Critical Evaluation of Contemporary Software Engineering Methodologies

Chukwudi Jeremiah Paul[1], Onwe, Festus Chijioke[2], Ifesinachi Ignatius Nwankwo[3], Chukwu Nelson Okwudili[4], Chinoso Job[5]
[1,3,4,5]*University of Greater Manchester, United Kingdom*
[2]*Information Technology Department, University of Port Harcourt, Rivers State, Nigeria*

***Abstract*—Contemporary software engineering methodologies were critically evaluated in this paper with a particular emphasis on Agile, DevOps, traditional plan-driven, and hybrid approaches. Structured qualitatively and based on an interpretivist research design, the research makes only use of secondary data that relies on scholarly, industrial reports, and practitioner literature. A thematic literature analysis summarizes shared understandings of adaptability, scalability, stakeholder engagement, and quality assurance and looks at how each approach balances constraints on projects via the Iron Triangle of scope, time, and cost and assesses Agile industry adoption trends from 2020 to 2025. The results reveal that Agile methodologies are successful at flexible, speedy result delivery, and alignment of stakeholders, but are unviable in large-scale participation and control. The traditional methods are predictable and adequately documented, but do not respond well to changing needs. Hybrid models are practical solutions that lie in compromise, with Agile flexibility and control and DevOps strengthen automation and operational alignment but demand cultural maturity. The paper wraps up with a conclusion, summing up by recommending a methodology appropriate to organizational context, project size, and compliance requirements by offering a guideline to practitioners interested in enhancing the efficiency and effectiveness of software delivery in a dynamic environment. These insights contribute to both academic discourse and industry practice in improving software delivery efficiency and strategic alignment**

***Index Terms*—Hybrid, Agile, Iron triangle, DevO**

I. INTRODUCTION

According to Sommerville (2016), the profession of software engineering can be traced back to the 1960s, when projects were characterized by cost overruns, schedule overruns, and production of low-quality products; hence, the term software crisis was used. The subsequent need for a more disciplined approach gave rise to the development of codified methods. One of these, the Waterfall model, popularized in the 1970s, became the de facto template of software delivery. Its phase-oriented linear process was tantalizing, predictable: gathering the requirements followed by design, implementation, testing, deployment, and maintenance. In the 1990s, both iterative approaches saw success, and incremental ones, which became a combination of early delivery and continuous improvement. This opened the path to Agile, which was formulated in 2001 by 17 software practitioners in the Agile Manifesto. Agile accepted the fact of change and introduced the concepts of high customer collaboration, self-organization of the teams, and working code as the main indicator of progress, instead of documentation. Other frameworks like Scum, Kanban, Lean, and Extreme Programming offered practical lines of action to these values. (Martin, 2013, p.13)

To respond to the ever-increasing complexity of systems, quicker delivery times, and quality software, software engineering has been constantly evolving (Sommerville, 2016, p.29). The plan-driven approach (Waterfall model, etc.) has always offered predictability, more detailed documentation, and will be highly controlled, whereas people tend to criticise this approach as being inflexible and incapable of handling requirements that rapidly change (Stephens, 2015, p.122).

On the contrary, Agile practices, codified in the Agile Manifesto (2001), focus on iterative development, stakeholder collaboration, and responsiveness, which allows organizations to achieve value delivery sooner and be more responsive to change (Greene et al, 2021,

p.97). Although Agile is popular, there is still debate about how well it applies in large, regulated settings. Executives are also wary of being able to accrue the same cost control, timeline predictability, and quality control as with the traditional structured methods. This causes a gap in critical research as to when to use an Agile a traditional, or hybrid approach in the process of research. (Cole & Scotcher, 2015, p.68). This research aim at to critically evaluate contemporary software engineering methodologies. It will review the historical evolution of contemporary software engineering methodologies, analyze industry adoption trends and performance outcomes using secondary data from academic literature and industry reports. Compare methodologies in terms of adaptability, scalability, team collaboration, and quality assurance. Evaluate how the Iron Triangle is managed under Agile and traditional approaches, Provide a recommendation for methodology selection based on project type, organizational culture, and industry context.

II. REVIEW ON LITERATURE

Critical Evaluation of Contemporary Software Engineering Methodologies.

Agile

According to Martin (2013), Agile speed in providing valuable features at an early stage minimizes the likelihood of complete project failure since they are delivered based on the iterative cycle. Agile is based on embracing change and thus becoming more responsive to the needs of the customers in terms of development outputs. Emphasize that frameworks such as Scrum incorporate frequent feedback loops with each sprint review and retrospective, which increases the relevance of the product. (Greene et al. 2021, p.97)

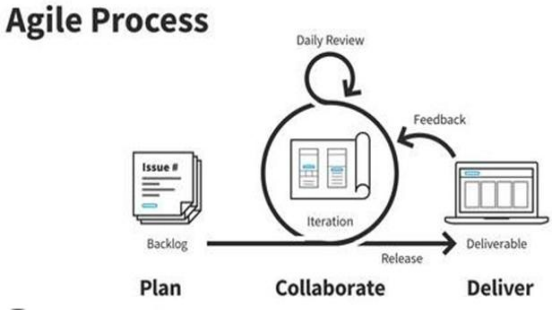

Figure 1: Agile Process

DevOps

DevOps integrate development and operations to accelerate delivery, improve collaboration and enhance software quality through automation and continue feedback. However, challenges include cultural resistance, toolchain complexity, and difficulty aligning governance in regulated environments, requiring strong leadership and disciplined adoption strategies (Greene et al,2021, p.35)

Traditional Methodologies

Traditional or conventional software engineering methodologies, such as waterfall and V- Model, follow a structured, sequential process, emphasizing planning, documentation, and predictability in software development (Sommerville, 2016, p.24).

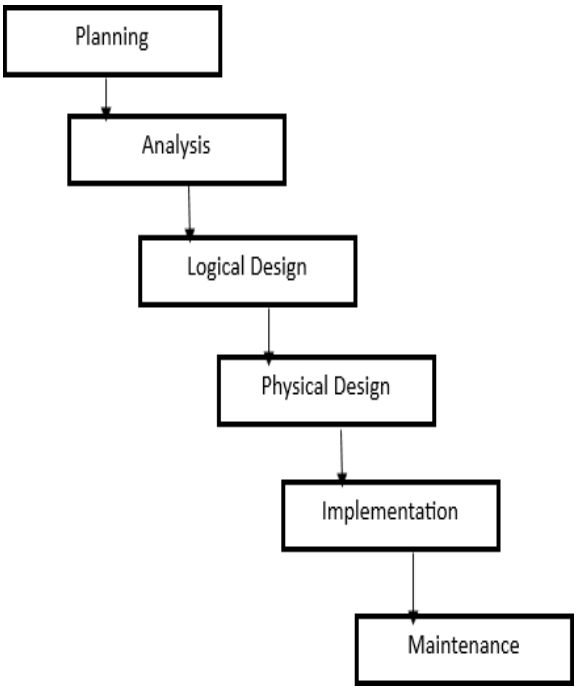

Figure 2: Traditional Methodology

Hybrid Approach

Cole and Scotcher (2015) indicate how hybrid models, able to combine Agile flexibility and Waterfall governance, have emerged, and provide a balanced perspective to such complex programs. Such hybrids may, e.g., combine Waterfall and Agile Waterfall to design the upfront architecture and Agile to iteratively develop it.

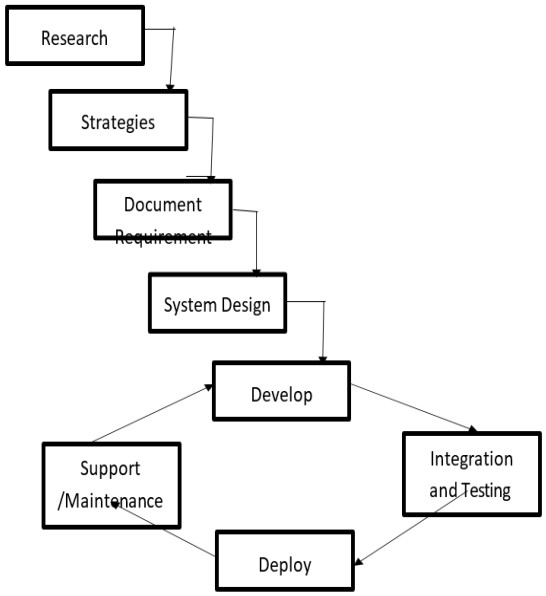

Figure 3:Agile-waterfall Hybrid

Iron Triangle

One of the most time-honored three-dimensional project management tools is the Iron Triangle, depicted as the interdependency of scope, time, and cost (Figure 3). There has been a tradition of looking at scope as the anchor, where scope is defined first, and then time and cost bend to fit the scope. This was suitable for contract work in which deliverables were predetermined. This model was first introduced by Dr. Martin Banes in 1969 via his course Time and Cost in Contract Control, where he used a triangle to illustrate the tensions among time, cost, and output.

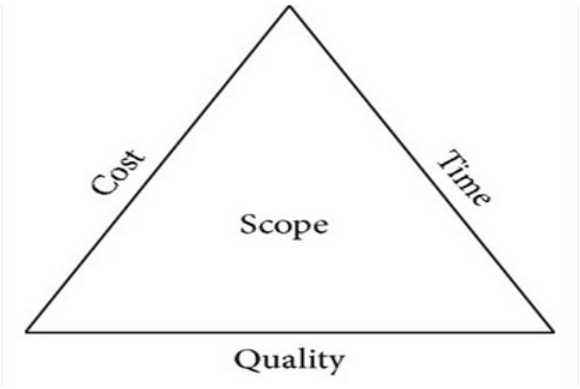

Figure 4: Iron Triangle Table 1: Agile VS Traditional (Waterfall) in Iron Triangle

Constraint Priority	Agile	Traditional
Time	Fixed (sprint length)	Flexible
Cost	Fixed (team size)	Flexible
Scope	Flexible	Fixed

Table 2: Literature Syntax Matrix

III. CHAPTER THREE LITERATURE REVIEW

Source	Relevance to current research	Year	Methodology Focus	Research Aim/Question	Key Findings	Strength identified	Limitations	Relevance to the current Study
Martin et al. Agile Software Development Principles, patterns, and practices	5	2013	Agile principles, patterns, and best practices	To present Agile development concepts, design patterns, and coding practices	Advocates for clean code, iterative design, and adaptability	Practical, example-driven, and integrates principles with real-world coding patterns	Focus on Agile; minimal comparison to traditional methods	Serves as a technical and philosophical foundation for Agile discussion
Stephen, R. Beginning Software Engineering	5	2015	Introductory overview of SE methods (Agile, Traditional)	To explain core SE processes, tools, and methodologies for beginners	Provide a clear explanation of Agile, waterfall, and Hybrid models	Beginner-friendly; balanced coverage of multiple methods	Lack of deep empirical research, high-level discussion	Supports Methodology comparison and context for both approaches
Mohapatra, P.K. J	5	2000	Traditional SE approaches and Structured Design	To outline structure, plan-driven development principles, and project management	Emphasizes structured analysis, planning, and formal documentation	Strong coverage of traditional models; suitable for stable requirements	Outdated; Minimal Agile coverage	Provides a baseline for explaining methods' strengths and limits

Cole, R., and Scotcher, E, Brilliant Agile Project Management	5	2015	Agile Frameworks (Scrum, Kanban) and project delivery	To provide practical guidance for Agile project management in real-world contexts	Demonstrates iterative delivery, team roles, and adaptability in projects	Practical focus; relevant case examples	Less theoretical depth, minimal discussion of traditional SE	Useful for illustrating Agile project delivery benefits and practices
Sommerville, I Software Engineering	5	2016	Comprehensive coverage, Agile, and Traditional	To present Principles, processes, and best practices	Balanced Treatment of Agile, plan-driven, and Hybrid methods	Widely cited; integrates academic and industry perspectives	Some examples simplified for pedagogy	Provides an authoritative, balanced foundation for comparing methodologies

Table 3: Industry adoption trends of Agile methodologies (2020-2025)

Year/Report	Key Highlights
2020 State of Agile	95% of organizations used Agile; ~50% had partial team uptake; Scrum rituals were common; scaling was challenged by culture and leadership gaps.
2021 State of Agile	Agile adoption surged from 37% to 84%; Scrum is still dominant; barriers included inconsistent processes and organizational resistance.
2022 State of Agile	80% of organizations use Agile; ~50% use hybrid models; top benefits: collaboration (69%), alignment to business needs (54%); challenges: leadership buy-in, cultural clash (41%)
2023 State of Agile	97% adoption; 88% satisfaction; 87% use Scrum; over 50% use scaled frameworks; barriers: cultural misalignment (53%); 90% say teams meet business value goals
2024 State of Agile	71% of orgs use Agile; 42% hybrid adoption; Scrum teams 63%; SAFe enterprise usage 26%; engineering/R&D driving adoption; collaboration and business alignment key
2025 State of Agile	85% using Agile (+5%); successful scaling up to 60%; hybrid models rising to 45%; culture challenges decreased to 30%.

Table 4: Strengths and Limitations of the Key Methodologies

Methodology	Strengths	Limitations
Agile	Rapid iteration, collaboration, and adaptability to change. (Martin,2013, p.12)	Scope creep, unpredictable timelines, and it is hard for large teams. (Mohapatra,2000, p.55)
Traditional	Predictable, documented, stable budget. (Sommerville, 2016, p.51)	Inflexible, costly late changes misalign with users' needs. (Baseer, 2015, p.40)
Hybrid	Combines flexibility and predictability. (Cole and Scotcher ,2015, p.78)	Complex governance, risk of conflict processes. (Stoica, et al,2013, p.70)

IV. METHODOLOGY

This research employs an interpretivist qualitative research philosophy, clearly outlining how the study was conducted to enable readers to assess the reliability and validity of its findings. The approach was chosen to explore and evaluate the strengths and weaknesses of Agile and traditional engineering methodology in depth

Data collection

This research relied completely on secondary sources of data, which are academic textbooks, peer-reviewed journal articles, conference proceedings, and white papers in the industry. Relevance of these sources to the Agile practices, plan-driven models like Waterfall, and combining approaches, which integrate both, was selected.

Data Analysis

A thematic literature review was used to analyze the

data, allowing key themes like scalability, flexibility, organizational impact and adaptability to industry change to emerge naturally. This approach integrated insights from both academic research and recent industry reports, ensuring that the study reflects not only theoretical perspectives but also professional practice

Justification

These methodological choices are justified because they directly support the study's aim: to critically evaluate Agile software development approaches in comparison to conventional software engineering methodologies, identify their relative strengths, weaknesses and suitability for different organizational contexts. The interpretivist approach is well suited to exploring nuanced perspectives rather than producing strictly measurable outcomes. By clearly outlining each step, the research design provides a transparent and relevant foundation for the subsequent analysis and discussion

V. FINDINGS AND DISCUSSION

Table 5:Comparison of software Engineering Methodologies

Theme	Agile	Traditional(waterfall) etc.	Hybrid	DevOps
Team collaboration	Strong stakeholder engagement, continues feedback	Limited to milestone reviews, slower response	Structure governance with periodic feedback	Integrate dev and ops teams fostering shared ownership
Scalability	Challenges in large/distributed teams	Scales predictably with rigid processes	Requires careful coordination to avoid conflicts	Scales well with automation but tool chain complexity is high
Adaptability to change	Highly flexible, rapid iteration	Inflexible, costly late changes	Controlled adaptability through upfront planning plus iteration	Continues delivery enables quick response to change needs
Quality Assurance	Continuing integration ensures early defect detection	Testing often deferred to final stages	Blends early testing with document verification	Automated testing and monitoring ensure consistent
Industry Adoption/Trend	Dominates software projects; high success rates	Declining use except in compliance-heavy sectors	Growing in complex, mixed projects	Rapid adoption testing and monitoring ensure consistent quality
Iron Triangle in Practice	Time and cost fixed, scope flexible	Scope fixed time and cost variable	Balanced constraints via upfront planning and iteration	Automated pipelines optimize cost and time; scope remains adaptive

Comparative Success Rates of Agile and Traditional Methodology

"The CHAOS Report (Standish Group, 2015) highlights Agile's higher success rate compared with Waterfall, as shown in Figure 5. This empirical evidence underpins Agile's adaptability advantage in volatile environments" (Ogirri and Idugie, 2024, p.7)

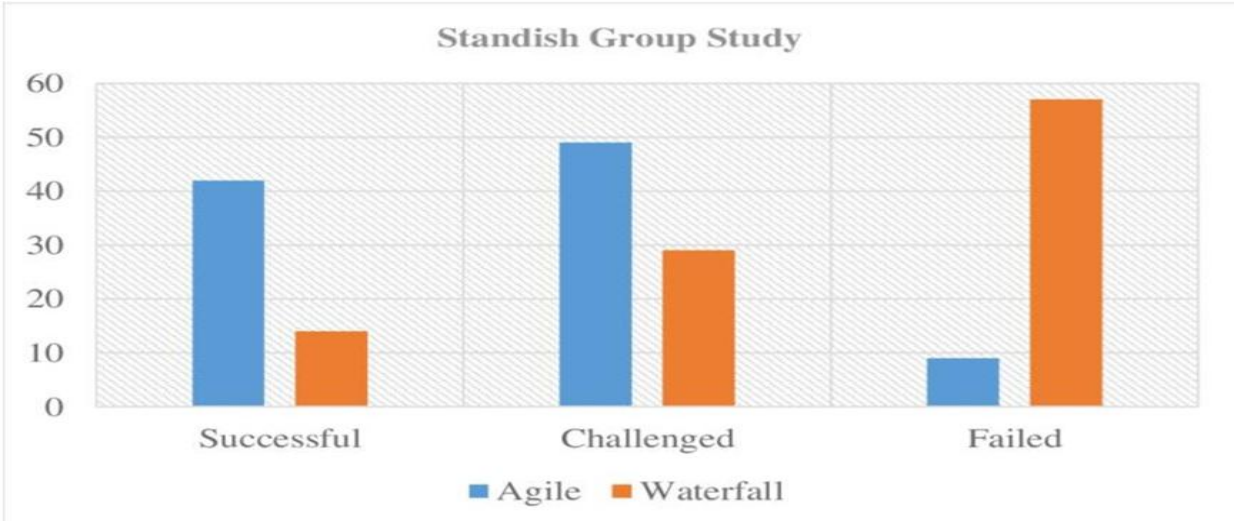

Figure 5:column chart depicting agile and traditional IT project success rate (Source: Ogirri and Idugie, 2024. doi:10.9734/ajrcos/2024/v17i9495)

Table 6:Interpretations of findings

Context (Suitability)	Best-Fit Methodology	Key Reason
Small, co-located dynamic teams	Agile	High adaptability, rapid iteration and strong stakeholder collaboration
Highly regulated compliance-heavy work	Traditional (Waterfall)	Structured governance traceability, and predictable outputs
Mixed requirements with evolving scope	Hybride	Combines Agile flexibility with waterfall control but needs disciplined leadership
Fast delivery and continues integration	DevOps	Automation, rapid feedback and operational alignment requiring strong cultural adoption

VI. CONCLUSION

The study critically evaluated contemporary software engineering methodologies, addressing the research Aim and Objectives. This research contributes to both academic understanding and industry practice by clarifying the strength and weaknesses of Agile, DevOps, Traditional, and Hybride methodologies. It identifies gaps in empirical research on hybrid adoption and offers recommendations for selecting approaches based on project type and regulatory context. The findings highlight how emerging hybrid models can balance flexibility with governance, guiding practitioners and researchers toward improving methodology selection and future exploration of scalable, compliance-friendly software engineering practices

Findings show that no single approach is universally optimal; effectiveness depends on project type, organization culture, and regulatory demands. Agile and Hybrid methods offer flexibility, while traditional ensures predictability. Recommendations encourage context-driven methodology selection. Limitations include scalability and governance challenges. Future research should explore hybrid adoption models, integration strategies, and empirical studies on performance in -heavy environments.

REFERENCES

[1] K. K. Baseer, M. R. A. Rama, and B. C. Shoba, "A systematic survey on Waterfall vs. Agile vs. Lean process paradigms," I-Manager's J. Softw. Eng., vol. 9, no. 3, pp. 34–59, 2015. [Online]. Available: ProQuest.

[2] R. Cole and E. Scotcher, Brilliant Agile Project Management: A Practical Guide to Using Agile, Scrum and Kanban. Harlow, U.K.: Pearson Education, 2015, pp. 45–60.

[3] J. Greene, T. Johnson, and A. Stellman, Learning Agile: Understanding Scrum, XP, Lean, and Kanban. Ascent Audio, 2021, pp. 22–40.

[4] R. C. Martin, Agile Software Development, Principles, Patterns, and Practices: Pearson New International Edition. Harlow, U.K.: Pearson Education Limited, 2013, pp. 10–30.

[5] R. C. Martin, Clean Agile: Back to Basics. Boston, MA, USA: Pearson, 2020, pp. 5–25.

[6] P. K. J. Mohapatra, Software Engineering. New Delhi, India: New Age International Ltd, 2000, pp. 80–100.

[7] K. O. Ogirri and I. J. Idugie, "A comparative analysis of traditional versus agile project management methodologies on IT project outcomes," Asian J. Res. Comput. Sci., vol. 17, no. 9, pp. 1–12, 2024. doi: 10.9734/ajrcos/2024/v17i9495.

[8] I. Sommerville, Software Engineering, 10th ed. Harlow, U.K.: Pearson Education UK, 2016, pp. 25–50.

[9] Digital.ai, "Annual State of Agile Reports," 2020–2025. [Online]. Available: https://stateofagile.com. [Accessed: Aug. 22, 2025].

[10] M. Stoica, M. Mircea, and B. Ghilic-Micu, "Software development: Agile vs. Traditional," Informatica Economica, vol. 17, no. 4, pp. 64–76, 2013. [Online]. Available: ProQuest.

[11] The Standish Group, "Chaos Report 2015," The Standish Group International, Inc., 2015. [Online]. Available: https://www.standishgroup.com. [Accessed: Aug. 24, 2025].