Policy Research Journal
ISSN (E): 3006-7030 ISSN (P) : 3006-7022 Volume 3, Issue 3, 2025

OPTIMIZING CI/CD THROUGH AGILE-DEVOPS COLLABORATION

Hadia Hafeez[*1], Muhammad Arif[2], Hamza Zahid[3], Muhammad Bilal Qureshi[4]

[*1,2,3,4]*Department of Computer Science & IT, Superior University, 10 KM Lahore- Sargodha Rd, Sargodha, Punjab 40100, Pakistan.*

[1]hadiamehar09@gmail.com, [2]md.arif@superior.edu.pk, [3]hamzashaikhcan@gmail.com, [4]bilalshah1728@gmail.com

DOI: https://doi.org/10.5281/zenodo.15010250

Keywords
Agile-DevOps integration, CI/CD pipeline, software development, automation.

Article History
Received on 05 February 2025
Accepted on 05 March 2025
Published on 12 March 2025

Corresponding Author: *

Abstract
In contemporary software development, Continuous Integration and Continuous Deployment (CI/CD) play a crucial role in ensuring fast and reliable software delivery. The integration of Agile methodologies with DevOps principles enhances CI/CD performance by promoting teamwork, automation, and iterative enhancements. This study explores the effects of Agile-DevOps integration on CI/CD efficiency, evaluates key performance metrics, identifies challenges along with possible solutions, and offers practical recommendations for development teams. The results indicate that Agile-DevOps collaboration significantly improves CI/CD pipeline efficiency by minimizing deployment time, enhancing software quality, and boosting team productivity. Metrics such as deployment frequency, lead time, and defect rates show notable advancements. Despite these benefits, challenges like cultural barriers, tool compatibility issues, and security risks remain. To address these concerns, a theoretical framework for Agile-DevOps integration is proposed to optimize CI/CD workflows. The study concludes that organizations leveraging Agile-DevOps collaboration can enhance software delivery by implementing automation, continuous feedback mechanisms, and effective change management practices.

INTRODUCTION

Over the past two decades, there has been increasing interest in the concept of "smart" innovations, spanning diverse areas such as Smart Aging and Smart Z-wave home monitoring, as outlined by Alter (2019) in his detailed review of smart technologies. This special issue adopted a more holistic perspective, focusing on 'smart working, living and organizing' by integrating various smart initiatives to achieve broader objectives. Some studies have specifically examined the elements of smart working and smart organizing in the context of the DevOps approach to software development (Hemon et al., 2020). Version One (2018) estimated that over 90% of companies now adopt agile methodologies for software development. These approaches aim to strengthen collaboration between customers and developers, ensuring the software meets market needs while enabling faster release cycles. However, despite the accelerated development process, a bottleneck has arisen due to misalignment between the Development (Dev) and Operations (Ops) teams. The Operations team, which manages software releases, often operates independently of the development process, resulting in significant delays in delivering software to customers. To overcome this challenge, Debois introduced the concept of DevOps, advocating for a closer integration of the

Dev and Ops functions (Dhal et al., 2022 ; Brunnert et al., 2015).

Companies have increasingly adopted the DevOps approach. Several authors have examined the challenges involved in transitioning from traditional waterfall methods to agile practices, followed by continuous integration and continuous deployment (Holmstrom Olsson et al., 2012). There is considerable variation in how extensively companies implement continuous integration (CI) and continuous deployment (CD) practices (Dixit & Jangid, 2024; Stahl & Bosch, 2014). These activities are fundamental to DevOps. Since DevOps emerged from practical application, there has been relatively little focus on its conceptual or theoretical foundations. Numerous definitions of DevOps exist, with some aspects being well-established and familiar, while others are more recent developments (Jagdish, 2024; Huttermann, 2012). The primary innovation of DevOps lies in merging two traditionally distinct organizational groups: Development and Operations. Historically, the separation between these functions led to isolated silos, resulting in significant inefficiencies in software delivery. By integrating Development and Operations, DevOps has introduced new collaboration dynamics among roles such as Developers, Architects, Scrum Masters, Product Owners, Release Engineers and Testers. These collaborative patterns diverge from those traditionally observed in Scrum practices, such as Daily Standups, Backlog Grooming and Sprint Reviews or Retrospectives. This analysis focused on five essential roles within a DevOps framework—Release Manager, Architect, Product Owner, Department/Project Manager and Production Engineer— emphasizing their key interactions with other DevOps roles. It also identifies the skills required for each role, distinguishing between 'hard' and 'soft' skills (Sachin & Jagdish, 2024; Gallivan et al., 2004; Robles, 2012; Wong et al., 2006). As technologies and methodologies continue to evolve, there is a growing expectation for employees to regularly enhance and expand their skill sets (Goles et al. 2009).

The DevOps methodology emphasized continuous and effective collaboration between Development and Operations teams to achieve seamless integration and faster software delivery (Fitzgerald and Stol, 2017). It promotes the creation of crossfunctional teams where members are expected to anticipate and understand one another's responsibilities. For instance, developers must consider the real-world production environments where their code will be deployed, while Operations teams need to align their workflows with the methods developers use to create, test and package code for release. A crucial component of DevOps is the increased use of test automation (Lwakatare et al., 2015), which helps streamline and optimize the entire deployment process. As Wettinger et al. (2014) highlighted, automation plays a vital role in fostering efficient collaboration and ensuring tight integration between development and operations. Humble and Molesky (2011) outlined four core DevOps values: Culture, Automation, Measurement, and Sharing. Similarly, Kim et al. (2016), in the DevOps Handbook, emphasized the significance of culture, advocating for "a high-trust culture where departments collaborate effectively, work is transparently prioritized, and systems have enough flexibility to ensure high-priority tasks are completed swiftly."

The growing adoption of continuous integration, continuous delivery, and DevOps practices highlights the need for a close connection between software development, its physical implementation, and the supporting infrastructure. This is reflected in the idea of treating infrastructure as code. In the past, operations engineers manually configured hardware and software systems. However, with the growing scale and complexity of modern infrastructures, there has been a growing shift toward automating the scripts used for system configuration and management. In today's large-scale server farms, often comprising thousands of machines, challenges such as configuration drift or unauthorized changes can result in significant disruptions and expensive downtime (Morris, 2020).

1.1. Being smarter with DevOps

Alter (2019) defines smartness in socio-technical systems as the capacity of a purposefully designed entity to perform and control functions that yield directly observable results for people through automated information processing, interpretation, and learning. Applying this definition to DevOps

reveals that it was intentionally created to bridge the gap between Development and Operations, thereby fulfilling the purposefully designed aspect. DevOps seeks to control the interaction between these traditionally separate functions, producing measurable outcomes assessed by cross-functional teams via tools like sprint burndown charts. Automation plays a critical role, with continuous integration builds triggered by code commits and automated testing conducted with specialized tools. Furthermore, the display of information, such as indicators for build status, enhances visibility in DevOps environments. Lastly, as smartness is context-dependent, DevOps teams leverage context-specific information to ensure effective communication and collaboration between Development and Operations.

A SWOT analysis, as illustrated in Figure 1, is a widely used framework for assessing both the internal and external environments of an organization to support decision-making. This tool utilizes a diagnostic approach to pinpoint critical factors that contribute to the success or failure of a strategy, plan, or product. Additionally, conducting a SWOT analysis can help identify potential weaknesses within a business that may hinder progress or provide competitors with an advantage if not addressed (Lee et al., 2021). Commonly applied in strategic planning, SWOT categorizes influential factors into four distinct groups: strengths, weaknesses, opportunities, and threats. The strengths and weaknesses focus on evaluating an organization's internal environment, whereas opportunities and threats analyze the external environment of the system. These internal and external variables, which represent key influences within and outside the system, are among the most crucial considerations. Consequently, the SWOT analysis serves as a diagnostic tool to determine the essential elements that impact the effectiveness of a strategy or system (Elavarasan et al., 2020).

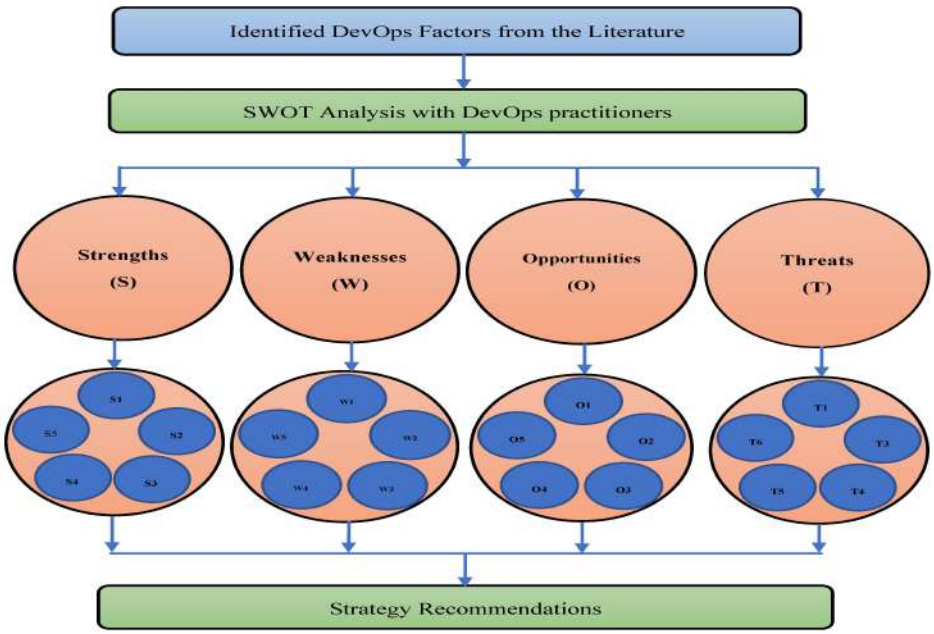

Fig.1. The SWOT Structure

1.2. Roles, skills and competencies of IT jobs in transition to DevOps

1.2.1. Roles

There are five key roles of interest:

1) Product Owner
2) Manager
3) Architect
4) Production Engineer, who conducts testing as part of the Operations team (notably, developers also perform testing in a DevOps environment)
5) Release Manager.

The **Product Owner** (PO) role is crucial in an agile environment using Scrum. POs serve a dual purpose, acting as advocates for client needs while also fulfilling an operational function that connects

business objectives to project management. They are tasked with maximizing the value generated by development teams. To be effective, POs require autonomy, and their decisions should be honored by all stakeholders involved (Coyle et al., 2015). There is debate over whether Product Owners (POs) can function as project managers (PMs). While POs manage the Product Backlog, the Scrum guide emphasizes that Scrum Teams are self-organizing and cross-functional, avoiding external direction or traditional PM roles. Consequently, POs focus more on business ownership than conventional project management (Schwaber and Sutherland 2011). While Product Owners (POs) are theoretically managers focused on business ownership rather than traditional project management, they may also perform PM duties, with some studies merging the PO and Scrum Master roles to define their competencies (Oomen et al. 2017).

The role of **Manager** (Department & Project Manager) is crucial and has various definitions in the literature. We define a manager as an individual responsible for managing both tangible and intangible resources to achieve a specific goal. Managers can act as project leaders, team leaders, or both, often due to hierarchical structures that combine these roles. Other roles, like Scrum Masters, may not always be present or are sometimes fulfilled by individuals in more traditional positions, such as developers or PMs. Team Managers encompass various professions, including development or operational team leaders and qualification-integration managers. Project Managers oversee an IT project from inception to delivery, aiming to achieve optimal results that meet customer requirements for quality, performance, cost, time and security (Coyle et al., 2015).

The role of the **architect** (AR) is evolving, with architects often developing a range of skills, whether they focus on technical, software-application, or functional aspects. On the Operations side, **Production Engineers** (PEs) or production integrators and testers are tasked with overseeing production, operations, incident monitoring, and user support. PEs contribute to creating architectural documentation and application development while providing expertise and assistance in resolving incidents. Additionally, the Release Manager (RM) plays a crucial role in ensuring project success by managing deployment processes, tracking different versions, and coordinating between development, testing, and deployment teams. Within the organization, the RM's role aligns with that of a PM Implementation, which is vital, although its scope is sometimes questioned within the DevOps framework (Hemon et al., 2020).

1.2.2. Skills and competencies

Gallivan et al. (2004) emphasized the extensive research history on IT skills, noting that soft, non-technical skills can sometimes surpass technical skills in importance. Wong et al. (2006) further highlighted the value of attributes like adaptability, flexibility, motivation and strong communication. Non-technical skills include interpersonal abilities, leadership, organization, independence, motivation and creativity. Despite their significance, research reveals that these skills are often underrepresented in recruitment advertisements, which tend to prioritize easily measurable "hard skills." This focus has perpetuated a "recruitment gap" identified in earlier studies. Such neglect is particularly problematic, as some soft skills are crucial for the success of agile software teams (Vivian et al., 2015).

Wiedemann and Schulz (2017) examined critical capabilities of DevOps teams that could offer a competitive edge, highlighting seven key areas such as Change Readiness, Decision-Making and Collaboration. However, their work lacked a clear distinction between capabilities and skills, leaving the specific skills driving competitive advantage undefined. While they proposed strategies to improve collaboration within DevOps teams, the depth or extent of this collaboration was not thoroughly explored. Similarly, Wiedemann (2017) investigated emerging forms of collaboration but did not delve into the level of collaboration within DevOps teams.

1.2.3. Collaboration pattern

As organizations shift to DevOps, collaboration patterns and skill sets will evolve, fostering teamwork and cooperative behaviors essential for effective collaboration. 10 Developers, while specialized, will increasingly rely on diverse skill sets and become

more versatile to meet the demands of faster development cycles. This transition will promote direct interaction among team members, enhancing understanding of each other's roles beyond manager-led communication. However, increased automation may lead to fewer distinct roles within DevOps teams (Nerkar and Paruchuri 2005).

Knowledge sharing and coordination are expected to change with the adoption of DevOps (Humble and Molesky 2011). DevOps emphasizes sharing practices, establishing common ground, and bridging cultural differences. While agile teams interact more frequently than those in traditional plan-driven approaches, these interactions are primarily among development roles, which share similar cultural values such as speed, creativity, and innovation. Agile practices serve as coordination mechanisms that promote knowledge sharing among team members (Strode 2016). Nevertheless, while teamwork is vital in agile settings, individualism and competition can still exist. It can be hypothesized that development and operations teams often work separately with infrequent coordination, leading to a lack of shared mental models. Conversely, in a DevOps environment, even if some tasks are performed independently, mental models and awareness of respective constraints are more commonly shared during collaborative activities. This aligned with Bruns' (2013) concept of "working alone together" in R&D teams, suggesting that even when separated and team member's work collaboratively. Additionally, the frequency of joint activities with operations teams significantly increases in a DevOps framework.

This study aimed to investigate the synergistic integration of Agile and DevOps methodologies, focusing on how these frameworks can be effectively combined to optimize Continuous Integration and Continuous Deployment (CI/CD) pipelines. By merging the iterative flexibility of Agile with the automation and collaboration principles of DevOps, this research seeks to enhance both the efficiency and quality of software delivery processes. Additionally, through a mixed-method research approach, the study aimed to provide empirical evidence that highlights the practical benefits of this integration, particularly in terms of improved deployment speed, stronger team collaboration and increased customer satisfaction. Lastly, the research aspired to bridge the gap between academic theory and industry practices, offering actionable insights and guidelines for technology companies. By delivering practical strategies for enhancing CI/CD processes through the Agile-DevOps synergy, the study hoped to support technology companies in achieving more streamlined and effective software deployment.

2. LITERATURE REVIEW

Leite et al. (2019) presented a talk titled "10+ Deploys per Day: Dev and Ops Cooperation at Flickr," showcasing how collaboration between development (Dev) and operations (Ops) teams could enable more agile and scalable software development. Their innovative approach involved closely integrating Dev and Ops to achieve multiple software deployments—often exceeding ten releases per day—while maintaining safety and efficiency. This concept marked a significant shift in software development practices and its ongoing evolution. They later coined the term DevOps, short for Development and Operations and organized the first DevOps Day event. Despite being a central topic of discussion for over a decade, the DevOps movement still lacks a singular, universally accepted definition.

According to Wiedemann et al. (2019), the absence of a singular, standardized definition for DevOps may be intentional, allowing each team to adopt a version that best aligns with its specific needs. Nonetheless, various authors have proposed definitions, including Leite et al. (2019) defined DevOps as "a collaborative and multidisciplinary effort within an organization aimed at automating the continuous delivery of new software versions while maintaining their accuracy and reliability." Other perspectives view DevOps as a combination of values, principles, methodologies, practices and tools. Further widely recognized definitions are also available.

Effective project execution depends heavily on the management approach used. While the waterfall model follows a linear, event-based sequence, the iterative model develops software in stages, refining until it meets desired outcomes (Larman and Basili, 2003). DevOps bridged development and operations, fostering collaboration across the software lifecycle to

ensure agile, stable and frequent releases. More than a technical strategy, DevOps addressed organizational and human challenges, promoting teamwork and efficient delivery. Rajapakse et al. (2022) highlighted DevOps as a widely adopted approach, known for accelerating deployment rates.

Continuous Integration involves developers frequently merging their tested code into the main project, typically once a day. Continuous Delivery ensures the software is 13 always in a deployable state, allowing releases to be made at any time, with each change generating a release candidate that can be evaluated. In this approach, a team member with the necessary authority decides when and which candidate to release manually. Continuous Deployment, on the other hand, automatically releases every change to production without requiring team intervention (Stahl et al., 2017).

According to Lwakatare et al. (2016), DevOps represented an evolution of Agile development, shaped gradually through practical implementation experiences. While Agile primarily focuses on the software development process itself, DevOps extends this by involving the development team in both the deployment and operational phases of the software, even while still in development. This illustrated how DevOps processes are integrated within Agile practices.

As noted in (Faustino et al., 2020), DevOps culture can support incident management for deliverable products, enabling continuous monitoring of Agile development teams by integrating deployment with operational monitoring through the DevOps framework. This integration enhanced stability within the Agile cycle; while Agile emphasized productivity in deliverables from a technical ICT perspective, DevOps focuses on assessing the effectiveness of outputs throughout development cycles. This key integration is expected to contribute to qualitative advancements in Software Engineering.

From study discussion, it is evident that Agile methodologies provide the foundational structure for DevOps (Hemon et al., 2020). Combining these methodologies enhances the intelligence of the information system generated within work cycles, involving a variety of roles such as Developers, System Architects, Product Owners, Release Engineers, and Testers. This approach boosts collaboration by incorporating professionals beyond traditional roles—such as those seen in SCRUM within Agile— thereby fostering cross-functional teams within the DevOps framework (Dornenburg, 2018).

When development teams using Agile methodologies collaborate closely with operations teams as promoted by DevOps, the software release process is typically accelerated, with studies (Cespedes et al., 2020) showing gains not only in speed but also in software quality, specifically in reliability and maintainability. Consequently, the final product aligned with key project objectives across development (Agile) and 14 deployment and testing (DevOps) stages. However, while DevOps bridged the gap between Developers and Operators, it lacks a straightforward roadmap or standardized approach for implementation within organizations (Nybom et al., 2016), leaving companies responsible for setting their standards and metrics. This reliance on organizational and team maturity can complicate the definition of specific integration processes within a DevOps framework.

The joint adoption of Agile and DevOps allowed organizations to manage the growing complexity of customer requirements and requests more effectively. It promoted a collaborative and Agile framework that supersedes traditional waterfall models and the isolation of development and operations teams. This study investigated the advantages of integrating both methodologies through a qualitative approach that includes twelve case studies from international software engineering firms. Thematic analysis is used to identify the benefits of this combined adoption. The findings reveal twelve key advantages, including process automation, enhanced communication between teams and reduced time to market due to integrated processes and shorter software delivery cycles. Although Agile and DevOps target different goals and challenges, when effectively combined and aligned, they can provide significant benefits to organizations. This study's novelty lies in systematically outlining these benefits from various perspectives within the software engineering business landscape (Almeida et al., 2022).

Galup et al. (2020) described that implementation of DevOps faced several challenges and concerns that can hinder its adoption, such as resistance to change,

organizational vision and legacy systems. Issues like the misuse of the term, unclear guidelines and the absence of a definitive definition contribute to the confusion surrounding the application of DevOps principles. These principles suggested that development and operations teams previously operated independently with minimal understanding of each other's roles; however, the extent of this knowledge gap is often not as significant as DevOps assumes. While better collaboration between teams can enhance the overall software development process, it does not imply that DevOps teams lacked cooperation in the past. Additionally, a notable concern is that the rate of DevOps adoption remains relatively low.

The collaborative aspects of Agile and DevOps, as highlighted in the research by Fernandez-Diego et al. (2020), played a crucial role in improving software quality and speeding up development cycles. This combination harnesses Agile's flexibility and customer-centric approach alongside DevOps' focus on rapid delivery, thereby streamlining the development process. As a result, there is a noticeable decrease in the time to market for software products. The literature suggests a direct link between this integration and the essential principles of effective Continuous Integration/Continuous Delivery (CI/CD), including rapid and reliable testing and deployment, which are fundamental to DevOps and reinforced by Agile methodologies.

The literature reviewed highlighted that the integration of Agile and DevOps offers numerous advantages for the success of Continuous Integration/Continuous Delivery (CI/CD) practices. This synergy not only improves technical processes but also positively impacts the cultural and operational dynamics of software development teams. However, it is important to note the necessity for ongoing adaptation and evolution of this integration model to keep up with technological changes and shifting market needs. Future research could benefit from a more in-depth analysis of empirical data related to CI/CD outcomes following the integration of Agile and DevOps. Furthermore, longitudinal studies (Theurich et al., 2023) examining the long-term effects of this integration on organizational efficiency and software quality would provide valuable insights into this evolving area.

3. METHODOLOGY

3.1. Research Objective and Research Questions

The primary research objective was to understand how teams collaborate effectively and what skills are essential for optimizing Continuous Integration/Continuous Delivery (CI/CD) through Agile DevOps collaboration.

This objective was broken down into the following research questions:

- **RQ1**: What skills are necessary for effective collaboration in teams working within an Agile DevOps environment?
- **RQ2**: How do the skills and collaboration patterns differ among teams operating at various levels of CI/CD automation?

The first question focuses on analyzing collaboration and skill requirements at an individual level, while the second explores team-level dynamics in different automation contexts.

3.2. Case Study Approach

This study utilized a case study methodology, incorporating interviews, observations, and document analysis to examine Agile DevOps practices and the optimization of CI/CD processes. A case study is an empirical research approach that explores a current phenomenon within its real-world context, especially when the distinction between the phenomenon and its context is ambiguous (Yin, 1994).

3.3. Study Context

The research was conducted within a large European IT services firm employing approximately 15,000 IT staff, known for its structured hierarchy and a bureaucratic culture that presents challenges to agility (Strode et al., 2009). The company's Information Systems Division consists of development and operations functions, which have adopted Agile methodologies for over 15 years and have integrated DevOps for eight years, positioning the company as an early adopter of the DevOps model.

3.4. Teams and Job Roles

The study examined 12 application development teams characterized by the use of Agile engineering practices and varying levels of CI/CD automation. Analyzed five job roles within these teams to understand their collaboration patterns and skill requirements (Runeson and Host, 2008). Differences in infrastructure and test automation methods were noted across the teams. The transition to DevOps and increased automation are hypothesized to influence the necessary human skills and collaboration patterns within teams.

3.5. Data Collection and Analysis

Data collection spanned 12 months, involving 59 individual, in-depth interviews lasting approximately 90 minutes each. These were conducted face-to-face, recorded, transcribed and coded. Data analysis followed Braun and Clarke's (2006) six-step thematic analysis process to identify significant themes. The study emphasized the "keyness" rule, which prioritized the relevance of a theme to the research question over frequency. Thematic prevalence was assessed both at the data and individual levels. Reliability was ensured through Lincoln and Guba's (1985) trustworthiness criteria and findings were validated by sharing interpretations with participants.

3.6. Inductive Research Study Phase

The inductive research comprised two phases:

Phase 1:

A literature review helped define criteria for selecting 12 sample teams.

Observations and interviews were conducted over 15 days with four teams to align practical observations with literature-based criteria, such as team size, level of automation and outsourcing extent. Interviews with 12 strategists from Information Systems and HR departments further supported sample validation.

Phase 2:

1. **Team Size:**

Teams were categorized as small (≤14 members) or large (>15 members).

2. **Outsourcing Policy:**

Projects were assessed based on outsourcing nature, comparing fixed-price contract activities against projects without outsourcing or those using technical assistance contracts.

3. **Automation Level:**

Automation levels in development and operations workflows were analyzed, referencing "infrastructure as code".

3.7. Automation Level

- Automation Level 1 (Agile - Aut.1): Represented the initial stage of automation compared to the traditional V-model. Agile enables more frequent releases but does not achieve full DevOps integration. Development and Operations teams remain siloed with minimal collaboration and no automated release processes.
- Automation Level 2 (Continuous Integration - Aut.2): Development and Operations synchronize testing activities, including unit and non-regression tests, aligning with code development and maximizing test automation (Stahl and Bosch, 2014).
- Automation Level 3 (Continuous Delivery - Aut.3): Teams collaboratively conduct integration, end-to-end, performance and user acceptance tests, ideally automating these processes. Greater automation requires enhanced collaboration, facilitating contextually relevant performance indicators (Chen, 2017).

3.8. Interpretation of Interviews

Phase 1 insights guided the interpretation of the 59 interviews in Phase 2. Interviewees identified primary collaborators to assess collaboration scope, minimizing bias by not probing about all potential job roles. Collaboration enrichment was evaluated through satisfaction or dissatisfaction feedback during semi-structured interviews (Hemon et al., 2018). Skills were explored by questioning the necessary know-how and behavioral skills for evolving roles. Findings were compared against hard and soft skills defined by Maram et al. (2009) and Robles (2012).

4. FINDINGS

4.1 Hard Skills and DevOps Maturity in the Context of Optimizing CI/CD Through Agile DevOps Collaboration

In the pursuit of optimizing CI/CD pipelines through Agile DevOps collaboration, a detailed

analysis of hard skills (HSk) becomes essential. The findings from Maram et al. (2009), as reflected through the experiences of 59 interviewees, highlight key aspects relevant to CI/CD pipeline enhancement:

4.1.1 Importance of Technical Skills Regardless of the Level of Automation

A significant insight from the study is that hard technical skills remain vital across all automation levels in a CI/CD context. For product engineers (PE), the emphasis on technical skills such as Testing (TST) and Quality, Security (QSY) is pronounced. This focus aligned with the necessity for continuous testing and quality assurance within CI/CD workflows, where automated tools are pivotal. PEs noted that testing automation and orchestration tools are crucial to maintain reliable, repeatable and swift deployment processes. Statements from engineers highlighted the challenge and importance of creating effective automation for test and delivery processes, ensuring code is properly versioned and tested as part of routine operations. This insight supported the idea that optimizing CI/CD requires robust technical capabilities that cater to automation and quality standards.

4.1.2 Evolution of Management (MNG) Skills with Automation

DevOps project managers (DPM) demonstrated a balance between managerial (MNG) and technical skills like software engineering (SWE). However, the study showed that higher levels of automation reduce the emphasis on traditional managerial skills. As automation in CI/CD grows, teams tend to self-organize, reducing the necessity for direct management intervention. This self-sufficiency is facilitated by the Agile and DevOps emphasis on autonomy and collaboration, which supports more streamlined and adaptive CI/CD practices.

DPMs recognized that their role shifts towards enabling teams to self-manage and collaborate effectively, which is essential in optimizing CI/CD. One DPM highlighted that enabling team autonomy and encouraging individuals to take initiative is crucial for successful project delivery. At higher automation levels, management becomes less about control and more about supporting seamless integration and collaborative culture, which aligns with the principles of Agile DevOps.

4.1.3 Negotiation (NEG) Skills and Team Alignment in High Automation Levels

Negotiation skills (NEG) appeared less significant as automation increases. This trend could be attributed to the Agile DevOps practice of continuous collaboration, fostering natural alignment among CI/CD stakeholders. In lower automation levels, negotiation is necessary to balance differing team priorities. However, as automation enhances and the DevOps culture of frequent interaction takes hold, shared understanding and alignment become more intuitive, reducing the emphasis on formal negotiation.

An interviewee mentioned the importance of understanding the different roles within a team to improve collaboration, especially in integrated Agile DevOps teams working on CI/CD pipelines. When teams are aligned in their understanding of responsibilities and goals, they work more cohesively, which directly supports efficient CI/CD pipeline execution. The DevOps approach inherently promotes this cross-functional awareness, thus optimizing CI/CD through stronger team integration and minimized friction.

These findings indicated that the successful optimization of CI/CD through Agile DevOps collaboration hinges on the technical skills to manage automated processes, evolving management practices to foster self-organized teams and shared alignment among team members to ensure fluid operations.

4.2 Perceptions of Soft Skills in the Context of CI/CD Optimization Through Agile DevOps Collaboration

The codification of interviewees' perceptions of soft skills (SSk) illuminated key changes as automation levels evolve, aligning with the optimization of CI/CD (Continuous Integration/Continuous Deployment) processes in Agile DevOps environments. Drawing from Robles' (2012) groups of soft skills, certain insights emerged:

4.2.1 Core Representations of Communication (COM) Skills

Communication skills (COM) are foundational to successful Agile DevOps collaboration, essential for

facilitating CI/CD practices. At every level of automation, interviewees highlighted the importance of clear, effective communication to foster seamless integration and deployment. For instance, a Release Manager (RM, Aut.1) emphasized, "You have to know how to communicate," underlining communication as a bridge between development and operations. This view was reinforced as automation increased; at Aut.3, communication evolved beyond basic information sharing to comprehensive team-wide understanding, enhancing CI/CD by ensuring all stakeholders grasp deployment stages and integrate feedback efficiently.

4.2.2 Interpersonal Skills (IPS) in Enhancing Collaborative Efficiency

Interpersonal skills (IPS), encompassing empathy, patience and sociability, were repeatedly noted across all automation levels. A Product Owner (PO, Aut.1) mentioned the necessity of balancing stakeholder demands and maintaining harmony within the CI/CD pipeline. By Aut.3, these skills facilitated smoother cross-functional collaboration, critical for reducing cycle time and preventing bottlenecks in CI/CD workflows. An interviewee (DPM, Aut.3) highlighted the need for quick relational assessments to build strong working relationships, an essential asset for maintaining momentum in iterative deployment cycles.

4.2.3 Flexibility (FLX) as a Driver of Continuous Improvement

Flexibility (FLX), the capacity to adapt and learn, is integral to CI/CD processed that demand rapid iteration and continuous learning. At lower automation levels (Aut.1), adaptability was associated with basic adjustments to new tools and processes. However, by Aut.3, flexibility was described in broader terms: embracing change, learning from mistakes and adjusting swiftly to new CI/CD protocols. A DPM (Aut.3) noted, "Everything evolved so quickly, it is better to know how to learn," illustrating that in high-automation environments, the ability to pivot and absorb new strategies ensures smoother and faster deployments.

4.2.4 Teamwork (TWK) as the Backbone of DevOps Culture

Teamwork (TWK) emerged as a cornerstone of Agile DevOps collaboration and CI/CD efficiency. The DevOps philosophy inherently breaks down silos between development (Dev) and operations (Ops), fostering collaborative environments where shared responsibility accelerates CI/CD processes. At Aut.1, team members were encouraged to understand each other's progress and align efforts. By Aut.3, collaboration reached a maturity where teams operated as unified entities, minimizing delays and enhancing deployment frequency. An Ops professional (RM, Aut.2) emphasized, "We do not ask for technical expertise anymore…we want the problem solved by a community."

4.2.5 Responsibility (RES) in Maintaining CI/CD Accountability

Responsibility (RES) skills, encompassing conscientiousness and reliability, were crucial for maintaining high standards in CI/CD practices. At early stages of automation, interviewees stressed the importance of individual commitment to getting tasks done. By Aut.3, the collective sense of accountability strengthened, driving teams to take ownership of their code, deployments, and problem-solving. A DPM (Aut.3) mentioned, "If they are self-organized, they must get in deep…when they fail, they repair," underscoring that high-functioning CI/CD pipelines depend on individuals who ensure continuous delivery without compromising quality.

The transition from basic to high automation levels in CI/CD processes through Agile DevOps collaboration highlighted evolving soft skill representations. Communication, interpersonal skills, flexibility, teamwork and responsibility are continuously adapted to meet the demands of more advanced CI/CD environments. As automation increases, the collaborative culture strengthened by Agile principles emphasizes skills that support seamless integration, rapid iteration, and efficient feedback loops, ultimately optimizing CI/CD workflows.

4.3 Roles and Collaboration Analysis

The analysis of collaboration roles within Agile DevOps environments reveals significant insights into the dynamics of cross-functional teamwork as influenced by Continuous Integration/Continuous

Deployment (CI/CD) practices. The results demonstrated that the evolution from traditional development models to automated, agile DevOps settings reshapes the collaboration patterns across different roles, enhancing both their frequency and quality.

Cross-Functional Collaborations and CI/CD Automation Levels:

Interview data indicated that higher levels of automation in CI/CD practices are directly correlated with increased collaboration between development (Dev) and operations (Ops) teams. Specifically, Table 4.1. highlighted that in automation level 1, cross-functional DevOps collaborations were fewer (20) compared to internal Dev or Ops collaborations (31). However, as teams progressed to automation level 3, these crossfunctional interactions became nearly equal (31 vs. 32), signifying balanced collaboration as the CI/CD automation matured. This progression indicates that the more automated the environment, the more integrated and balanced cross-functional interactions become, emphasizing the collaborative nature of DevOps for optimizing CI/CD processes.

Table 4.1. Collaboration analysis by level of automation

	Automation 1	Automation 2		Automation 3	
	Ops	Dev	Ops	Dev	Ops
Dev	11	23	11	25	12
Ops	10	15	6	19	7

Key Roles and Their Collaboration Patterns: Table 4.2. presented a detailed breakdown of collaborations by role, revealing both commonalities and unique aspects in how different functions interact:

- **Release Manager (RM):** Positioned centrally in collaborative structures, RMs work closely with project managers and production engineers to prioritize tasks and ensure project success. Their role in CI/CD-focused teams involves facilitating regular meetings and common work plans to maintain seamless progress. This reflected the critical nature of RMs in orchestrating the synchronization of deployment pipelines, ensuring smooth transitions between development and operations and enhancing CI/CD reliability.
- **Product Owner (PO):** POs were noted for their interactions with development project managers (DPMs) and developers, particularly during sprint reviews and demos that are integral to agile methodologies. POs highlighted that the transition to higher CI/CD automation levels fostered an integrated team approach, facilitating closer interaction with marketing and technical teams. This cross-functional engagement is vital for continuous feedback loops, enabling swift adjustments to CI/CD pipelines based on evolving product requirements and user feedback.
- **Architect (AR):** Architects frequently collaborate with POs, DPMs, and development teams to align on project goals and technical feasibilities. The shift toward DevOps, combined with agile practices, enhances these collaborations by promoting shared decision-making, thus optimizing architectural adjustments within CI/CD processes. This collaboration is essential for maintaining a consistent infrastructure that supports frequent deployments without compromising system integrity.
- **Production Engineer (PE):** PEs cited substantial collaboration with developers, aligning with the core philosophy of DevOps that blurs traditional operational boundaries. PEs play a pivotal role in ensuring CI/CD systems are operational, collaborating on shared responsibilities for documentation and process management. As CI/CD automation increases, PEs and developers often function as a unified DevOps team, fostering a culture of shared accountability and continuous improvement.

- **Department and Project Managers (DPM):** DPMs manage comprehensive collaboration networks, involving POs, RMs, and functional architects. They are integral to coordinating budgets, tracking progress and prioritizing development efforts, all of which are crucial for optimizing CI/CD pipelines. 26 Agile DevOps practices redefine DPM roles to incorporate elements of Scrum Master Responsibilities, thereby streamlining planning and project management within CI/CD frameworks.

Table 4.2. Main collaborations reported by each role

OPS Main Collaborations cited ↑ ↓	PO (N=12)	DPM (N=13)	AR (N=11)	PE (N=12)	RM (N=11)
Product Owner, N	3	13	11	11	10
Department & Project Manager, N	10	13	11	12	11
Architect, N	3	1	1	4	2
Scrum Master, N		3			
Developer, N	9	8	8	12	4
DevOps animator, N		1			
Technical expert, N		1	1	3	2
Quality expert, N	1				
Multiple collaborators, N		2	1		
Qualifier, N		1			
Production Engineer, N	1	4	2	1	8
Administrator Infrastructure Director, N			1		
System Engineer, N				1	
Operators, N		3	2	5	4
Release Manager, N	2	1	6	6	

Improvements in Collaboration Quality:
The shift to agile DevOps practices, supported by CI/CD, has led to notable improvements in the quality of collaborations. Interviewees attributed these enhancements to the structured setup of daily meetings, sprint reviews, and the adoption of collaborative tools like Mingle. While these formalized interactions contribute to synchronization and proactive problem-solving, they also pose potential challenges such as meeting fatigue, which requires strategic management to sustain productivity and morale.

The Role of Automation in Collaborative Dynamics:
As automation advances from level 1 to level 3, teams experience an increase in cross-functional interactions that become as significant as intra-functional collaborations. This shift underscores that optimizing CI/CD through agile DevOps is as much about fostering seamless collaboration as it is about implementing robust technical practices. The evolving role of project participants—from architects and engineers to product owners and managers—demonstrates that CI/CD optimization relies heavily on integrated communication and shared responsibilities.

Optimizing CI/CD through agile DevOps collaboration is characterized by a balance between cross-functional and intra-functional teamwork, with increasing automation driving greater cohesion. Roles such as RMs, POs, and PEs adapt to new collaborative dynamics, fostering a culture where interaction and mutual support enhance deployment speed and reliability. Agile practices, paired with CI/CD processes, enable continuous improvement through strengthened partnerships, ultimately

contributing to efficient software delivery and operational excellence.

5. CONCLUSION

In this study, it was explored how the progression towards DevOps enhances Continuous Integration/Continuous Delivery (CI/CD) through improved collaboration and smartness. It was identified three distinct stages that underpin the transition from Agile practices to DevOps: Agile (Aut.1), Continuous Integration (Aut.2) and Continuous Delivery (Aut.3). Each stage emphasized the significance of skills and collaboration patterns essential for optimizing CI/CD processes. Study findings indicated a significant shift in the required soft skills within software teams and their collaborative dynamics. As teams advance to Automation Level 3, collaboration between Development (Dev) and Operations (Ops) becomes more balanced, reflecting richer interactions and deeper understandings among team members. Study highlighted the critical role of responsibilities and communication skills at Automation Level 1, aligning with Agile methodologies, which naturally encourage these competencies. Interestingly, flexibility and teamwork skills were less emphasized in interviews, suggesting potential areas for further development in CI/CD practices. As teams progress to Automation Level 2, collaborations expand, but observed a decrease in the emphasis on responsibility-related skills at Automation Level 3. By applying Alter's (2018) criteria for smartness, it was demonstrated that adopting DevOps fosters automation and enhances flexibility, resulting in faster and more efficient delivery cycles. Therefore, DevOps not only contributed to greater smartness in the Information System function but also serves as a critical enabler of optimized CI/CD. Despite the existence of maturity models in Software Engineering, a dedicated model for analyzing the transition from Agile to DevOps, particularly regarding CI/CD optimization, is still lacking. Existing commercial models, such as the simplified DevOps Maturity Model from Atlassian-GitLab, tend to focus more on technical aspects without offering substantial guidance on managing this transition or achieving full adoption of DevOps practices. Overall, this research sheds light on the practical implications of transitioning to DevOps for optimizing CI/CD processes and suggests a simple yet effective framework for visualizing this path. While findings contributed valuable insights and based on an exploratory embedded case study within a single 32 large organization. Consequently, further research across diverse contexts is necessary to validate and expand upon these results. In addition, study methodology did not fully capture the diversity of collaborations among team members. Future studies that employ observational methods to analyze interactions and collaborative networks would enhance our understanding of the factors influencing CI/CD optimization in a DevOps context. Finally, it would be beneficial to delve deeper into identifying elements that may limit the smartness of DevOps initiatives, as these factors may not always be direct opposites of those contributing to enhanced smartness.

6. REFERENCES

Abuamoud, I., Lillywhite, J., Simonsen, J., & Al-Oun, M. (2016). Factors influencing food security in less popular tourists sites in Jordan's Northern Badia. International Review of Social Sciences and Humanities, 11(2), 20-36.

Almeida, F., Simões, J., & Lopes, S. (2022). Exploring the benefits of combining DevOps and agile. Future Internet, 14(2), 63.

Alter, S. (2018). Making sense of smart living, working, and organizing enhanced by supposedly smart objects and systems. In Proceedings of the IFIP WG8.6 Working Conference: Smart Working, Living and Organizing (pp. 1–14). Presented at the IFIP WG8.6 working conference: Smart working, living and organizing, Portsmouth. UK: Springer.

Alter, S. (2019). Making sense of smart living, working, and organizing enhanced by supposedly smart objects and systems. In Smart Working, Living and Organising: IFIP WG 8.6 International Conference on Transfer and Diffusion of IT, TDIT 2018, Portsmouth, UK, June 25, 2018, Proceedings (pp. 247-260). Springer International Publishing.

Ambler, S. W., & Lines, M. (2012). Disciplined agile delivery: A practitioner's guide to agile software delivery in the enterprise. Indianapolis, IN, USA: IBM Press.

Braun, V., & Clarke, V. (2006). Using thematic analysis in psychology. Qualitative Research in Psychology, 3(2), 77–101.

Brunnert, A., van Hoorn, A., Willnecker, F., Danciu, A., Hasselbring, W., Heger, C., ... & Wert, A. (2015). Performance-oriented DevOps: A research agenda. arXiv preprint arXiv:1508.04752.

Bruns, H. C. (2013). Working alone together: Coordination in collaboration across domains of expertise. Academy of Management journal, 56(1), 62-83.

Céspedes, D., Angeleri, P., Melendez, K., & Dávila, A. (2020). Software product quality in DevOps contexts: A systematic literature review. In Trends and Applications in Software Engineering: Proceedings of the 8th International Conference on Software Process Improvement (CIMPS 2019) (pp. 51-64). Springer International Publishing.

Chen, L. (2017). Continuous delivery: Overcoming adoption challenges. Journal of Systems and Software, 128, 72–86.

Coyle, S., Conboy, K., & Acton, T. (2015). An exploration of the relationship between contribution behaviours and the decision making process in agile teams.

Dhal, K., Karmokar, P., Chakravarthy, A. *et al.* Vision-Based Guidance for Tracking Multiple Dynamic Objects. *J Intell Robot Syst* 105, 66 (2022). https://doi.org/10.1007/s10846-022-01657-6

Dixit, S., & Jangid, J. (2024). Exploring Smart Contracts and Artificial Intelligence in FinTech. https://jisem-journal.com/index.php/journal/article/view/2208

Dörnenburg, E. (2018). The path to devops. IEEE software, 35(5), 71-75. Faustino, J., Pereira, R., Alturas, B., & Silva, M. M. D. (2020). Agile information technology service management with DevOps: An incident management case study. International Journal of Agile Systems and Management, 13(4), 339-389.

Easwaran, V., Orayj, K., Goruntla, N., Mekala, J. S., Bommireddy, B. R., Mopuri, B., ... & Bandaru, V. (2025). Depression, anxiety, and stress among HIV-positive pregnant women during the COVID-19 pandemic: a hospital-based cross-sectional study in India. BMC Pregnancy and Childbirth, 25(1), 134.

Elavarasan, R. M., Afridhis, S., Vijayaraghavan, R. R., Subramaniam, U., & Nurunnabi, M. (2020). SWOT analysis: A framework for comprehensive evaluation of drivers and barriers for renewable energy development in significant countries. *Energy Reports*, *6*, 1838-1864.

Fernández-Diego, M., Méndez, E. R., González-Ladrón-De-Guevara, F., Abrahão, S., & Insfran, E. (2020). An update on effort estimation in agile software development: A systematic literature review. IEEE Access, 8, 166768-166800.

Fitzgerald, B., & Stol, K. J. (2017). Continuous software engineering: A roadmap and agenda. Journal of Systems and Software, 123, 176-189.

Gallivan, M. J., Truex III, D. P., & Kvasny, L. (2004). Changing patterns in IT skill sets 1988- 2003: a content analysis of classified advertising. ACM SIGMIS Database: the DATABASE for Advances in Information Systems, 35(3), 64-87.

Galup, S., Dattero, R., & Quan, J. (2020). What do agile, lean, and ITIL mean to DevOps?. Communications of the ACM, 63(10), 48-53.

Goles, T., Hawk, S., & Kaiser, K. M. (2009). Information technology workforce skills: The software and IT services provider perspective. Information Systems Outsourcing: Enduring Themes, Global Challenges, and Process Opportunities, 105-125.

Guimarães, Portugal. Rajapakse, R. N., Zahedi, M., Babar, M. A., & Shen, H. (2022). Challenges and solutions when adopting DevSecOps: A systematic review.

Information and software technology, 141, 106700.

Hemon, A., Lyonnet, B., Rowe, F., & Fitzgerald, B. (2020). Fromagile to DevOps: Smart skills and collaborations. Information Systems Frontiers, 22(4), 927-945.

Hemon, A., Monnier-Senicourt, L., & Rowe, F. (2018). Job satisfaction factors and risks perception: an embedded case study of DevOps and agile teams. In Proceedings of the 39th International Conference on Information Systems (ICIS), San Francisco, CA, USA.

Holmström Olsson, H., Alahyari, H., & Bosch, J. (2012). Climbing the" Stairway to Heaven"–A Mulitiple-Case Study Exploring Barriers in the Transition from Agile Development towards Continuous Deployment of Software. In 2012 38th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA) (pp. 392–399). IEEE.

Hood, K., & Al-Oun, M. (2014). Changing performance traditions and Bedouin identity in the North Badiya, Jordan. Nomadic Peoples, 18(2), 78-99.

Humble, J., & Molesky, J. (2011). Why enterprises must adopt devops to enable continuous delivery. Cutter IT Journal, 24(8), 6.

Hüttermann, M. (2012). Beginning devops for developers. In DevOps for Developers (pp. 3-13). Berkeley, CA: Apress.

Jagdish Jangid. (2023). Enhancing Security and Efficiency in Wireless Mobile Networks through Blockchain. International Journal of Intelligent Systems and Applications in Engineering, 11(4), 958–969. Retrieved from https://ijisae.org/index.php/IJISAE/article/view/7309

Kim, G., Humble, J., Debois, P., Willis, J., & Forsgren, N. (2021). The DevOps handbook: Howto create world-class agility, reliability, & security in technology organizations. It Revolution.

Larman, C., & Basili, V. R. (2003). Iterative and incremental developments. a brief history. Computer, 36(6), 47-56.

Lee, J., Kim, I., Kim, H., & Kang, J. (2021). SWOT-AHP analysis of the Korean satellite and space industry: Strategy recommendations for development. *Technological Forecasting and Social Change*, *164*, 120515.

Leite, L., Rocha, C., Kon, F., Milojicic, D., & Meirelles, P. (2019). A survey of DevOps concepts and challenges. ACM Computing Surveys (CSUR), 52(6), 1-35.

Lincoln, Y. S., & Guba, E. G. (1985). Naturalistic inquiry. Beverly Hills: Sage Publications.

Lwakatare, L. E., Kuvaja, P., & Oivo, M. (2015). Dimensions of devops. In International Conference on Agile Software Development (pp. 212–217). Helsinki, Finland: Springer.

Lwakatare, L. E., Kuvaja, P., & Oivo, M. (2016). Relationship of devops to agile, lean and continuous deployment: A multivocal literature review study. In Product-Focused Software Process Improvement: 17th International Conference, PROFES 2016, Trondheim, Norway, November 22-24, 2016, Proceedings 17 (pp. 399-415). Springer International Publishing.

Maram, M., Prabhakaran, P., Murthy, S., & Domala, N. (2009). Sixteen roles performed by software engineers in first one year. In 22nd Conference on Software Engineering Education and Training (pp. 212–215). IEEE. Morris, K. (2020). Infrastructure as code.

Nguyen, H. U., Trinh, T. X., Duong, K. H., & Tran, V. H. (2018). Effectiveness of green muscardine fungus Metarhizium anisopliae and some insecticides on lesser coconut weevil Diocalandra frumenti Fabricius (Coleoptera: Curculionidae). CTU Journal of Innovation and Sustainable Development, (10), 1-7.

Nguyen, L., Trinh, X. T., Trinh, H., Tran, D. H., & Nguyen, C. (2018). BWTaligner: a genome short-read aligner. Vietnam Journal of Science, Technology and Engineering, 60(2), 73-77.

Nybom, K., Smeds, J., & Porres, I. (2016). On the impact of mixing responsibilities between devs and ops. In Agile Processes, in Software Engineering, and Extreme Programming:

17th International Conference, XP 2016, Edinburgh, UK, May 24-27, 2016, Proceedings 17 (pp. 131-143). Springer International Publishing.

Oliveira Martins, S. R. (2017). High Radicality of product innovation and high flexibility and high agility of system of manufacturing: Towards the smart factories. Procedia Manufacturing, 11, 1324– 1334.

Oomen, S., De Waal, B., Albertin, A., & Ravesteyn, P. (2017). How can scrum be successful? Competences of the scrum product owner. In Proceedings of the 25th ECIS European Conference on Information Systems (pp. 131–142).

O'Reilly Media. Nerkar, A., & Paruchuri, S. (2005). Evolution of R&D capabilities: The role of knowledge networks within a firm. Management science, 51(5), 771-785.

Robles, M. M. (2012). Executive perceptions of the top 10 soft skills needed in today's workplace. Business communication quarterly, 75(4), 453-465.

Robles, M. M. (2012). Executive perceptions of the top 10 soft skills needed in today's workplace. Business Communication Quarterly, 75(4), 453–465.

Runeson, P., & Höst, M. (2008). Guidelines for conducting and reporting case study research in software engineering. Empirical Software Engineering, 14(2), 131.

Sachin Dixit, & Jagdish Jangid. (2024). Asynchronous SCIM Profile for Security Event Tokens. Journal of Computational Analysis and Applications (JoCAAA), 33(06), 1357–1371. Retrieved from https://eudoxuspress.com/index.php/pub/article/view/1935

Sawyer, S., Ellers, S., Kakumanu, M. S., Bommireddy, B., Pasgar, M., Susan-Kurian, D., ... & Jurdi, A. A. (2025). Trial in progress for a colorectal cancer detection blood test. https://ascopubs.org/doi/10.1200/JCO.2025.43.4_suppl.TPS306

Schwaber, K., & Sutherland, J. (2011). The scrum guide. Scrum Alliance, 21. Stahl, D., & Bosch, J. (2014). Modeling continuous integration practice differences in industry software development. Journal of Systems and Software, 87, 48–59.

Ståhl, D., & Bosch, J. (2014). Modeling continuous integration practice differences in industry software development. Journal of Systems and Software, 87, 48-59.

Stahl, D., Martensson, T., & Bosch, J. (2017, August). Continuous practices and devops: beyond the buzz, what does it all mean?. In 2017 43rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA) (pp. 440-448). IEEE.

Strode, D. E. (2016). A dependency taxonomy for agile software development projects. Information Systems Frontiers, 18(1), 23- 46.

Strode, D. E., Huff, S. L., & Tretiakov, A. (2009). The impact of organizational culture on agile method use. In 42nd Hawaii International Conference on System Sciences (HICSS) (pp. 1– 9).

Theurich, P., Witt, J., & Richter, S. (2023). Practices and challenges of threat modelling in agile environments. Informatik Spektrum, 46(4), 220-229.

Vivian, R., Tarmazdi, H., Falkner, K., Falkner, N., & Szabo, C. (2015). The development of a dashboard tool for visualising online teamwork discussions. In IEEE/ACM 37th IEEE International Conference on Software Engineering (Vol. 2, pp. 380–388). IEEE.

Wettinger, J., Breitenbücher, U., & Leymann, F. (2014). Standardsbased DevOps automation and integration using TOSCA. In Proceedings of the 2014 IEEE/ACM (pp. 59–68). Presented at the 7th International Conference on Utility and Cloud Computing, London, UK: IEEE Computer Society.

Wiedemann, A. M., & Schulz, T. (2017). Key Capabilities of DevOps Teams and their Influence on Software Process Innovation: A Resource-Based View. In Proceedings of 23rd ACIS Americas Conference on Information Systems (pp. 1–10). Boston.

Wiedemann, A. M., & Schulz, T. (2017). Key Capabilities of DevOps Teams and their Influence on Software Process Innovation: A

Resource-Based View. In Proceedings of 23rd ACIS Americas Conference on Information Systems (pp. 1–10). Boston.

Wiedemann, A., Forsgren, N., Wiesche, M., Gewald, H., & Krcmar, H. (2019). Research for practice: the DevOps phenomenon. Communications of the ACM, 62(8), 44-49.

Wong, S., von Hellens, L., & Orr, J. (2006). Non-technical skills and personal attributes: The soft skills matter Most. In Proceedings of the 6th Australiasian Women in Computing Workshop (pp. 27–33). Brisbane, Australia.

Yin, R. K. (1994). Case StudyResearch: Design and Methods (2nd edn). Thousand Oaks, CA: Sage.