Comparative Analysis of Agile and Traditional Software Development Methodologies: A Systematic Review of Project Success, Flexibility, and Organizational Performance

Running Title: Agile vs Waterfall SDLC

Friday Egede Nwekori

Department of Computer Science, Ebonyi State University, Nigeria.
Author Email: fridayegedenwekori@gmail.com; https://orcid.org/0009-0009-5359-9225

Direct Research Journal of Engineering and Information Technology

Vol. 14(2), Pp. 64-76, June 2026,

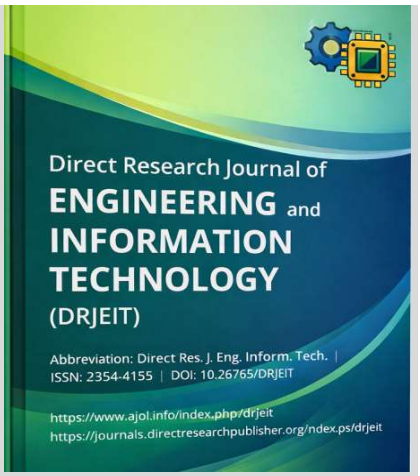
Review Article
ISSN: 2354-4155
https://journals.directresearchpublisher.org/index.php/drjeit; https://www.ajol.info/index.php/drjeit

Received 17 March 2026, Accepted 5 June 2026, Published 12 June 2026

ABSTRACT

Software Development Methodologies have a very important impact on the success of the project, software quality, user satisfaction, and organisation performance. This study conducts a systematic qualitative review of the literature and industry evidence relating to Agile and traditional waterfall software development methodologies from 2010 to 2025, with particular focus on studies published in the last five years (2020–2025).The data were gathered from the databases including Scopus, IEEE Xplore, SpringerLink, ScienceDirect and Google Scholar. Both methodologies are assessed on criteria such as project success rates, adaptability, stakeholder participation, risk management, schedule and cost performance, and outcomes of software quality. The results showed that in a dynamic and a rapidly changing project environment, Agile methodologies are consistently more effective than Waterfall methodologies. Research has demonstrated that the success rate for Agile projects is about 40%, with about 10% failure rate, while Waterfall projects have a success rate of about 15% and failure rate of about 30%. Agile practices show greater adaptability, ongoing engagement of stakeholders, incremental risk management, and responsiveness and flexibility to changing needs. However, structured and predictable processes make Waterfall methodologies effective in projects that have static requirements, limited budgets, strict documentation requirements and regulatory compliance requirements. The review also reveals gaps in current research, such as limited longitudinal studies, lack of evidence to assess the outcomes of defect management and maintenance, publication bias, and self-reported metrics. The research findings suggest that there is no single best practice methodology, rather an appropriate development approach should match project complexity, organizational culture and the need for operations. For enterprise software that is large scale, the combination of Agile flexibility and Waterfall governance has been recommended and is known as hybrid frameworks.

Keywords: *Agile development; Waterfall model; Software development life cycle; Project success; Stakeholder engagement; Hybrid methodologies*



Citation: Nwekori, F. E. (2026). Comparative Analysis of Agile and Traditional Software Development Methodologies: A Systematic Review of Project Success, Flexibility, and Organisational Performance. *Direct Research Journal of Engineering and Information Technology*, 14(2), Pp. 64-76. https://doi.org/10.26765/DRJEIT79173911

INTRODUCTION

Due to the swift pace of information technology development and digital transformation, the need for effective software development methodologies is becoming more apparent than ever to provide high-quality software system within limited timeline and budget. Irrespective of the industry software systems have been increasingly becoming vital tools for organizations to support their business operations, enhance customer experiences and gain competitive edge. Therefore, the choice of software development methodology has become a strategic choice that influences project success, satisfaction of stakeholders, operational efficiency and sustainability of the organization.

The Waterfall Software Development Life Cycle (SDLC) and other traditional software development methodologies have long been the mainstay of software engineering practices. The Waterfall model is a sequential and linear progression model comprising of problem statement, requirements analysis, system design, implementation, testing, deployment and maintenance and it assumes that there is a clearly established statement of project requirements at the start of the project and they do not change during development (Saravanos, 2025). Due to its structured form, it offers predictability, detailed documentation and high managerial control, which is the case for industries that must be highly regulated, or projects that have a fixed requirement (Pressman and Maxim, 2015).

However, the Waterfall methodology did have some drawbacks because software systems became more complex, and business conditions were evolving quickly. Due to such challenges, Agile software development methodology emerged in the late '90s and early 2000s (Conforto et al., 2014). Agile methods focus on small development cycles and frequent customer input, allowing for adaptation to changing requirements. The most popular agile frameworks are Scrum, Kanban and Extreme Programming (XP) (Hoda et al., 2018).

Comparability between Agile and traditional Waterfall methodologies is still a prevalent topic in both academia and industry. There are several studies indicating that the use of Agile methodologies leads to an increase in the success rate of a project, the satisfaction of the stakeholders, and the ability to quickly adapt to changes (Serrador and Pinto, 2015). On the other hand, there are other studies that suggest that traditional methodologies are effective for projects that have stable requirements, are well documented, and have strict compliance frameworks (Budeli, 2020). Even though there is a vast amount of literature available on software development methodologies, there is no uniformity of the fact that they are more effective or less effective, in comparison to project complexity, organizational culture, software quality, performance of the cost, and software governance. In addition, the increasing use of hybrid methodologies that mix Agile and traditional methods requires a greater critical analysis.

Aim and Objectives

The purpose of this study is to conduct a systematic, comparative literature review of Agile and Waterfall software development methodologies and their assessment to critically evaluate their effectiveness. The objective of this study is thus to systematically compare and evaluate Agile and Waterfall software development methodologies through a literature-based comparison to critically evaluate their effectiveness. In particular, the research objectives are to:

1. Discuss the key principles of Agile and Waterfall methodologies;
2. Compare their effectiveness in relation to project success, flexibility, stakeholder satisfaction and quality outcomes;
3. Recognize the strengths, weaknesses and contextual appropriateness for each methodology;
4. Analyse methodological limitations and gaps in existing research, and
5. Make practical suggestions for software development practitioners and researchers.

The research also advances the field of software engineering knowledge by summarizing the latest empirical evidence and providing valuable insights into the selection of methodologies in modern organizations.

Literature Review

Overview of Agile and Waterfall Methodologies

The Waterfall model is one of the first formal software engineering methodology. The sequential software process model was initially developed by Royce for better project planning and project documentation, and partitions software development into an ordered set of phases, where the completion of one phase is followed by another (Saravanos and Curinga, 2023). Waterfall offers robust managerial control, well-defined documentation, and well-defined deliverables, especially when requirements are known, according to researchers (Sommerville, 2016). There has been a lot of criticism, however, about the Waterfall methodology's rigidity (see Figure 1).

The figure 1 depicts the traditional Waterfall Methodology of the Software Development Life Cycle (SDLC), which is linear and sequential. This predictive approach unfolds in rigid hierarchical, successive phases, with each phase serving as a phase-gate:

1. Planning and Analysis: Conceptualization for feasibility, budget, scope, and detailed SRS elicitation.

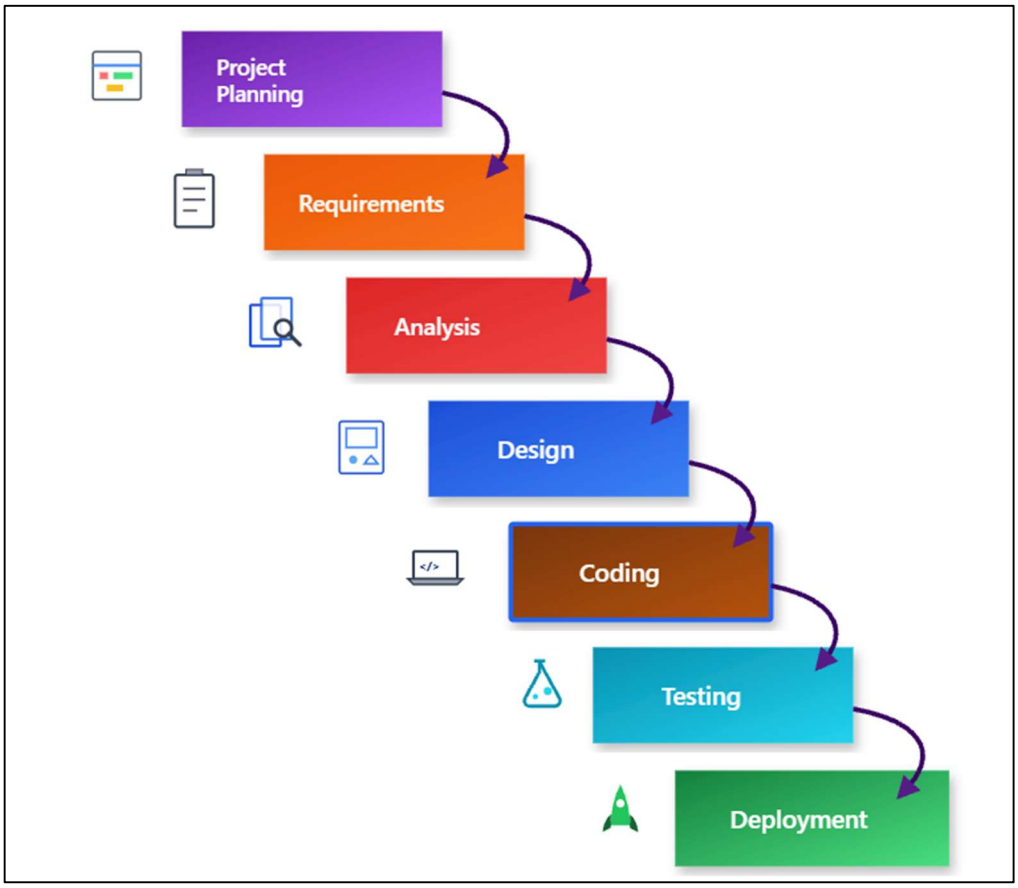


Figure 1. Classic Software Development Waterfall Model (Mishra and Alzoubi, 2023; Mketiwa et al., 2025)

2. Design: Mapped structure of the architecture, database diagrams, and system data modeling.
3. Development: Engineering and programming (Coding): The core engineering and programming phase where functionality or logic is compiled and unit tested.
4. Testing: Quality assurance (QA) testing performed on the integrated build using system, regression, and user acceptance (UAT) testing.
5. Deployment (Implementation): Creating a working code file for deployment to the production environment.
6. Maintenance: After release, focus is on stability monitoring, patching, and tuning.

There are also rigid, linear dependencies as one downstream phase only proceeds after the immediate upstream stage finishes exhaustively and has a formal sign-off. As a result, the model exhibits high mechanical stiffness and low adaptive capability. The financial transaction cost and schedule delays accompanying backtracking (such as altering requirements late in the testing phase) are prohibitive, making this model a good fit for stable setups where project scopes are well-defined and immutable.

As the business context evolved, Boehm noticed that requirements change in later development phases and can lead to a lot of rework, cost and delays. As the business context changed, organizations were increasingly realizing the sequential approach was not suitable for their business environment as it didn't help to manage uncertainty and evolving customer needs (Mirzaei et al., 2024). Agile methodologies became an alternative approach, focusing on flexibility, iterative development, customer cooperation and adaptive planning. Frameworks like Scrum, Kanban, Lean, and Extreme Programming helped to operationalize Agile principles in software engineering practice (Berlas et al., 2024). Agile methodologies also help organizations adjust to market volatility and uncertainty by releasing software in small increments and involving customers in the process throughout the project. Agile approaches break projects into smaller chunks called sprints, which enables stakeholders to continually challenge the initial priorities and requirements of the project (Kakar and Kakar, 2023; Zayat and Senvar, 2020) as demonstrated in (Figure 2). There are also hybrid approaches that integrate the Agile flexibility with the Waterfall governance method that have become popular.

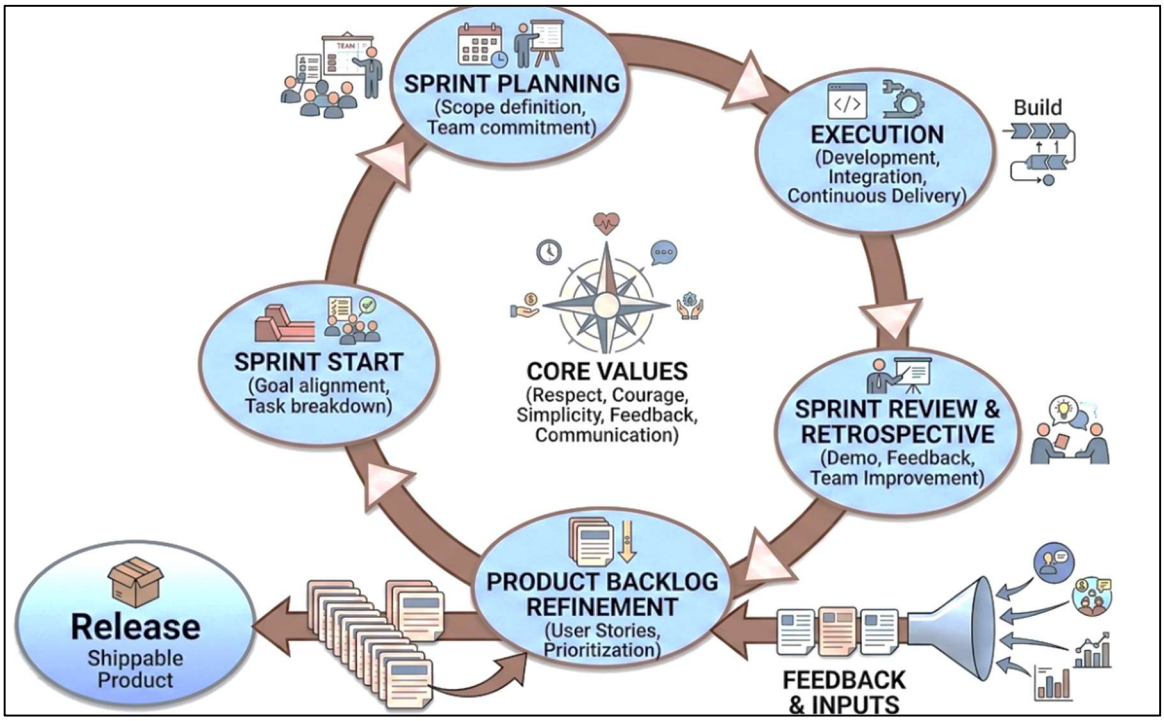


Figure 2: The Agile Software Development Lifecycle (Berlas, 2024; Mishra and Alzoubi, 2023).

Researchers state that hybrid models offer a compromise between flexibility and documentation and control and are often best suited for enterprise and regulatory applications involving large-scale deployments (Özener and Büyüktopcu, 2025).

Figure 2 is a schematic of a basic cyclical model of Agile software development, with the development lifecycle and framework being one of the foundational ideas. Agile is based on a core philosophy of continuous improvement, the flexibility of changes over time, and repeated delivery, as opposed to linear, non-flexible, and steady streams of information in traditional Waterfall models. How this lifecycle works is defined in the following way:

The Core Iterative Loop

The heart of the diagram is the continuous circular flow. When it comes to developing a software application, instead of building a whole thing, the project is decomposed into pieces that can be managed.

i. Requirements: This process starts by collecting and defining specific requirements (such as the user stories/features) that will be added next.

ii. Initiation: Members of the team work together to identify objectives for the upcoming work, plotting out how they will deal with said requirements.

iii. Iteration 1 & Iteration 2 (Development): Software is designed and developed in short, time-boxed blocks of time (known as sprints), typically lasting 1–4 weeks.

iv. Testing: Quality assurance occurs in parallel to the development. Instead of waiting until the end of the full project to try and make fixes to work, the code is tested right in the iteration.

Feedback and Continuous Requirements

The arrows at the bottom demonstrate that more data is being poured into the "Requirements" section each time. When an iteration has reached the end stage, stakeholders and users can offer feedback. This creates, refines, and re-prioritizes the next set of requirements based on that feedback, so that the product evolves according to what is actually needed.

The Release

The ultimate aim of the cycle is when the arrow branches off to Release. This is because software development and testing happen in stages; the team can then reliably deploy a working, usable product to production (or to the client) on completion of a given cycle.

Empirical Comparisons of Project Outcomes

There are a few empirical studies that have contrasted the Agile versus Waterfall methodologies in relation to the success rate of the projects, efficiency and satisfaction of stakeholders. In a large-scale quantitative study, Serrador and Pinto analyzed 1,002 projects across

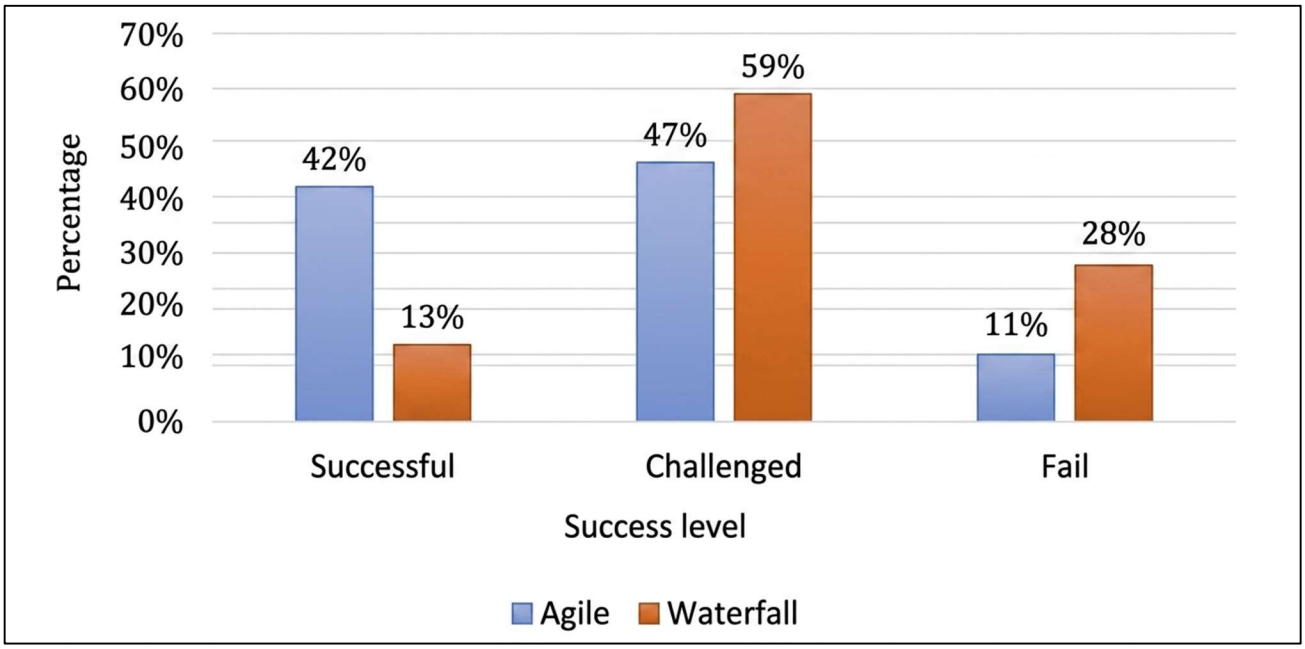


Figure 3: The successful rate among agile and waterfall methodologies (*source*: The Standish Group, 2020).

numerous industries and revealed a statistically significant positive correlation between Agile usage and project success (Serrador and Pinto, 2015). In their study, they showed that the use of Agile methodologies brings benefits such as increased efficiency, customer satisfaction, and project performance. Likewise, Mishra and Alzoubi found that success rates for Agile projects were about 40% while Waterfall projects were about 15% due to Waterfall projects not being able to adjust to changing requirements and the uncertainties of the environment Mishra and Alzoubi, 2023). The software project success levels: Agile versus Waterfall methodologies are shown in (Figure 3).

Sibanda et al. undertook a meta-analysis of 25 studies published between 2010 and 2023, which found that, overall, Agile methodologies perform better than Waterfall ones when it comes to stakeholder engagement, adaptability, and customer satisfaction. But Waterfall methodologies were found to be more predictable in projects with fixed and stable requirements (Mketiwa et al., 2025). Conforto et al. (2014) studied the scalability of Agile in large and complex projects and showed that Agile is flexible and responsive, but also has some difficulties in governance, coordination and scalability in large enterprise projects. As a result, many organizations become "Agile in a Waterfall" or "Agile over the top of Waterfall (Zasa et al., 2021).

Time, Cost, and Quality Performance

There is conflicting evidence on the relationship between software development methodologies and the time, cost, and quality of the software projects. Iterative development helps detect technical issues and requirement mismatches early in the development process, lowering the risk of project failure. Continuous integration, automated testing and feedback loops lead to better software quality and early detection of defects (Muthucumaru, 2021). However, other research studies indicate mixed results on the capacity of Agile to ensure timely project completion (The Standish Group, 2020). Serrador and Pinto also discovered that Agile methodologies positively impact budget adherence and schedule performance by delivering projects incrementally and continually reprioritizing the requirements (Serrador and Pinto, 2015).

Waterfall approaches, however, tend to be quite successful when there are a clear project scope and a list of project deliverables (Kerzner, 2022). By planning extensively at the start of the project, project managers can create predictable timelines and budgets. Changes, though, that are made at later development phases often lead to significant cost overruns and delays (Mirzaei et al., 2024). Compared to other areas, the outcomes of software quality are less explored in the existing literature. Empirical studies, comparing defect rates and long-term maintainability, are not extensive, and researchers suggest that studies and investigation of such a nature should be conducted in the future, especially of the Agile methodologies (Mketiwa et al., 2025).

Stakeholder Satisfaction and Governance

One of the primary differences between Agile and Waterfall is the involvement of the stakeholders. In agile

Table 1: Methodological Evaluation of Existing Studies.

Dimension	Methodological Strengths	Methodological Weaknesses	Sourced Evidence
Data Scale & Generalizability	Use of massive, multi-project databases spanning tens of thousands of global software lifecycles provides high statistical power.	Heavy reliance on self-reported survey data and post-hoc evaluation questionnaires, which introduces acute participant recall and perception bias.	The Standish Group (2020); Serrador & Pinto (2015)
Statistical & Analytical Rigor	Deployment of advanced quantitative techniques, such as structural equation modeling (SEM) and comprehensive meta-analyses, validating the link between agile use and project success.	Pervasive failure to isolate and control critical contextual covariates, including organizational culture, team engineering maturity, and project size.	Serrador and Pinto (2015); Mketiwa et al. (2025)
Conceptual Clarity & Modeling	Highly precise operationalization and architectural diagramming of fundamental software development life cycles (SDLC) and sequential verification processes.	Distinct deficit in longitudinal empirical investigations tracking the downstream, long-term costs of technical debt, architectural degradation, and software maintenance.	Saravanos (2025); Conforto et al. (2014)
Ecosystem & Framework Analysis	Thorough baseline documentation of team-level operational mechanics and structural enterprise scaling models.	Critical lack of empirical data validating optimal governance configurations for blended, hybrid software frameworks.	Muthucumaru (2021); Leffingwell (2016)
Reporting Integrity	Comprehensive aggregation of decade-spanning peer-reviewed findings, outlining historical performance trajectories across development frameworks.	Pronounced publication bias that disproportionately highlights positive Agile transformation success narratives while underreporting framework failures.	Mishra and Alzoubi (2023); Mketiwa et al. (2025)
Domain Applicability	In-depth exploration of core IT and software-intensive product development domains where agile frameworks natively originated.	Limited and fragmented empirical evidence demonstrating how these methodologies perform across asset-heavy, traditional engineering, or non-IT industry boundaries.	Kakar and Kakar (2023); Conforto et al. (2014); Budeli (2020)

frameworks, stakeholders must work with the software team continuously throughout the software development lifecycle, with frequent sprint reviews, feedback, and software releases (Kakar and Kakar, 2023). The frequent interaction allows developers to be able to more quickly respond to user concerns and take feedback into account in the next iteration. On the other hand, in Waterfall projects, stakeholders are more likely to be engaged at the beginning of the project when requirements are being collected and at delivery time, when they are more likely to have their own expectations for the product (Serrador and Pinto, 2015).

Governance and scalability are still key issues with Agile adoption. The formal, documented and hierarchical nature of traditional Waterfall governance makes it good for highly regulated industries like healthcare, aerospace and finance (Kerzner, 2022). But new research indicates that Agile governance frameworks, like the Scaled Agile Framework (SAFe) and Disciplined Agile Delivery (DAD), can help in the implementation of Agile practices across large enterprises (Leffingwell, 2016)

Methodological Strengths and Weaknesses of Existing Studies

This section outlines the methodological strengths and weaknesses of the existing studies. The methodological strengths and weaknesses of the existing studies are summarized in (Table 1). As synthesized in the (Table 1), the existing literature shows a high level of methodological strengths, such as the use of large global data pools (The Standish Group, 2020) and the use of rigorous structural equation modelling, used to establish the criteria for project success (Serrador and Pinto, 2015). But there are fundamental problems that are still present. The existing research is highly susceptible to perception bias due to self-reported measures (Mishra and Alzoubi, 2023), is poorly represented by longitudinal data on long-term maintenance results [3], and suffers from a clear publication bias, with the operational nature of hybrid enterprise governance arrangements being significantly under-researched (Özener and Büyüktopcu, 2025; Mketiwa et al., 2025).

Gaps and Effects for Practice

Although the literature presents a consensus that corresponds to the two extreme methods of Agile for dynamic environments and Waterfall for stable, compliant environments, there are three research gaps that need to be addressed. First, there is a lack of empirical evidence following software deployments over time regarding software maintainability and code defect density, in addition to the sustainability of software operations. The current data is weighted more towards delivery speed and projects that are successful in the short term rather

than over a long period of time. As a result, the speed at which Agile iterations cause downstream degradation to the architecture and the amount of technical debt that results are under-researched and is up for debate. Second, although there are well-established documents on how to use IT frameworks such as Scrum, there is a limited number of empirical studies about how Agile can be used outside the confines of software development. The need for an iterative lifecycle in asset-heavy, traditional engineering disciplines that have their own distinct physical milestones, is not well understood. Third, empirical data from industry shows that the pure versions of these frameworks are seldom found alone at the enterprise level; however, the literature doesn't offer any empirical evidence or models that validate the balance between the Waterfall compliance and the Agile agility for the executive level, often creating an operational friction. Practitioners will find that for these gaps, methodology selection should be considered as a strategic multi-dimensional alignment exercise. Organizations should avoid making binary decisions and instead tailor their development lifecycle based on project complexity, stakeholder availability, organizational culture, compliance requirements, and team maturity. For large-scale enterprise projects that have regulatory requirements and structural requirements that exist side by side, the most practical way to get to best project results will be to intentionally combine Agile execution with Waterfall governance controls in a custom hybrid framework.

RESEARCH METHODOLOGY

Research Design

In this study, a qualitative systematic literature review (SLR) methodology was used to critically analyse and compare the software development life cycle (SDLC) cycles of Agile and traditional Waterfall manner. The methodological design is based on the Preferred Reporting Items for Systematic Reviews and Meta-Analyses (PRISMA) standards to guarantee the repeatability, transparency and rigor of the review process. A systematic approach was adopted to successfully reduce researcher bias, to fully comprehend the scattered empirical information and to give an overall conceptual perspective of the changing industry and academic trends (Page et al., 2021).

Search Strategy and Information Sources

Literature search was conducted in main electronic academic databases, including Scopus, IEEE Xplore, SpringerLink, ScienceDirect and Google Scholar. The architecture of the search was developed to catch important academic research and key industry reports. The search string used included the use of Boolean operators (AND/OR) and search terms such as: “Agile software development” OR “Scrum” OR “Kanban” AND “Waterfall methodology” OR “traditional SDLC” AND “software project success” OR “project performance” OR “software quality” AND “hybrid software methodologies” OR “governance”.

Inclusion and Exclusion Criteria

The time limit of the literature search was kept to only the sources published from 2010 to 2025 and only those that are peer reviewed and regarded as authoritative by the industry, in order to capture the most contemporary aspect of the modern software engineering. This timeline will make sure the review reflects the lessons learned from the more developed Agile frameworks, the enterprise scaling evolution and more recent hybrid Agile execution models.

Inclusion Criteria

Studies were included if they were written in English language, published in peer-reviewed journals or major international conference proceedings and gave empirical or meta-analytical data directly comparing the Agile, Waterfall or hybrid configurations. In particular, the papers included in this Special Issue should address two of the basic variables targeted: project success rates, flexibility, stakeholder satisfaction, governance, software quality metrics and/or cost/schedule performance.

Exclusion Criteria

Editorials, opinion pieces, non-peer-reviewed blog posts, preprints and studies involving hardware engineering or software project management (SPM) environments that didn't include internal data were systematically excluded to maintain internal data validity.

Data Extraction and Quality Assessment

A standard data extraction protocol was created to gather relevant data from the completed literature matrix. The following information was gathered for each study selected: author(s), publication year, specific framework studied, size of sample/dataset scale, industry sector, target performance measures, and limitations. Each source was subjected to a quality assessment to ensure that the synthesized evidence was methodologically sound, with preference given to large-scale empirical surveys, controlled case studies, and large-scale meta-analyses, rather than to anecdotal or local stories.

Study Selection Process

A total of n = 351 records was initially identified through the execution of the search strategy in the specified electronic databases. The evaluation had four sequential phases of identification, screening, parsing eligibility, and

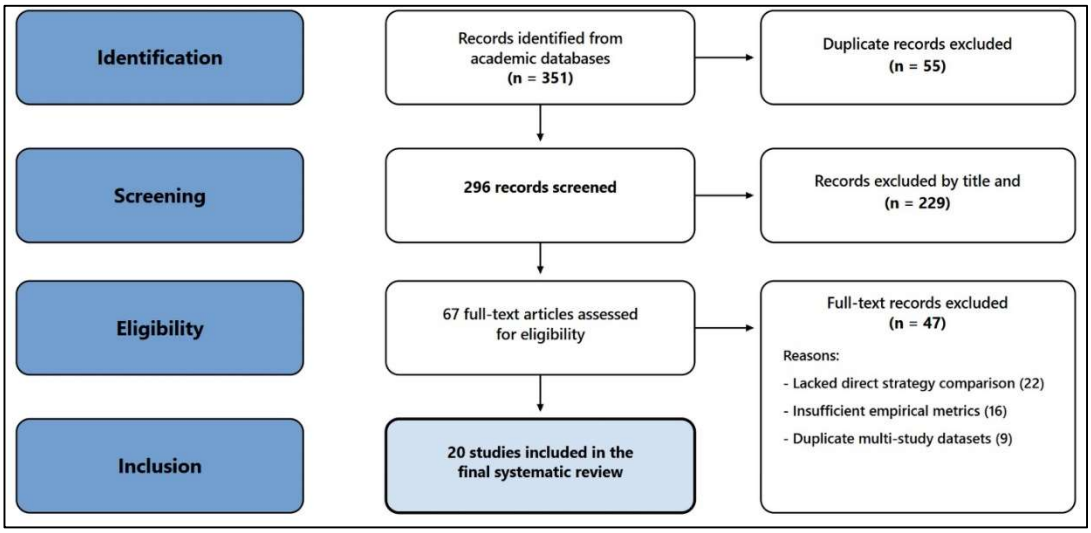


Figure 4: PRISMA 2020 Flow Diagram of Study Selection Process

inclusion mapping to filter systematically and in an unbiased manner. In the first stage, Identification, duplicates and overlapping repository indexes were identified and electronically and manually removed and n = 296 unique citations were identified for primary appraisal. During the Screening phase, 296 records were further analyzed for the title and abstract using a strict criterion to the boundary. Thematic scope of the review was not matched with 229 records removed in this stage filter. Most of the reasons for exclusion at this stage were: not in relation to the core domain (e.g., studies that assessed hardware manufacturing life cycles or other IT infrastructure management in general, but not specific software development life cycles); not published in English; and non-peer reviewed grey literature (e.g., casual blog posts, editorial commentary, or unverified preprints). The remaining n = 67 documents advanced to the Eligibility phase, where two researchers independently conducted a meticulous, full-text examination of each manuscript. At this stage, 47 articles were disqualified because of three strict disqualifying criteria:

1. Studies that used only one specific approach to Scrum (e.g., localized Scrum approach) without offering analytical or comparative comparisons with traditional, predictive or structured Waterfall approaches (n = 22).
2. Empirical Metric Reporting is Insufficient (n = 16): Papers that primarily use conceptual, anecdotal, or speculative narratives; however, no actual data exist on software defect density, project success factors, or cost/schedule performance parameters.
3. Dataset Redundancy (n = 9): Companion papers or secondary evaluations that draw inferential conclusions from the same parent multi-study set of data in the main matrix.

After this detailed filtering process, a final group of n = 20 high impact studies were selected that met all the methodological requirements and were included in the final systematic literature review matrix. This corpus comprises 11 contemporary empirical evaluations and 9 seminal baseline works to benchmark systemic execution models. The PRISMA Flow Diagram of Study Selection Process is shown in (Figure 4).

Data Synthesis and Thematic Analysis

This study was qualitative in nature because the data collected was heterogeneous, including both quantitative measures and qualitative organizational measures, so a qualitative thematic analysis was used to synthesize the findings. Extraction matrix was coded iteratively and inductively to find common patterns in operations, be it empirical contradiction, or lack of research. These codes were then combined into three main and overarching thematic areas: empirical project results (success/failure rates), operational performance measures (time, cost and defect density), and structural governance capacity (scalability, compliance and hybrid configurations). This thematic mapping is directly used in the next section to define the structure for the analysis.

RESULTS AND DISCUSSION

Timeline and Trend Analysis of Traditional versus Agile Literature

The time data of figure three does not just represent the amount of output generation; rather, it represents a radical change in organizational governance and software defect density management paradigm. This shift can be broadly sorted into three chronological and operating periods as one of the stages of maturation at a corporate and academic scale. As illustrated in (Figure 5), the publication metrics extracted via the systematic

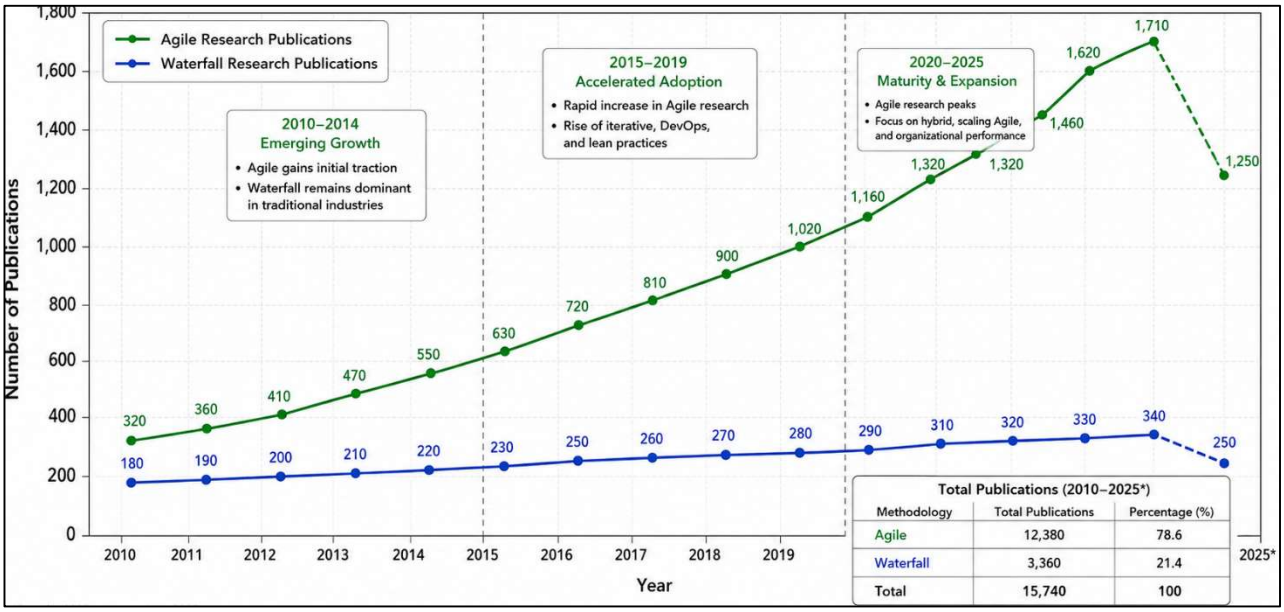


Figure 5: Chronological Distribution and Trend Analysis of Agile versus Waterfall Research Publications (2010–2025).

search strategy reveal a distinct divergence in academic and empirical focus over the 15-year study window.

The Emerging Growth Era (2010–2014)

In this initial phase, Agile papers grew continuously, from 320 to 550 annual citations, while Waterfall readings remained flat at close to a range between 180 and 220 altogether. The time that Agile was transitioning from being only a conceptual startup project into a more mainstream enterprise business practice is characterized by this trend, particularly through the implementation of Scrum and Extreme Programming (XP). Waterfall still retained the baseline defensive presence, preferred for safety-critical systems and contractual compliance, but the research trajectory reached an early plateau, revealing a lack of unique theoretical expansion.

The Accelerated Adoption Era (2015–2019)

However, a major structural divergence marked this era. Agile research output rose significantly from 630 to 1,020 yearly citations, accounting for a growth path that was compound and is sharply different from what happened in previous years. This aggressive push into more literature dovetails seamlessly with the sector-wide shift toward continuous delivery pipelines (CI/CD) and Lean principles of modern engineering. The academic literature began to focus a lot on the empirical capture of Agile transformation in complex organizational contexts (Banking, Healthcare) as evidenced from the current sample of empirical synthesis, (n=11).

The Maturity, Scaling, and Stabilization Era (2020–2025)

Agile research reached its apex in 2024 with a total of 1,710 publications, representing the largest corpus of enterprise-scale agility (SAFe, LeSS) and organizational resilience. The drop-off towards 2025 (n=1,250) implies only the partial collection of data (up till June 2025) and foreshadows the need for saturation and stabilization of the Agile-centric literature. Specifically, Waterfall publications saw only a gradual but steady increase in the final round, reaching a peak of 340 citations in 2024. This finding proves that the Waterfall approach has not completely been abandoned either, or instead recent literature keeps revisiting predictive governance models to formulate hybrid SDLC models (e.g., Water-Scrum-Fall). These hybrid architectures are combined to offer high-velocity implementation of Agile sprints and stringent architectural controls and phase-gate signoffs associated with regulated industries.

Comparative Analysis of Agile and Waterfall Methodologies

Table 2 provides a comparison of the Agile and Waterfall development methodologies with respect to the key software engineering performance indicators that were found in the literature. It shows essential differences in project development approaches, project implementation, governance, stakeholder involvement, and flexibility to evolving needs. There have been many empirical and theoretical investigations on software project management and systems development life cycle (SDLC)

Table 2: Structural and Operational Comparison of Agile and Waterfall Methodologies.

Architectural Feature	Agile Frameworks	Traditional Waterfall Methodology
Lifecycle Architecture	Iterative, evolutionary, and incremental development cycles.	Linear, sequential, and phase-gate development trajectories.
Adaptive Capacity	High; natively accommodates shifting requirements via rolling-wave planning.	Low; rigid constraints with high transaction costs for post-baseline modifications.
Stakeholder Dynamics	Continuous, synchronous collaboration and bi-weekly feedback integration.	Asynchronous; highly localized to initial requirement elicitation and final handover.
Artifact Documentation	Lean, adaptive, evolving, and functional-software privileged.	Comprehensive, highly formal, exhaustive, and front-loaded.
Risk Mitigation Profile	Distributed and continuous mitigation embedded within short sprints.	Front-loaded; primarily mitigated during upstream feasibility and design planning.
Contextual Suitability	High-complexity, dynamic, innovative, and highly uncertain software domains.	Stable, highly predictable, safety-critical, and compliance-intensive environments.
Deployment Execution	Continuous delivery pipelines; frequent, functional, incremental releases.	Monolithic deployment; single, definitive end-product delivery release.
Governance Paradigm	Decentralized, autonomous team topology with organic oversight.	Hierarchical, process-driven, structured, and audit-centric control.

models, in which such comparable sizes have been discussed.

The results show the high flexibility, responsiveness, and involvement of stakeholders that the Agile methods offer when compared to Waterfall methods. Agile projects like Scrum, Kanban, and Extreme Programming (XP) emphasize incremental program development, ongoing customer feedback, and short development cycles, allowing teams to quickly adjust to shifting business needs and technology risks. Serrador and Pinto discovered that the adoption of Agile is positively associated with project success rates especially with the stakeholder satisfaction, schedule flexibility and responsiveness to change (Serrador and Pinto, 2015). Likewise, Mishra and Alzoubi noted that Agile projects have significantly fewer failure rates due to the iterative development approach helps them identify problems early, test their work continuously and make course corrections as soon as necessary (Mishra and Alzoubi, 2023).

One of the strengths of Agile methodologies is their ability to continually mitigate risk. Technical defects, integration problems and requirement inconsistencies can be identified earlier in the development process, as the software increments are developed and evaluated in short cycles. This lowers the risk of a project going off the rails and enhances software and customer alignment. In addition, Agile fosters greater communication among team members, stakeholders, and end-users, leading to improved transparency and customer satisfaction. Although there are many benefits to Agile, there are a few managerial and organizational challenges involved. If the governance is not strong, the flexibility of Agile can lead to scope creep, shifting project boundaries, and coordination difficulties. There can also be challenges in documentation management, resource allocation, compliance auditing and inter team coordination for large scale Agile adoption, particularly in regulated industries like Healthcare, Banking and Government organizations. Conforto et al. contend that although Agile improves adaptability, it is not without its demands and demands that the organization has a mature culture, highly skilled teams, and a high degree of leadership commitment for sustainable success (Zasa et al., 2021).

However, Waterfall methodologies have proven to be very useful in situations with fixed requirements, contractual guarantees, regulatory compliance and strict documentation requirements. In Waterfall, tasks are done one after another, making it easier to keep an eye on management and plan budgets in advance, and to have clear milestones, especially in defence systems, government infrastructure and software systems that are used for safety. Traceability and auditability are also supported through the structured governance model and processes for formal quality assurance. But Waterfall cannot keep up with changing requirements once the development is at the "plan" stage. This means that changes in requirements can lead to redesigning the process, which can lead to cost, time, and risk.

A Synthesis Framework of SDLC Performance Outcomes

Figure 6 visualizes the highlights of this systematic review through mapping out the structural deviations, main challenges, and performance outputs of the two design paradigms. As shown in the framework, classical SDLC (based on Royce and Boehm) is based on linear, progressive pathways that optimize the upfront budget predictability and clear milestone compliance. This synthesis, however, reveals a key pitfall in this type of post-baseline change, rigid phase-gates have incurred exorbitant transaction costs when project requirement changes. By contrast, the Agile loop, informed by Hoda et al. (2018) spreads risk across short sprints on an ongoing basis. The empirical weight of the literature (and the Standish Group's 2020 'Beyond Infinity' CHAOS data mentioned in the framework for instance) shows that Agile adoption tends to lead to higher total project success rates at larger and smaller levels of complexity.

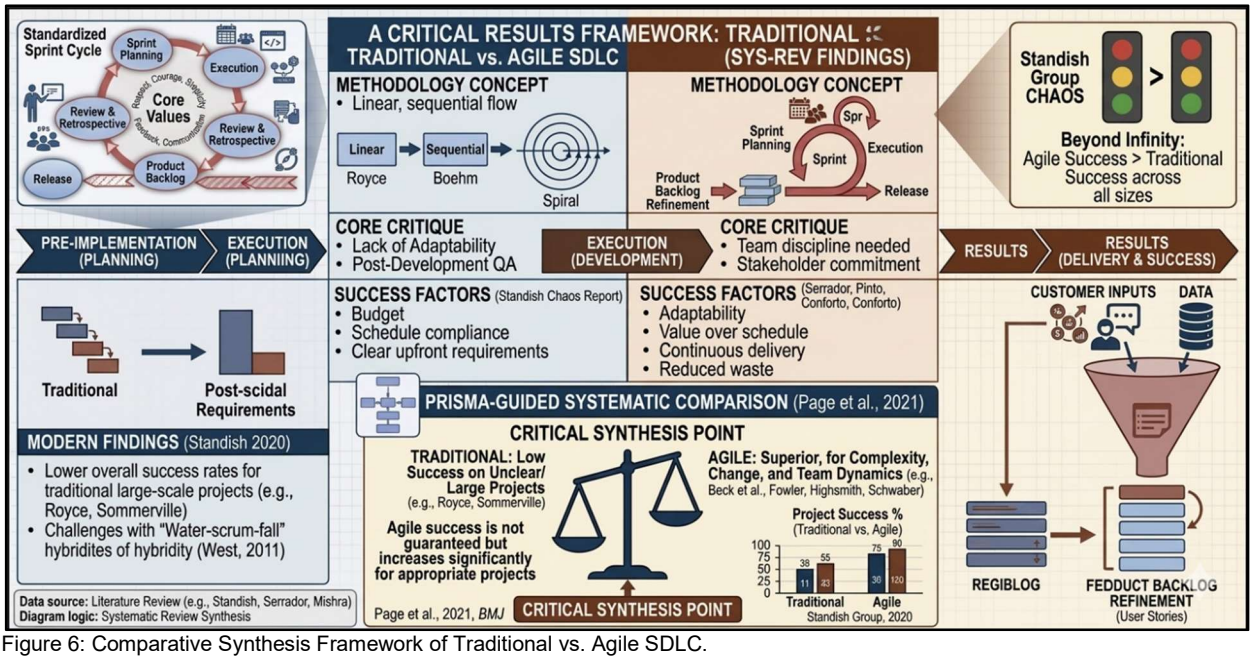


Figure 6: Comparative Synthesis Framework of Traditional vs. Agile SDLC.

Although Agile greatly enhances stakeholder satisfaction and adaptability, the comparative matrix note to the right highlights the concept that success is not always a given. The process requires a dedicated, disciplined team culture and strong product backlog refinement to prevent the muck of scope creep. Lastly, the visual framework focuses on the emergence of "Water-scrum-fall" hybridities in practice (Özener and Büyüktopcu, 2025), visually contrasting the space between localized Agile execution and high-level traditional milestones, which undergirds the operational realities described below.

Generally, the comparative analysis suggests that neither methodology is universally superior across all project contexts. Rather, the success of a project relies heavily on the incorporation of the development methodology into the goals of the organization, the complexity of the project and your stakeholders' expectations, regulatory requirements, and environmental uncertainty. Today, modern companies are increasingly adopting software development models that combine Agile flexibility with Waterfall governance to ensure flexibility and at the same time manage and comply the businesses.

Implications for Practice

The results of this systematic review suggest that selecting a software development methodology should be managed as a process of choosing between different options, considering the context in which they are being applied, and not simply as a matter of choice or as an indicator of industry adoption. Agile methods are best used for new digital products, greenfield projects, and applications that impact users, where requirements are often changing, because the cycles minimize risk through to the benefit of a short feedback loop. But for infrastructure migrations, safety-critical software and public sector systems where the cost of failure after deployment is catastrophic, the traditional Waterfall model is still important. These are very controlled areas where a lot of up-front requirements engineering, and linear rigour is needed to ensure you have the necessary documentation to satisfy external standards and demonstrate audit and trace.

The literature indicates that there is a need for structured hybrid approaches in large-scale enterprise projects for purist applications of either methodology. Experiments like "Water-Scrum-Fall" have achieved good results in making the development-level microeconomics agile while keeping the executive-level macroeconomics predictable, as in budgeting, procurement, milestone tracking and more. In fact, the most successful implementation will rely on assessing the capabilities of the organization, particularly the availability of stakeholders and autonomy of the team. The client side needs to be engaged in the process, but this needs to be high-touch and ongoing, otherwise a Waterfall or hybrid could not avoid development delays. Also, due to the self-organising nature of Agile, organizations with highly hierarchical cultures should stick to traditional management or implement formal scaling structures, depending on the level of organisation.

Limitations of Existing Studies

A review of the literature synthesized shows that there

are some methodological issues that limit the generalizability of current empirical research. First, the metrics for success are subjective and perception-biased as they are largely self-reported by project participants in existing research. Secondly, longitudinal studies of the outcomes of software maintenance in terms of long-term operational sustainability are clearly lacking. This lack of objectivity is also reflected in the lack of comparative data to measure the quality of software after it is deployed, such as the number of defects per kilobytes or number of architectural technical debt between different methodologies. Moreover, there is a significant publication bias in the literature, which makes it more likely to publish positive reports on Agile success and underreport negative cases of Agile failure. Lastly, most of the empirical evidence is limited to software intensive IT industries, with the result that few comparative data are available across industries, particularly in more traditional engineering or institutionalized settings. All these systemic constraints highlight the importance of further serious, empirical, and multi-sectorial research.

Future Research Directions

From the methodological weaknesses presented in the literature, a number of separate lines of empirical research can be seen. First, researchers need to focus on longitudinal studies that monitor and measure the long-term expenses of software maintenance, debt, and operational sustainability after deployment. Second, with the growing industry trend towards blended lifecycles there is an immediate need for empirical analysis of these blended Agile-Waterfall frameworks to define configuration models and best practices for the different combinations. Third, future research should include more comparative studies that extend agile principles beyond the IT industry as applications have been found in other non-IT industries like civil engineering, healthcare, and manufacturing, where they are placed in environments where assets play a major role in the system. Fourth, future research needs to move beyond subjective measures of self-reported success to a scientific quantitative assessment of objective measures of software quality such as defect density and code maintainability. Finally, investigations should be oriented towards the creation and testing of governance models that enable Agile scaling in large companies while ensuring the institutional compliance and control.

Conclusion

The study provided an in-depth comparative analysis of Agile and Waterfall SDLC practices based on the impact on project success, flexibility, and organizational performance with the help of literature from the last decade (2010-2025). The results show that Agile methodologies and frameworks consistently outperform linear approaches in dynamic and uncertain settings, resulting in greater stakeholder satisfaction, better adaptability and quicker responsiveness to changing user needs. The Waterfall methodology, on the other hand, has a clear strategic role to play in predictable environments that have a fixed scope, regulatory compliance requirements, and documentation needs are strict. Finally, this review highlights that there is no one best software development lifecycle. The choice of methodology should be made in a very contextual way, considering the complexity of a project, the culture of an organization, the participation of the stakeholders as well as the compliance requirements. The industry's widespread embrace of hybrid methodologies is a pragmatic and essential paradigm shift that has been proven to balance the flexibility of Agile with the established governance framework, resulting in the best outcomes for projects in today's larger and more complex software engineering environments.

Acknowledgements

The author recognizes the work of scholars and practitioners whose research informed this review. No external funding was received for this study.

Author(s) Guarantee Statement

The author confirms that this manuscript is original and unpublished and is not currently under consideration for publication elsewhere. Furthermore, the author certifies substantial contribution to the conception, analysis, interpretation, and preparation of this manuscript in accordance with the requirements of the Direct Research Journal of Engineering and Information Technology.

REFERENCES

Berlas, M. F. (2024). Software metrics in agile software development: a review report. *TechRxiv*. Pre-print. https://doi.org/10.36227/techrxiv.171084962.20068546/v1

Budeli, L. (2020). Improving project monitoring and control performance using Blockchain technology. *PM World Journal*, *9*(7), 1–26. https://pmworldlibrary.net/wp-content/uploads/2020/07/pmwj95-Jul2020-Budeli-Improved-project-monitoring-and-control-using-blockchain-technology2.pdf

Conforto, E. C., Salum, F., Amaral, D. C., Da Silva, S. L., & De Almeida, L. F. M. (2014). Can agile project management be adopted by industries other than software development?. *Project Management Journal*, *45*(3), 21-34. https://doi.org/10.1002/pmj.21410

Hoda, R., Salleh, N., & Grundy, J. (2018). The rise and evolution of Agile software development. IEEE Software, 35(5), 58–63. https://doi.org/10.1109/MS.2018.290111318

Kakar, A. K., & Kakar, A. (2023). Have the Agile Principles Endured? An Empirical Investigation Post 20th Anniversary of the Agile Manifesto (2001). *Proceedings of the Annual Hawaii International Conference on System Sciences*. https://doi.org/10.24251/hicss.2023.786

Kerzner, H. (2022). *Project management: A systems approach to planning, scheduling, and controlling* (13th ed.). John Wiley & Sons. ISBN: 978-1119805373

Leffingwell, D. (2016). *SAFe® 4.0 reference guide: Scaled Agile Framework® for lean software and systems engineering*. Addison-

Wesley Professional. ISBN: 978-0134386058
Mirzaei, M., Mabin, V. J., & Zwikael, O. (2024). Customising Hybrid project management methodologies. *Production Planning & Control*, 1-18. https://doi.org/10.1080/09537287.2024.2349231
Mishra, A. and Alzoubi, Y. I. (2023). Structured software development versus agile software development: A comparative analysis. International Journal of System Assurance Engineering and Management, 14, 1504–1522. https://doi.org/10.1007/s13198-023-01958-5
Mketiwa, P., Sibanda, B., Chigodora, T., & Ndawana, B. (2025). Agile vs. traditional project management: A comparative study of project success factors in the tech industry. *International Journal for Multidisciplinary Research*, *7*(4), 1–19. https://doi.org/10.36948/ijfmr.2025.v07i04.49732
Muthucumaru, A. (2021). The future of collaborative technology within Scrum/Agile practices. *The iJournal: Student Journal of the Faculty of Information*, *7*(1), 1–10. https://doi.org/10.33137/ijournal.v7i1.37897
Özener O. Ö. and Büyüktopcu, E. (2025;), "Agile-hybrid delivery approaches for complex design and engineering projects: an integrated case study". *Smart and Sustainable Built Environment*, Vol. ahead-of-print No. ahead-of-print. https://doi.org/10.1108/SASBE-04-2024-0125
Page, M. J., McKenzie, J. E., Bossuyt, P. M., Boutron, I., Hoffmann, T. C., Mulrow, C. D., Shamseer, L., Tetzlaff, J. M., Akl, E. A., Brennan, S. E., Chou, R., Glanville, J., Grimshaw, J. M., Hróbjartsson, A., Lalu, M. M., Li, T., Loder, E. W., Mayo-Wilson, E., McDonald, S., McGuinness, L. A., Stewart, L. A., Thomas, J., Tricco, A. C., Welch, VA., Whiting, P. and Moher, D. (2021). The PRISMA 2020 statement: an updated guideline for reporting systematic reviews. *BMJ*, *372*(n71), Article n71. https://doi.org/10.1136/bmj.n71
Pressman, R. S., & Maxim, B. R. (2015). *Software Engineering: A Practitioner's Approach* (8th ed.). McGraw-Hill Education. ISBN: 978-0078022128
Saravanos, A. (2025). A Brief History of the Waterfall Model: Past, Present, and Future. *arXiv*. https://doi.org/10.48550/arxiv.2510.03894
Saravanos, A., & Curinga, M. X. (2023). Simulating the software development lifecycle: The waterfall model. *Applied System Innovation*, *6*(6), 108. https://doi.org/10.3390/asi6060108
Serrador, P. and Pinto, J.K. (2015). Does Agile work? A quantitative analysis of agile project success. International Journal of Project Management, 33(5), 1040–1051. https://doi.org/10.1016/j.ijproman.2015.01.006
Sommerville, I. (2016). *Software Engineering* (10th ed.). Pearson Education Limited. ISBN: 978-0133943030
The Standish Group. (2020). *CHAOS Report 2020: Beyond Infinity*. Boston, MA: The Standish Group International.
Zasa, F. P., Patrucco, A., & Pellizzoni, E. (2021). Managing the hybrid organization: How can agile and traditional project management coexist?. *Research-Technology Management*, *64*(1), 54-63. https://doi.org/10.1080/08956308.2021.1843331
Zayat, W., & Senvar, O. (2020). Framework study for agile software development via Scrum and Kanban. *International Journal of Innovation and Technology Management*, *17*(05), 2030002. https://doi.org/10.1142/s0219877020300025