DevOps and Agile Integration Software Development and Deployment Process.

The Department of Computer Science Engineering
Institute of Technology, Nirma University (NU)
Ahemdabad, Gujarat, India
Keval Savaliya 21BCE263
Adhishek Rathod 21BCE249

Abstract

Agile and DevOps techniques together may help organizations better meet the requirements and expectations of their clients. This methodology expedites the development process and improves software quality by replacing waterfall models and separating development teams from operations. With the help of DevOps, software development and delivery are fully automated. When moving from agile to DevOps, there are three distinct phases to consider: agile, continuous integration, and continuous delivery. The objective of this research is to investigate the advantages of merging the two models and suggests a hybrid DevOps Enabled Agile approach for software development. As part of the research, a small-scale experimental pilot project will be conducted to show how software development teams may integrate Agile and DevOps approaches to enhance and expedite the software development process while upholding realistic skill requirements.

Keywords: Agile, DevOps, Automation, software engineering,„waterfall models Continuous Integration,Continuous Delivery, configuration management,hybrid DevOps, Agile methodology,integrate Agile and DevOps

I. INTRODUCTION

The four fundamental tenets of the agile methodology are customer participation, functioning software, individual engagement, and change reaction. One particularly successful software development approach is the agile methodology. Scrum is a very adaptable framework that can be used with a range of approaches, including DAD, SAFe, XP, and others. It is especially useful for small teams. Agile has advantages including faster delivery, greater customer cooperation, less documentation required, and ongoing business needs alignment.

However, there are a number of shortcomings, including a lack of cross-functionality and the division of the development, testing, and delivery sectors. The pace and effectiveness of software development are constrained by this lack of feedback integration throughout the development process. In addition to creating fragmented and chaotic sectors, development teams without cross-functionality may also delay project completion beyond initial projections.

DevOps is an extension of the Agile methodology that aims to enable cross-functionality in teams via the alignment of operations and development processes. It emphasizes developer and operator cooperation, leading to very synchronized development, testing, and delivery. [18]

However, due to their lack of in-depth training and competency in DevOps culture and practices, software engineers are ill-prepared to fulfill the high skill requirements of pure DevOps and its inability to adapt to constantly changing client expectations. Therefore, it would not be ideal for an efficient and successful software development process to completely include DevOps into public sectors, such as mid- and low-level software development projects, since this would create challenges and downsides.

DevOps and Agile methodologies have broad definitions and emphasize concepts over concrete actions, which may make it difficult to understand the differences between them. Project management is essential to the design, implementation, and supervision of software projects. Task execution is critical to the success of these efforts. In order to overcome these obstacles and achieve client expectations, agile methodologies have been extensively embraced around the globe. While agile development provides a wide range of options at every stage of the development process, DevOps techniques manage the application development process in fewer iterations, or sprints.

The purpose of this research is to investigate how people react to Agile and DevOps approaches from a qualitative standpoint. The literature will be specifically examined to look at the findings made by earlier studies about the impacts of Agile and the implications of automation in DevOps and Agile on the state of IT. The goal of this research project is to improve knowledge of the information technology sector and the efficacy of information systems as a result of DevOps' contributions [29].

Table 1. Literature Review Table

Authors	Year of publication	P1	P2	P3	P4	P5	Advantages	Disadvantages
Mohammad *et al [23].*	2023	✓	×	✓	✓	✓	Process model for BDA and how it is an improve CRISP-DM	Managing data governance and compliance becomes more challenging with Big Data.
Ellen *et al [21].*	2023	✓	✓	✓	✓	×	Use of popular metrics such as deployment	Popular metrics NOt capture broader organizational goals.give importance to customer stisfaction.
Srivastava *et al [27].*	2023	×	✓	×	✓	✓	DevOps gain a competitive edge by delivering software rapidly than their compe,	Defining meaningful metrics and (KPIs) to assess the success of DevOps it can be challenging.
Almeida *et al [1].*	2022	×	×	×	×	×	Problems with Agile development and DevOps solution	Businesses may not provide a completely objective picture of the advantages.
Singh *et al [13].*	2022	×	×	✓	✓	✓	Challenges of an Agile organization Design	Interaction of multiple services and components in a dynamic environment can be complex.
Md.Masnun *et al [22].*	2022	×	×	×	✓	✓	They has been focused on discovering gaps of the existing DevOps and Agile Methodologies	They haven't used methods and instruments to confirm if their approach can
M.Jayakody *et al [16].*	2021	✓	×	✓	×	×	The identifi mitigating strategies practiced by IT project teams using the question survey	Traditional IT cultures resist the cultural shift towards automation,shared responsibility that DevOps promotes
Bildirici *et al [28].*	2021	✓	✓	×	✓	✓	Working with Havelsan can provide valuable exposure to these specialized domains.	Havelsan can sometimes be bureaucratic, which may slow down decision-making and innovation.
Govil *et al [12].*	2020	✓	×	✓	✓	×	Agile methodologies and DevOps processes are acquainted in e–commerce application.	It is worthy to mention that DevOps culture is not successful without agile.
Necmettin *et al [24].*	2020	×	×	✓	×	✓	COBIT focuses KPIs and metrics, while Scrum relies on different metrics like velocity	OBIT's lead to rigid processes that may not be well-suited for rapidly changing environments or agile
Carla *et al [7].*	2019	✓	✓	×	×	✓	The use of Agile-DevOps in tandem is always appropriate approach to achieve change.	Barriers exist in large customer and project partnerships to these practices

Parameter: P1=Cultural Alignment, P2=Cross Functional Year, P3= CI/CD, P4=Feedback Loop, P5=Automation

II. LITERATURE SURVEY

A. *Literature Review*

Software development professionals often use Agile and DevOps as business paradigms, however there are sometimes misunderstandings or overlaps between them. These methods work well together rather than against one another. To bring about change in groups, divisions, or entities, they work in concert.Adaptability to ongoing change and the realization that no one solution can address every organizational requirement are prerequisites for accepting these ideas [5].

B. *Agile methodology*

By accepting end-user input, software engineers want to simplify and increase the agility of program modification. Software development saw the birth of lightweight ideas like Scrum and Kanban in the 1990s. These ideas were made official in 2001 by the Agile Software Development Manifesto [26].

1) People: Placing more emphasis on relationships between coworkers, customers, and stakeholders than on procedures and instruments.

2) Immediacy: putting practical applications ahead of thorough documentation.

3) Flexibility: Accepting change and adapting to it instead than following set plans to the letter.

C. *The Modern DevOps*

While they are complimentary to waterfall processes, agile and devops are not the same. Agile is more concerned with dependability, safety, and compliance than DevOps is with designing and delivering new products to customers. These fields are crucial to contemporary enterprises because they provide adaptability, in-depth analysis, and communication throughout the software installation process. Although each theory has advantages and disadvantages, they complement one another to increase effectiveness and performance [15].

1) Culture: The culture change aims to foster collaboration and integration between the developers' and operations' teams.

2) Automation: It refers to the use of high levels of automation to ensure continuous delivery of code changes through automated tests.

3) Lean: It is the systematic application of principles to enhance efficiency and minimize complexity.

4)Measurement: The process of measurement involves maintaining key performance indicators to gauge performance and pinpoint areas for improvement.

5) Sharing: It is important to disseminate knowledge and optimal methodologies across the business to guarantee efficient communication and cooperation.

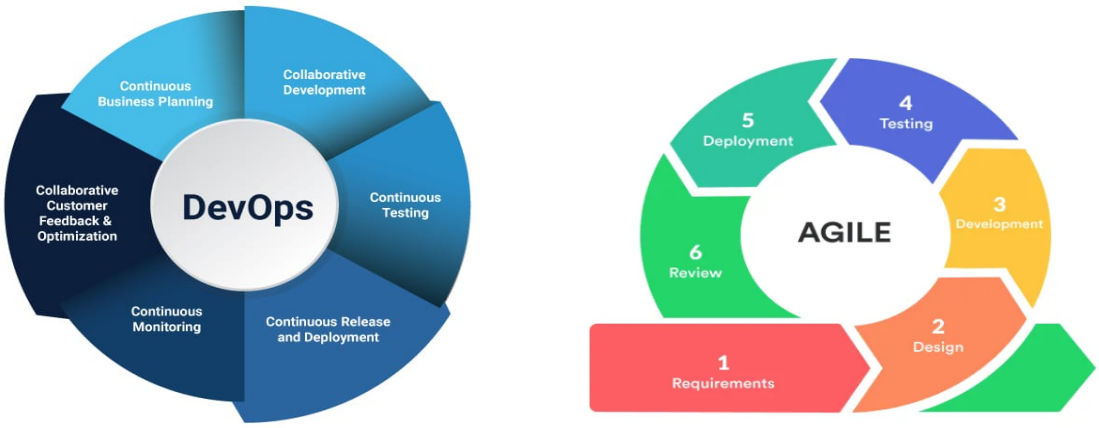


Fig. 1. DevOps & Agile Process Model

In the last ten years, companies have either moved to a DevOps team from their original IT staff or started using agile methodologies for their software development projects. Test automation, continuous integration (CI), and automated design are the main areas of emphasis for agile teams. DevOps teams often employ CI. Agile was seen as a novel approach by developers who were fed up with the limitations of a waterfall technique. Nevertheless, there are drawbacks to agile planning, such missing deadlines, parts that are done but are incompatible, and new additions that break apart older features because there was no cooperation with DevOps and ITOps. These problems are associated with failures in Agile development's communication.

*D. **Similarities***

The objective of both DevOps and Agile methodologies is to produce software quickly without compromising customers or operations. They are both flexible, prioritize consistency and speed, and can be used into a range of business structures and sectors. Agile is more concerned with speed, flexibility, and high-quality business value delivery than DevOps is with providing the technical foundation. Both approaches work well together and need informed stakeholders and productive sprints [14].

Agile	Devops
Recursive development in brief bursts	Continuous improvement in brief bursts development method that follows a step-by-step plan
Methodology that is adaptable and flexible	The procedure is recorded and adheres to the predetermined standards and structure that were established at the outset.
Feedback-based approach: Development processes are guided by sprints, which provide brief build updates that are assessed and used to inform future directions.	Restrictions on feedback: Customer and tester input is sought at the last stage of software development processes, which is when the fulfillment is assessed.
A clause for flexibility: During the iterative development process, it is anticipated that the needs and scope of the project would change.	Once decided upon, the specifications and scope are final.
Plan, requirements, design, build, test, and maintain are the stages of the Software Development Life Cycle (SDLC), and they start early in the process.	There is no overlap in the sequence in which the SDLC stages are followed. Employees in one functional group do not participate in any other phase outside of their designated work duties.
adopts a cooperative and communicative approach. Everyone involved talks about the needs, difficulties, developments, and adjustments on an ongoing basis.	Adopts a project-focused approach with the intention of finishing the SDLC procedure completely.
The goal for each team member was to see the projects through to completion, each step completed in order.	Only throughout the stages of their individual SDLC projects were team members focused on their duties.
Ideal for quick initiatives with a lot of risk.	Ideal for simple jobs with known conditions.
Few dependencies since the emphasis is more on mentality than on implementation details.	strict relationships between people, projects, technology, and procedures.

Table 2. Agile Vs DevOps

Agile approaches have been used to large-scale projects with success, assisting companies in meeting present demands and promoting a happy workplace. On the other hand, not much research has been done on the topic of how Agile techniques work with OPI programs. Proactive development, concurrent design, multisectoral management, and corporate business strategies might influence future technical developments. One of the main concerns during the 2020 XP summit in Trondheim, Norway, was how to incorporate Agile approaches into large-scale projects [19].

Problem with Agile	DevOps Solution
Customers often experience delays in receiving new features.	New features are tested and released as soon as they are finished using DevOps technologies.
Software components that have been completed do not work with one another.	Development may be divided into separate but compatible components thanks to open APIs and test automation.
Before being released, the product's quality is not adequately guaranteed.	Automation of quality assurance and a reduction in the requirement for tedious human labor are both benefits of DevOps technologies and processes.
Old functionality are broken by new features.	Following every modification, the quality of the already in use functionality is promptly and automatically guaranteed.
Deadlines and budget objectives are missed.	The development process is more transparent and predictable thanks to the DevOps technologies and processes.
Teams working on IT operations and development are not collaborating.	Tasks are jointly agreed upon both IT operations teams and development teams. They have a same objective.

Table 3. Issues with DevOps and Agile development solutions

III. PROBLEM STATEMENT

The problem of integrating the agility of Agile approaches with the effectiveness and dependability of DevOps practices is one that modern software development firms must overcome. While Agile promotes quick iterations and frequent feature updates, DevOps seeks to automate and streamline the software delivery process. The combination of these two methods frequently raises a number of significant problems [8].

1) Goal misalignment (G): DevOps stresses automation, monitoring, and reliability, whereas Agile prioritizes flexible development and client feedback. Conflicts in the development process may result from these divergent agendas.

2) Cross-Functional Teams (CFT): While DevOps necessitates close cooperation between the development and operations teams, Agile largely relies on cross-functional teams and regular communication. It can be hard to bridge the gap between these teams.

3) Continuous Integration and Continuous Deployment (CI/CD): Agile and DevOps both advocate CI/CD pipelines, although it can be difficult to synchronize these pipelines to provide incremental features. It is a huge effort to maintain a quick release cycle while making sure that every code change is adequately tested and reliably deployed.

4) Agile Frameworks (AF): Decide on a framework, such as Scrum, Kanban, or a hybrid strategy, that supports DevOps.Use Agile ceremonies to keep your attention on iterative development and feedback, such as stand-ups, sprint planning, and retrospectives.

5) User Stories (US) and the backlog (B): Create user stories, and keep track of your to-do list.Prioritize user stories according to their business value, then divide them into more manageable development jobs.

6) Sprints (s) and Iterations (I): Produce working software incrementally by working in brief sprints or iterations.Make sure that each sprint yields deliverable increments.

7) Automation (A): Automate repetitive operations, such as testing, deployment, and provisioning of infrastructure.To manage and version infrastructure configurations, use infrastructure as code (IaC).

8) Monitoring (M) and Feedback (F): Implement reliable monitoring and feedback methods to collect information on the effectiveness of the application, user input, and operational metrics.Make adjustments to development priorities and use this data to promote ongoing improvement.

9) Security (S) and compliance (C): Include security procedures like static code analysis, vulnerability scanning, and security testing in the development process.Ensure that you are adhering to all rules and laws.

10) Integration of the toolchain (T): Choosing and integrating the right technologies for Agile and DevOps might be difficult. Problems with training, maintaining a cohesive toolchain, and compatibility exist.

11) Training and skill development (TR): Make sure team members are competent in both Agile and DevOps approaches by offering them training and support.Encourage continual skill improvement and learning.

12) Change Management (CM): While DevOps emphasizes stability and predictability, Agile accepts shifting requirements. It becomes crucial to manage changes effectively and without interfering with business operations.

13) Scalability (SC) and Flexibility (FLX): It's essential to make sure that Agile and DevOps processes can scale while still being flexible to meet evolving business requirements as projects expand.

14) Feedback Loop (FL), Reviews (R) and Governance (G): Establish feedback loops, conduct regular reviews, and implement governance to enhance communication, address issues, and track progress in integrated Agile and DevOps processes.

IV. METHODOLOGY

This methodology offers a structured strategy for integrating Agile and DevOps in a way that leverages the advantages of each while resolving the difficulties that may result from their divergent goals and methods. It's vital to remember that the specific procedures and methods may change depending on the particular needs and restrictions of the company.

The mathematical equation for the methodology can be represented as:

$$G = CFT + (CI + CD) + AF + (US + B) + (I + S) + A + (M + F) + (S + C) + T + Tr + CM + FL + (Sc + Flx) + (R + G)$$

Using mathematical symbols, we are mixing different methodological components in this equation. The depth and complexity of the actual integration process are not captured by this portrayal, which is quite abstract and oversimplified. DevOps and Agile integration in practice is a complex, dynamic, and iterative process that is impossible to adequately capture in a single mathematical formula.While many components of DevOps and Agile processes (such CI/CD automation) can be optimized

using mathematics and algorithms, the whole methodology is a concoction of behaviors, technologies, and culture that is inherently challenging to define as a mathematical model [30].

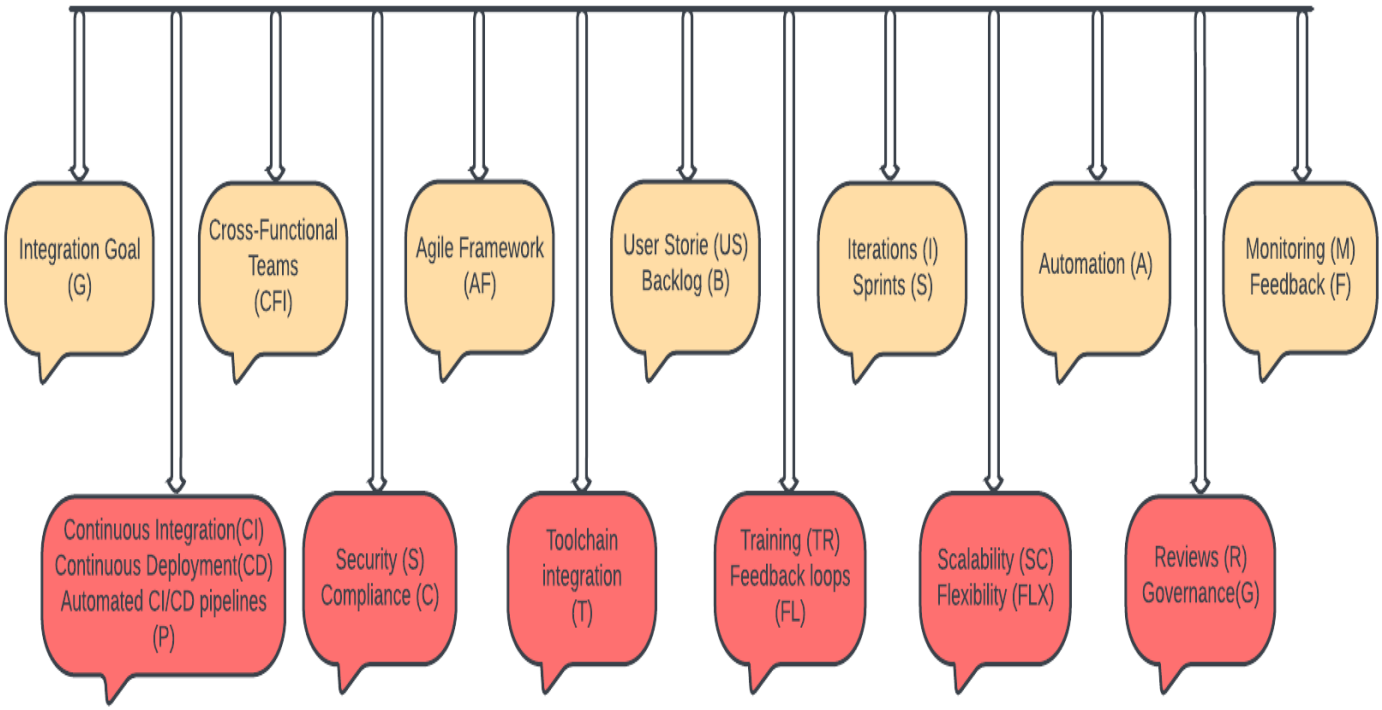


Fig. 2. Terms used in methodology of Agile & DevOps

A. Use Case: Resource Allocation Optimization

With this methodology we can Maximize the Efficiency of Resource Allocation

	x_i: Resource allocation
Sum(x_i) ¡= R, for all i	R: Total available resources
x_i ¡= C_i, for all i	p_i: Productivity, C_i: Capacity
U_i * C_i = p_i * x_i, for all i	U_i: Utilization rate
x_i = 0, for all i	W_i: Priority weight for team i
Maximize: Z = Sum(W_i * p_i), for all i	Z: Objective function value

In the first part of the code you have a constraint that ensures that the sum of the allocations (x_i) of all groups does not exceed the total number of available resources (R) This constraint ensures that you do not allocate excessive resources, a it can lead to items items that are scarce or malfunctioning. This is a fundamental limitation in resource allocation problems [2].

The second part of the code describes the relationship between performance, capacity, and level of activity. It mandates that each group's activity level (U_i) is the product of their productivity (p_i) and their resource allocation (x_i). This limit helps ensure that resources are used efficiently and that teams do not waste resources. It also indirectly relates the value of the objective function (Z) to total employment and resource allocation.

The objective function is to maximize Z , which is the sum of the weighted priority results of all groups. This objective encourages the allocation of resources to groups with greater priority and greater efficiency, potentially leading to more efficient use of resources

While taking into account resource limitations, team capacity, and utilization rates, the objective is to maximize productivity across Agile teams. A decision variable, x_i, which represents resource allocation, and a priority weight, W_i, which reflects the significance of the project, are supplied to the teams. Real-world resource allocation issues in Agile and DevOps are more complex and may involve additional aspects than this simplified mathematical model suggests. The particular characteristics and objectives of the company determine the use of mathematical models and algorithms [6].

V. SOLUTION STATEMENT

A. *Algorithm*

Algorithm 1 Integration Process

```
IntegrationGoal ← DefineIntegrationGoal()
CrossFunctionalTeams ← CreateCrossFunctionalTeams()
CI/CD_Pipelines ← ImplementContinuousIntegrationAndDeployment()
AgileFramework ← ChooseAgileFramework()
Backlog ← CreateBacklog()
while SprintNotComplete do
  Sprint ← PlanSprint()
  for UserStory in Sprint do
    Develop(UserStory)
    Test(UserStory)
    Deploy(UserStory)
    Monitor(UserStory)
  end for
end while
AutomationTools ← ImplementAutomation()
for Task in RepetitiveTasks do
  AutomationTools.Execute(Task)
end for
MonitoringTools ← ImplementMonitoring()
while ApplicationIsLive do
  Metrics ← MonitoringTools.CollectMetrics()
  Feedback ← GatherUserFeedback()
  if IssuesFound then
    ResolveIssues()
  end if
end while
SecurityTools ← ImplementSecurity()
ComplianceTools ← EnsureCompliance()
SecurityTools.ScanCode()
ComplianceTools.CheckStandards()
Toolchain ← SelectAndIntegrateTools()
TrainingPrograms ← ProvideTraining()
TrainingPrograms.Execute()
ChangeManagement ← ImplementChangeManagement()
ChangeManagement.EnsureAdoption()
FeedbackLoops ← CreateFeedbackLoops()
FeedbackLoops.MonitorPerformance()
FeedbackLoops.ContinuousImprovement()
DesignForScalability()
DesignForFlexibility()
Reviews ← ConductRegularReviews()
Governance ← ImplementGovernance()
EnsureCompletionOfIntegration(IntegrationGoal)
```

The ultimate goal of this Algorithm is to achieve successful integration by aligning development practices with Agile principles and DevOps practices, resulting in faster delivery of high-quality software, improved collaboration, and greater adaptability to changing market needs [11].

a) Paragraph 1: Overview: The presented algorithm, named "Integration Process," outlines a structured approach for achieving the integration of DevOps and Agile practices in a software development and deployment environment. This process is designed to streamline and automate various stages of software development and deployment. It begins by defining the integration goal using the equation:

$$\text{IntegrationGoal} = \text{DefineIntegrationGoal}()$$

This equation represents the initial step in the process, where the integration objective is set. The subsequent paragraphs will detail the core components and steps of this integration.

b) Paragraph 2: Agile and DevOps Integration: The algorithm incorporates key elements of Agile methodologies and DevOps practices to ensure efficient and collaborative software development. Cross-functional teams are formed, as indicated by the equation:

$$\text{CrossFunctionalTeams} = \text{CreateCrossFunctionalTeams}()$$

This signifies the creation of teams comprising individuals with diverse skills to promote collaborative development. Furthermore, continuous integration and deployment pipelines are implemented, which can be expressed as:

$$\text{CI/CD\_Pipelines} = \text{ImplementContinuousIntegrationAndDeployment}()$$

This step aims to automate the software build and deployment process, enhancing development efficiency.

c) Paragraph 3: Agile Framework and Iterative Development: To embrace Agile principles, an Agile framework is chosen, outlined by the equation:

$$\text{AgileFramework} = \text{ChooseAgileFramework}()$$

This ensures adherence to Agile practices such as iterative development and user story prioritization. The algorithm then delves into backlog creation and sprint planning. During sprints, the process develops, tests, deploys, and monitors user stories, which can be summarized using mathematical notation:

$$\begin{aligned}
\text{Backlog} &= \text{CreateBacklog}() \\
\text{Sprint} &= \text{PlanSprint}() \\
\text{UserStory} &= \text{Sprint.Tasks[0]} \text{ (a specific user story)} \\
&\text{Develop(UserStory)} \\
&\text{Test(UserStory)} \\
&\text{Deploy(UserStory)} \\
&\text{Monitor(UserStory)}
\end{aligned}$$

These equations represent the Agile components of the algorithm, focusing on iterative development and incremental improvement.

d) Paragraph 4: Automation and Feedback Loops: The Integration Process emphasizes automation, enabling the execution of repetitive tasks through automation tools. This can be expressed as:

$$\text{AutomationTools.Execute(Task)}$$

Additionally, the algorithm incorporates feedback loops during the monitoring and user feedback gathering phases, as shown by the equation:

$$\text{FeedbackLoops.MonitorPerformance}()$$

These feedback loops ensure ongoing improvement and alignment with the integration goal. The algorithm also addresses security, compliance, toolchain selection, training, change management, scalability, flexibility, reviews, and governance to create a comprehensive approach to software development and deployment.

Agile and DevOps may be seen as an endless cycle of operations in which tasks are performed one after the other sequentially. There are seven processes in this DevOps and Agile endless loop: planning, writing code, testing, verifying, deploying, operating, and monitoring. The goal and specifications of the software product are established during the planning phase. The development phase, which focuses on software development and code review, comes after this section. The code is often committed to code repositories and is subjected to unit tests based on build automation and integration.

The artifacts are examined and assessed in accordance with the specified criteria during the verification or construction step.Automation testing is done throughout the testing process to make sure that software artifact standards are followed. Trial versions that are linked to end consumers may also be issued during this testing period.To lower risk and assess the new program for a limited user base, for instance, canary testing might be used. The code is put into production during the deploy phase.

Automated code deployment is required, and in the event of any significant modifications, the codes should be put in production first for oversight. The configuration and administration of software applications take place during the operation stage, which follows the deployment stage. The monitoring step involves evaluating the deployed apps' performance. Data are gathered and examined for this reason, aiding in the identification of issues and the gathering of input for the software's iterative development.

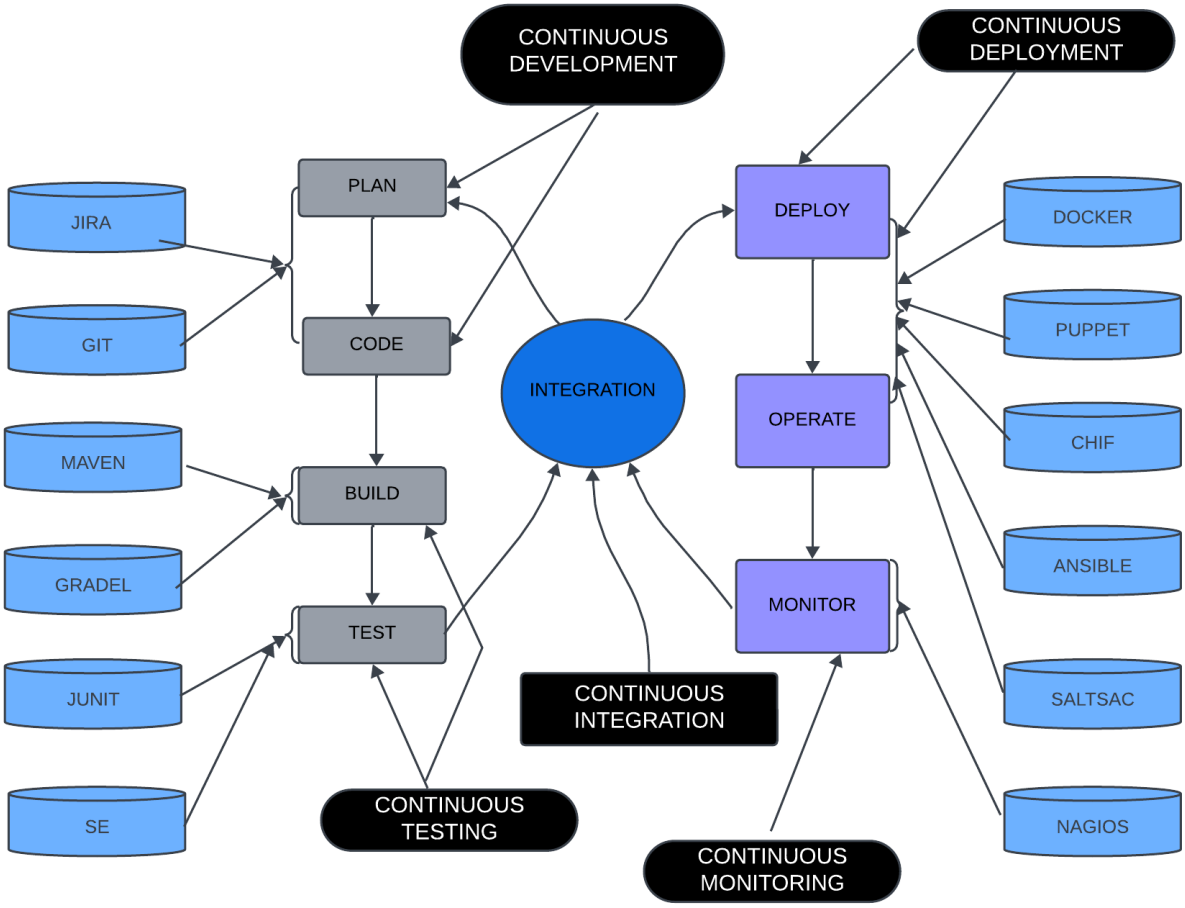


Fig. 3. DevOps & Agile life-cycle, including various phases

continuous development: Planning and coding are crucial stages in continuous development, where project goals are established and developers begin creating code for the application, using tools like JIRA and Git.

Continuous testing: Strategic test automation, as suggested by Zimmerer, is a crucial step in application development, ensuring continuous testing throughout the entire lifecycle, modifying tests according to requirements.

Continuous integration: Deployment packages are created, code is compile, tested, and adhered to, enabling quick feedback on integration issues and ensuring system functionality. Regular merging onto a shared repository is crucial for good practice.

Continuous Monitoring: It is a crucial phase in the DevOps lifecycle, where developers regularly monitor the performance of a software system. This allows for rapid analysis of crucial information, enabling the identification of the application, and ensures that the system's behavior is appropriate.

Technical Factors: The 'Technical Success Factors' category includes five subcategories.Performance engineering requiring expertise in the software life cycle. Continuous integration tools automate tasks, facilitating pipeline system cooperation.microservices used in cloud infrastructure.

Organization Success Factor: Azad and Hyrynsalmi's approach identifies three subcategories of organizational success, including hierarchy, strategic planning, and intra-organizational cooperation or communication, which are crucial for DevOps success.

Social & Culture Success Factor: DevOps success relies heavily on social and cultural factors, with team dynamics and cultural change being subcategories of these elements.

We propose that the effectiveness of DevOps would be directly impacted by organizational and technological elements in this paradigm, including hierarchy, intra-organizational communication, and strategic planning. The theory known as the technology-organization-environment (TOE) supports these two links by stating that the adoption of technological innovations in an organization is directly influenced by organizational and technical variables. Previous work on information systems and

managerial science has also experimentally validated these two correlations. We conclude from these results that cloud-based businesses may also be motivated by these two motivations. It is suggested that the impact of organizational and technological elements on success would be mitigated by social and cultural factors, such as team relationships and cultural shifts. Reasons for such moderating effects include cultural variations in distance from power, avoiding uncertainty, individualism vs collectivism, and masculinity against femininity. Tradition has also been shown to be an influence in earlier research on IT adoption. It follows from this that the influence of organizational and technological elements might differ according on the culture. Moreover, social and cultural elements may have a direct bearing on achievement.

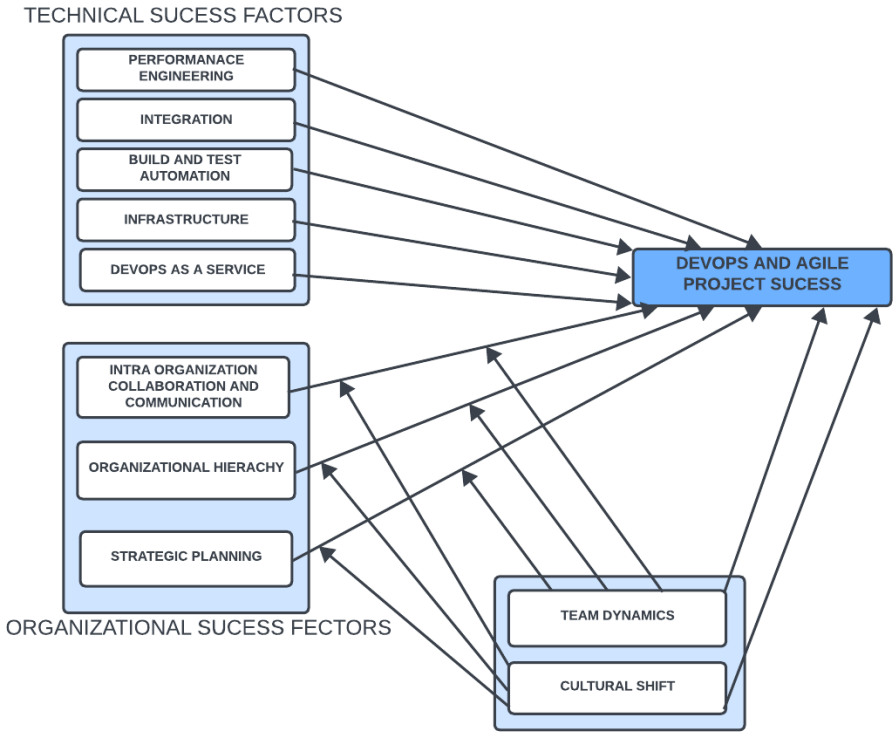


Fig. 4. The synthesized framework of critical success factors.

We do point out, however, that by gathering empirical data from DevOps-using firms, the real links may be investigated in further studies. Because the present assessment is based primarily on peer-reviewed literature, it's also possible that not all significant CSFs have previously been found. The model may also need to be revised after the field has gained more experience.The proposed architecture aims to provide a foundation for understanding DevOps CSFs and addressing company issues, enhancing the awareness of DevOps techniques among practitioners and experts. However, it requires empirical evidence from surveys and interviews with DevOps experts to validate its applicability. The framework may be modified or certain aspects removed based on empirical validation and the field's maturity. This will ensure the structure's relevance and effectiveness in the field.

Organizational decision-making is influenced by both external and internal technologies, with DevOps's entry considered a new innovation if organizational success factors are major. DevOps adoption for CSFs is influenced by size and management support, with technically-oriented aspects having greater significance than agile or plan-driven methodologies.This study aims to address the fragmented CSF literature on DevOps, a new software process paradigm, by identifying and expanding on the need for a comprehensive model.

The studies examined did not include key success criteria like training, top management support, and personal qualities, possibly due to their limited role or overemphasis on DevOps-specific features. Further research is needed to address these gaps.The study suggests that DevOps success characteristics can be used by businesses to adapt their attitudes and culture, impacting public sector evaluation, industry size, and IT application. DevOps aims to create an organizational culture that enhances collaboration, communication, and team goals, eliminating obstacles and silos.

Table 4. Critical Success Factors and Individual Factors

Category	Critical Success Factor (CSF)	Individual Factors
Technical Success Factor	Performance engineering	Performance engineering complexity, Performance regressions effects Lightweight approaches for performance engineering Software release Seamless upgrades
	Integration	a seamless interface with current tools, It is important to make the most of the limited mechanisms for quality feedback, AISD, and a deeper integration of ISD with IS. Considering the difficulties in operations and support young system, Process of continuous integration, deployment of continuous integration, and continuous integration
	Build and Test Automation	an increase in test automation, to promote change management, (Soundness) verification particular building automation tools, tests that are automatically run, test and development automation, Process automation, automation, not automating enough processes.
	Infrastructure & Deployment	The infrastructure is essential for the integration of software processes. Infrastructure aides DevOps teams in managing hardware and software concerns and ensures that software is delivered as needed. For a better execution of DevOps practices in the organization, it is necessary to have infrastructure provisioning and infrastructure development
	DevOps as a Service	DevOps approaches must include microservices. In order to sustain the life cycle process through continuous delivery and continuous release, DevOps teams collaborate. With reference to micro services, we have discovered a solution.
Organizational Success Factor	Intra-Organizational Collaboration and Communication	The success of DevOps in an organization relies on effective communication between development and operations teams, as poor communication significantly hinders the process.
	Organizational Hierarchy	DevOps experts rely on superiors for guidance and team assignments, as a team leader ensures the organization's hierarchy remains intact and aligns with the plan.
	Strategic Planning	Strategic planning entails balancing business and technical needs, taking into account resources, budgets, and goals, and making sure that delivery and development are ongoing to preserve the quality of the final result.
Social & Cultural Success Factor	Team Dynamics	Team dynamics in DevOps success are crucial for efficient communication, collaboration, and efficient software development, fostering a cohesive and secure team environment.
	Cultural Shift	Cultural shift significantly impacts DevOps performance in firms, requiring organizational culture, information sharing, and cultural diversity for team development and faster task completion.

VI. PERFORMANCE EVALUATION

Table 5. Case studies.

ID	COUNTRY	DESCRIPTION
Cs1	India	A company that develops technology solutions for the IT outsourcing services industry in areas such as web applications, mobile apps, cloud strategy, analytics and business intelligence and testing, quality assurance services and Agile project management Their research explores multiple approaches enables DevOps in Agile context.
Cs2	USA	The company specializes in information technology and provides comprehensive training for IT professionals on systems and technology upgrades. The benefits of Agile thinking for deployment work have been explored in their study.
Cs3	Canada	A multinational firm that provides consulting services to assist businesses in transforming their development teams by incorporating new methodologies and technology.They examine how Agile principles should be revised to take into account the requirements of operations teams in enterprises in their study.
Cs4	USA	Technology infrastructure based on public, private, hybrid, and multi-cloud infrastructure is provided by the cloud service provider. The differences and similarities between the two models are explored in their paper.
Cs5	Australia	An Australian company that develops software for project managers, software engineers, and other software development teams. In their newsletter, they show how the two devices can be used simultaneously for automation.
Cs6	USA	An international consulting firm specializes in assisting firms in utilizing cutting-edge technology and business management principles to gain competitive advantage and accomplish their missions. They generally cater to CEOs and provide consultation and training services. Explored throughout the essay are points of divergence and convergence.
Cs7	India	In the sphere of information and communication technology, a company that specialized in the transmission of technological information. Important findings from merging DevOps and Agile are discussed in their news release.
Cs8	USA	Adaptive cybersecurity, corporate DevOps, automation, and data-driven business are just a few of the sophisticated knowledge-creating domains in which this international information technology organization excels. Their study looks at how Agile methodologies may benefit DevOps.
Cs9	Switzerland	A company that offers appealing, scalable cloud-based solutions that primarily targets the European market. They look at the benefits of each paradigm independently in an effort to predict the advantages of using both.
Cs10	India	The SAFe methodology is used by a company that manufactures electronic educational materials, which is part of the Agile scalability paradigm. Their research looks at change-driven approaches and how SCRUM and DevOps relate to this issue.
Cs11	Germany	A global IT company is using cloud solutions, DevOps, software testing, quality assurance, artificial intelligence and big data. In their talk, they discuss software engineering problems that can be solved by combining the two models.
Cs12	UK	a consultancy company that streamlines corporate operations with the aid of the cloud, Slack, and Trello technology. They highlight how the DevOps paradigm has been established in their news release, citing the cloud and Agile methodologies as key factors.

Table 6. Details of Benefits

ID	BENEFITS	DETAILS
Bf1	Time to market	Better team communication results in faster software delivery.
Bf2	Automation	Assembly planning and manufacturing processes are highly automated to meet signal standards.
Bf3	Communication	Encourage ongoing communication between the operational team and the development team.
Bf4	Mindset and culture	Establishing cooperation amongst teams
Bf5	Planning	Services are things that need to be delivered, modified, maintained, maintained, and supported as a service, and are now added to the backlog.
Bf6	Visibility	More transparency of the release and upgrade procedures
Bf7	Risk mitigation	There is a lot of danger found in the details of any race
Bf8	Software quality	Less software bugs exist, which makes it easier to provide updates more quickly.
Bf9	Cost	People become more multi-skilled when people and activities are combined, and human resource costs will eventually go down.
Bf10	Software quality	Both functional and load tests are considered.
Bf11	Efficiency	Performance measures from mixed methods in both areas are evaluated by project managers.
Bf12	Flexibility	To remain competitive, the business has to be able to act quickly in response to constant demands for adjustment.

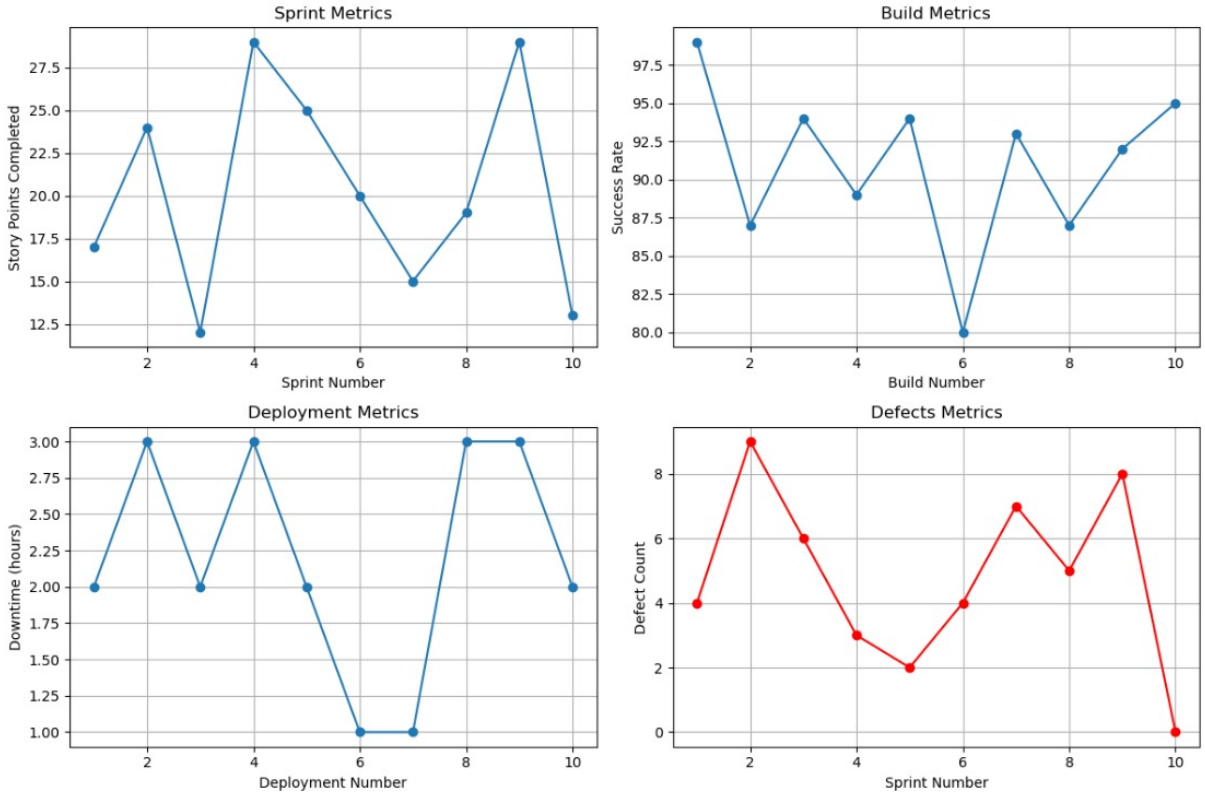
*A. **Results***

The twelve case study papers for each firm in Table 5 were thoroughly examined, and each benefit was assigned a unique identity, denoted by ”Bf” and accompanied by a number.Common themes in case studies share an acronym and are explained in detail, while Table 6 outlines the process for locating these themes in each case study. Table 6 presents the themes for each case study in alphabetical order, as provided in Table 5. We call attention to themes that appear in a number of case studies.The case studies reveal varying views on universal themes, with some highlighting positive aspects of the institutions. For instance, improved team communication leads to faster time to market in Cs1, while process integration is highlighted in Cs8. Cost reduction is also a challenge, with Cs6 focusing on multi-skilled human resources, while Cs8 is driven by increased team performance [25].

Table 7. comparative evaluation of the advantages and rankings.

Benefit	Cs1	Cs2	Cs3	Cs4	Cs5	Cs6	Cs7	Cs8	Cs9	Cs10	Cs11	Cs12	Rank
Bf1	×		×			×		×					#3
Bf2	×	×			×		×		×	×	×	×	#1
Bf3		×			×		×		×	×			#2
Bf4		×		×	×								#4
Bf5			×		×								#5
Bf6				×				×					#5
Bf7				×									#10
Bf8				×									#10
Bf9						×		×					#5
Bf10							×				×		#5
Bf11							×						#10
Bf12									×			×	#5

A comparison of the advantages that were noticed is summarized in Table 6. Mapped out are all previously determined benefits. The ”ranking” feature allows us to evaluate how important each advantage is in relation to each other and identify benefits that cross many categories.case studies as well as those that are only observed in a specific situation inside each organization.Benefits related to communication, automation, and time to market (BF1, BF2) immediately come to mind. These are the three key advantages of combining DevOps with Agile. On the other hand, certain advantages are only marginally shown.a variety of case studies, such as those on risk reduction (BF7), effectiveness (BF11), and software quality (BF8). It is more challenging to reproduce these advantages in other software firms since they are less pertinent and depend on the unique circumstances [17].



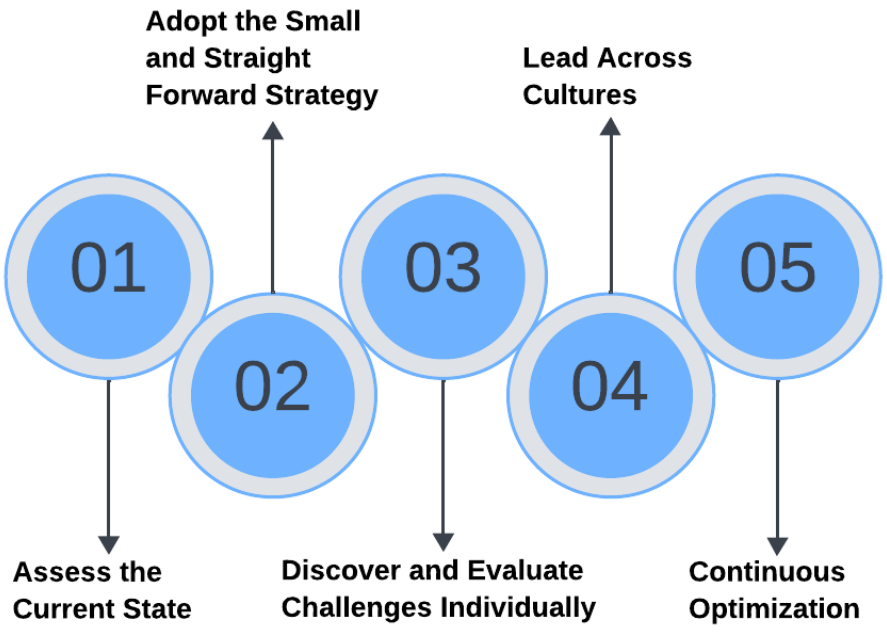


Fig. 5. DevOps and Agile Services Together

1. Assess the Current State

You may develop a road-map for transformations initiatives with the support of the clarity on the organization's existing status. The main components of the present condition are the IT service management process, leadership roles, past deployments, and cultural preparedness.

2. Adopt the Small and Straight Forward Strategy

Take it slow. Start by developing an MVP (Minimum Viable Product) that will benefit the company, its clients, and its staff via the use of efficient procedures and enabling technology.

3. Discover and Evaluate Challenges Individually

There is no issue that is too trivial to be disregarded. Utilize the most appropriate concepts and techniques as you deal with even the tiniest issues. The most well-known is automation, which lowers the likelihood of mistakes by boosting productivity and elevating employee happiness.

4. Lead Across Cultures

It's like combining two separate cultures when Agile and DevOps are together. The leaders now have a duty to facilitate and encourage this cultural shift in the two techniques. You provide your best to the process by assisting and encouraging workers to comprehend and adjust to the new adjustments.

5. Continuous Optimization

Perfect answers are impossible to find. A little room for development will always exist. It is essential to keep your edge and be ready for any unforeseen circumstances. In order to do this, it is necessary to continuously optimize software, processes, tools, and transformative activities.

You may successfully transition your company into a digital one by doing the steps listed above, and you can track your organization's development by measuring and analyzing its performance.

B. Discuss

Even though Agile and DevOps are widely accepted but distinct ideas, they may work well together and provide businesses with useful advantages. As previously said, since this is often a manual procedure, businesses have difficulties while installing and delivering new software versions. Furthermore, a large number and frequency of mistakes are produced by this method. Non-operational resources may be employed in non-production settings to decrease the frequency of issues and to boost automation and flexibility. The developer may have more control over the environment, infrastructure, and applications by combining the adoption of Agile and DevOps. The right combination of DevOps and Agile results in a more efficient collaboration and Agile process. To make model processes more meaningful and effective, this approach simplifies and automates. Shorter supply cycles, providing value to small, previously unrecognized development packages, provide a particular illustration of these benefits The importance of organizing musicians—enabling strategies to operate at a pragmatic approach to improving and eliminating conventional methods that do little to contribute to the development cycle—is likewise emphasized. Last but not least, automated testing throughout the Agile and DevOps chains makes tests more consistent and reusable across environments [9].

It is the primary reason why product delivery attempts fail. Teams that begin working together are better able to monitor how processes have changed since they were first implemented, which encourages the development of process enhancements. Cois et al. acknowledge that DevOps's capacity to maximize communication between the teams involved and the client is what makes it so unique. This makes it possible for the team to apply the ITOps paradigm, for instance, by including the operations team. Because of this, it can provide a development environment that is suitably safe. It has more promise, however, when connected to an organization's Agile teams. For instance, the marketing and sales divisions may participate in the activities related to the release delivery, enabling businesses to maximize the value of their resources while enhancing the product even more.The research also uncovered a wide range of advantages, including improved risk detection and mitigation, enhanced process visibility, and higher-quality software. When the two paradigms are combined, consolidation is encouraged, giving project managers a better understanding of the teams' work and the relationships between them. Additionally, continual client input creates value from the start of the project, minimizing the risks associated with development and operation, and iterative planning across teams facilitates adaptation in the event of changes. The combination Agile and DevOps paradigm requires collaboration between the two teams in order to accomplish the shared objectives of delivering high-quality, functioning code [20].

VII. CONCLUSIONS

According to this research, there is no inherent conflict between the Agile and DevOps concepts; rather, when correctly matched, they may be advantageous for enterprises. DevOps enhanced this system, whilst Agile introduced a rapid delivery approach in line with client demands. Adopting both approaches is, in this sense, a choice that often yields excellent results. Along with being a good match, they also support businesses in adapting to team changes. A team may find it difficult to make changes to its approach and strategy without much help. In order to prevent isolated silos that hinder collaborative work, businesses must handle this topic in a cross-cutting manner inside the company. DevOps establishes an atmosphere

that is helpful for managing these processes with good communication, whereas Agile makes room for more agile work with partial deliverable [3].

This work makes significant contributions to theory and practice. Through the use of many case studies of software businesses operating in foreign markets, this research has made it possible to identify a number of theoretical advantages of combining the adoption of both paradigms. The analysis enables us to investigate the relative importance of each of the 12 advantages that are identified. From an implementation standpoint, the advantages mentioned apply to businesses who have already implemented Agile and DevOps independently and have not yet moved to integrate the two approaches. Although the two models are not mutually exclusive, combining them may enhance their effects on companies, according to the study's results [10].

A. Limitations & Future Research Directions

There are many flaws in this analysis. To begin with, because the data generated in this study come from secondary sources, we cannot go into detail and use interviews to provide evidence for the use of both models around Furthermore, because case studies come from for-profit industries, it is difficult for them to accurately identify specific community groups or to provide a completely objective view of the benefits to organizations but our study internally and post-validation procedures were included to reduce the potential for bias. It is recommended that future research on the benefits of integrating DevOps and Agile incorporate a scientific perspective with a business perspective. To this end, the relevant literature can be thoroughly searched. Furthermore, due to the qualitative approach taken, we are unable to definitively identify and quantify any benefits. It is suggested that future research should be a quantitative study with larger data sets to demonstrate the advantages of using both models simultaneously, taking into account different business departments of the organisations. Since some benefits may be easier to achieve for organizations with lower system maturity, it will take Agile and DevOps implementation maturing in these organizations and, as a result, it will also be worth monitoring as it relates to the benefits received [4].

REFERENCES

[1] Fernando Almeida, Jorge Simoes, and Sergio Francisco Ferreira Lopes. Exploring the benefits of combining devops and agile. *Future Internet*, 14:63, 02 2022.

[2] Len Bass, Ingo Weber, and Liming Zhu. *DevOps: A Software Architect's Perspective*. Addison-Wesley Professional, 2015.

[3] Kent Beck et al. Manifesto for agile software development, 2001. Agile Manifesto Official Website.

[4] Boyuan Chen. Improving the software logging practices in devops. pages 194–197, 05 2019.

[5] Boyuan Chen and Jun Cui. Improving the software logging practices in devops. 07 2021.

[6] Kazuyuki Dobashi and Tomoyuki Kono. Agile devops practices: Perspectives from japan. *International Journal of Information Technology Project Management (IJITPM)*, 10(1):1–18, 2019.

[7] Leonardo Ferreira Leite, Carla Rocha, Fabio Kon, Dejan Milojicic, and Paulo Meirelles. A survey of devops concepts and challenges, 09 2019.

[8] Nicole Forsgren, Jez Humble, and Gene Kim. Accelerate: The science of lean software and devops: Building and scaling high performing technology organizations, 2018. Goodreads.

[9] Nicole Forsgren, Jez Humble, and Gene Kim. The devops handbook: How to create world-class agility, reliability, and security in technology organizations. *IEEE Transactions on Engineering Management*, 65(1):1–3, 2018.

[10] Martin Fowler. Continuous delivery. Technical Report 1, ThoughtWorks, 2013.

[11] Martin Fowler, Kent Beck, et al. Agile manifesto, 2001. Agile Manifesto Official Website.

[12] Nikhil Govil, Mayank Saurakhia, Pradyumna Agnihotri, Sachin Shukla, and Shivam Agarwal. Analyzing the behaviour of applying agile methodologies devops culture in e-commerce web application. In *2020 4th International Conference on Trends in Electronics and Informatics (ICOEI)(48184)*, pages 899–902, 2020.

[13] Nancy W. Grady, Jason A. Payne, and Huntley Parker. Agile big data analytics: Analyticsops for data science. In *2017 IEEE International Conference on Big Data (Big Data)*, pages 2331–2339, 2017.

[14] Jez Humble and David Farley. *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation*. Addison-Wesley Professional, 2010.

[15] Watts S. Humphrey. The agile-devops approach to software development, 2010. Technical Note.

[16] J.A.V.M.K Jayakody and Janaka Wijayanayake. Challenges for adopting devops in information technology projects. pages 203–210, 09 2021.

[17] Gene Kim, Kevin Behr, and George Spafford. *The Phoenix Project: A Novel about IT, DevOps, and Helping Your Business Win*. IT Revolution Press, 2018.

[18] Gene Kim, Patrick Debois, John Willis, and Jez Humble. *The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations*. IT Revolution Press, 2016.

[19] Dean Leffingwell. Scaling software agility: Best practices for large enterprises. *Addison-Wesley Professional*, 2007.

[20] Mika Lehtonen and Marko Paukkeri. Agile and devops as a continuum: An experience report from a large-scale organization. In *2016 IEEE 9th International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, pages 304–313. IEEE, 2016.

[21] Lucy Ellen Lwakatare, Pasi Kuvaja, and Markku Oivo. Relationship of devops to agile, lean and continuous deployment. pages 399–415, 11 2016.

[22] Shah Murtaza Rashid Al Masud, Md. Masnun, Afia Sultana, Anamika Sultana, Fahad Ahmed, and Nasima Begum. Devops enabled agile: Combining agile and devops methodologies for software development. *International Journal of Advanced Computer Science and Applications*, 13(11), 2022.

[23] Sikender Mohsienuddin Mohammad. Devops automation and agile methodology. *SSRN Electronic Journal*, 5:946–949, 08 2017.

[24] Necmettin Ozkan, Ayça Kolukısa, İsmail Filiz, Enis Özer, and Burak Gören. Harmonizing it frameworks and agile methods: Challenges and solutions for the case of cobit and scrum. 09 2020.

[25] Morgan Rowse and Jason Cohen. A survey of devops in the south african software context. 01 2021.

[26] Ken Schwaber. Agile project management with scrum. *Microsoft Corporation*, 2002.

[27] Sandeepak Singh and Vanshika Srivastava. Agile devops and system testing the evolving scene. September 2022.

[28] Sandeepak Singh and Vanshika Srivastava. Devops and agile methods integrated software configuration management experience, 06 2023.

[29] Mariela Stoyanova. Smart concept for project management – transition to devops. *Knowledge International Journal*, 34:93–97, 10 2019.

[30] Anna Wiedemann, Nicole Forsgren, Manuel Wiesche, Heiko Gewald, and Helmut Krcmar. The devops phenomenon: An executive crash course. *Queue*, 17:93–112, 04 2019.