Data Analytics for Large-scale Wireless Networks: Challenges and Opportunities. *ACM Computing Surveys*, *52*(5), 1–36. https://doi.org/10.1145/3337065

Debbiche, A., Dienér, M., & Berntsson Svensson, R. (2014). *Challenges When Adopting Continuous Integration: A Case Study*. *8892*, 17–32. https://doi.org/10.1007/978-3-319-13835-0_2

Dhudasia, M. B., Grundmeier, R. W., & Mukhopadhyay, S. (2023). Essentials of data management: An overview. *Pediatric Research*, *93*(1), 2–3. https://doi.org/10.1038/s41390-021-01389-7

Dickerson, J. E. (2022). Data, information, knowledge, wisdom and understanding. *Anaesthesia and Intensive Care Medicine*, *23*(11), 737–739. https://doi.org/10.1016/j.mpaic.2022.08.012

Duvall, P. M. (with Glover, A., & Matyas, S.). (2007). *Continuous integration: Improving software quality and reducing risk* (1st edition). Addison Wesley.

Elazhary, O., Werner, C., Li, Z. S., Lowlind, D., Ernst, N. A., & Storey, M.-A. (2022). Uncovering the Benefits and Challenges of Continuous Integration Practices. *IEEE Transactions on Software Engineering*, *48*(7), 2570–2583. IEEE Transactions on Software Engineering. https://doi.org/10.1109/TSE.2021.3064953

Fitzgerald, B., & Stol, K.-J. (2017). Continuous software engineering: A roadmap and agenda. *The Journal of Systems and Software*, *123*, 176–189. https://doi.org/10.1016/j.jss.2015.06.063

Fosso Wamba, S., Akter, S., Trinchera, L., & De Bourmont, M. (2019). Turning information quality into firm performance in the big data economy. *Management Decision*, *57*(8), 1756–1783. https://doi.org/10.1108/MD-04-2018-0394

Gandomi, A., & Haider, M. (2015). Beyond the hype: Big data concepts, methods, and analytics. *International Journal of Information Management*, *35*(2), 137–144. https://doi.org/10.1016/j.ijinfomgt.2014.10.007

Garg, A., Kumar Kaliyar, R., & Goswami, A. (2022). *PDRSD-A systematic review on plan-driven SDLC models for software development*. *1*, 739–744. https://doi.org/10.1109/ICACCS54159.2022.9785261

Gong, Y., Liu, G., Xue, Y., Li, R., & Meng, L. (2023). A survey on dataset quality in machine learning. *Information and Software Technology*, *162*, 107268-. https://doi.org/10.1016/j.infsof.2023.107268

Gordon, K. (with British Computer Society). (2007). *Principles of data management: Facilitating information sharing* (1st edition). British Computer Society.

Hassenstein, M. J., & Vanella, P. (2022). Data Quality—Concepts and Problems. *Encyclopedia*, *2*(1), 498–510. https://doi.org/10.3390/encyclopedia2010032

Haug, A. (2021). Understanding the differences across data quality classifications: A literature review and guidelines for future research. *Industrial Management & Data Systems*, *121*(12), 2651–2671. https://doi.org/10.1108/IMDS-12-2020-0756

Hilton, M., Nelson, N., Tunnell, T., Marinov, D., & Dig, D. (2017). Trade-offs in continuous integration: Assurance, security, and flexibility. *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*, 197–207. https://doi.org/10.1145/3106237.3106270

Hilton, M., Tunnell, T., Huang, K., Marinov, D., & Dig, D. (2016). *Usage, costs, and benefits of continuous integration in open-source projects*. 426–437. https://doi.org/10.1145/2970276.2970358

Hoppe, A., Seising, R., Nürnberger, A., & Wenzel, C. (2011, July 18). *Wisdom—The blurry top of human cognition in the DIKW-model?* https://doi.org/10.2991/eusflat.2011.91

Istifarulah, M. H. R., & Tiaharyadini, R. (2023). DevOps, Continuous Integration and Continuous Deployment Methods for Software Deployment Automation. *JISA (Jurnal Informatika Dan Sains)*, *6*(2), 116–123. https://doi.org/10.31326/jisa.v6i2.1751

Jain, N. (2011). Software Development Life Cycle: A Detailed Study. *International Journal of Advanced Research in Computer Science*, *2*(3).

Kazim, A. (2017). A Study of Software Development Life Cycle Process Models. *International Journal of Advanced Research in Computer Science*, *8*(1). https://www.proquest.com/docview/1901446145/abstract/B80732AC24604E5APQ/1

Khatri, V., & Brown, C. V. (2010). Designing data governance. In *Communications of the ACM* (Vol. 53, Issue 1, pp. 148–152). https://doi.org/10.1145/1629175.1629210

Kolosova, T. (with Berestizhevsky, S.). (2021). *Supervised machine learning: Optimization framework and applications with SAS and R* (1st ed.). CRC Press.

Konda, V. R., & Tsitsiklis, J. N. (2003). OnActor-Critic Algorithms. *SIAM Journal on Control and Optimization*, *42*(4), 24. https://doi.org/10.1137/S0363012901385691

Krichen, M. (2023). *Deep Reinforcement Learning*. 1–7. https://doi.org/10.1109/ICCCNT56998.2023.10306453

Lemahieu, W., vanden Broucke, S., & Baesens, B. (2018). Organizational Aspects of Data Management. In *Principles of Database Management* (pp. 79–90). https://doi.org/10.1017/9781316888773.006

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., & Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature (London)*, *518*(7540), 529–533. https://doi.org/10.1038/nature14236

Mohammed, S., Budach, L., Feuerpfeil, M., Ihde, N., Nathansen, A., Noack, N., Patzlaff, H., Naumann, F., & Harmouch, H. (2024). *The Effects of Data Quality on Machine Learning Performance* (arXiv:2207.14529). arXiv. https://doi.org/10.48550/arXiv.2207.14529

Naeem, S., Ali, A., Anam, S., & Ahmed, M. M. (2023). An Unsupervised Machine Learning Algorithms: Comprehensive Review. *International Journal of Computing and Digital System (Jāmiʿat al-Baḥrayn. Markaz al-Nashr al-ʿIlmī)*, *13*(1), 911–921. https://doi.org/10.12785/ijcds/130172

Rana, R., & Staron, M. (2015). *Machine learning approach for quality assessment and prediction in large software organizations*. 1098–1101. https://doi.org/10.1109/ICSESS.2015.7339243

Rossi, C., Shibley, E., Su, S., Beck, K., Savor, T., & Stumm, M. (2016). Continuous deployment of mobile software at facebook (showcase). *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, 12–23. https://doi.org/10.1145/2950290.2994157

Rowley, J. (2007). The wisdom hierarchy: Representations of the DIKW hierarchy. *Journal of Information Science*, *33*(2), 163–180. https://doi.org/10.1177/0165551506070706

Rüegg, J., Gries, C., Bond-Lamberty, B., Bowen, G. J., Felzer, B. S., McIntyre, N. E., Soranno, P. A., Vanderbilt, K. L., & Weathers, K. C. (2014). Completing the data life cycle: Using information management in macrosystems ecology research. *Frontiers in Ecology and the Environment*, *12*(1), 24–30. https://doi.org/10.1890/120375

Saunders, M. N. K. (with Lewis, P., & Thornhill, A.). (2019). *Research methods for business students* (Eighth edition.). Pearson Education.

Sayyadi, M. (2024). How to improve data quality to empower business decision-making process and business strategy agility in the AI age. *Business Information Review*, *41*(3), 124–129. https://doi.org/10.1177/02663821241264705

Sein, M. K., Henfridsson, O., Purao, S., Rossi, M., & Lindgren, R. (2011). Action Design Research. *MIS Quarterly*, *35*(1), 37–56. https://doi.org/10.2307/23043488

Shahin, M., Ali Babar, M., & Liming Zhu. (2017). Continuous Integration, Delivery and Deployment: A Systematic Review on Approaches, Tools, Challenges and Practices. *IEEE Access*, *5*, 3909–3943. https://doi.org/10.1109/ACCESS.2017.2685629

Shetty, M. Y., B S, P., Aditya, & Gadiyar, H. M. T. (2023). Software Development Life Cycle (SDLC) in Software Engineering – A Brief Review. *Journal of Computer Science and System Software*, 5–9. https://doi.org/10.48001/jocsss.2023.115-9

Soares, E., Sizilio, G., Santos, J., Da Costa, D. A., & Kulesza, U. (2022). The effects of continuous integration on software development: A systematic literature review. *Empirical Software Engineering*, *27*(3), 78. https://doi.org/10.1007/s10664-021-10114-1

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement Learning: An Introduction* (2nd ed.). MIT Press.

Sutton, R. S., McAllester, D., Singh, S., & Mansour, Y. (1999). Policy Gradient Methods for Reinforcement Learning with Function Approximation. *Advances in Neural Information Processing Systems*, *12*. https://proceedings.neurips.cc/paper_files/paper/1999/hash/464d828b85b0bed98e80ade0a5c43b0f-Abstract.html

Vasilescu, B., Yu, Y., Wang, H., Devanbu, P., & Filkov, V. (2015). *Quality and productivity outcomes relating to continuous integration in GitHub*. 805–816. https://doi.org/10.1145/2786805.2786850

Virmani, M. (2015). *Understanding DevOps & bridging the gap from continuous integration to continuous delivery*. 78–82. https://doi.org/10.1109/INTECH.2015.7173368

Wook, M., Hasbullah, N. A., Zainudin, N. M., Jabar, Z. Z. A., Ramli, S., Razali, N. A. M., & Yusop, N. M. M. (2021). Exploring big data traits and data quality dimensions for big data analytics application using partial least squares structural equation modelling. *Journal of Big Data*, *8*(1), 1–15. https://doi.org/10.1186/s40537-021-00439-5

Yangyang Zhao, Serebrenik, A., Yuming Zhou, Filkov, V., & Vasilescu, B. (2017). *The impact of continuous integration on other software development practices: A large-scale empirical study*. 60–71. https://doi.org/10.1109/ASE.2017.8115619

Yiran, W., Tongyang, Z., & Yidong, G. (2018). *Design and implementation of continuous integration scheme based on Jenkins and Ansible*. 245–249. https://doi.org/10.1109/ICAIBD.2018.8396203