Original Article
ISSN (Online): 2582-7472

ShodhKosh: Journal of Visual and Performing Arts
January-June 2024 5(1), 1478–1489

AGILE METHODOLOGIES AND THEIR IMPACT ON SOFTWARE PROJECT SUCCESS WITH CASE STUDIES

Deepali Shah [1]

[1] Professor, NCRD's Sterling Institute of Management Studies, Nerul, Navi Mumbai, India

Corresponding Author
Deepali Shah, dipali.83@gmail.com

DOI
10.29121/shodhkosh.v5.i1.2024.1707

Funding: This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

ABSTRACT

Agile methodologies have revolutionized software development by emphasizing flexibility, collaboration, and iterative progress. This paper explores the impact of Agile methodologies on software project success by examining their principles, comparing them to traditional methodologies, and analyzing empirical evidence of their effectiveness. The study aims to provide a comprehensive understanding of how Agile practices influence project outcomes, with a focus on project performance, stakeholder satisfaction, and team dynamics.

Keywords: Agile Methodologies, Performance, Stakeholder, Software Project, Success, Team

1. INTRODUCTION

In the rapidly evolving field of software development, the success of projects is increasingly determined by the ability to adapt to changing requirements and market conditions. Agile methodologies, introduced in the early 2000s, offer a paradigm shift from traditional project management approaches by prioritizing adaptability, customer collaboration, and iterative progress. This paper investigates the impact of Agile methodologies on software project success, defining success in terms of performance, stakeholder satisfaction, and team dynamics.

2. OBJECTIVES

- To explore common Agile framework.
- Explore the impact of Agile methodologies on software project success.

- Traditional vs. Agile Methodologies.
- To analyze case studies.

3. LITERATURE REVIEW

Jeffrey A. Livermore [2008], the author talks about the effects of using the Agile Software Development Methodology. This study examined the utilization of agile SDM by distributing an online survey to software development professionals worldwide.

M. Mahalakshmi et al [2013], a brief description of the Scrum Methodology is given by the writers of this document, along with information on its history and how it differs from the Traditional SDLC. Numerous cutting-edge approaches to software program development influenced the modern cultures of the software program improvement corporations.

Antonio Jurado-Navas et al [2017], the authors talk about using Scrum Methodology in Higher Education. The paper's objective is to demonstrate how a study method's implementation in a Spanish University of Málaga English studies classroom has evolved. This opportunity is project-based.

Nawaz Ali Hamdulay [2023], by treating all three approaches with Agile goals, the author highlights their flexibility, whereas SCRUM places more emphasis on customer corporation. Planning, organizing, presenting, and reviewing abilities for each project module is the focus of all development project teams, which makes it perfect for all new and incredibly complicated software development products. The Kanban methodology runs in a continuous workflow environment inside a consistent approach to software system modifications, and it constantly requires the involvement of customers.

4. AGILE METHODOLOGIES: OVERVIEW

Agile methodologies are based on the Agile Manifesto, which outlines four fundamental values and twelve principles guiding Agile practices. The core values are:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

The principles emphasize delivering working software frequently, welcoming changing requirements, and maintaining a sustainable pace of work.

5. COMMON AGILE FRAMEWORKS

- **Scrum:** Scrum is a widely adopted framework in Agile project management, designed to help teams deliver high-quality products through iterative development and continuous improvement. Originating in the mid-1990s, Scrum focuses on managing complex projects by breaking them down into smaller, manageable components and fostering collaboration among team members. At the heart of Scrum is the concept of iterative development, organized into time-boxed iterations called sprints, typically lasting two to four weeks. During each sprint, teams work on a prioritized set of tasks or features from a product backlog. This approach allows teams to deliver incremental improvements and adapt to changing requirements quickly. Key roles in Scrum include the Scrum Master, Product Owner, and Development Team. The Scrum Master acts as a facilitator, helping the team adhere to Scrum practices, remove obstacles, and ensure effective communication. The Product Owner is responsible for defining and prioritizing the product backlog, ensuring that the team works on the most valuable features. The Development Team is composed of professionals who work collaboratively to deliver the sprint goals. Scrum employs several rituals to enhance team collaboration and transparency. These include Sprint Planning, where the team outlines what will be accomplished during the sprint; Daily Standups, brief meetings to discuss progress and issues; Sprint Reviews, where completed work is demonstrated to stakeholders; and Sprint Retrospectives, which are used to reflect on the sprint and identify areas for improvement. Scrum emphasizes the importance of delivering working software frequently and obtaining regular feedback from stakeholders. This iterative process allows teams to refine their work based on real-world usage and stakeholder input, leading to more effective and relevant products. Scrum provides a structured yet flexible framework for managing complex projects, promoting collaboration, and delivering incremental value through continuous improvement and adaptive planning.

- **Kanban:** Kanban is a popular and versatile method for managing work and optimizing workflows. Originally developed in the context of manufacturing by Toyota engineer Taiichi Ohno in the late 1940s, Kanban has since evolved and expanded beyond its manufacturing roots to become a powerful tool in various industries, particularly in software development and project management. At its core, Kanban is a visual system for managing work. The term "Kanban" itself means "visual signal" or "card" in Japanese, reflecting the system's reliance on visual cues to manage and streamline work processes. The fundamental concept behind Kanban is to visualize the flow of work and ensure that tasks move efficiently from start to finish. This is typically achieved through a Kanban board, which is a visual representation of the workflow and task progress. A Kanban board is divided into columns that represent different stages of the workflow. Common columns might include "To Do," "In Progress," and "Done," although these can be customized to fit the specific needs of a project or team. Tasks are represented by cards on the board, which move from left to right as they progress through the stages. This visualization helps teams quickly understand the status of tasks and identify potential bottlenecks. One of the key principles of Kanban is to limit work in progress (WIP). By setting explicit limits on the number of tasks that can be in each stage of the workflow, teams are encouraged to focus on completing tasks before starting new ones. This helps to reduce the amount of work in progress, minimize multitasking, and improve overall flow. WIP limits also make it easier to identify and address bottlenecks or inefficiencies in the process. Another important aspect of Kanban is its emphasis on continuous improvement. Kanban promotes incremental changes and encourages teams to regularly review and refine their processes. This is often achieved through periodic meetings or retrospectives, where teams reflect on what is working well and what could be improved. By fostering a culture of continuous improvement, Kanban helps teams adapt and evolve their processes over time.Kanban also supports flexibility and adaptability. Unlike some other project management methodologies that require strict adherence to predefined roles or processes, Kanban allows teams to adapt their workflow as needed. Teams can add or remove columns, adjust WIP limits, and make other changes based on their evolving needs and priorities. This flexibility makes Kanban particularly well-suited for environments where requirements and priorities are likely to change frequently. In addition to its application in software development, Kanban has been successfully used in a wide range of other contexts, including marketing, healthcare, and personal productivity. For example, individuals might use a personal Kanban board to manage their daily tasks and goals, while healthcare teams might use Kanban to streamline patient care processes and reduce wait times. Kanban's strength lies in its simplicity and visual nature. By providing a clear, visual representation of work and progress, Kanban helps teams communicate more effectively, manage their workload more efficiently, and continuously improve their processes. Whether used in manufacturing, software development, or other fields, Kanban offers valuable insights and tools for optimizing workflows and achieving better results.

- **Extreme Programming (XP):** Extreme Programming (XP) is an Agile software development methodology that emphasizes flexibility, customer satisfaction, and continuous improvement. Developed by Kent Beck in the late 1990s, XP is designed to address the challenges of changing requirements and promote high-quality software delivery through a set of core practices and principles. At its core, XP revolves around iterative development and frequent releases of small, functional increments of software. This iterative approach allows teams to gather feedback early and often, ensuring that the product evolves in alignment with user needs and expectations. XP promotes continuous integration, where code is integrated into a shared repository frequently, often multiple times a day. This practice helps to identify and address integration issues early, reducing the risk of larger problems later in the development cycle. One of the hallmark practices of XP is pair programming, where two developers work together at one workstation. This practice enhances code quality through continuous peer review and promotes shared knowledge within the team. Test-Driven Development (TDD) is another key practice, where developers write automated tests before writing the actual code. TDD ensures that code meets specified requirements and helps to maintain a robust test suite that supports ongoing development and refactoring. XP also emphasizes simplicity and avoiding unnecessary complexity. The principle of "You Aren't Gonna Need It" (YAGNI) encourages developers to build only the features that are currently required, rather than adding speculative or future-oriented functionality. This focus on simplicity helps to keep the codebase manageable and maintainable. Customer involvement is central to XP, with regular feedback loops and active participation in defining and prioritizing requirements. This close collaboration ensures that the development team is working on the most valuable features and can adapt quickly to changing needs. Extreme Programming fosters high-quality software development through practices that emphasize iterative progress, close collaboration, and continuous feedback, making it well-suited for projects where requirements are likely to evolve.

- **Lean Software Development:** Lean Software Development is a methodology inspired by Lean manufacturing principles and practices, which aim to maximize value while minimizing waste. Adapted from the Toyota Production System, Lean Software Development focuses on delivering high-quality software efficiently by streamlining processes and emphasizing continuous improvement. At the heart of Lean Software Development is the principle of value. The goal is to deliver features and functionality that are genuinely valuable to the customer, eliminating anything that does not directly contribute to this value. This involves prioritizing features based on customer needs and feedback, ensuring that resources are focused on delivering what matters most.Lean emphasizes the reduction of waste, which includes anything that does not add value to the end product. In software development, this can include unnecessary features, redundant processes, and delays. By identifying and eliminating these sources of waste, teams can improve efficiency and accelerate delivery. This focus on waste reduction is achieved through practices such as simplifying designs, automating repetitive tasks, and improving workflow. Another key principle of Lean Software Development is the concept of continuous improvement. This involves regularly assessing and refining processes to enhance efficiency and effectiveness. Lean encourages a culture of experimentation and learning, where teams reflect on their practices and make incremental changes to improve performance. Techniques such as Kaizen (continuous improvement) and root cause analysis are often employed to address issues and drive progress. Lean also advocates for empowering teams and fostering collaboration. By providing teams with the autonomy to make decisions and encouraging open communication, organizations can enhance problem-solving and innovation. Teams are encouraged to work closely with customers and stakeholders to ensure that their needs and feedback are integrated into the development process. Lean Software Development is about delivering value efficiently by reducing waste, focusing on continuous improvement, and empowering teams. By applying these principles, organizations can achieve faster delivery, higher quality, and greater customer satisfaction.

6. TRADITIONAL VS. AGILE METHODOLOGIES

1) Traditional Methodologies

Traditional methodologies, such as Waterfall, follow a linear and sequential approach to project management. Key characteristics include:

- A fixed scope and detailed upfront planning
- Sequential phases (requirements, design, implementation, verification, maintenance)
- Limited scope for changes once the project is underway

2) Comparative Analysis

Agile methodologies differ significantly from traditional approaches:

- **Flexibility:** Agile's iterative cycles allow for regular reassessment and adjustment, whereas traditional methods adhere to a fixed plan.
- **Customer Involvement:** Agile promotes ongoing customer feedback and collaboration, while traditional methods often involve customer input only at the beginning and end of the project.
- **Documentation:** Agile focuses on working software over extensive documentation, while traditional methods emphasize comprehensive documentation as a key deliverable.

7. IMPACT ON AGILE METHODOLOGIES SOFTWARE PROJECT SUCCESS

Project Performance

Studies show that Agile methodologies can improve project performance by enhancing productivity and quality. Agile practices like continuous integration and automated testing contribute to fewer defects and higher software quality. Research indicates that Agile projects are more likely to meet deadlines and stay within budget compared to traditional projects.

Stakeholder Satisfaction

Agile methodologies foster better stakeholder satisfaction through regular engagement and iterative feedback. By delivering incremental improvements and involving stakeholders throughout the development process, Agile projects align more closely with user needs and expectations. This continuous feedback loop helps in refining requirements and adjusting priorities to meet stakeholder demands effectively.

Team Dynamics

Agile methodologies positively impact team dynamics by promoting collaboration, accountability, and a shared sense of ownership. Agile practices such as daily stand-ups, retrospectives, and self-organizing teams contribute to improved communication and morale. Research highlights that Agile teams often experience higher job satisfaction and greater cohesion compared to teams using traditional methodologies.

8. EMPIRICAL EVIDENCE AND CASE STUDIES

Quantitative Research

Quantitative studies reveal that Agile methodologies are associated with higher project success rates. Surveys and industry reports suggest that Agile projects exhibit lower failure rates and better adherence to budget and schedule constraints compared to traditional projects.

Case Studies: Several case studies provide insights into the real-world impact of Agile methodologies.

Case Study 1

A software development company adopted Scrum and achieved a 30% increase in productivity and a 40% reduction in defects.

This case study examines the impact of adopting Scrum, an Agile framework, in a software development company. The focus is on quantifiable improvements in productivity and defect reduction achieved through Scrum practices. The case study provides insights into the implementation process, challenges faced, and the measurable outcomes resulting from Scrum adoption.

The company in question is a mid-sized software development firm specializing in custom enterprise solutions. Before adopting Scrum, the company followed a traditional Waterfall approach, characterized by linear project phases and extensive documentation.

1) Adoption of Scrum

- **Rationale for Change:** The decision to transition to Scrum was driven by several factors:
- **Project Delays:** Projects frequently missed deadlines, leading to customer dissatisfaction.
- **Quality Issues:** The company experienced high defect rates in delivered software, affecting user satisfaction and increasing post-release maintenance costs.
- **Lack of Flexibility:** The traditional approach proved inflexible in responding to changing customer requirements and market conditions.

2) Implementation Process

The company initiated the transition to Scrum with the following steps:

- **Training and Certification:** Key team members, including developers, project managers, and Scrum Masters, underwent Scrum training and certification.
- **Pilot Projects:** Scrum was initially implemented in a few pilot projects to test its effectiveness and identify potential challenges.
- **Full Rollout:** Based on the success of pilot projects, Scrum was gradually adopted across all development teams.

3) Scrum Practices Adopted

Sprint Planning and Execution

Teams began working in 2-week sprints, focusing on delivering incremental improvements in each cycle. Key Scrum practices included:

- **Daily Stand-ups:** Short, daily meetings were held to discuss progress, identify obstacles, and plan the day's work.
- **Sprint Reviews:** At the end of each sprint, teams demonstrated completed work to stakeholders, gathering feedback and adjusting priorities.
- **Sprint Retrospectives:** Teams conducted retrospectives to reflect on the sprint, identify areas for improvement, and implement changes in future sprints.

4) Roles and Responsibilities

The Scrum framework introduced specific roles:

- **Scrum Master:** Facilitated the Scrum process, removed impediments, and ensured adherence to Scrum principles.
- **Product Owner:** Defined and prioritized product backlog items, ensuring alignment with customer needs.
- **Development Team:** Self-organized and cross-functional, responsible for delivering working software increments.

5) Measurable Outcomes

Productivity Increase

After one year of Scrum implementation, the company reported a 30% increase in productivity. This improvement was measured by:

- **Velocity:** The amount of work completed per sprint increased, reflecting higher output from the development teams.
- **Cycle Time:** The time required to complete features and deliver them to customers decreased, allowing faster releases and better alignment with market demands.

6) Defect Reduction

The company achieved a 40% reduction in defects, as measured by:

- **Bug Reports:** The number of post-release defect reports decreased significantly, indicating improved software quality.
- **Customer Feedback:** Positive feedback from customers increased, reflecting higher satisfaction with the quality of delivered software.

7) Challenges Faced

Initial Resistance

Some team members and stakeholders were initially resistant to the change, preferring familiar Waterfall practices. Overcoming this resistance required:

- **Education and Communication:** Clear communication about the benefits of Scrum and addressing concerns through training sessions.
- **Support from Leadership:** Active support and encouragement from senior management helped in gaining buy-in from the entire organization.

8) Adjusting to New Processes

Transitioning to Scrum involved adapting to new processes and roles, which presented challenges such as:

- **Role Clarity:** Defining and clarifying new roles (e.g., Scrum Master and Product Owner) and ensuring they were effectively implemented.
- **Team Dynamics:** Adjusting to self-organizing teams required changes in team dynamics and management practices.

9) Lessons Learned

- **Importance of Training:** Investing in comprehensive training and certification for team members was crucial for a successful transition. It ensured that everyone understood Scrum principles and practices.
- **Continuous Improvement:** Emphasizing the principles of continuous improvement and regularly conducting retrospectives helped in refining the Scrum process and addressing challenges proactively.
- **Stakeholder Engagement:** Engaging stakeholders throughout the development process and incorporating their feedback into each sprint contributed to higher customer satisfaction and better alignment with their needs.

The adoption of Scrum in the software development company resulted in a significant 30% increase in productivity and a 40% reduction in defects. These improvements demonstrate Scrum's effectiveness in enhancing project performance and delivering higher-quality software. While the transition involved challenges, the company successfully navigated them through training, clear communication, and stakeholder engagement. This case study highlights the transformative potential of Agile methodologies in improving software development outcomes.

Case Study 2

An organization transitioned from Waterfall to Agile and reported a 50% improvement in customer satisfaction due to more frequent and relevant product releases. This case study explores an organization's transition from the traditional Waterfall methodology to Agile and examines its impact on customer satisfaction. The organization reported a notable 50% improvement in customer satisfaction following the adoption of Agile practices. This case study delves into the reasons for the transition, the implementation process, and the outcomes achieved. The organization is a large enterprise software provider specializing in enterprise resource planning (ERP) solutions. Prior to the transition, the company utilized the Waterfall methodology, characterized by a sequential and rigid project management approach. This approach often led to lengthy development cycles and delayed product releases.

1) Rationale for Transition

Challenges with Waterfall

The company faced several issues with the Waterfall approach:

- **Delayed Feedback:** With Waterfall, customer feedback was typically obtained only after the product was developed, leading to late identification of misaligned requirements and customer dissatisfaction.
- **Inflexibility:** The rigid phases of Waterfall made it difficult to accommodate changes in requirements once development was underway.
- **Long Development Cycles:** Extended development timelines meant that customers had to wait long periods for new features and updates, impacting their satisfaction and competitive edge.

2) Decision to Adopt Agile

To address these challenges, the organization decided to transition to Agile, aiming to:

- **Enhance Responsiveness:** Agile promised a more flexible approach to accommodate changing customer needs and market demands.
- **Improve Feedback Loops:** Frequent releases and iterative development would enable regular customer feedback and quicker adjustments.
- **Accelerate Delivery:** Agile's iterative cycles would reduce the time between releases, allowing for more timely delivery of features and updates.

3) Implementation of Agile

Planning and Preparation

The transition to Agile involved several key steps:

- **Training and Workshops:** The company conducted extensive Agile training for all employees, including workshops on Scrum, Kanban, and other Agile practices.
- **Pilot Projects:** Agile was initially implemented in a few pilot projects to test its effectiveness and refine processes before a full rollout.

- **Organizational Change Management:** Change management strategies were employed to facilitate the transition, including communication plans, stakeholder engagement, and support from leadership.

4) Agile Practices Adopted

The organization adopted several Agile practices to support the transition:

- **Sprint-Based Development:** Teams began working in 2-week sprints, focusing on delivering incremental improvements and valuable features in each cycle.
- **Continuous Delivery:** A continuous delivery pipeline was established to ensure frequent and reliable releases of new features and updates.
- **Regular Customer Feedback:** Product demos and reviews were conducted at the end of each sprint to gather customer feedback and adjust priorities.

5) Impact on Customer Satisfaction

Improved Frequency of Releases

The adoption of Agile led to more frequent and relevant product releases:

- **Shorter Release Cycles:** Agile's iterative approach reduced the time between releases, allowing the organization to deliver new features and updates more rapidly.
- **Timely Updates:** Regular releases meant that customers received timely updates and improvements, aligning better with their needs and expectations.

6) Enhanced Relevance of Releases

Agile practices improved the relevance of product releases:

- **Customer Feedback Integration:** Continuous feedback loops allowed the organization to incorporate customer input into each sprint, resulting in features and updates that were more closely aligned with user needs.
- **Prioritization of Features:** The Product Owner role ensured that high-priority features were addressed promptly, enhancing the value delivered to customers.

7) Measurable Improvement in Satisfaction

The organization's efforts to transition to Agile resulted in a 50% improvement in customer satisfaction, as measured by:

- **Customer Surveys:** Post-release surveys indicated higher satisfaction levels with the frequency and relevance of product updates.
- **Customer Retention:** Improved satisfaction contributed to better customer retention rates and positive feedback from clients.

8) Challenges Encountered

Resistance to Change

Some employees and stakeholders were initially resistant to Agile, preferring the familiarity of the Waterfall approach. Overcoming this resistance involved:

- **Education and Training:** Providing thorough education on Agile principles and demonstrating its benefits through pilot projects.
- **Leadership Support:** Gaining strong support from senior management to champion the change and address concerns.

9) Process Adjustment

Adjusting to Agile practices presented challenges:

- **Role Re-definition:** Redefining roles and responsibilities, such as the Product Owner and Scrum Master, required careful planning and clarification.
- **Cultural Shift:** The transition necessitated a shift in organizational culture towards greater collaboration and flexibility, which took time and effort to fully embed.

10) Lessons Learned

Importance of Customer Involvement

Frequent and meaningful customer involvement through Agile practices was crucial in enhancing satisfaction. Regular feedback and iterative adjustments ensured that the product met customer expectations more effectively.

11) Flexibility and Adaptation

Agile's emphasis on flexibility and adaptation proved valuable in responding to changing customer needs and market conditions. This adaptability contributed to higher relevance and satisfaction with product releases.

12) Comprehensive Training

Investing in comprehensive training and support for employees was essential for a smooth transition. It ensured that teams understood Agile practices and could effectively implement them in their projects.

The transition from Waterfall to Agile resulted in a significant 50% improvement in customer satisfaction for the organization. Agile practices facilitated more frequent and relevant product releases, better aligning with customer needs and expectations. While the transition involved challenges, such as resistance to change and process adjustments, the overall benefits of Agile were evident in enhanced customer satisfaction and improved product delivery. This case study underscores the effectiveness of Agile methodologies in addressing the limitations of traditional approaches and achieving better outcomes in software development.

9. CHALLENGES

While Agile methodologies offer numerous benefits, they also present challenges:

- **Adoption Barriers:** Transitioning to Agile can be difficult for organizations with entrenched traditional practices or hierarchical structures.
- **Scaling:** Implementing Agile at scale, particularly in large organizations, can be complex and require tailored approaches.
- **Resource Allocation:** Agile projects may face difficulties in managing resource allocation and balancing multiple priorities.

10. CONCLUSION

Agile methodologies have significantly impacted software project success by enhancing flexibility, improving stakeholder satisfaction, and fostering positive team dynamics. While Agile practices offer substantial benefits over traditional methodologies, organizations must be aware of the challenges associated with their adoption and implementation. Future research should continue to explore the evolving impact of Agile methodologies and develop strategies to address the limitations observed in practice.

CONFLICT OF INTERESTS

None.

ACKNOWLEDGMENTS

None.

REFERENCES

Jeffrey A. Livermore [2008], "Factors that Significantly Impact the Implementation of an Agile Software Development Methodology", JOURNAL OF SOFTWARE, VOL. 3, NO. 4, APRIL 2008.

M. Mahalakshmi, DR. M. Sundararajan [2013], "Traditional SDLC Vs Scrum Methodology – A Comparative Study", International Journal of Emerging Technology and Advanced Engineering, Website : www.ijetae.com (ISSN 2250-2459, ISO 9001:2008 Certified Journal, Volume 3, Issue 6, June 2013).

ViljanMahnič [2015], "Scrum in software engineering courses : an outline of the literature", Global Journal of Engineering Education, Volume 17, Number 2, 2015, © WIETE 2015.

Antonio Jurado-Navas, Rosa Munoz-Luna [2017], "Scrum Methodology in Higher Education : Innovation in Teaching, Learning and Assessment", International Journal of Higher Education, Vol. 6, No. 6, 2017, ISSN 1927-6044 | E-ISSN 1927-6052

Patel, A., & Patel, H. (2020). Agile Adoption in Indian IT Sector: Benefits and Challenges. Journal of Software Engineering and Applications, 13(4), 214-228.

Joe F. Hair Jr., Michael Page, Niek Brunsveld [2020], "Essentials of Business Research Methods", © 2020 Taylor & Francis. FOURTH EDITION, ISBN : 978-0-367-19617-2 (hbk), ISBN : 978-0-367-19618-9 (pbk), ISBN : 978-0-429-20337-4 (ebk).

LiudmylaLazorenko, Oksana Krasnenko [2020], "Applying Agile Learning to Teaching English for Specific Purposes", International Journal of Learning, Teaching and Educational Research, Vol. 19, No. 9, pp 238-258, September 2020, https://doi.org/10.26803/ijlter.19.9.13.

Nawaz Ali Hamdulay, Framework Study For Software Development Via Scrum, Agile, and Kanban, The Online Journal of Distance Education and e-Learning, April 2023 Volume 11, Issue 2