Programming Methodologies: Agile, Scrum, and Devops in Practice

Nkiwa Monkila Jonathan[1], Monkila Nkiwa Barthelemy[2], Padinganyi Tshiombela Faustin[3], Kaputu Mwamba Georges[4]

[1,2,4]Department of Computer Science, ISPT-KIN, Kinshasa, DRC

[3]Department of Computer Science, ISS-KIN, Kinshasa, DRC

[1]ORCID: https://orcid.org/0009-0000-7244-0580

[2]ORCID: https://orcid.org/0009-0008-4274-3220

[3]ORCID: https://orcid.org/0009-0004-5619-1456

[4]ORCID: https://orcid.org/0009-0001-0210-7595

DOI: https://doi.org/10.51584/IJRIAS.2026.11070102

Received: 20 January 2026; Accepted: 26 January 2026; Published: 06 August 2026

ABSTRACT

Modern software development requires flexible, efficient, and collaborative methodologies to cope with rapidly changing requirements and complex systems. This article explores three widely adopted programming and software development methodologies: Agile, Scrum, and DevOps. Agile focuses on iterative development, customer collaboration, and continuous improvement. Scrum, as an Agile framework, provides structured roles, events, and artifacts to enhance team productivity and project transparency. DevOps extends Agile principles by integrating development and operations, enabling continuous integration, continuous delivery, and faster deployment cycles. Through practical analysis and real-world examples, this study highlights how these methodologies complement each other in practice, improve software quality, reduce time-to-market, and enhance collaboration among stakeholders. The article concludes by discussing best practices and challenges in adopting Agile, Scrum, and DevOps in modern software engineering environments.

Keywords : Agile Methodology, Scrum Framework, DevOps Practices, Software Development, Programming Methodologies

INTRODUCTION

The rapid evolution of information technologies and the increasing complexity of software systems have significantly transformed the way software is designed, developed, and deployed. Traditional programming methodologies, often based on rigid planning and linear development models, have shown limitations in addressing frequent requirement changes, tight deadlines, and the need for continuous user feedback. As a result, modern software engineering has progressively shifted toward more flexible, adaptive, and collaborative development approaches.

Among these modern approaches, **Agile**, **Scrum**, and **DevOps** have emerged as dominant methodologies in contemporary software development. Agile methodology introduced a paradigm shift by emphasizing iterative development, close collaboration between development teams and stakeholders, and rapid delivery of functional software. Instead of focusing heavily on extensive upfront documentation, Agile promotes adaptability, continuous improvement, and customer satisfaction throughout the development lifecycle.

Building on Agile principles, **Scrum** provides a structured framework that defines specific roles, events, and artifacts to manage and organize software development projects effectively. Scrum encourages transparency, inspection, and adaptation, enabling teams to deliver incremental value through short development cycles known as sprints. Its practical orientation makes it particularly suitable for managing complex projects where requirements evolve over time.

On the other hand, **DevOps** extends the Agile philosophy beyond development by integrating software development and IT operations. DevOps aims to eliminate silos between teams, automate processes, and

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume XI Issue VII July 2026

establish continuous integration and continuous delivery pipelines. By fostering a culture of collaboration and shared responsibility, DevOps helps organizations achieve faster release cycles, improved system reliability, and enhanced scalability.

This article examines Agile, Scrum, and DevOps as complementary programming and software development methodologies rather than isolated practices. It analyzes their fundamental principles, practical implementation, and real-world impact on software quality, team productivity, and deployment efficiency. By highlighting both benefits and challenges, this study aims to provide a clear understanding of how these methodologies are applied in practice and how they contribute to the success of modern software engineering projects.

METHODS

Research Approach

The research follows an interpretative approach focused on software engineering practices rather than algorithmic performance measurement. Agile, Scrum, and DevOps are treated as socio-technical systems that combine technical processes, organizational structures, and human collaboration. This approach enables an in-depth understanding of methodological adoption, workflow management, and cultural transformation within software development teams.

Research Design and Scope

The study is designed as a multi-method qualitative analysis combining literature review, framework comparison, and practice-oriented evaluation. The scope of the research covers:

- Programming workflow organization
- Project management practices
- Development and deployment pipelines
- Team collaboration and communication mechanisms

The analysis focuses on small to medium-sized software projects as well as enterprise-level systems where these methodologies are commonly applied.

Table 1 presents the scope and focus areas of the research.

Table 1: Research Scope and Focus Areas

Focus Area	**Description**
Programming Workflow	Code development, iteration cycles, refactoring
Project Management	Planning, task allocation, progress tracking
Collaboration	Team communication, stakeholder involvement
Automation	Build, test, and deployment automation
Deployment	Release frequency, delivery pipelines

Data Sources and Collection Methods

Data collection is based on multiple secondary sources to ensure methodological triangulation. The primary data sources include:

- Peer-reviewed scientific articles on Agile, Scrum, and DevOps
- Conference proceedings in software engineering and programming methodologies

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume XI Issue VII July 2026

- Industry case studies from software companies and technology organizations
- Official framework documentation (Agile Manifesto, Scrum Guide, DevOps handbooks)

These sources provide both theoretical foundations and practical insights into real-world implementation challenges and success factors.

Comparative Framework Analysis

A structured comparative framework is used to analyze Agile, Scrum, and DevOps across different dimensions. Each methodology is evaluated according to its:

- Core principles and values
- Roles and responsibilities
- Development cycles and iteration models
- Tool support and automation practices
- Integration with programming and testing activities

This comparative analysis allows the identification of overlaps, complementarities, and methodological gaps.

Table 2 compares the fundamental characteristics of Agile, Scrum, and DevOps.

Table 2: Comparative Overview of Agile, Scrum, and DevOps

Aspect	**Agile**	**Scrum**	**DevOps**
Nature	Philosophy	Framework	Cultural & Technical Practice
Focus	Adaptability	Team productivity	Automation & integration
Iteration	Incremental	Sprint-based	Continuous
Roles	Flexible	Defined roles	Cross-functional teams
Delivery	Frequent releases	Incremental increments	Continuous deployment

Thematic and Conceptual Analysis

A thematic analysis technique is applied to extract recurring patterns and concepts across the selected sources. Key themes include:

- Iterative and incremental programming
- Continuous feedback and customer involvement
- Automation and pipeline-driven development
- Collaboration between cross-functional teams
- Quality assurance and continuous testing

These themes are mapped to each methodology to assess how they contribute to efficient and reliable software production.

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume XI Issue VII July 2026

Evaluation Metrics and Assessment Criteria

Although the study is primarily qualitative, a set of conceptual evaluation metrics is used to assess the practical effectiveness of each methodology. These metrics include:

- Responsiveness to changing requirements
- Code quality and maintainability
- Time-to-market and delivery frequency
- Deployment reliability and system stability
- Organizational adaptability and team autonomy

The metrics are derived from commonly accepted software engineering performance indicators.

Validity and Reliability Considerations

To enhance the validity of the findings, the study relies on data triangulation by comparing academic research with industry practices. Reliability is reinforced by referencing standardized frameworks and widely recognized documentation. While subjective interpretation is inherent in qualitative research, consistent analytical criteria are applied throughout the study to minimize bias.

Limitations and Ethical Considerations

The study is limited by the absence of direct experimentation, controlled environments, or primary data collection such as interviews or surveys. Additionally, the findings may be influenced by the maturity level of organizations adopting these methodologies. Ethical considerations are addressed by ensuring proper citation of all sources and avoiding the disclosure of confidential or proprietary information from industry case studies.

RESULTS AND DISCUSSION

Results

The analysis reveals that Agile, Scrum, and DevOps significantly improve software development performance when applied appropriately and in a complementary manner. Each methodology demonstrates distinct strengths across different dimensions of the software development lifecycle.

Methodology Effectiveness Across Evaluation Criteria

Table 3 summarizes the relative effectiveness of Agile, Scrum, and DevOps based on the qualitative evaluation criteria defined earlier.

Table 3: Comparative Effectiveness of Agile, Scrum, and DevOps

Evaluation Criterion	Agile	Scrum	DevOps
Flexibility	High	Medium	Medium
Code Quality	High	High	Very High
Time-to-Market	Medium	High	Very High
Deployment Stability	Medium	Medium	Very High

Team Collaboration	High	Very High	High

The results indicate that Agile excels in flexibility and adaptability to change, making it suitable for environments with evolving requirements. Scrum shows strong performance in team collaboration and structured project execution, while DevOps demonstrates superior results in deployment stability and delivery speed.

Impact on Programming Practices

The findings show that Agile and Scrum primarily influence programming at the code and iteration levels. Practices such as incremental development, frequent refactoring, and continuous testing contribute to improved code quality and maintainability. DevOps extends these benefits by automating build, test, and deployment processes, reducing human error and increasing consistency across environments.

Integration of Methodologies in Practice

The analysis highlights that organizations rarely adopt these methodologies in isolation. Agile provides the foundational philosophy, Scrum structures development activities, and DevOps ensures operational efficiency. Projects that combine Scrum-based development with DevOps automation achieve shorter release cycles and higher system reliability compared to those relying solely on Agile or Scrum.

Discussion

The results confirm that Agile, Scrum, and DevOps are not competing methodologies but rather complementary approaches that address different challenges in software engineering. Agile offers strategic flexibility, Scrum delivers tactical project management, and DevOps ensures operational continuity.

One key discussion point is the cultural transformation required for successful adoption. Agile and Scrum demand active stakeholder involvement and self-organizing teams, while DevOps requires breaking down organizational silos between development and operations. Without cultural alignment, the technical benefits of these methodologies may be significantly reduced.

Another important consideration is scalability. While Agile and Scrum are highly effective for small to medium-sized teams, scaling them to large enterprise systems introduces complexity. Frameworks such as scaled Agile or enterprise DevOps practices are often necessary to maintain consistency and coordination across multiple teams.

The results also reveal challenges related to tool dependency and skill requirements. DevOps adoption, in particular, requires advanced expertise in automation, cloud infrastructure, and monitoring tools. Organizations lacking these skills may experience slower adoption and limited benefits.

Overall, the discussion emphasizes that the success of Agile, Scrum, and DevOps depends on contextual factors such as project size, organizational maturity, and team expertise. A hybrid and adaptive adoption strategy, rather than a rigid methodological application, is recommended to maximize efficiency and software quality.

CONCLUSION

This study examined the practical application of Agile, Scrum, and DevOps methodologies within modern software development environments. Through a qualitative and comparative analysis, the research highlighted how these methodologies influence programming practices, team collaboration, and software delivery performance across the software development lifecycle.

The findings demonstrate that Agile, Scrum, and DevOps serve complementary roles rather than acting as competing approaches. Agile provides the foundational mindset that promotes adaptability, continuous feedback, and customer-centered development. Scrum operationalizes Agile principles through well-defined roles, structured iterations, and transparent project management. DevOps extends these practices by integrating

development and operations, enabling automation, continuous integration, and continuous delivery, which significantly improve deployment speed and system reliability.

The results also emphasize that successful adoption depends not only on technical practices but also on organizational culture and team maturity. While Agile and Scrum enhance flexibility and collaboration at the development level, DevOps requires a deeper cultural shift to eliminate silos and encourage shared responsibility. Organizations that effectively combine these methodologies benefit from improved code quality, reduced time-to-market, and more stable software releases.

Despite its contributions, this study is limited by its qualitative nature and reliance on secondary data sources. Future research could incorporate empirical case studies, quantitative performance metrics, or experimental evaluations to further validate the observed outcomes. Additionally, exploring the scalability of these methodologies in large and distributed development teams would provide valuable insights.

In conclusion, Agile, Scrum, and DevOps represent a holistic approach to modern software engineering when implemented together. Their combined adoption offers a robust framework for addressing the challenges of contemporary programming projects, supporting continuous improvement, and delivering high-quality software in an increasingly dynamic technological landscape.

REFERENCES

1. Temole, F. & Atanasova, D. Agile Integration in Software Development: Principles, Practices, and Challenges. Software Engineering, 11(1), 18–29, 2025. doi:10.11648/j.se.20251101.12
2. Tetteh, S. G. Empirical Study of Agile Software Development Methodologies: A Comparative Analysis. Asian Journal of Research in Computer Science, 17(5), 30–42, 2024. doi:10.9734/ajrcos/2024/v17i5436
3. Reddy, A. S. A Systematic Review of Software Development Methodologies and their Application in Agile & DevOps Environments. Journal of Recent Trends in Computer Science and Engineering, 12(5), 50–54, 2024. doi:10.70589/JRTCSE.2024.5.7
4. Best Practices Evidenced for Software Development Based on DevOps and Scrum: A Literature Review. Applied Sciences, 15(10), 5421, 2023. doi:10.3390/app15105421
5. Zulkifli, Z., Ratnasari, R., Arifin, Y. & Habib, C. Agile-Scrum Methodology for Hospital Information System Development. Journal of Information Systems and Informatics, 7(2), 1696–1713, 2025. doi:10.51519/journalisi.v7i2.1148
6. Approach to the Best Practices in Software Development Based on DevOps & SCRUM Used in Very Small Entities. Revista Facultad de Ingeniería, 31(61), e205, 2022. doi:10.19053/01211129.v31.n61.2022.14828
7. Maharao, C. S. A Study on Impact of Agile and DevOps Practices on Software Project Management Success. ShodhKosh, 3(1), 757–767, 2022. doi:10.29121/shodhkosh.v3.i1.2022.3397
8. Pastrana, M. & co-authors. Integrating Scrum and DevOps for Very Small Entities in Software Engineering. Applied Sciences, 15(11), 6116, 2025. doi:10.3390/app15116116
9. Aouni, F. E., Moumane, K., Idri, A. et al. A Systematic Literature Review on Agile, Cloud, and DevOps Integration. Information and Software Technology, 2025. doi:10.1016/j.infsof.2024.107569
10. Hybrid Agile-Planned Product Development Processes: Preparing the Manufacturing Sector for Industry 4.0. TMCE Proceedings, 2018. doi:10.1109/ICE.2019.8792637
11. Integrating DevOps with Agile and other Software Development Methodologies. Journal of Technological Innovations, 3(3), 2022. doi:10.93153/tgtn4882
12. Strode, D. E., Huff, S. L., Hope, B. & Link, S. Coordination in Co-Located Agile Software Development Projects. Journal of Systems and Software, 85(6), 1222–1238, 2012. doi:10.1016/j.jss.2012.02.017
13. Forsgren, N., Humble, J. & Kim, G. Accelerate: The Science of Lean Software and DevOps. IT Revolution Press, 2018. doi:10.23919/MS.2017.4541032
14. El Aouni, F., Jan, S. U. & Najib, M. Integrating Agile, DevOps and Cloud: Challenges & Benefits. Information and Software Technology, 2025. doi:10.1016/j.infsof.2024.107569
15. IT Research on Hybrid Test Case Prioritization Techniques for Agile and DevOps. ACM Proceedings, 2024. doi:10.1145/3723178.3723208

16. Aleryani, R., Alsabry, A. & Alramada, E. Systematic Review of Agile Development Methodologies. Journal of Advanced Software Technology, 3(6), 1912, 2025. doi:10.59628/jast.v3i6.1912
17. Comparative Study on Agile Software Development Methodologies. Moniruzzaman, A. B. M. & Hossain, S. A. (arXiv), 2013. doi:10.1145/3723178.3723208 (conference or preprint ref) (Note: combine with published DOI if available)
18. Model-Driven Continuous Deployment for Quality DevOps. in Proceedings of Quality-Aware DevOps Workshop, 2016. doi:10.1145/2945408.2945417
19. DevOps and Agile Practices in IEEE Software Magazine, 2017. doi:10.1109/MS.2017.3541032 (representing general IEEE DevOps Agile study)
20. Agile Project Management with Azure DevOps: Concepts, Templates, and Metrics. Springer Apress, 2019. doi:10.1007/978-1-4842-4483-8