Software Engineering
2025, Vol. 11, No. 1, pp. 18-29
https://doi.org/10.11648/j.se.20251101.12

Review Article

Agile Integration in Software Development: Principles, Practices, and Challenges

Felix Temole[1, *], **Desislava Atanasova**[2]

[1]Department of Computer Science, University of Ruse "Angel Kanchev", Ruse, Bulgaria
[2]Department of Informatics and Information Technologies, University of Ruse "Angel Kanchev", Ruse, Bulgaria

Abstract

Agility in software development has evolved from a niche methodology to a mainstream paradigm that enables organizations to respond rapidly to changing market demands and customer expectations. This paper explores the integration of agility into software development through modern agile methodologies such as Scrum, Extreme Programming (XP), and Kanban. It highlights the core benefits of agility, including improved adaptability, faster delivery cycles, and enhanced customer satisfaction, while also addressing persistent challenges such as documentation gaps, progress measurement, and organizational resistance to change. Beyond current practices, the paper examines the future trajectory of agile development, emphasizing the growing influence of emerging technologies such as artificial intelligence (AI) and DevOps. These technologies are reshaping agile workflows by enabling greater automation, predictive analytics, and continuous delivery. Drawing on recent scholarly and industry research, the paper outlines best practices for successful agile transformation, including the alignment of agile practices with strategic business goals, the cultivation of a supportive organizational culture, and the use of modern tools to enhance transparency and performance analytics. By synthesizing empirical findings and forward-looking insights, this study provides a comprehensive roadmap for agile adoption and continuous improvement, offering valuable guidance for practitioners, managers, and researchers aiming to build resilient, customer-centric software systems in increasingly complex and technology-driven environments.

Keywords

Agility, Agile Software Development, Agile Methodologies, Flexibility, Adaptability, Iterative Development, Software Engineering

1. Introduction

Digital transformation presents companies with the challenge of developing software products with greater speed and flexibil-ity, along with greater customer orientation. Traditional development models such as the waterfall model are increasingly reaching their limits. In this context, agile methods have emerged as an effective response: they enable iterative development processes, promote continuous feedback, and enhance adaptability to changing market and customer requirements [1].

Agility refers to an organization's ability to respond dynamically and effectively to change, particularly in complex and unpredictable innovative processes. In software devel-

*Corresponding author: felix.temole@capgemini.com (Felix Temole)

Received: 31 July 2025; **Accepted:** 30 August 2025; **Published:** 19 September 2025

opment, this agility is operationalized through specific methods such as Scrum, Kanban, or Extreme Programming. Scrum, for instance, structures work into sprints and promotes regular reflection and adaptation, while Kanban focuses on visualizing workflows and limiting work in progress. Extreme Programming (XP), on the other hand, emphasizes technical excellence through practices such as pair programming and test-driven development. These emphasize flexibility, transparency, collaboration, and continuous improvement, and are especially suitable for IT projects with a high degree of innovation and technical complexity [2, 48].

The aim of this paper is to systematically analyze the role of agile methods in modern software development. It identifies key advantages, such as enhanced adaptability, accelerated delivery, and improved customer satisfaction, as well as persistent challenges including documentation gaps, progress tracking, and organizational resistance. Based on recent empirical and industry research, the paper derives practice-oriented recommendations for successful agile transformation. Furthermore, it provides a forward-looking perspective on emerging trends, particularly the rise of hybrid development models that blend agile and traditional approaches, and the integration of enabling technologies such as Artificial Intelligence (AI) and DevOps — a set of practices, principles, and cultural philosophies designed to unify software development (Dev) and IT operations (Ops) — thereby extending agility across the entire software lifecycle [3, 4].

2. Literature Search and Selection Methodology

A substantial body of literature exists on the topic of Agile Integration in Software Development. To ensure the relevance and accuracy of the data, as well as to maintain the integrity and reliability of the information included in this review, a literature search was conducted across several academic databases, including:

1) ScienceDirect
2) IEEE Xplore
3) ACM Digital Library
4) PubMed
5) Google Scholar

Search Strategy

The search was limited to works or papers written in English, academic papers published from January 2020 onwards, academic papers with a strong focus on Agile Integration in Software Development, academic papers that provide insight into the benefits and challenges of Agile Integration in Software Development, and publications from reputable academic journals, conference papers, and peer-reviewed literature were considered. Sources that are not peer-reviewed or lack an academic foundation, such as news articles, blogs, and opinion pieces; sources that are not fully accessible; and any publications dated before January 2020 were excluded. The following keywords were used and combined in various ways for analysis:

1) Agility, Agile Software Development, Agile Methodologies
2) Flexibility, Adaptability, Iterative Development
3) Software Engineering, Agile Integration

Boolean operators (AND, OR) and filters (e.g., publication date, document type) were applied to refine the search results.

Selection Criteria

To uphold the quality and reliability of the review, the following inclusion criteria were applied:

1) Publications written in English
2) Peer-reviewed journal articles, conference papers, and academic publications
3) Studies with a strong focus on Agile Integration in software development
4) Papers that provide insights into the benefits, challenges, or practical applications of agile methodologies

Conversely, the following exclusion criteria were enforced:

1) Non-academic sources such as blogs, news articles, and opinion pieces
2) Publications not fully accessible
3) Studies published before January 2020

Data Extraction Process

A standardized qualitative form was used to extract the necessary data, which included the following elements:

1) Author(s)
2) Title of the publication
3) Source (journal/conference)
4) Date of publication
5) Main findings
6) Relevance and significance to the review topic

Quality Assessment

To evaluate the quality of the included studies, a set of predefined criteria was applied (see Table 1 below). These criteria focused on:

1) Reliability of findings
2) Alignment with the review's objectives
3) Methodological rigor

Studies deemed to be of high quality were given greater emphasis in the synthesis and interpretation of the literature.

Table 1. *Evaluation Criteria for Literature Review.*

Criteria	Description
Date of Publication	Assesses the timeliness of the source, ensuring relevance to recent developments in agile software

Criteria	Description
	development (focus on publications from 2020 onward).
Relevance to Review Objectives	Evaluates how well the source aligns with the goals of the literature review and its contribution to the existing body of knowledge. Sources are classified as Useful, Important, or Critical based on their significance.
Main Findings	Summarizes the key outcomes, insights, and implications presented in the source, highlighting its value to the research topic.

3. Agile Paradigm and Manifesto for Agile Software Development

Origins and Core Values of the Agile Manifesto

Agile software development is based on the "Agile Manifesto" formulated in 2001, which defines four core values and twelve principles. These values prioritize individuals and interactions over processes and tools, working software over comprehensive documentation, customer collaboration over contract negotiation, and responding to change over following a plan [5, 49]. These values, as shown in figure 1 below, form the foundation of a flexible, iterative, and customer-centric development practice that stands in stark contrast to traditional, plan-driven models.

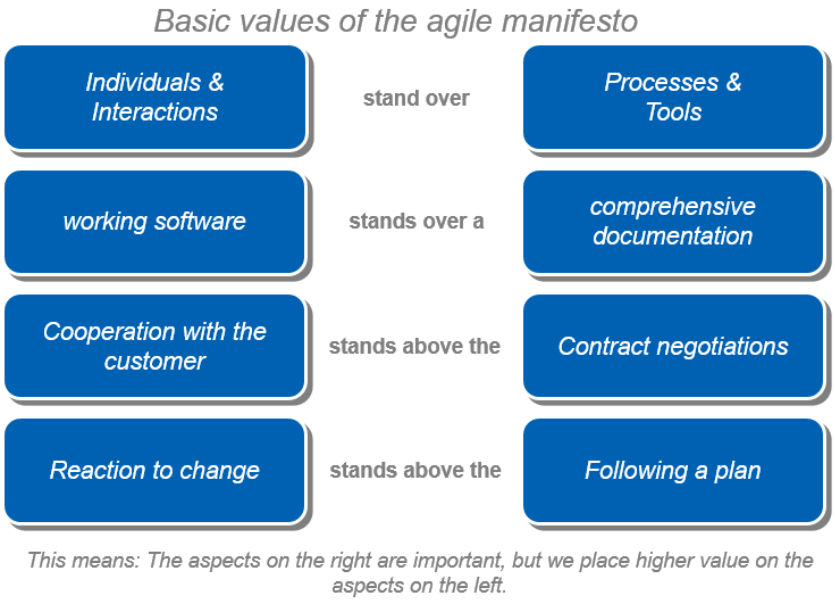

Figure 1. *Core values of the agile manifesto (Source: Own representation according to [5]).*

The twelve Principles of Agile

The twelve principles of the Agile Manifesto elaborate on these values, emphasizing, among other things, the continuous delivery of working software, openness to changing requirements, close collaboration within the team and with the customer, technical excellence, and self-organizing teams [6]. These principles foster a culture of continuous improvement and adaptability, which is becoming increasingly important in dynamic markets [7]. Recent studies highlight the importance of agile mindsets for a company's capacity to innovate [8].

The Role of the Agile Mindset

A key prerequisite for the successful implementation of agile methods is an organization-wide agile mindset. Studies show that agility is not achieved merely by introducing agile methods, but through a profound transformation of thinking and corporate culture [9]. To fully incorporate agile methods, the company should have a so-called "growth mindset," the belief that abilities can be developed through effort and learning [10].

Impact on Innovation and Organizational Performance

The adoption of agile methods has been shown to positively impact a company's innovation capabilities. A recent study demonstrates that agile practices, such as those employed by Scrum or Kanban, not only accelerate product development but also improve quality, reduce risks, and increase employee satisfaction [11, 12]. This is especially evident in hybrid organizational forms — those combining

classical and agile approaches — which offer the advantage of combining flexibility with structure [11, 13].

Comparison with Traditional Development Models

Compared to traditional models such as the Waterfall or V-Model, agility offers greater adaptability to changing requirements and stronger customer involvement in the development process. While classical models are characterized by linear workflows and extensive planning, agile methods enable incremental development with regular feedback loops and continuous improvement [9].

Agility as a Strategic Principle

Overall, agility is now increasingly understood as a strategic principle that extends far beyond software development. It influences organizational structures, leadership styles, and innovation processes, and is increasingly regarded as a key competency for competitiveness in a dynamic, digital world.

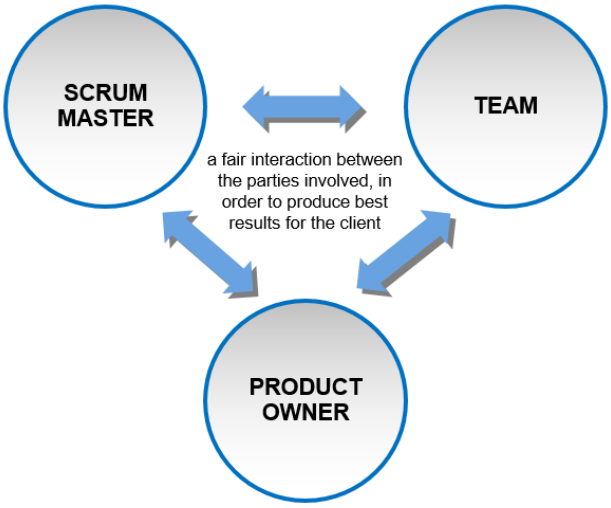

Figure 2. *Three roles of responsible people in Scrum (Source: Own representation).*

4. Agile Methodologies in Practice

The most widely adopted agile methodologies include Scrum, Kanban, and Extreme Programming (XP). Each offers distinct strengths and is suited to different project environments:

Scrum

Scrum is the most widely adopted agile framework, utilized by over 66% of agile teams worldwide. It structures development into time-boxed iterations called sprints and defines clear roles such as Product Owner, Scrum Master, and Development Team [14]. The Scrum Master acts as a servant-leader, enabling collaboration between the Product Owner, who defines value, and the Development Team, who delivers it. Their interaction is guided by transparency, frequent communication, and shared accountability, ensuring that the team remains aligned with business goals while continuously improving [15].

Scrum promotes cross-functional collaboration and emphasizes transparency through key artifacts like the Product Backlog, Sprint Backlog, and Sprint Review, enabling continuous feedback and iterative improvement. Recent trends highlight a growing focus on scaling Scrum across large enterprises using frameworks such as Scaled Agile Framework (SAFe) and Large-Scale Scrum (LeSS) [14, 16].

The table below illustrates how the three core principles of empirical process control — transparency, inspection, and adaptation — are embedded within the Scrum framework. It maps these principles to Scrum's roles, artifacts, and events, and provides practical examples to demonstrate how they are applied in real-world agile teams. This structured overview highlights how Scrum fosters continuous learning, collaboration, and value delivery through iterative feedback and improvement [15, 17, 47].

Table 2. *Mapping Empirical Process Control Principles to Scrum Elements (Own representation).*

Principle	Roles	Artifacts	Events	Examples
Transparency	Product Owner shares backlog priorities with stakeholders Scrum Master ensures team understands Scrum Developers update task boards daily	Product Backlog: visible to all Sprint Backlog: updated daily Increment: meets Definition of Done	Sprint Planning: team sees what's planned Daily Scrum: team shares progress Sprint Review: stakeholders see results Retrospective: open feedback Sprint: ongoing visibility	A shared Jira board shows all tasks and their status Sprint goal is visible to the whole team Stakeholders attend Sprint Review to see the product increment
Inspection	Scrum Master facilitates retrospectives Product Owner reviews progress Developers check	Sprint Backlog: reviewed daily Increment: tested and demoed Product Backlog:	Daily Scrum: team inspects progress Sprint Review: feedback on increment Retrospective: team	Team notices a recurring bug during Daily Scrum Stakeholders suggest UI changes during Sprint Review Retrospective reveals

Principle	Roles	Artifacts	Events	Examples
	build/test results	refined regularly	inspects process	communication gaps
Adaptation	Scrum Master coaches team to improve Product Owner reprioritizes backlog Developers adjust tasks	Product Backlog: reprioritized Sprint Backlog: updated as needed Increment: refined based on feedback	Sprint Planning: adjusts scope Daily Scrum: adapts plan Sprint Review: updates backlog Retrospective: improves process	Team drops a low-priority feature mid-sprint to focus on a blocker Product Owner adds a new user story after customer feedback Team agrees to shorten Daily Scrum to improve focus

Kanban

Kanban is an agile methodology that emphasizes visualizing workflows, limiting work in progress (WIP), and optimizing flow efficiency. By making tasks and their statuses visible — typically through a Kanban board — teams can identify bottlenecks, manage capacity, and improve delivery predictability [18]. The WIP limits prevent team members from being overloaded and ensure a smooth, continuous flow of work. Kanban is particularly effective in environments with continuous delivery require-ments, frequent priority shifts, or maintenance and support contexts, where flexibility and responsiveness are crucial. It sup-ports incremental change, allowing teams to evolve their processes without the need for disruptive overhauls [14].

In practice, Kanban is increasingly combined with Scrum in hybrid models like Scrumban [13], which leverages the structured planning of Scrum with the flow-based adaptability of Kanban. This combination enhances flexibility, improves workflow transparency, and helps teams manage unplanned work more effectively. Unlike time-boxed frameworks, Kanban does not prescribe fixed iterations, making it ideal for teams that require real-time prioritization and continuous delivery pipelines [14, 18].

Extreme Programming (XP)

Extreme Programming (XP) is an agile methodology that emphasizes engineering excellence and developer-centric practices to ensure high-quality, maintainable code in fast-paced, high-risk environments. XP is particularly effective in contexts where requirements change frequently, and code quality is critical to success.

Key practices in XP include:

1) Test-Driven Development (TDD): Writing automated tests before the actual code to ensure functionality and reduce defects.
2) Pair Programming: Two developers work together at one workstation, enhancing code quality, knowledge sharing, and team collaboration.
3) Continuous Integration (CI): Code is integrated and tested frequently — often multiple times a day — to detect issues early and maintain a stable codebase.
4) Refactoring: Continuous improvement of code structure without changing its behavior, ensuring long-term maintainability and adaptability.

XP promotes rapid feedback loops, customer involvement, and simplicity in design, making it ideal for projects that demand both speed and precision. Unlike some other agile methods, XP places a strong emphasis on technical discipline, which helps teams deliver robust software even under pressure [14, 19, 48, 49].

The comparative table below outlines three of the most widely adopted agile methodologies — Scrum, Kanban, and Extreme Programming (XP) — highlighting their distinct focuses, structures, and strengths [47, 48]. While Scrum emphasizes time-boxed sprints and defined roles to manage iterative development, Kanban optimizes workflow through visual management and continuous delivery, and XP prioritizes technical excellence with practices like test-driven development and pair programming, making it ideal for high-risk, quality-critical environments.

Table 3. *Comparative overview of core agile methods.*

Aspect	Scrum	Kanban	Extreme Programming (XP)
Primary Focus	Iterative development, team roles	Workflow visualization and flow optimization	Engineering excellence and rapid feedback
Structure	Time-boxed sprints (2–4 weeks)	Continuous flow, no fixed iterations	Iterations with strong technical discipline
Key Roles	Product Owner, Scrum Master,	No prescribed roles	Coach, Customer, Developers

Aspect	Scrum	Kanban	Extreme Programming (XP)
	Development Team		
Core Practices	Sprint Planning, Daily Stand-ups, Reviews, Retrospectives	Visual boards, WIP limits, flow metrics	TDD, Pair Programming, CI, Refactoring
Artifacts	Product Backlog, Sprint Backlog, Increment	Kanban Board, Cumulative Flow Diagram	Automated Tests, Codebase, User Stories
Best Use Cases	Product development with evolving requirements	Support, maintenance, continuous delivery	High-risk projects needing high code quality
Flexibility	Moderate – structured but adaptable	High – continuous prioritization and delivery	Moderate – strict technical practices
Scalability	Scalable via SAFe, LeSS, Nexus	Scales organically, often used in hybrid models	Less scalable; best for small teams
Strengths	Clear structure, team accountability, stakeholder visibility	Visual clarity, adaptability, continuous delivery	High-quality code, rapid feedback, strong developer discipline
Challenges	Can become rigid if misapplied; role confusion	Risk of lack of planning; metrics misinterpretation	Demands high discipline; steep learning curve

In modern agile environments, XP practices are often integrated into broader frameworks like Scrum or DevOps pipelines, contributing to a culture of continuous improvement and technical agility [14].

Modern research underscores that agility is not merely a set of tools or practices - it is a strategic orientation that permeates organizational culture, leadership, and decision-making [20, 21, 49]. Agile transformation involves:

1) Cultural alignment: fostering trust, collaboration, and psychological safety.
2) Process adaptation: iterative planning, continuous delivery, and adaptive governance.
3) Leadership evolution: shifting from command-and-control to servant leadership and empowerment.

Organizations that embed agility at a strategic level report higher responsiveness to market changes, improved innovation capacity, and enhanced employee engagement [20, 21, 45].

Empirical studies show that the choice of scaling framework (e.g., SAFe, LeSS) is less critical than the organizational readiness and cultural fit. Hybrid approaches that blend agile and traditional models are increasingly common and often more effective in complex environments [13, 22, 51].

5. Integration of Agility into Software Development and Derivation of the Concept of Agile Software Development

Contextualizing Agility in Modern IT Projects

The integration of agility into software development must be understood within the broader context of modern IT project environments [23, 51], which are characterized by:

1) High complexity
2) Significant internal resource demands
3) A strong emphasis on innovation

These characteristics vary depending on the project phase — whether it's initiation, implementation, or maintenance — and influence the degree to which agile methodologies are applicable. Traditional linear models, such as the Waterfall model, have proven inadequate in addressing the dynamic and iterative nature of modern software development. This realization, emerging in the early 1990s, led to the evolution of agile approaches that prioritize flexibility, customer collaboration, and incremental delivery [23, 51].

Agility as a Strategic Enabler

To further elaborate on the integration of agility into software development, recent scholarly research emphasizes that agility is not merely a set of practices but a strategic orientation that permeates organizational culture and decision-making [12, 24, 51]. Agile methodologies such as Scrum, Extreme Programming (XP), and Kanban have evolved significantly, with frameworks like SAFe and LeSS enabling scalability across large enterprises [16, 47].

Scrum remains the most widely adopted agile framework, structuring work into time-boxed sprints and emphasizing transparency through artifacts like product backlogs and sprint reviews [12]. XP, on the other hand, focuses on engineering excellence through practices such as pair programming, test-driven development, and continuous integration [15]. Kanban enhances workflow visualization and limits work in progress to optimize delivery pipelines [18].

Role of Automation and AI in Agile Integration

The integration of agility also involves leveraging automation and Artificial Intelligence (AI) tools to streamline testing, deployment, and monitoring processes [25, 26]. AI-driven analytics are increasingly used to predict project

risks and optimize team performance [25, 26]. Moreover, agile transformation requires fostering a collaborative culture, embracing continuous improvement, and aligning leadership with agile values [23].

Challenges in Agile Adoption

Despite its benefits, agile adoption faces challenges such as documentation deficits, difficulty in measuring progress, and resistance to change. Addressing these requires balancing lean documentation with clarity, using value-based metrics like velocity and burndown charts, and promoting mindset shifts through coaching and training [27].

Strategic Impact on Agile Integration

In conclusion, integrating agility into software development is a multifaceted endeavor that demands alignment across technical practices, organizational culture, and strategic vision. When implemented effectively, agile methodologies enhance adaptability, accelerate delivery, and improve customer satisfaction, positioning organizations to thrive in dynamic and uncertain environments.

6. Benefits of Agile Software Development

Agile software development offers a wide range of benefits for organizations, both internally and externally. These benefits stem from its core principles of adaptability, iterative delivery, and customer collaboration.

Market Responsiveness and Customer Satisfaction

Externally, agile methods enhance an organization's ability to respond rapidly to market changes and evolving customer needs. By emphasizing short development cycles and continuous delivery, agile enables faster time-to-market and more frequent product releases [28, 45, 49]. This responsiveness leads to higher customer satisfaction, as feedback is integrated early and often into the development process [29]. Agile practices also contribute to employer branding by fostering a modern, flexible work culture that attracts skilled professionals [30, 46].

Team Dynamics and Organizational Culture

Internally, agile promotes transparency, collaboration, and a culture of continuous improvement. Teams benefit from increased autonomy, shared responsibility, and iterative learning through retrospectives and feedback loops [28, 51]. These practices not only improve product quality but also enhance employee motivation and engagement [30]. Agile also supports the emergence of a constructive error culture, where failures are seen as learning opportunities rather than setbacks [30].

Modularity and Risk Mitigation

From a technical perspective, agile encourages modular development and prioritization based on business value. Features are often developed independently and integrated later, allowing for flexible reprioritization and risk mitigation throughout the project lifecycle [28, 45]. This modularity, combined with continuous testing and refinement, contributes to higher software quality and reduced delivery risk [29].

Key Benefits

In summary, agile methodologies provide organizations with:

1) Faster Time-to-Market through iterative development and early delivery.
2) Improved Quality via continuous testing and customer feedback.
3) Greater Customer Satisfaction by aligning development with user needs.
4) Organizational Agility through cultural and structural adaptability.
5) Enhanced Employee Engagement by promoting autonomy and learning.

7. Challenges in Agile Adoption

Despite its benefits, agile adoption is not without obstacles:

Documentation Deficits

While agile methodologies prioritize working software over comprehensive documentation, this emphasis can inadvertently lead to knowledge silos, onboarding challenges, and loss of institutional memory [27]. The reduced focus on formal documentation often results in critical information being retained informally — within teams or individuals — making it harder for new members to be onboarded easily or for cross-functional collaboration to thrive. Striking the right balance between lean documentation and clarity remains a persistent challenge in agile environments [27, 46, 50]. Effective strategies include maintaining living documentation (e.g., wikis, annotated codebases), integrating documentation into the development workflow, and fostering a culture of knowledge sharing through pair programming, reviews, and retrospectives [31, 50].

Measuring Progress

While agile teams commonly use metrics like velocity, burndown charts, and cycle time to track progress, these tools are often misunderstood or misapplied, leading to inaccurate performance assessments and misguided decision-making. Unlike tradi-tional metrics such as lines of code or feature counts, which are inadequate in agile contexts, modern agile metrics aim to reflect value delivery, team capacity, and predictability [31, 32].

However, challenges persist. For example, velocity is frequently misused as a performance benchmark rather than a planning aid, and burndown charts can be misleading if scope changes are not properly accounted for. Moreover, over-reliance on quantitative metrics can obscure qualitative factors like team morale, collaboration quality, and customer satisfaction [31, 33].

To address these issues, high-performing agile teams adopt a balanced metrics approach, combining quantitative

indicators (e.g., lead time, throughput, escaped defects) with qualitative insights (e.g., retrospective outcomes, stakeholder feedback). The goal is to foster transparency, continuous improvement, and value-driven delivery—not to enforce rigid performance controls [31, 33, 46].

Resistance to Change

One of the most persistent barriers to successful agile adoption is resistance to change, often rooted in cultural inertia, hierar-chical organizational structures, and a lack of agile literacy. These factors can undermine transformation efforts by fostering skepticism, protecting outdated processes, and discouraging experimentation [34].

Agile transformation requires more than just implementing new tools or frameworks — it demands a fundamental shift in mindset, leadership style, and organizational behavior. Without executive sponsorship, psychological safety, and cross-functional collaboration, agile initiatives often stall or regress [34].

To overcome these challenges, organizations should invest in agile coaching, targeted training programs, and change manage-ment strategies that promote shared ownership, transparency, and continuous learning. Leadership plays a critical role in mod-eling agile values, empowering teams, and creating an environment where change is not only ac-cepted but embraced [34, 46].

8. Best Practices for Agile Integration

Foster a Collaborative Culture

Agile software development depends on strong communi-cation, collaboration, and knowledge sharing within cross-functional teams. Direct, transparent communication across all levels fosters shared understanding and psycho-logical safety, enabling team members to contribute openly and learn from one another [35, 36]. Agile teams prioritize collective intelligence over individual expertise to enhance decision-making and innovation [36].

Customer collaboration is also central to agile, with fre-quent interactions ensuring that evolving requirements are continuously addressed [37]. In large-scale agile environ-ments, structured and informal knowledge-sharing practices help maintain alignment across distributed teams [16, 38]. Agile thrives in cultures that support open dialogue, trust, and continuous learning — key conditions for adaptability and value delivery.

Embrace Continuous Improvement

Agile software development is grounded in a culture of continuous improvement and learning, which is essential for maintain-ing adaptability and delivering sustained value. This culture is fostered through iterative reflection, perfor-mance analysis, and incremental adjustments to processes, tools, and team dynamics [33, 50]. Practices such as retro-spectives, sprint reviews, and feedback loops are central to identifying inefficiencies and implementing targeted im-provements [33, 50].

Technical excellence and simplicity are also key enablers of agility. High-quality design and clean codebases enhance maintainability and responsiveness to change, while mini-mizing work in progress reduces complexity and accelerates delivery [39]. Agile teams are encouraged to adopt lean prin-ciples, focusing on delivering only what is necessary to meet customer needs efficiently [39].

In the context of Scrum, continuous improvement is em-bedded in the framework through structured events like Sprint Retrospectives and Sprint Reviews, which promote team learning and process refinement. These mechanisms help Scrum teams enhance both their internal collaboration and the quality of the products they deliver [33].

Equally important is the integration of customer feedback throughout the development lifecycle. Agile teams prioritize customer satisfaction by incorporating feedback early and often, using methods such as usability testing, product usage analytics, and direct communication channels. This hu-man-centered approach ensures that development efforts align with user expectations and evolving market demands [29].

Ultimately, agile success depends on creating an environ-ment that supports learning, technical discipline, and a cus-tomer-centric approach — all of which contribute to contin-uous value delivery and organizational resilience.

Leverage Automation, AI and DevOps

Modern agile teams increasingly leverage automation and artificial intelligence (AI) to enhance efficiency, reduce manu-al effort, and support data-driven decision-making across the software development lifecycle. The integration of automation, AI, and DevOps into agile workflows is reshaping how soft-ware is developed, tested, and maintained [40, 41]:

1) Automation in testing, deployment, and monitoring accelerates delivery cycles and reduces human error. It enables the implementation of robust Continuous Inte-gration and Continuous Delivery (CI/CD) pipelines, which are essential for maintaining and sustaining agility at scale [6].
2) AI-powered tools assist in backlog prioritization, sprint planning, and risk prediction by analyzing historical project data, team performance metrics, and user be-havior patterns. These tools support more informed and adaptive planning processes [5].
3) Generative AI is increasingly used to produce code snippets, generate documentation, and create test cases. This not only boosts developer productivity but also reduces cognitive load, allowing teams to focus on higher-level problem-solving [5].
4) DevOps integration extends agile principles beyond development into operations and maintenance. By fos-tering automation, collaboration, and continuous feed-back across the entire software delivery pipeline, DevOps enhances organizational agility and supports the seamless deployment of software in production en-vironments [39, 40].

Despite these advancements, challenges remain. Ensuring data quality, maintaining model transparency, and fostering team trust in AI-generated outputs are critical concerns that must be addressed to fully realize the potential of these technologies. Moreover, the adoption of such tools requires upskilling teams and adapting organizational structures to support new workflows and responsibilities.

Scale Thoughtfully

To extend agile practices beyond individual teams, frameworks like SAFe (Scaled Agile Framework) and LeSS (Large-Scale Scrum) have emerged:

1) SAFe provides structured guidance for aligning multiple agile teams with enterprise-level strategy, incorporating roles like Release Train Engineer and Agile Portfolio Management [42, 43].
2) LeSS simplifies scaling by extending Scrum principles across multiple teams working on a shared product, emphasizing minimal overhead and empirical process control [42, 43, 49].

Empirical studies show that the choice of scaling framework (e.g., SAFe, LeSS, Nexus) is less critical than the organizational readiness and cultural fit. However, scaling should not compromise core agile values. Hybrid approaches that blend agile and traditional models are increasingly common and often more effective in complex environments [13, 16].

9. Conclusion

Agile methodologies have become a cornerstone of modern software development, offering organizations the ability to deliver value rapidly, adapt to change, and maintain a strong customer focus. Their iterative nature, characterized by short release cycles and continuous feedback, enables early delivery of usable software — even when functionality is initially limited — while allowing for rapid identification and resolution of issues in real-world use [28].

The integration of agile practices such as Scrum, Kanban, and Extreme Programming (XP) supports dynamic prioritization based on business value, enabling teams to respond flexibly to shifting requirements and market conditions. This adaptability is particularly valuable in volatile environments, where responsiveness and speed are critical to maintaining competitive advantage [43].

Agile's benefits extend beyond technical outcomes. Internally, it fosters transparency, collaboration, and a culture of continuous learning. Externally, it enhances customer satisfaction by embedding feedback loops throughout the development lifecycle. These dual impacts contribute to improved product quality, faster time-to-market, and increased organizational agility [28].

However, the successful integration of agile methodologies is not without challenges. It requires alignment across technical practices, organizational culture, and strategic vision. Barriers such as resistance to change, lack of agile maturity, and inadequate communication structures can hinder effectiveness if not addressed through deliberate change management, leadership support, and ongoing team development [13].

Looking ahead, agile software development is being reshaped by emerging technologies and evolving organizational needs. The integration of DevOps practices extends agile principles into operations and maintenance, promoting automation, continuous delivery, and cross-functional collaboration. Simultaneously, artificial intelligence (AI) is transforming agile workflows by enabling predictive analytics, intelligent backlog prioritization, and generative capabilities such as automated code and test generation.

Moreover, the rise of hybrid models — which blend agile and traditional project management approaches — reflects a growing need for flexibility in tailoring development strategies to specific organizational contexts and project complexities. These developments open new opportunities for innovation but also introduce new demands on organizational structures, employee skillsets, and governance models.

In conclusion, while agile methodologies have already proven their value in enhancing software development outcomes, their continued evolution — driven by technological integration and hybridization — marks the next phase in the digital transformation of organizations. When supported by modern tools, empowered teams, and a commitment to continuous improvement, agile practices will remain essential for building resilient, customer-centric, flexible, and ever-evolving software systems [43, 44].

Abbreviations

AI	Artificial Intelligence
CI/CD	Continuous Integration and Continuous Delivery
CIO	Chief Information Officer
DevOps	Development and Operations
IIT	Informatics and Information Technologies
IT	Information Technology
LeSS	Large-Scale Scrum
SAFe	Scaled Agile Framework
TDD	Test-Driven Development
XP	Extreme Programming

Acknowledgments

I would like to express my deepest gratitude to Professor D. Atanasova for her support throughout this research.

Funding

The authors declare that they have no funding Support.

Data Availability Statement

Data related to the search methodology are available in:
Location: MagentaCloud Share
Link: https://magentacloud.de/s/LGtbtAA4aXXdaCH
Password: Publication072025

Conflicts of Interest

The authors declare that they have no conflicts of interest to this work.

References

[1] A. Behrens, M. Ofori, C. Noteboom and D. Bishop, "A systematic literature review: How agile is agile project management?", Information Systems, vol. 22, no. 3, pp. 298–316, https://doi.org/10.48009/3_iis_2021_298-316, 2021.

[2] S. G. Tetteh, "Empirical Study of Agile Software Development Methodologies: A Comparative Analysis," Asian Journal of Research in Computer Science, vol. 17, no. 5, pp. 30–42, https://doi.org/10.9734/ajrcos/2024/v17i5436, 2024.

[3] N. Prenner, C. Unger-Windeler and K. Schneider, " Goals and challenges in hybrid software development approaches," Journal of Software: Evolution and Process, vol. 33(11), https://doi.org/10.1002/smr.2382, 2021.

[4] R. Pereira, I. Bianchi and A. Rocha, "Digital Technologies and Transformation in Business, Industry and Organizations," vol. 3, Studies in Systems, Decision and Control, Springer, https://doi.org/10.1007/978-3-031-78412-5, 2025.

[5] K. Schache, "20 Jahre Agiles Manifest – Ein Überblick über Werte & Prinzipien," cc-bei.news, https://ccecosystems.news/20-jahre-agiles-manifest-ein-ueberblick-ueber-werte-prinzipien/, 2021.

[6] C. Drumond, "Agile Manifest für die Softwareentwicklung," Atlassian, https://www.atlassian.com/de/agile/manifesto, 2021.

[7] A. Dautovic, "Einführung von Agilen Methoden im Unternehmen," in Agilität in Unternehmen, vol. 1, pp. 1-18, Wiesbaden, Springer, https://doi.org/10.1007/978-3-658-31001-1_1, 2021.

[8] L. Barroca, T. Dingsøyr and M. Mikalsen, "Agile Transformation: A Summary and Research Agenda from the First International Workshop," arXiv preprint arXiv: 1907.10312, https://doi.org/10.48550/arXiv.1907.10312, 2019.

[9] A. Kaack, "Eine empirische Studie zum Stand von agiler Software-Entwicklung in der Praxis," HAW Hamburg, https://users.informatik.haw-hamburg.de/~ubicomp/arbeiten/master/kaack.pdf, 2021.

[10] M. Dujmovic-Bracak and D. Harder, "Kann ein Growth Mindset die Agilität im Unternehmen erhöhen?," in Resilienz durch Organisationsentwicklung, pp. 203-223, Springer, https://doi.org/10.1007/978-3-658-36022-1_9, 2022.

[11] R. Hoda and J. Noble, "Becoming agile: A grounded theory of agile transitions in practice," IEEE Transactions on Software Engineering, vol. 49, no. 1, pp. 1-17, https://doi.org/10.1109/ICSE.2017.21, 2023.

[12] P. F. Soares, "Risks and Challenges of Scrum: A Systematic Literature Review," Digital Technologies and Transformation in Business, Industry and Organizations, vol. 210, pp. 181-196, https://doi.org/10.1007/978-3-031-07626-8_9, 2022.

[13] F. Almeida and B. Blaskovics, "Approaches for Hybrid Scaling of Agile in the IT Industry: A Systematic Literature Review," Information, vol. 15, no. 10, https://doi.org/10.3390/info15100592, 2024.

[14] N. Ozkan et al., "Scrum, Kanban or a Mix of Both? A Systematic Literature Review," Annals of Computer Science and Information Systems, vol. 30, pp. 883-892, https://doi.org/10.15439/2022F143 2022.

[15] D. S. Pashchenko, "Refining the Scrum Paradigm: A Comprehensive Research of Software Development Practices (2020–2023)," Computing & AI Connect, vol. 1, no. Article ID: 2024.0005, https://doi.org/10.69709/CAIC.2024.103102, 2024.

[16] M. Paasivaara, B. Behm, C. Lassenius and M. Hallikainen, "Large-scale agile transformation at Ericsson: A case study", Empirical Software Engineering, 23(5), 2550–2596. https://doi.org/10.1007/s10664-017-9555-8, 2018.

[17] A. C. Sassa, I. A. de Almeida, T. N. F. Pereira and M. S. de Oliveira, "Scrum: A Systematic Literature Review," International Journal of Advanced Computer Science and Applications, 14(4), https://doi.org/10.14569/IJACSA.2023.0140420, 2023.

[18] D. Anderson, "Kanban: Successful Evolutionary Change for Your Technology Business," Seattle, WA, USA: Blue Hole Press, 2021.

[19] V. Stray, R. Hoda, M. Paasivaara and P. Kruchten, "Agile Processes in Software Engineering and Extreme Programming: Proceedings of the 21st International Conference on Agile Software Development," XP 2020, Copenhagen, Denmark, June 8–12, 2020. Lecture Notes in Business Information Processing, 383. Springer, https://doi.org/10.1007/978-3-030-49392-9, 2020.

[20] S. Rathor, W. Xia and D. Batra, "Achieving software development agility: different roles of team, methodological and process factors," Information Technology & People, vol. 37, no. 2, pp. 456-478, https://doi.org/10.1108/ITP-10-2021-0832, 2024.

[21] S. Magistretti and D. Trabucchi, "Agile-as-a-tool and agile-as-a-culture: A comprehensive review of agile approaches adopting contingency and configuration theories," Review of Managerial Science, 19, 223–253, https://doi.org/10.1007/s11846-024-00745-1, 2024.

[22] Z. Hoy and M. Xu, "Agile Software Requirements Engineering Challenges-Solutions—A Conceptual Framework from Systematic Literature Review," Information, 14(6), 322., https://doi.org/10.3390/info14060322, 2023.

[23] K. Dikert, M. Paasivaara and C. Lassenius, "Challenges and success factors for large-scale agile transformations: A systematic literature review," Information and Software Technology, vol. 119, pp. 87-108, https://doi.org/10.1016/j.jss.2016.06.013, 2016.

[24] B. Gezici and A. K. Tarhan, "Systematic literature review on software quality for AI-based software," Empirical Software Engineering, vol. 27, no. 66, https://doi.org/10.1007/s10664-021-10105-2, 2022.

[25] S. S. Almalki, "AI-Driven Decision Support Systems in Agile Software Project Management: Enhancing Risk Mitigation and Resource Allocation," Systems, MDPI, 13(3), 208. https://doi.org/10.3390/systems13030208, 2022.

[26] P. Weichbroth, "A Case Study on Implementing Agile Techniques and Practices: Rationale, Benefits, Barriers and Business Implications for Hardware Development," Applied Sciences, 12(17), 8457, https://doi.org/10.3390/app12178457, 2022.

[27] T. Dingsøyr, F. O.lav Bjørnson, J. Schrof and T. Sporse, " A longitudinal explanatory case study of coordination in a very large development programme: the impact of transitioning from a first- to a second-generation large-scale agile development method," Empirical Software Engineering, vol. 28, no. 1, https://doi.org/10.1007/s10664-022-10230-6, 2023.

[28] M. Moloto, A. Harmse and T. Zuva, "Impact of Agile Methodology Use on Project Success in Organizations – A Systematic Literature Review," Software Engineering Perspectives in Intelligent Systems, vol. 1294, CoMeSySo 2020, Springer, https://doi.org/10.1007/978-3-030-63322-6_21, 2020.

[29] IEEE/ISO/IEC 12207-2017, "Systems and software engineering — Software life cycle processes," IEEE Standards Association, https://standards.ieee.org/ieee/12207/5672/, 2017.

[30] P. Burton, "Agile and Organizational Change Management to Increase Project Success," PM World Journal, vol. XII, no. VII, https://pmworldjournal.com/article/agile-and-organizational-change-management, 2023.

[31] K. P. Pham and M. Neumann, "Optimizing Agile Performance Metrics: Practical Challenges and Strategic Implementation Insights," Software, System, and Service Engineering, vol. LNBIP 542, p. 3–29, https://doi.org/10.1007/978-3-031-84913-8_1, 2025.

[32] M. Huss, D. R. Herber and J. M. Borky, "Comparing Measured Agile Software Development Metrics Using an Agile Model-Based Software Engineering Approach versus Scrum Only," Software, 2(3), 310–331, https://doi.org/10.3390/software2030015, 2023.

[33] N. Yang, X. Wang, Z. Zhang, D. Siemon and S. Hyrynsalmi, "Core Theories in Agile Software Development," XP 2025: Agile Processes in Software Engineering and Extreme Programming, LNBIP 545, pp. 3–18, Springer, https://doi.org/10.1007/978-3-031-94544-1_1, 2025.

[34] D. Baxter et al., "Institutional challenges in agile adoption: Evidence from a public sector IT project," Government Information Quarterly 40(4), Article 101858, https://doi.org/10.1016/j.giq.2023.101858, 2023.

[35] D. Strode, T. Dingsøyr and Y. Lindsjørn, "A teamwork effectiveness model for agile software development," Empirical Software Engineering, vol. 27, no. 56, https://doi.org/10.1007/s10664-021-10115-0, 2022.

[36] S. Ghobadi and L. Mathiassen, "Perceived barriers to effective knowledge sharing in agile software teams," Information Systems Journal, 25(2), pp. 95-125, https://doi.org/10.1111/isj.12053, 2014.

[37] C. Khalil, V. Fernandez and T. Houy, "Can Agile Collaboration Practices Enhance Knowledge Creation between Cross-Functional Teams?" Digital Enterprise Design and Management 2013, AISC, vol. 205, pp. 123-133, Springer, https://doi.org/10.1007/978-3-642-37317-6_11, 2013.

[38] F. Tobisch and F. Matthes, "Knowledge Sharing and Coordination in Large-Scale Agile Software Development – A Systematic Literature Review and an Interview Study," XP 2025: Agile Processes in Software Engineering and Extreme Programming, LNBIP 545, pp. 81-99, Springer, https://doi.org/10.1007/978-3-031-94544-1_6, 2025.

[39] B. Fitzgerald and K. Stol, "Continuous software engineering: A roadmap and agenda," Journal of Systems and Software, vol. 123, pp. 176-189, https://doi.org/10.1016/j.jss.2015.06.063, 2021.

[40] M. H. Maturi, S. S. Meduri, H. Gonaygunta and G. S. Nadella, "A Systematic Literature Review: The Recent Advancements in AI Emerging Technologies and Agile DevOps," International Meridian Journal, 2(2), 1–23, https://meridianjournal.in/index.php/IMJ/article/view/75 2020.

[41] B. Cabrero-Daniel, "AI for Agile Development: A Meta-Analysis," arXiv preprint, https://doi.org/10.48550/arXiv.2305.08093, 2023.

[42] C. Verwijs and D. Russo, "Do Agile scaling approaches make a difference? An empirical comparison of team effectiveness," Empirical Software Engineering, vol. 29, no. 75, https://doi.org/10.1007/s10664-024-10481-5, 2024.

[43] K. Chakravarty and J. Singh, "A Dissection of Agile Software Development in Changing Scenario and the Sustainable Path Ahead," International Journal of System Assurance Engineering and Management, vol. 15, pp. 2606-2622, https://doi.org/10.1007/s13198-024-02283-1, 2024.

[44] Y. Zhang, H. Dong, D. Baxter and N. Dacre, "Agile Meets Digital: A Systematic Literature Review on the Interplay Between Agile Project Management and Digital Transformation," Presented at the British Academy of Management 37th Conference, University of Sussex Business School, Brighton, UK., https://eprints.soton.ac.uk/476132/ 2023.

[45] T. Dingsøyr, T. Dybå, and N. B. Moe, "Agile Software Development: Current Research and Future Directions," Berlin, Germany: Springer, https://doi.org/10.1007/978-3-642-12575-1, 2010.

[46] S. Hooda, V. M. Sood, Y. Singh, S. Dalal and M. Sood, Eds., "Agile Software Development: Trends, Challenges and Applications," Hoboken, NJ, USA: Wiley, https://doi.org/10.1002/9781119896838, 2023.

[47] B. Julian, J. Noble and C. Anslow, "Agile Practices in Practice: Towards a Theory of Agile Adoption and Process Evolution," Agile Processes in Software Engineering and Extreme Programming – XP 2019, pp. 3-18, Montréal, QC, Canada: Springer, https://doi.org/10.1007/978-3-030-19034-7_1 2019.

[48] D. Šmite, E. Guerra, X. Wang, M. Marchesi and P. Gregory, Eds., "Agile Processes in Software Engineering and Extreme Programming – XP 2024 Proceedings," 25th International Conference on Agile Software Development, Bozen-Bolzano, Italy: Springer, https://doi.org/10.1007/978-3-031-61154-4, 2024.

[49] M. Kuhrmann et al., "What Makes Agile Software Development Agile?," arXiv preprint arXiv: 2109. 11435, https://doi.org/10.48550/arXiv.2109.11435, 2021.

[50] D. Ghimire and S. Charters, "The Impact of Agile Development Practices on Project Outcomes," Software, vol. 1, no. 3, pp. 265–275, https://doi.org/10.3390/software1030012, 2022.

[51] H. Dong, N. Dacre, D. Baxter and S. Ceylan, "What is Agile Project Management? Developing a New Definition Following a Systematic Literature Review," Project Management Journal, vol. 55, no. 6, pp. 668–688, https://doi.org/10.1177/87569728241254095, 2024.