A Proposed Model (DOA) for DevOps Practice in Agile Development

Anum Bahar
Faculty of Computing
Riphah International University,
Lahore, Pakistan

Muhammad Yaseen
Faculty of Computing
Riphah International University
Lahore, Pakistan

Amara Parveen
Faculty of Computing
Riphah International University
Lahore, Pakistan

Muhammad Asif Nauman
Faculty of Computing
Riphah International University,
Lahore, Pakistan

ABSTRACT
In the modern world of software development, there is a growing trend that incorporates DevOps practice into agile software development methodology as a critical factor in improving software delivery processes. Moreover, this paper presents the DevOps in Agile (DOA) model as a possible solution for unification between development and operational teams and as a way to shorten the cycle time and increase the quality of software products. Thus, the DOA model also highlights communication, integration, and automation as key success factor. This model aims to combine the best of both the Agile concept and the DevOps to create a model that is flexible enough to adapt to market forces while at the same time meeting high standards of software quality. The paper identifies the issue of culture and tool choice as key themes and discusses how they may relate to the integration of these methodologies, then offers a systematic process by which organizations may successfully implement the DOA model.

Keywords
Agile, DevOps, Software Development, integration, Testing automation, DOA model

1. INTRODUCTION
Recently, there has been an integration of DevOps practice and Agile methodologies have become a new methodology in the software development life cycle that serves as a boost way of developing software. DevOps integration in the agile environment also solves the problems that arise from having development and operation as two separate terms in a project. DevOps and agile are two terms or practices frequently used in today's software development, and their engagement is intended to enhance the quality, velocity, and productivity of software development. Although Agile highlights the concept of iterative and incremental development and the ability to incorporate changing user needs, on the other hand, DevOps is more focused on the aspect of automation and collaboration between the development and operations teams. In this paper, DOA (DevOps in Agile) model has been proposed which should define its primary DevOps practices and principles of agile software development.

1.1 Agile
Agile software development is a methodology used in information systems (IS) and software development concentrates on the user and is designed to be developed in an incremental and iterative style. The term Agile software development methodology dates back to 2001. Agile was developed as a process model in response to the problems of the previous approach called the waterfall model of software development which was often planned-driven and was described as a predictive model [1]. Agile organizations are networked teams with short learning cycles focusing on a clear community vision and delegating decision-making to the closest teams, unlike traditional hierarchical structures. Agile methodologies prioritize adaptation over innovation for management and efficiency improvement in the networked knowledge-based economy and are developed and adopted in current research work [2]. Agile methodology is a highly efficient software development approach based on four core principles: individual interactions, working software, customer collaboration, and change response. It can be adapted based on project size, requirements, and factors like DAD, SAFe, XP, and Scrum, which are easy to use and highly adopted [3]. Agile software development offers several advantages over traditional software development methods, including limited documentation, intensive customer collaboration, rapid delivery, and continuous alignment with business goals or requirements [4]. As the need for fast delivery and the ability to adapt to market needs in organizations is realized, Agile has been embraced by organizations not only in the IT sector but also in other industries [5].

1.2 DevOps Practices
In 2009, Patrick Debois founded the first DevOps Days conference in Ghent, Belgium. This meeting later expanded to other countries. DevOps is a culture that fosters collaboration between development and operational teams to deploy code for faster, automated production. It involves all developers in the production phase and system engineers, administrators, and security professionals. This practice increases an organization's speed, improves customer service, and strengthens market competition [6]. Companies are implementing DevOps to enhance software delivery processes, reducing gaps between operation and development teams once successfully implemented and adopted within software organizations [7]. DevOps is a framework that comprises four dimensions, i.e. Collaboration, automation, measurement, and monitoring. DevOps is an extension of the agile software development method that emphasizes continuous software delivery and continuous integration with automation reducing latency and improving communication, collaboration, and reliability [8]. DevOps

recover software delivery, improves communication, faster time to market, responsiveness, and reduced interruption. It updates product development, increases responsibility, and fosters broader skillsets and roles [9]. DevOps is a combination of people, process, and tooling, focusing on Continuous Integration and Continuous Delivery (CI/CD) for faster and more secure code delivery. It encourages open-mindedness, predictability, cross-skill training, and shared tasks. DevOps tools play a critical role in integrating different environments and tools, enabling changes to propagate across different environments. Organizations often face challenges in selecting appropriate tools and frameworks to automate and integrate different systems [10]. The DevOps Lifecycle is a continuous cycle that optimizes product plans, design, testing, and repairs by combining development and operations teams. It fosters communication and automation, ensuring efficiency from planning to delivery, and enhancing effectiveness [11].

The DevOps methodology involves a simple lifecycle including Continuous Development, Continuous Integrations|, Continuous Testing, and Continuous Monitoring [12]. DevOps fosters a culture of continuous improvement, resulting in shorter iterative cycles and shorter update cycles. This reduces time to market and simplifies flaw detection. Teams should have open tools and guidelines for effective implementation [13]. The technique enables the rapid production of high-quality and reliable software through a specific cycle illustrated in Figure 1.

Figure 1: The Life Cycle of DevOps

The success of DevOps relies on understanding the DevOps lifecycle. So,

***Plan*:** At this stage establish clear objectives for development and operations teams, foster collaboration for effective planning, and conduct risk assessments to identify potential challenges and risks in the development procedure.

Code and Build: In this stage code development involves collaborative writing, continuous integration, automated testing, and peer review to maintain quality and identify issues early in the development process. Software developers release code in the groups and use products such as Git for easy versioning of the code. Automated CI tools are used to compile and execute test cases for the code.

***Test*:** At this stage continuous testing including unit tests, integration tests, and acceptance tests helps maintain code quality and identify bugs early. Automated testing tools specifically software testing tools check code modifications for potential functionality breakage and ensure code integrity.

Deploy: In this stage Continuous Deployment (CD) and Infrastructure as Code (IaC) are automated deployment procedures that use only tested code changes in web application and software production environments. CD involves testing code changes while IaC handles infrastructure configurations as code, automating provisioning and management.

Operate: In this stage application monitoring and logging are involved. Application monitoring involves real-time tools gathering data and creating logs for problem-solving purposes. Incident management is achieved through alerts, allowing fast response without involving multiple people or systems. Feedback loops are used for monitoring and planning future development and operations team improvements related to incidents. This process ensures efficient and effective communication within the system.

Monitoring: In this stage performance monitoring includes daily inspection of the web application for optimal operation, server status, and other parameters. User experience monitoring focuses on user engagement and satisfaction while security monitoring ensures data and web application protection.

DevOps supported Tools: DevOps utilizes various tools and technologies to enhance communication, integration, and optimization, supporting process integration and project management, all of which are crucial for effective DevOps practices.

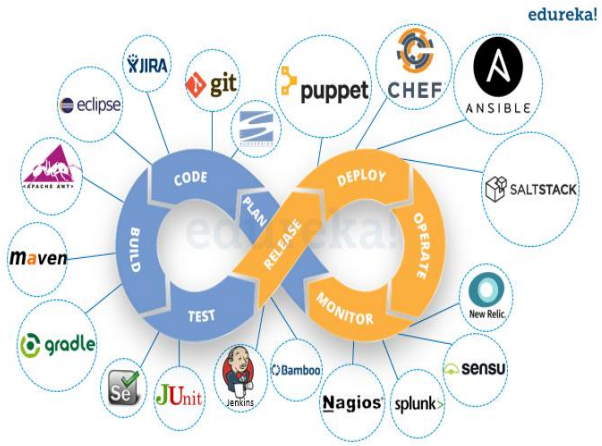

Figure 2: DevOps ToolChain

Figure 2 shows the different tools of DevOps and these tools are discussed further:

Version Control Systems Tools: Git tool is a version control system that allows collaboration among developers and making it crucial for managing code changes. The GitHub/GitLab offers hosting for Git repositories, issue tracking, CI/CD, and project management, facilitating collaboration among several members.

CI/CD Tools*:* In CI/CD Jenkins and CircleCI tools are used. The Jenkins tool is an open-source automation server for building, deploying, and automating software development processes. It simplifies the configuration of CI/CD workflows with its plugins. CircleCI is a cloud-based CI/CD tool that enables quick application building, testing, and deployment.

Containerization Tools: Docker is a container-based tool for developers to generate, deploy, and run applications while Kubernetes is a management tool for containerized applications, enabling automated deployments and scaling.

Infrastructure as Code (IaC) Tools*:* Terraform is an open-source tool for provisioning infrastructure using declarative configuration language, while Ansible automates application deployment and ensures consistent environment setup.

The Monitoring and Logging Tools: Prometheus is an

open-source monitoring toolkit for Kubernetes-based environments, while the ELK Stack, comprising Elasticsearch, Logstash, and Kibana, analyzes and visualizes real-time log data for app performance and user behavior.

Project Management Tools: JIRA is a widely used project management tool that supports Agile methodologies like Scrum and Kanban, aiding software development teams in project planning, tracking, and management. Trello is a project management visualization software that aids task setting and teamwork development.

1.3 DevOps in Agile Context

DevOps practices within Agile software development is a radical evolution for organizations to change the way they engineer software. Bringing those two together, by accessing the iterations and discipline that Agile inherits with collaboration principles of DevOps as well as an automation-centric approach, it should enable organizations to have a more coherent lifecycle [5]. Agile methodology enhances user-developer communication, while DevOps is a team-based approach that divides silos into teams. While they share similarities, they are not identical. Some argue that DevOps is superior to agile methodology. Understanding the nuances of both methodologies is crucial to eliminate ambiguity [14]. Agile and DevOps are important methodologies for successful software development. Twenty years ago, the focus was on development, implementation, and integrated IT [3]. The integration of DevOps with Agile and other software development techniques can transform software engineering practices, enabling organizations to overcome cultural barriers, simplify processes, and encourage teams. This approach fosters innovation, reliability, and efficiency, enabling rapid market response while meeting high-quality requirements. The implementation of DevOps and agile methodologies is crucial for continuous enhancement and stakeholder satisfaction. Combining DevOps and Agile strengths can lead to organizational success [15].

The integration of a DevOps culture into agile contexts enhances the alignment between development and operations enabling teams to work together efficiently and deliver high-quality software. According to [16], the integration of DevOps into Agile practices fosters continuous feedback loops, a crucial element for iterative improvement and innovation

The integration of agile software development methodology and DevOps practices is becoming a crucial strategy for organizations to enhance their software development capabilities.

2. AIM AND OBJECTIVE

The main objective of this research article is to introduce a new model applying the DevOps approach to Agile software development known as the DevOps in Agile (DOA). This model is mainly meant to increase communications and integration between development and IT operations teams to reduce cycle time and increase the quality of the final software product. Through the flexible integration of Agile and DevOps methodologies, the DOA model aims at filling what it considers the existing gaps in software development practices more so in CI/CD and automated testing. The study will explore the implications of this model for organizations seeking an integrated approach to software development and operation to enhance and maintain an innovative culture

3. RELATED WORK

Modern approaches to software development have divided them into two primary categories Agile and DevOps. Agile is centered on iterative development, customer interaction, and responding to change, and DevOps is based on the efficient shift between the development and deployment process for more frequent and effective deliveries [3]. The combined use of these two models has been acknowledged as one of the key success factors for organizations that seek to improve the efficiency and quality of their software delivery [17]. The studies presented in recent research indicate that to manage the complexity of software projects, it is crucial to integrate Agile and DevOps models for project delivery in competitive markets [18]. In traditional Agile environments, various problems could be encountered like lack of communication between teams working on different modules and slow feedback, which can be addressed with the help of DevOps practices, using automation, integration, and feedback [19]. DevOps, a 2009 concept, focuses on enhancing software development methodologies by fostering a relationship between development and IT operations teams. Successful DevOps results in increased deployment frequency, shorter change lead times, improved recovery times, and increased organizational resilience. [20] Implementing DevOps practices in an Agile software development environment can be challenging due to cultural resistance, tool choice, communication barriers, skills gaps, and inconsistencies between theory and practice. Enablers of integration difficulty include management support, team cohesiveness, and learning. These factors enhance the chances of success in integration. Research shows that integrating DevOps practices with agile approaches can positively impact project performance. Therefore, it is essential to identify and address these challenges to ensure successful integration and successful project outcomes [21]. To overcome challenges, focus on critical success factors and top management support for DevOps practices in Agile software development. Provide resources and promote a culture of change. Improve collaboration between development and operations by creating cross-functional teams. Adequate communication and organizational improvements such as process automation, aligned objectives, effective communication, and KPIs will ensure goals and progress [22]. However, DevOps practices in Agile software development to enhance the software quality by emphasizing the value for customers, collaboration, automation, and optimization. The integration of both frameworks enables organizations to get the job done faster, at a cheaper price, and in better quality since the practices do not encourage the creation of silos [23]. Additionally, an empirical assessment of DevOps frameworks revealed that the introduction of the DevOps culture can result in notable enhancements in both software quality and team productivity [24]. The proposed DOA model emerges from this understanding of the present world as a fast-paced digital environment that requires organizations to create collaborative learning cultures. DOA model integrates DevOps into Agile and combines them into a single framework to help make the environment better for both the development and operations teams to facilitate optimal functionality and efficient delivery of

quality software within the shortest time possible and improvement of the customer feedback [25].

This research will, therefore, fill this gap by offering a framework for the adoption of DevOps in Agile software development, enhancing its applicability for organizations willing to make the transition.

4. RESEARCH QUESTIONS

The work presented in this paper is based on the following seven research questions:

RQ1: What are the critical challenges of DevOps in agile software development?

RQ2: What are the critical challenges in the real industrial practice of DevOps in the context of agile software development?

RQ3: What are the critical success factors and best practices of DevOps in agile software development?

RQ4: What are the critical success factors and best practices in the real industrial practice of DevOps in the context of agile software development?

RQ5: To design a robust model for DevOps practice in agile software development based on identified best practices.

RQ6: To Implement and validate the proposed model for DevOps practice in agile software development via case studies.

5. RESEARCH METHODOLOGY

The methodology of the proposed model (DOA) DevOps practice in Agile software development consists of the following three phases. The methodology of model is designed in previous studies [26] [27] [28]. SLR will be conducted for the identification of challenges and best practices for DevOps in Agile [29] [30].

Phase#1: SLR will be conducted for data collection.

Phase#2: Empirical studies will be conducted to validate the result of SLR and to find the practices for the mentioned factors.

Phase#3: For evaluation and validation of DOA, a Case study will be conducted. To explain the aforesaid three phases the following subsections are added.

DOA development life cycle is shown in Figure 3.

5.1 Collection of Data and its analysis:

5.1.1. Critical success factors (CSFs)

These are critical factors when adopting DevOps within Agile environments. They include communication, integration, and collaboration, which are all key aspects that boost team performance and project success.

5.1.2 Critical Challenges (Cs)

Critical Challenges (Cs) are challenges and risks that can impede Agile development, including lack of training, change resistance, and miscommunication between development and operations teams which can impede the project and increase costs.

5.1.3 Best Practices

The activities will be outlined in a set of best practices to ensure effective implementation of CSFs and help organizations make recommendations for DevOps implementation in organizations adopting Agile frameworks. Studies in the software industry.

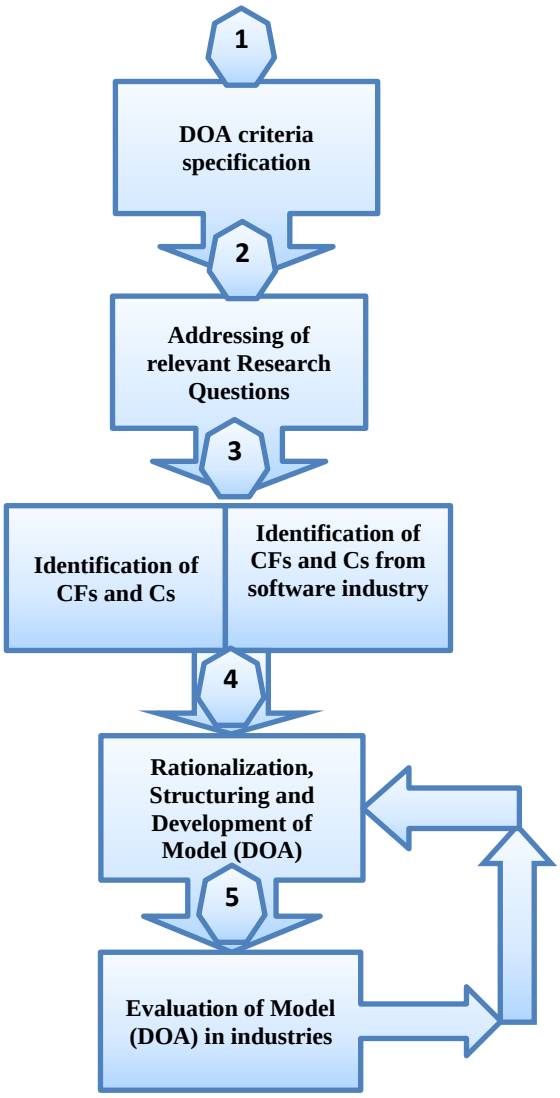

Figure 3: DOA Development Cycle Activities

The Systematic Literature Review (SLR) will be useful in finding out the kind of critical success factors (CSFs) and critical Challenges (Cs) that are linked to the intersection of DevOps and Agile. In contrast to the more traditional approach to conducting literature reviews. The systematic approach of the SLR enables efficient data extraction and analysis in line with our research questions. In this method, it is possible to capture all the factors that affect success and failure in these environments comprehensively. The identified factors will be analyzed from various perspectives, including project scopes, involved teams, and software types, as the fundamental factors causing success or failure may vary depending on contextual factors. An industry expert-based survey will be conducted with a panel of industry specialists to supplement the SLR findings, with the objectives of:

- Justification of the findings with the studies included in the SLR, to affirm the practical applicability of our recommendations.
- Identifying other CSFs and CRs that have not been explored in other research studies conducted in other organizations.
- Gathering best practices and lessons learned concerning the best ways of enhancing the implementation of the identified CSFs and CRs.

5.2 DOA Model Development

The DOA model will be developed through five stages, each ensuring a systematic approach to building a strong framework. The proposed model's general framework is illustrated in the figure 4 below.

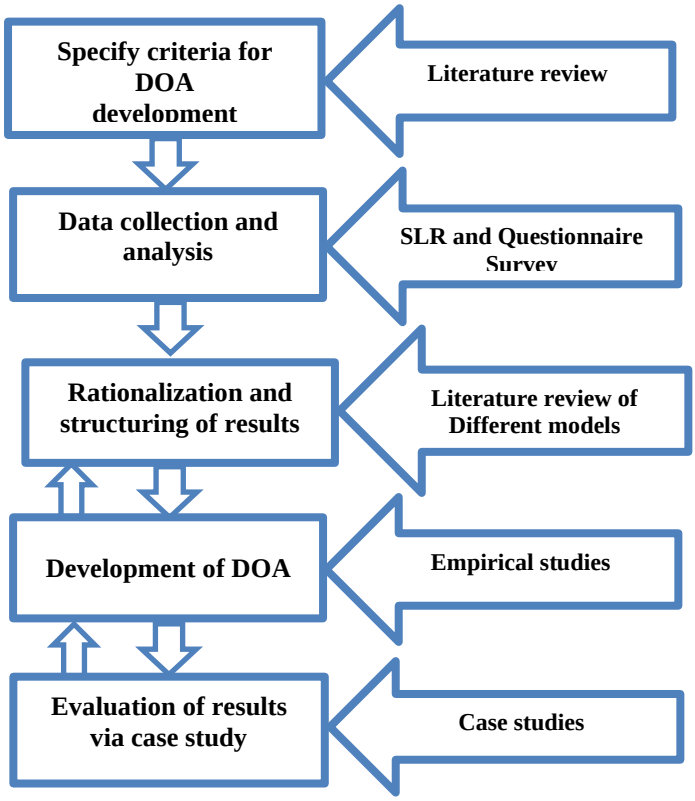

Figure 4: DOA Model Development

Assessment Criteria: The development and assessment of the DOA Model is based on two specific criteria.

5.2.1 User Satisfactions

The DOA model prioritizes user satisfaction by ensuring it is intuitive and aligns with operational needs. Feedback on features, navigation, and functionality is collected to create a model that meets user's requirements and enhances their workflow and productivity. This approach ensures the model's success and user satisfaction.

5.2.2 Usability

The DOA model aims to be user-friendly and easy to implement, focusing on simplicity and clarity. It will be designed with clear documentation, user guides, and training resources to facilitate understanding. The goal is to integrate seamlessly into existing processes, reducing the need for extensive training or resource allocation.

The study involves data collection, analysis, rationalization of findings, derivation of empirical models, and validation with case study references.

5.3 Structure of DOA Model

As it is mentioned below, the DOA model has been put forward to offer a structure to address the implementation of DevOps principles in Agile software development. Its structure is based on three primary components:

1. DOA Model Level
2. Factors CSFs and CRs at each level
3. Implementations Practices and Solutions

The division of CSFs and CRs into different levels helps define the interconnection between factors and helps organizations understand their maturity evolution. Organizations must follow practices related to each CSF and CR under that particular level similar to DevOps and Agile methodologies.

Figure 5 details the process of developing a DevOps Agile (DOA) model, based on literature review and expert opinions.

5.4 DOA Evaluation

The proposed DevOps in Agile (DOA) model will be assessed through industrial case studies of up to five organizations using DevOps practices in the Agile software development context. To gain a broader view of the model's effectiveness and feasibility, various data collection methods will be employed. These include faculty focus groups, structured interviews with primary stakeholders, and a self-administered questionnaire administered to a larger user population. The goal is to identify how effectively the model supports the Continuous Service Frameworks (CSFs), manages the Continuous Release Requests (CRs), and enhances development and operations team collaboration. The evaluation insights will not only enrich the improvements of the DOA model but also enhance the shared knowledge of the correct approaches to DevOps implementation in the Agile software development context. Real-life feedback collected through interviews or surveys can be incorporated to further improve the quality and efficiency of organizations seeking to enhance software delivery processes. This mixed-methods approach ensures a broader view of the model and its inadequacies, enhancing the shared knowledge of correct DevOps implementation approaches in the Agile software development context.

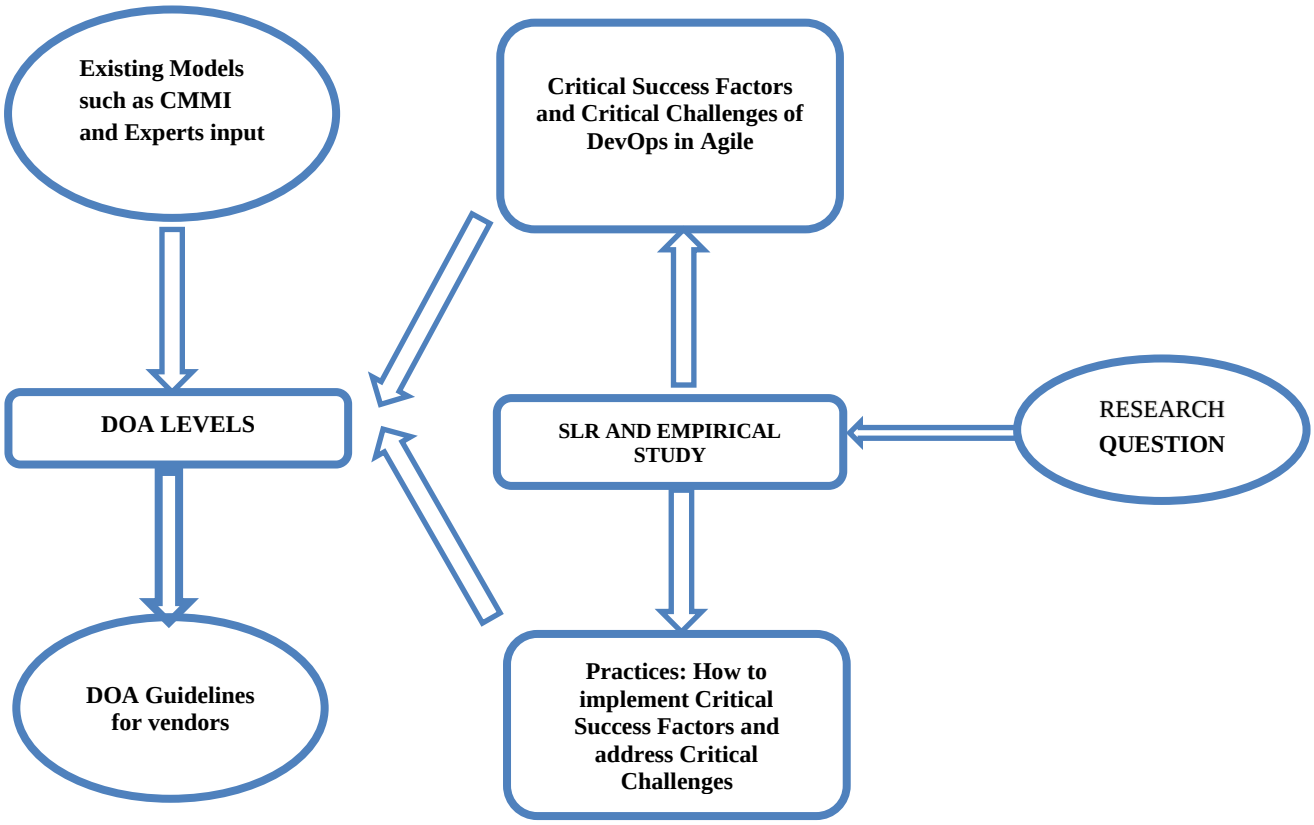

Figure 5. DOA Structure

6. RESEARCH MILESTONES ACHIEVED

Now complete the following research activities and these activities are:

- Problem identifications and Objective formulation
- Specification of Research Question
- Conducting a Systematic Literature Review (SLR)
- Selection of research methodology
- Defining DOA Model
- Evaluation method Selection

7. PRELIMINARY RESULTS

A Systematic Literature Review (SLR) has recently commenced as part of an ongoing research effort to investigate the challenges and best practices for implementing DevOps in Agile. The initial stage of this review has involved examining many research papers to identify the primary challenges faced by organizations and the best practices suggested in the literature. These early findings form the basis for additional analysis and refinement as the study continues. The integration of DevOps with Agile methodologies seeks to improve the speed of software delivery, enhance collaboration, and boost operational efficiency. Nevertheless, despite its best practices, incorporating DevOps into Agile frameworks presents several challenges. Many organizations tool with aligning processes, managing automation complexity, and navigating the cultural changes necessary for full DevOps adoption. The literature underscores significant gaps in seamlessly merging Agile and DevOps, highlighting the need for structured approaches to address challenges and implement industry-recommended practices. As the SLR is ongoing, researchers are currently evaluating existing studies to precise key challenges encountered by organizations integrating DevOps within Agile environments. Through the assessment of multiple scholarly sources, initial insights have been gathered regarding major challenges and corresponding best practices that organizations can employ to streamline their DevOps processes. The current findings offer preliminary perspectives on existing gaps, industry recommendations, and potential strategies to enhance Agile-DevOps synergy. However, additional analysis and validation are necessary to refine these results.

The table 1 below summarizes the primary challenges and best practices identified in the literature.

Table 1: List of Initial Challenges and best Practices

Challenges	Success Factors / Best Practices
Inefficient collaboration between Dev and Ops teams. [2] [10]	• Encourage cross-functional teams • Joint responsibilities • Frequent cross-team meetings.
Challenges of integration between Agile and DevOps workflow [5] [25] [16].	• Create CI/CD pipelines • Have automated feedback loops in place • Facilitate compatibility of Agile and DevOps tools.
Frequent releases causing system instability [1] [21] [23]	• Use feature toggles • Canary releases, • Blue-green deployments, • Automated rollback.
Rapid deployment processes raising security issues [9] [10]	• Embed DevSecOps practices • Pperform continuous security testing • Impose access control policies
Security challenges [11] [22] [13]	• Integrate security into each step of the DevOps pipeline (DevSecOps). • Automate security testing and include compliance gates in the CI/CD pipeline.
Difficulty to keep Agile sprint cycles with DevOps. [7] [15].	• Employ value stream mapping • Iterative enhancements, and automated sprint retrospectives
Toolchain complexity in DOA model [5] [12]	• Simplify toolchain management by embracing integrated DevOps platforms
Resistance to cultural change [6] [14] [23]	• Provide continuous learning and upskilling programs to bridge the knowledge gap

These best practices and challenges were identified from different research papers, serving as a basis for the suggested DevOps model in Agile. Due to the ongoing SLR, more analysis will be performed later to refine and affirm these findings to ensure practical applicability in real-world settings.

As the research continues, further insights will be added to improve the understanding of Agile-DevOps integration. Empirical research and case studies will be used to validate these findings and suggest a complete framework that covers areas of existing gaps and improves DevOps adoption for Agile environments. The ultimate results will deliver actionable suggestions for organizations willing to better their DevOps strategies in Agile frameworks.

8. CONCLUSION AND FUTURE WORK

DOA (DevOps in Agile) model integrates principles of DevOps into Agile development of software. This model presents an explicit approach to address challenges of software development in changing environments. It contributes to better coordination between development and operations teams, which results in better communication, automation, and continuous integration. This, consequently, results in better efficiency and quality of the software. DOA model reduces the complexity in delivering software using continuous feedback, reducing the deployment time, and facilitating easier modifications in requirements. This explicit approach not only mitigates operational lag but also supports the culture of continuous improvement and innovation within an organization. Follow-up studies will examine how to implement the DOA model based on actual studies, industry trends, and software development team feedback. We will examine how the model impacts software quality, deployment rate, and team efficiency in order to enhance its effectiveness. We will examine more sophisticated automation tools, AI-based testing, and security integration (DevSecOps) to make the model usable in other areas. The utilization of cloud-native DevOps practices and microservices architectures will be investigated to determine if they scale and remain stable. The study will also assess how organizations can enhance their Agile-DevOps adoption by addressing cultural resistance, tool selection issues, and skill gaps. Through continuous improvement and verification of the DOA model, this research hopes to contribute to the development of DevOps and Agile practices so that organizations can construct sustainable, high-quality software development practices.

9. REFERENCES

[1] A. M. S. Khan, A. W. Khan, F. Khan, M. A. Khan, and T. K. Whangbo, “Critical Challenges to Adopt DevOps Culture in Software Organizations: A Systematic Review,” IEEE Access, vol. 10, pp. 14339–14349, 2022, doi: https://doi.org/10.1109/ACCESS.2022.3145970.

[2] A. Omonije, “Agile Methodology: A Comprehensive Impact on Modern Business Operations,” International Journal of Science and Research, vol. 13, no. 2, 2024, doi: https://doi.org/10.21275/SR24130104148.

[3] S. M. R. A. Masud, Md. Masnun, A. Sultana, A. Sultana, F. Ahmed, and N. Begum, “DevOps Enabled Agile: Combining Agile and DevOps Methodologies for Software Development,” *Int. J. Adv. Comput. Sci. Appl.*, vol. 13, no. 11, 2022, doi 10.14569/ijacsa.2022.0131131.

[4] J. Beard, V. C. Storey, B. M. Samuel, and Fereshta Islamzada, “Agile Development: The Promise, the Reality, the Opportunity,” ResearchGate, pp. 1–20, May 2024, Accessed: Dec. 10, 2024.M. Choras *et al.*,

"Measuring and Improving Agile Processes in a Small-Size Software Development Company," *IEEE Access*, vol. 8, pp. 78452–78466, 2020, doi: 10.1109/access.2020.2990117.

[5] A. Nalamwar and P. Sirsat, "An Empirical Study on DevOps," vol. 3, no. 11, 2019.

[6] N. Azad, "Understanding DevOps critical successfactors and organizational practices," IEEE Xplore, May 01,2022.https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9808814, Doi: 10.1145/3524614.3528627.

[7] Harshali Rohit Kadaskar, "UNLEASHING THEPOWER OF DEVOPS IN SOFTWARE DEVELOPMENT," *Int. J. Sci. Res. Mod. Sci. Technol.*, vol. 3, no. 3, pp. 01–07, Mar. 2024, doi: 10.59828/ijsrmst.v3i3.185.

[8] M. Moeez *et al.*, "Comprehensive Analysis of DevOps: Integration, Automation, Collaboration, and Continuous Delivery," *Bull. Bus. Econ. BBE*, vol. 13, no. 1, Mar. 2024, doi: 10.61506/01.00253.

[9] M. Gokarna, "DevOps phases across Software Development Lifecycle," Jan. 06, 2021, *Institute of Electrical and Electronics Engineers (IEEE)*. doi: 10.36227/techrxiv.13207796.v2.

[10] L. Gren and P. Ralph, "What makes effective leadership in agile software development teams?," in *Proceedings of the 44th International Conference on Software Engineering*, Pittsburgh Pennsylvania: ACM, May 2022. doi: 10.1145/3510003.3510100.

[11] O. Justin Onyarin, "A Complete Guide to DevOps Best Practices," Mar. 2022, doi: 10.5281/ZENODO.6376787.

[12] J. A. Tsapa, "Integrating DevOps with Agile and other Software Development Methodologies," *Open Access*, vol. 3, no. 3, 2022.

[13] I. Indurangala, "Coalescing DevOps with Agile Methodologies in IT Industry; Insights from Case Studies".

[14] F. Bildirici, "DEVOPS AND AGILE METHODS INTEGRATED SOFTWARE CONFIGURATION MANAGEMENT: HAVELSAN EXPERIENCE".

[15] D. Nyale, "Unravelling DevOps Agile Methodologies: A Comprehensive Review of Recent Research," *Int. J. Res. Appl. Sci. Eng. Technol.*, vol. 11, no. 11, pp. 2012–2021, Nov. 2023, doi: 10.22214/ijraset.2023.56986.

[16] P. Raj and D. P. Sinha, "Project Management In Era Of Agile And Devops Methodologies," vol. 9, no. 01, 2020.

[17] G. Bou Ghantous and A. Q. Gill, "Evaluating the DevOps Reference Architecture for Multi-cloud IoT-Applications," *SN Comput. Sci.*, vol. 2, no. 2, Apr. 2021, doi: 10.1007/s42979021-00519-6.

[18] S. M. Mohammad, "DevOps automation and Agile methodology," vol. 5, no. 3, 2017. Sree , Moukthika Kameswari Mallela, Sree , Moukthika Kameswari Mallela, and Sree , Moukthika Kameswari Mallela, "A Transformative Approach to Agile-Driven DevOps in Software Project Management," Social Science Research Network, Jan. 2024, doi: https://doi.org/10.2139/ssrn.4766168.

[19] A. Hemon, B. Lyonnet, F. Rowe, and B. Fitzgerald, "From Agile to DevOps: Smart Skills and Collaborations," Information Systems Frontiers, Mar. 2019, doi: https://doi.org/10.1007/s10796-019-09905-1..

[20] M. Hüttermann and C. Rosenkranz, "The DevOps Funnel: Introducing DevOps as an Antecedent for Digitalization in Large-Scale Organizations," in *Proceedings of the Annual Hawaii International Conference on System Sciences*, Hawaii International Conference on System Sciences, 2023. doi: 10.24251/hicss.2023.683.

[21] A. Qumer Gill, A. Loumish, I. Riyat, and S. Han, "DevOps for information management systems," *VINE J. Inf. Knowl. Manag. Syst.*, vol. 48, no. 1, pp. 122–139, Feb. 2018, doi: 10.1108/vjikms-02-2017-0007.

[22] M. Yaseen, "Exploratory study of existing research on software requirements prioritization: A systematic literature review," Journal of Software Evolution and Process, vol. 36, no. 6, Sep. 2023, doi: https://doi.org/10.1002/smr.2613..

[23] M. Yaseen, "Requirement elicitation model for global software development vendors," Journal of Software: Evolution and Process, Nov. 2023, doi: https://doi.org/10.1002/smr.2628.

[24] M. Yaseen, R. Alroobaea, and H. Alsufyani, "Structural association of requirements engineering challenges in GSD: interpretive structural modelling (ISM) approach," Requirements Engineering, Feb. 2025, doi: https://doi.org/10.1007/s00766-025-00435-8..

[25] M. Yaseen, Samad Baseer, and S. Sherin, "Critical challenges for requirement implementation in context of global software development: A systematic literature review," Dec. 2015, doi: https://doi.org/10.1109/icosst.2015.7396413.

[26] M. A. Akbar, A. Alsanad, S. Mahmood, A. A. Alsanad, and A. Gumaei, "A Systematic Study to Improve the Requirements Engineering Process in the Domain of Global Software Development," IEEE Access, vol. 8, pp. 53374–53393, 2020, doi: https://doi.org/10.1109/access.2020.2979468.

[27] M. Krey, "DevOps Adoption: Challenges & Barriers," in Proceedings of the Annual Hawaii International Conference on System Sciences, Hawaii International Conference on System Sciences, 2022. doi: 10.24251/hicss.2022.8

[28] M. Yaseen, S. Ali, Abullah, and N. Ullah, "An Improved Framework for Requirement Implementation in the context of Global Software Development: A Systematic Literature Review Protocol," International Journal of Database Theory and Application, vol. 9, no. 6, pp. 161–170, Jun. 2016, doi: https://doi.org/10.14257/ijdta.2016.9.6.16