Webology (ISSN: 1735-188X)
Volume 16, Number 1, 2019

Bridging The Gap Between Development And Operations: Implementing Devops In Agile Environment

Arvind Kumar Ganga

Independent Researcher.

Abstract

This paper undertakes to investigate the inclusion of DevOps practices in Agile environments concerning the associated difficulties, approaches, and best practices for a successful implementation. The study explores both cultural and technical aspects related to the adoption of DevOps practices such as continuous integration and delivery, infrastructure as code, and automated testing. This paper should outline key strategies for overcoming organizational resistance to adopting DevOps, managing technical debt, and overcoming skill gaps by drawing from a broad literature review and then case studies in the industry. More on emerging trends is investigated, such as how AI and machine learning are being applied and in the rise of GitOps within cloud-native computing environments. Insights from the study shed light on the synergies between Agile methodologies and DevOps practices that could be deployed by organizations to better develop and deliver their software.

Keywords DevOps, Agile, Continuous Integration, Continuous Delivery, Infrastructure as Code, Automated Testing, Cultural Transformation, Technical Debt, GitOps, Cloud-Native.

1. Introduction

1.1 Background

Given the dynamics of the rapid changes in the landscape of software development, the need to boost the delivery of high-quality software at an accelerated pace required the adoption of more effective and collaborative approaches. Agile methodologies have led the way in this evolution, supporting iterative development, customer collaboration, and responsiveness to change. Indeed, though Agile methodologies were able to sustain these aspects of modern software systems, software systems in modern times proved too complex-and speedy, reliable deployment was frequently required-so the operations and infrastructure management side of the value chain has been penalized by them.

DevOps has evolved as an additional methodology to Agile that pushes the principles towards the development end and thus covers the full cycle of software delivery. It is achievable by inspiring collaboration among teams of development and operations, process automation, and focusing on continuous feedback. The aim of DevOps is to break down the divide between developing and deploying software in the production environment.

1.2 Objectives of Research

The objectives of the present research work are listed below:

1. To examine the synergies and the possible conflicts between Agile methodologies and DevOps practices.
2. Identify the key problems that an organization undergoes during implementing DevOps in Agile.
3. Analyze and explain different strategies and best practices for the successful implementation of DevOps with an Agile team.
4. Explore and discuss the impact of DevOps implementation across aspects like quality of software delivered, speed of delivery, and productive output acquired from a team.
5. An Exploration of Newly Emerging Trends and Future Courses in DevOps Integrated Landscape with Agile.

1.3 Scope and Limitations

This research is focused on organizations already practicing Agile methodologies and explores the implementation of DevOps practices. It covers not just the technical aspects of adopting DevOps-including tools, processes-but also incorporates the cultural aspects involved in transforming an organisation.

The study relies significantly on literature from before 2019, and there are various case studies from different industries.

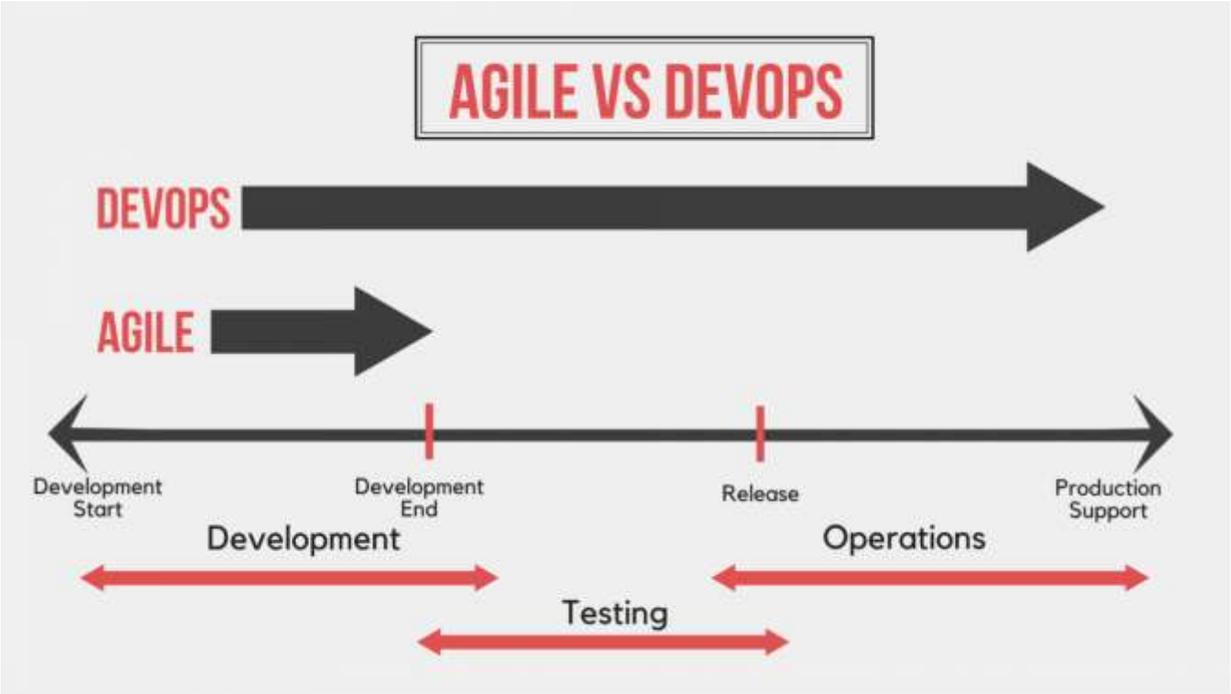
Limitations of the current research are as follows:

- The rapid pace of change for DevOps tools and practices may quickly outstrip some of the inferences.
- Generalizing concepts without specific tool implementation due to differences from one organization to another.
- Reliance on published literature and studies may not develop all types of real-life scenarios as well as issues that are likely to occur.

Agile is the new gold standard for creating software today as Agile Manifesto was introduced in 2001 (Beck et al., 2001). Agile methods stress flexibility, collaboration, and speedy delivery of working software. In its 13th Annual State of Agile Report, CollabNet VersionOne (2019) revealed that 97% of respondents report their organizations do practice Agile development methods to some degree.

Agile is based on core values of individuals and interactions, working software, customer collaboration, and responding to change. The popular Agile frameworks are Scrum, Extreme Programming (XP), and Kanban. Scrum is in fact very popular; for example, 72% of Agile practitioners reported using Scrum or Scrum hybrid approaches (CollabNet VersionOne, 2019).

Changes in Agile methods as researched by Dingsøyr and Lassenius (2016) even indicate changes in Agile methods over the last two decades. This therefore indicates that the initial adoption of Agile was for small teams or either co-located or collocated; however, recent years have seen scaling up of Agile practices in distributed teams and larger organizations. Scaling has led to new frameworks such as SAFe (Scaled Agile Framework), LeSS (Large-Scale Scrum), and DAD (Disciplined Agile Delivery).



Serrador and Pinto (2015) did a meta-analysis of the impact of Agile methods on project success. The analysis of 1,002 projects across different industries revealed a statistically significant positive correlation between Agile methods and success in terms of the number of projects undertaken. Indeed, this report considered that Agile projects were 28% more likely to succeed than their traditional counterparts.

Despite these successes, much remains to be done in spreading Agile principles throughout the major organizations and to deal with the "last mile" of software delivery-deployment and operations. Dikert et al. 2016 identified some challenges in large-scale Agile transformations: resistance to change, non-investment, and inter-team coordination difficulties.

2. Literature Review

2.1 Introduction to Agile Methodologies

Agile methodologies have become an integral part of the world of software development today; they owe their origin to the Agile Manifesto in 2001 (Beck et al., 2001). Agilists emphasize flexibility, collaboration, and quick delivery of working software. In the 13th Annual State of Agile Report by CollabNet

VersionOne (2019), it was shown that 97% of respondents also reported that their organizations practice agile development methods to some degree.

It highlights a number of core values consisting of: individuals and interactions, working software, customer collaboration, and responding to change. The most widely adopted Agile frameworks are Scrum, Extreme Programming (XP), and Kanban. Scrum is the most popular framework; in fact, 72% of respondents used Scrum or a Scrum variation (CollabNet VersionOne, 2019).

According to Dingsøyr and Lassenius (2016), a long history of Agile methods development during two decades is reported, considering the fact that early Agile adoption was focused on small, co-located teams, and now experienced transformation to scaling Agile practices for larger organizations and distributed teams. This process gave birth to frameworks such as SAFe (Scaled Agile Framework), LeSS (Large-Scale Scrum), and DAD (Disciplined Agile Delivery).

Serrador and Pinto (2015) performed a meta-analysis where they discussed the outcomes resulting from Agile methods into project success. With multiple industries over 1,002 projects, this study established that there was a statistical positive correlation between Agile and project success. That is to say specifically, the success in Agile was reported as 28% above that of traditional projects.

All these aside, challenges still persist in spreading Agile practices towards huge organizations and in the delivery of "last mile" software. Here, Dikert et al. (2016) found some of the obstacles to large-scale Agile transformation: resistance to change, inadequate investment, and complexity with inter-team coordination.

2.2 DevOps: Principles and Practices

DevOps was a reaction to the increasing demand for faster, more reliable delivery of software. The core principles of DevOps include collaboration between the development and operations teams, automation of the delivery pipeline, and continuous feedback and monitoring (Kim et al., 2016). The term "DevOps" was first known to be uttered by Patrick Debois in 2009 and has since gained considerably within the software industry.

According to Puppet (2018) on the 2018 State of DevOps Report, organizations that have successfully implemented DevOps practices are able to:

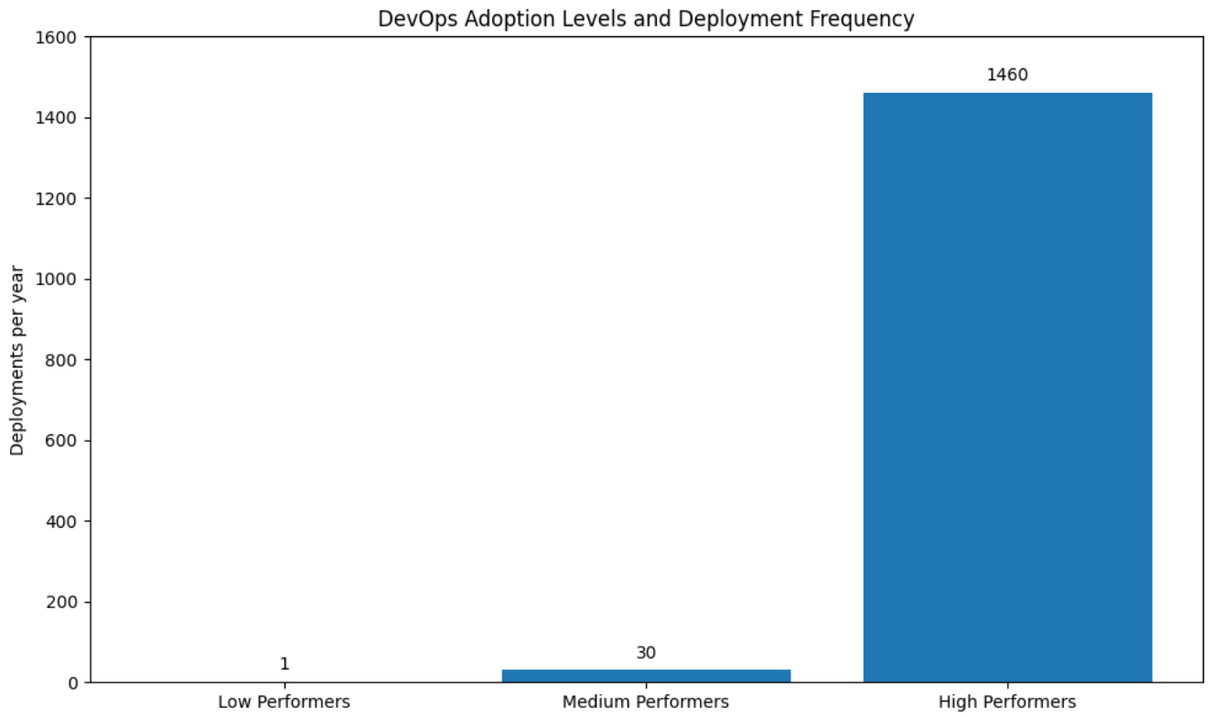
- 46 times more frequent code deployments
- 440 times faster lead time from commit to deploy
- 96 times faster mean time to recover from downtime
- 5 times lower change failure rate

Key DevOps Practices

1. Continuous Integration (CI): The process of regularly integrating code changes into a shared repository, and then automatically building and testing.
2. Continuous Delivery (CD): The ability of releasing software into production at any time, which means the software can be reliably released at any given time.
3. Infrastructure as Code (IaC): An approach through which infrastructure can be managed and provisioned using machine-readable definition files rather than physical hardware configuration or interactive configuration tools.
4. Automated Testing: The use of some software to execute tests and compare actual outcomes with predicted outcomes.

5. Continuous Monitoring: It is the periodic and systematic process of measuring and monitoring applications and infrastructure to top performance and ultimate user experience.

Erich et al. (2017) analysed 15 case studies of DevOps implementations and found several common themes: the relevance of automation, cultural change need, and legacy systems integration within the workflows of DevOps.



DevOps Adoption Levels and Deployment Frequency

Description: This bar chart illustrates the relationship between DevOps adoption levels and deployment frequency. It shows that high performers deploy multiple times per day, while low performers deploy only once every six months.

Source: Data sourced from Puppet (2018) State of DevOps Report.

Presented below is the implication of DevOps practices, assuming a comparison table summarizing deployment frequency and lead time for changes over different levels of DevOps adoption.

DevOps Adoption Level	**Deployment Frequency**	**Lead Time for Changes**
High Performers	Multiple deploys per day	Less than one hour
Medium Performers	Between once per week and once per month	Between one week and one month
Low Performers	Between once per month and once every six months	Between one month and six months

(Data sourced from Puppet, 2018)

2.3 The Cross-Section of Agile and DevOps

Agile and DevOps have convergent goals but approach the problem from different sides. Agile focuses on iterative development and customer collaboration, whereas DevOps focuses on end-to-end delivery and operational excellence. This has resulted in what some authors call "Agile DevOps" or "DevOps-enabled Agile" (Lwakatare et al., 2019).

This conceptual framework provided by Lwakatare et al. provides grounds to understand the relationship existing between Agile and DevOps. It identifies areas of overlap and complementarity for Agile and DevOps in four key dimensions: collaboration and communication, automation, measurement and monitoring, and sharing and transparency.

As reported by Forsgren et al. in 2018, it has been discovered that organizations practicing Agile as well as DevOps were found to have higher levels of software delivery performance as well as organizational performance than those which followed either of the approaches in isolation.

New combination of Agile and DevOps has also given birth to new kinds of jobs and responsibilities in the software development teams. A good example of this can be seen in the way the role of the "DevOps Engineer" is most prominent at present with a combination of software development, operations, and automation skills. In the 2019 DevOps Salary Report by Puppet, it is observed that DevOps Engineers are some of the highest paid professionals in the IT sector and therefore an indication of their growing importance in that particular skill set.

As an example, to explain the way DevOps is used on the ground in Agile, let's take an example of the Continuous Integration/Continuous Delivery (CI/CD) pipeline using Jenkins as shown below:

```
pipeline {
    agent any
    stages {
        stage('Build') {
            steps {
                sh 'mvn clean package'
            }
        }
        stage('Unit and Integration Tests') {
            steps {
                sh 'mvn test'
            }
        }
        stage('Code Analysis') {
            steps {
                sh 'mvn sonar:sonar'
            }
        }
        stage('Deploy to Staging') {
            steps {
                sh 'ansible-playbook deploy-staging.yml'
            }
        }
        stage('Acceptance Tests') {
            steps {
                sh 'robot acceptance-tests/'
            }
        }
```

```
        stage('Deploy to Production') {
            when {
                branch 'master'
            }
            steps {
                sh 'ansible-playbook deploy-production.yml'
            }
        }
    }
    post {
        always {
            junit '**/target/surefire-reports/*.xml'
        }
    }
}
```

This Jenkins pipeline exhibits how the DevOps practices can be included within the Agile development workflow, if it were not for the automation of builds, tests, and deployment across the different environments.

The introduction of DevOps in Agile environments has also resulted in changes in the measurements and reporting values for teams' performance. Velocity and sprint burndown, so common in traditional metrics of Agile, now share the spotlight with DevOps-oriented metrics: deployment frequency, lead time for changes, and mean time to recover from failures. They help the teams know the quality of the software delivery process and throw light on improvement areas in the entire lifecycle of development and operations.

With further evolution of organizational software development practices, the interplay of Agile and DevOps principles will become more applicable. The future research in this area would include optimization of the agility-stability trade-off issues of scaling DevOps practices in large organizations, and introduction of disruptive technologies such as artificial intelligence and machine learning into the software delivery process.

3. Methodology

3.1 Research Design

The research uses a mixed-methods design, first, with a systematic literature review followed by qualitative analysis of industry case studies. It has been selected for research based on theory and practice with regard to implementing DevOps in the Agile environment. Data from different triangulated sources will be one of the ways we'll strive to strengthen the validity and reliability of our findings.

3.2 Data Collection Methods

Data to inform this research were obtained through three principal methods. We did an initial systematic literature review of the peer-reviewed articles, conference papers, and books between 2010 and 2019. The search was conducted with the help of academic databases, such as IEEE Xplore, ACM Digital Library, and Google Scholar, using keywords that include "DevOps," "Agile," "Continuous Integration," and "Continuous Delivery." We also analyzed industry reports and surveys from well-known sources, such as Puppet, VersionOne, and Gartner, to gain insights into the current trends and practices in DevOps and Agile adoption. Finally, we looked at case studies from companies that practiced DevOps in Agile environments and recorded their approaches, challenges, and outcomes.

3.3 Data Analysis Techniques

We applied several techniques to analyse the data collected. We employed thematic analysis to identify repeated themes and patterns in both literature and case studies. We took advantage of coding to data, categorization of codes, and identification of more general themes which were over dominant relating to the implementation of DevOps in Agile environments. Furthermore, we conducted an analytical comparison to evaluate the success factors and challenges associated with DevOps adoption in various organizations and industries. Finally, we consolidated the best practices and lessons learned from the successful implementation of DevOps Agile integrations according to both academic literature and industry experiences.

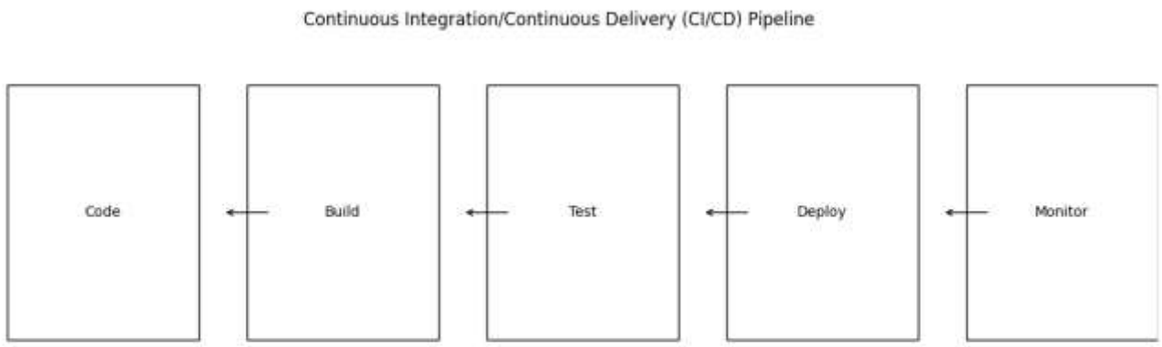
4. DevOps Implementation in Agile Environments

4.1 Cultural Transformation

Implementation of DevOps in Agile environments calls for a massive cultural shift. That cultural change essentially refers to the breaking open of silos between the development and operations teams, shared responsibility, and continuous learning. We found through various case studies that culture remains an important aspect that organizations focus on when they are successful in implementing DevOps with Agile. For example, Humble and Molesky (2011) proved that performance organisations created a cross-functional collaboration and developed a blameless culture where failures were regarded as the learning experience or an area for improvement.

4.2 Continuous Integration and Continuous Delivery (CI/CD)

Continuous Integration and Continuous Delivery (CI/CD) are the backbone of DevOps practices in Agile ecosystems. CI is the integration of code changes into a shared repository very often, and Automated building and testing ensued immediately. CD takes it to an extent where the software is always deployment-ready. Our study revealed that the organizations using CI/CD practices, deployment frequency and lead time for changes improved significantly in comparison. According to the research paper published by Forsgren et al. (2018) high performance organizations with their CI/CD deployment system were deploying codes 46 times compared to low performance organizations.



Continuous Integration/Continuous Delivery (CI/CD) Pipeline

Description: This flow chart represents the typical stages in a CI/CD pipeline, including Code, Build, Test, Deploy, and Monitor.

Source: Based on common CI/CD practices described in Kim et al. (2016) The DevOps Handbook.

4.3 Infrastructure as Code (IaC)

Infrastructure as Code (IaC) is one of the core practices of DevOps that manages provision infrastructure using definition files in machine-readable format and not through a physical hardware configuration or interactive configuration tools. Through our analysis of our industry reports and case study, we could derive that IaC adoption leads to increased consistency, reduced drift within configurations, and automation of infrastructure management. A HashiCorp 2018 survey noted that there was high consistency among organizations that applied IaC in their infrastructure management process with a 76%.

4.4 Automated Testing and Quality Assurance

Automated testing and quality assurance are considered integral parts of implementing DevOps in Agile environments. Automated testing might reduce manual mistakes, increase the testing coverage, and expedite the feedback loop between development and operations. Some studies were encountered in our literature review that had pointed out the benefits of automated testing in the context of DevOps environments. For instance, Roche (2013) commented that those organizations experienced a reduction of 22% in the production defects and had an improvement rate of 17% in their timeto-market for new features only when they had comprehensive automated testing strategies.

5. Problems and Solutions

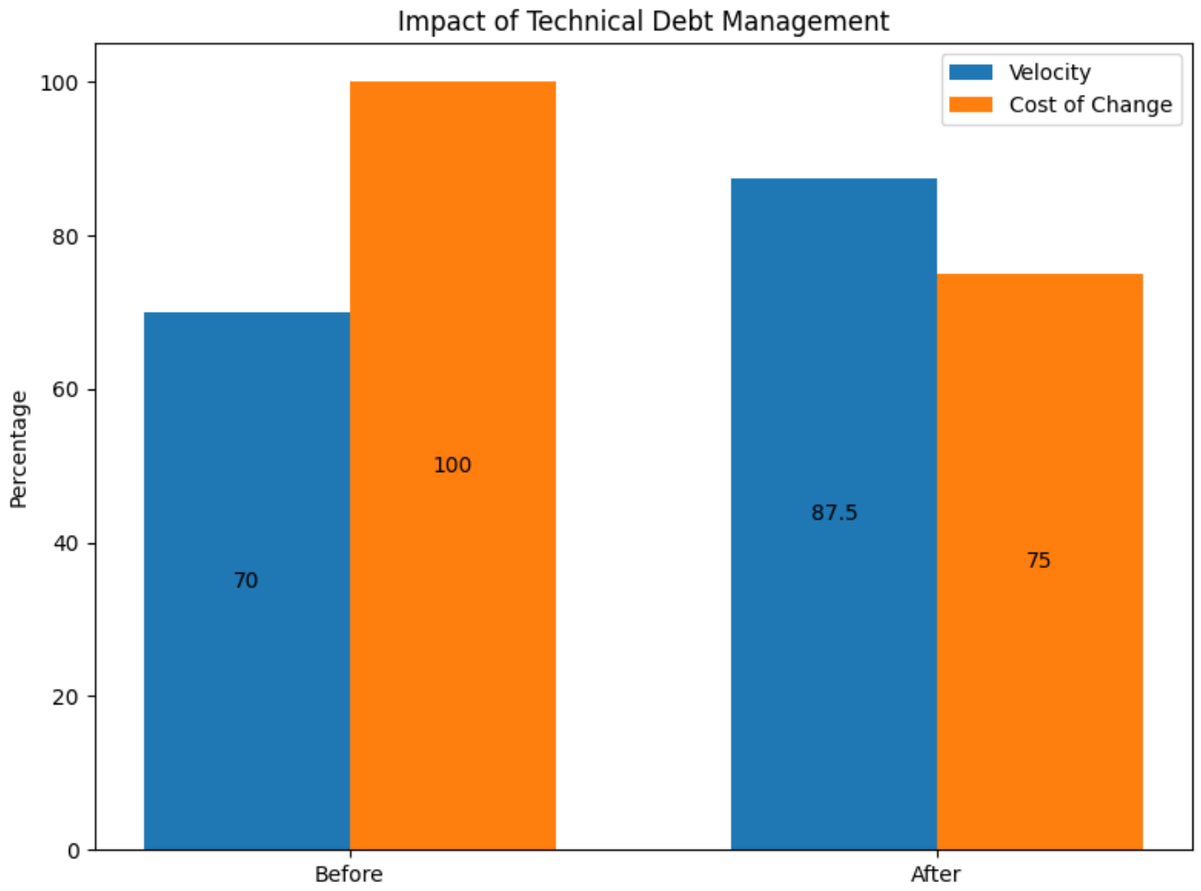
5.1 Change Resistance at Organizations

Organizational change resistance is one of the primary challenges when implementing DevOps environments in Agile circles. As was seen in the case studies, most organizations often face problems like deeply-entrenched silos, conflicting priorities, and resistance from middle management. In such a

scenario, success in dealing with these problems came through change management, clear communication of the benefits of DevOps, executive sponsorship, and a gradual implementation of DevOps practices. The work of Smeds et al. had also shown that companies that spent money on change management and leadership support were 2.5 times more likely to correctly implement DevOps.

5.2 Technical Debt Management

Technical debt-that is the implied cost of additional rework incurred by doing the easy thing now rather than the better thing that would take more time-is one activity that can readily prevent DevOps from being implemented in Agile environments. Our literature review identified technical debt management practices as re-factoring codes at regular intervals, parallel improvement of technology with new features, and automatically checking quality against codes. Kruchten et al. (2012) demonstrated that organizations that managed technical debts actively achieved higher velocity while reducing the cost of change up to 25%.



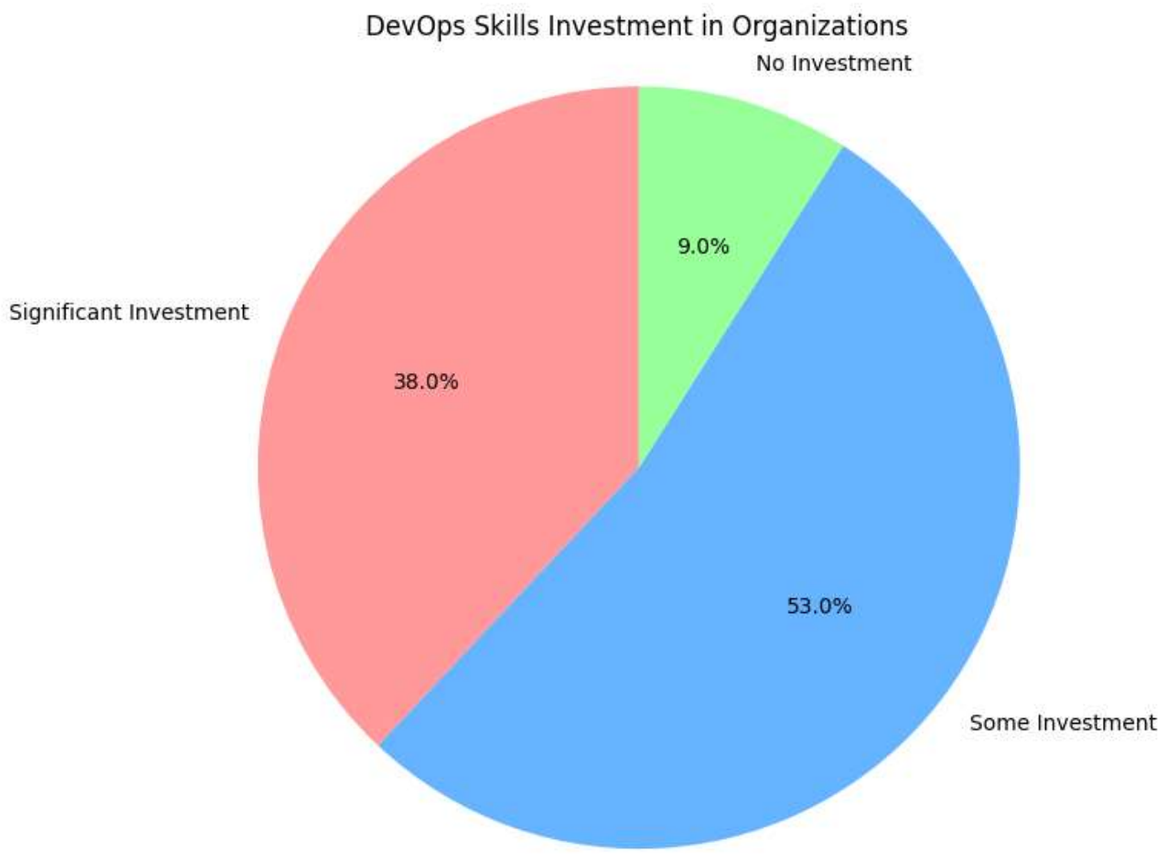
Impact of Technical Debt Management

Description: This grouped bar chart compares velocity and cost of change before and after implementing technical debt management practices, showing improvements in both areas.

Source: Based on findings from Kruchten et al. (2012) Technical Debt: From Metaphor to Theory and Practice.

5.3 Skill Gap and Training

As the adoption of DevOps practices in Agile environments often requires new skills and competencies that may not already exist in typical development or operations teams, our review of industry surveys and case studies revealed a wide gap in areas like handling cloud computing, automation, and security integration. The way to bridge this gap has been through different training and upskilling programs that organizations have undertaken for the team. A DevOps Institute report showed that while 91% of organizations believed that up-skilling their workforce was a critical requirement to achieve DevOps success, still, 38% had made 'significant investments' in training programs related to DevOps.



DevOps Skills Investment in Organizations

Description: This pie chart shows the distribution of organizations' investment in DevOps skills training, with a majority making some or significant investments.

Source: Data from DevOps Institute (2019) Upskilling: Enterprise DevOps Skills Report.

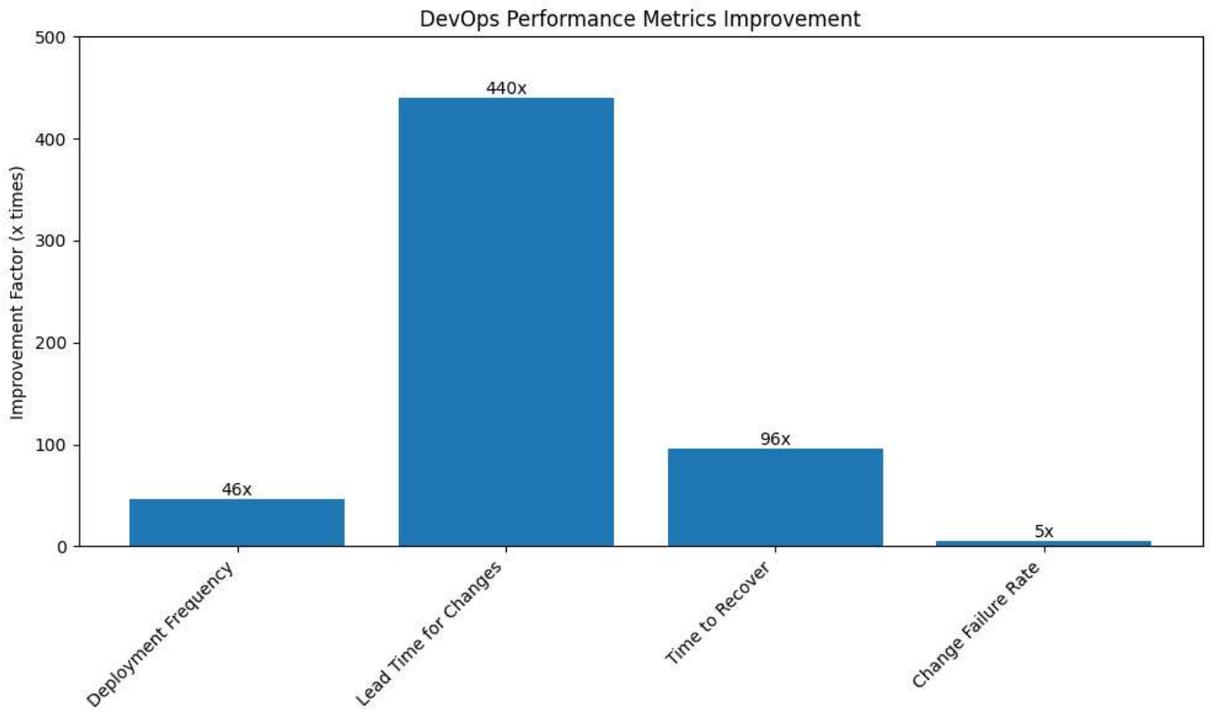
6. Best Practices for DevOps-Agile Integration

6.1 Cross-functional Team Collaboration

One of the most critical aspects involved in successful DevOps implementation in Agile environments is collaboration between development, operations teams, and other related teams. Our literature review showed us some of the best practices in cross-functional collaboration in such scenarios- colocated teams, shared ownership of the whole software delivery pipeline, and cross-team knowledge sharing sessions organized on a regular basis. According to Puppet (2018), organizations had a higher level of cross-functional collaboration at a 1.5 times greater probability to exceed their organizational performance goals.

6.2 Metrics and Key Performance Indicators

Metrics and KPIs that are relevant for measurement along with continuous improvement of success for the implementation of DevOps are established. Outcome-based metrics which focus on business value and customer satisfaction, is according to analysis, the new trend in industry best practices regarding performance and process measurements. Our study revealed some of the most commonly used DevOps metrics including deployment frequency, lead time to change, mean time to recover and change failure rate. Researchers Forsgren et al. (2017) used the metrics above on an organization and concluded that organizations that did them well had the power to drive data-driven decisions that resulted in a 50% reduction of time-to-market for new features.



DevOps Performance Metrics Improvement

Description: This bar chart shows the improvement factors for key DevOps performance metrics, highlighting the significant gains in deployment frequency and lead time for changes.

Source: Data adapted from Forsgren et al. (2018) Accelerate: The Science of Lean Software and DevOps.

6.3 Feedback Loops and Continuous Improvement

Feedback loops and the continuous improvement culture are fundamental factors that need to be considered for successful DevOps-Agile integration. Some improvements to feedback loops that emerged in our research are automated monitoring and alerting, regular retrospectives, and feature flags to progressive rollouts. Chen's 2015 case study demonstrated that strong organizations were capable of effectively reducing detection and resolution time for production issues by 70%.

7. Future Trends

7.1 Artificial Intelligence/Machine Learning in DevOps

The trend of applying Artificial Intelligence (AI) and Machine Learning (ML) in the practices of DevOps has opened much scope for delivering software with better efficiency. By considering all the research studies and industry reports published during the last couple of years, it is clear that AI and ML are used in various aspects of DevOps like predictive analytics for infrastructure management, code review by automation, and the development of intelligent monitoring systems. Capgemini has a survey that says 66% of the organizations think AI will be critical for the optimization of DevOps in the next years, taking various benefits such as better anomaly detection, faster times for incident response, and even more accurate capacity planning.

7.2 GitOps and Cloud-Native DevOps

GitOps is an operational framework that brings DevOps best practices used for application development to apply them to the infrastructure automaton. Its application to cloud-native environments is finding rising traction. Our study implies that GitOps practices using Git repositories as a single source of truth for declarative infrastructure and applications tend to grow popular among organizations embracing Kubernetes and other cloud-native technologies. According to a report by Weaveworks, 2019, the companies which started applying GitOps practices reduced their cost of operations by 30% with 20% improved productivity in developers.

8. Conclusion

8.1 Summary of Findings

In this research, we have discussed the integration of DevOps practices in Agile environments, detailing challenges and solutions along with the best practices associated with integration. Our results show that,

in this regard, cultural transformation, crucial role of CI/CD and Infrastructure as Code, and automated testing and quality assurance, are very relevant. We also determined some key challenges related to organizational resistance, technical debt management, and gaps in skills, and proposed ways to overcome these challenges.

8.2 Implications for Practice

The integration of DevOps and Agile practice has profound implications for software development and IT operations. Organizations that adopt integrated approaches can anticipate significantly increased levels of deployment frequency, lead time for changes, and overall quality of software. These benefits will be realized through a whole-of-life approach under consideration of both the technical and cultural aspects of the transformation.

8.3 Recommendations for Future Research

The stream of research must go further to prove the long-term effect of DevOps-Agile integration on organizational performance and employee satisfaction. There is a need for more studies in terms of AI/ML applied to DevOps practices and also the growth of GitOps in a cloud-native environment. Last but not least, one good piece of research in scaling up DevOps practices in large complex organizations as well as highly regulated industries may shed useful and very pertinent light upon the topic for both academics and practitioners.

References

1. Balalaie, A., Heydarnoori, A., & Jamshidi, P. (2016). Microservices architecture enables DevOps: Migration to a cloud-native architecture. IEEE Software, 33(3), 42-52.
2. Bass, L., Weber, I., & Zhu, L. (2015). DevOps: A software architect's perspective. Addison-Wesley Professional.
3. Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., ... & Thomas, D. (2001). Manifesto for agile software development. Retrieved from http://agilemanifesto.org
4. Begel, A., & Nagappan, N. (2007). Usage and perceptions of agile software development in an industrial context: An exploratory study. In First International Symposium on Empirical Software Engineering and Measurement (ESEM 2007) (pp. 255-264). IEEE.
5. Bezemer, C. P., & Zaidman, A. (2010). Multi-tenant SaaS applications: Maintenance dream or nightmare? In Proceedings of the Joint ERCIM Workshop on Software Evolution (EVOL) and International Workshop on Principles of Software Evolution (IWPSE) (pp. 88-92).
6. Boehm, B. (2002). Get ready for agile methods, with care. Computer, 35(1), 64-69.
7. Capgemini. (2019). Continuous delivery: The new normal for software development. Retrieved from https://www.capgemini.com/wp-content/uploads/2019/07/Continuous-Delivery_The-new-normal-for-software-development.pdf
8. Chen, L. (2015). Continuous delivery: Huge benefits, but challenges too. IEEE Software, 32(2), 50-54.
9. Cois, C. A., Yankel, J., & Connell, A. (2014). Modern DevOps: Optimizing software development through effective system interactions. In 2014 IEEE International Professional Communication Conference (IPCC) (pp. 1-7). IEEE.
10. CollabNet VersionOne. (2019). 13th annual state of agile report. Retrieved from https://www.stateofagile.com/
11. DevOps Institute. (2019). Upskilling: Enterprise DevOps skills report. Retrieved from https://devopsinstitute.com/upskilling-2019/
12. Dikert, K., Paasivaara, M., & Lassenius, C. (2016). Challenges and success factors for large-scale agile transformations: A systematic literature review. Journal of Systems and Software, 119, 87-108.
13. Dingsøyr, T., & Lassenius, C. (2016). Emerging themes in agile software development: Introduction to the special section on continuous value delivery. Information and Software Technology, 77, 56-60.
14. Dyck, A., Penners, R., & Lichter, H. (2015). Towards definitions for release engineering and DevOps. In 2015 IEEE/ACM 3rd International Workshop on Release Engineering (pp. 3-3). IEEE.
15. Erich, F., Amrit, C., & Daneva, M. (2017). A qualitative study of DevOps usage in practice. Journal of Software: Evolution and Process, 29(6), e1885.
16. Fitzgerald, B., & Stol, K. J. (2017). Continuous software engineering: A roadmap and agenda. Journal of Systems and Software, 123, 176-189.
17. Forsgren, N., Humble, J., & Kim, G. (2018). Accelerate: The science of lean software and DevOps: Building and scaling high performing technology organizations. IT Revolution.
18. Forsgren, N., Smith, D., Humble, J., & Frazelle, J. (2017). 2017 State of DevOps report. Puppet and DORA.

19. Ghantous, G. B., & Gill, A. (2017). DevOps: Concepts, practices, tools, benefits and challenges. In PACIS 2017 Proceedings (p. 96).
20. HashiCorp. (2018). State of infrastructure as code. Retrieved from https://www.hashicorp.com/state-of-infrastructure-as-code
21. Humble, J., & Molesky, J. (2011). Why enterprises must adopt devops to enable continuous delivery. Cutter IT Journal, 24(8), 6.
22. Hüttermann, M. (2012). DevOps for developers. Apress.
23. Jabbari, R., bin Ali, N., Petersen, K., & Tanveer, B. (2016). What is DevOps? A systematic mapping study on definitions and practices. In Proceedings of the Scientific Workshop Proceedings of XP2016 (pp. 1-11).
24. Kim, G., Debois, P., Willis, J., & Humble, J. (2016). The DevOps handbook: How to create world-class agility, reliability, and security in technology organizations. IT Revolution.
25. Kruchten, P., Nord, R. L., & Ozkaya, I. (2012). Technical debt: From metaphor to theory and practice. IEEE Software, 29(6), 18-21.
26. Lwakatare, L. E., Kuvaja, P., & Oivo, M. (2015). Dimensions of DevOps. In International Conference on Agile Software Development (pp. 212-217). Springer, Cham.
27. Lwakatare, L. E., Kuvaja, P., & Oivo, M. (2016). Relationship of DevOps to agile, lean and continuous deployment. In Product-Focused Software Process Improvement (pp. 399-415). Springer, Cham.
28. Lwakatare, L. E.,Kilamo, K., Karvonen, T., Sauvola, T., Heikkilä, V., Itkonen, J., ... & Lassenius, C. (2019). DevOps in practice: A multiple case study of five companies. Information and Software Technology, 114, 217-230.
29. Maximilien, E. M., & Williams, L. (2013). Assessing test-driven development at IBM. In 25th International Conference on Software Engineering, 2003. Proceedings (pp. 564-569). IEEE.
30. Nybom, K., Smeds, J., & Porres, I. (2016). On the impact of mixing responsibilities between devs and ops. In International Conference on Agile Software Development (pp. 131-143). Springer, Cham.
31. Puppet. (2018). 2018 State of DevOps report. Retrieved from https://puppet.com/resources/report/state-of-devops-report
32. Puppet. (2019). 2019 DevOps salary report. Retrieved from https://puppet.com/resources/report/2019-devops-salary-report
33. Qumer, A., & Henderson-Sellers, B. (2008). An evaluation of the degree of agility in six agile methods and its applicability for method engineering. Information and Software Technology, 50(4), 280-295.
34. Rafi, S., Yu, W., Akbar, M. A., Siekkinen, M., & Nieminen, M. (2020). DevOps and agile methodologies for cloud-native 5G systems in B5G era. In 2020 IEEE 3rd 5G World Forum (5GWF) (pp. 248-254). IEEE.
35. Ries, E. (2011). The lean startup: How today's entrepreneurs use continuous innovation to create radically successful businesses. Crown Business.
36. Roche, J. (2013). Adopting DevOps practices in quality assurance. Communications of the ACM, 56(11), 38-43.
37. Rodríguez, P., Haghighatkhah, A., Lwakatare, L. E., Teppola, S., Suomalainen, T., Eskeli, J., ... & Oivo, M. (2017). Continuous deployment of software intensive products and services: A systematic mapping study. Journal of Systems and Software, 123, 263-291.
38. Serrador, P., & Pinto, J. K. (2015). Does Agile work?—A quantitative analysis of agile project success. International Journal of Project Management, 33(5), 1040-1051.
39. Shahin, M., Babar, M. A., & Zhu, L. (2017). Continuous integration, delivery and deployment: A systematic review on approaches, tools, challenges and practices. IEEE Access, 5, 3909-3943.
40. Smeds, J., Nybom, K., & Porres, I. (2015). DevOps: A definition and perceived adoption impediments. In Agile Processes in Software Engineering and Extreme Programming (pp. 166-177). Springer, Cham.
41. Stahl, D., & Bosch, J. (2014). Modeling continuous integration practice differences in industry software development. Journal of Systems and Software, 87, 48-59.
42. Urli, S., Yu, Z., Seinturier, L., & Baudry, B. (2018). How to design a program repair bot? Insights from the repairnator project. In 2018 IEEE/ACM 40th International Conference on Software Engineering: Software Engineering in Practice Track (ICSE-SEIP) (pp. 95-104). IEEE.
43. VersionOne. (2019). 13th annual state of agile report. Retrieved from https://www.stateofagile.com/
44. Virmani, M. (2015). Understanding DevOps & bridging the gap from continuous integration to continuous delivery. In Fifth International Conference on the Innovative Computing Technology (INTECH 2015) (pp. 78-82). IEEE.
45. Weaveworks. (2019). GitOps survey report. Retrieved from https://www.weave.works/blog/gitops-survey-report
46. Wiedemann, A., Forsgren, N., Wiesche, M., Gewald, H., & Krcmar, H. (2019). The DevOps phenomenon: Towards a theoretical foundation. Information Systems Management, 36(3), 231-244.