Agile in Software Development for Enhancing Collaboration and Quality

2024F CS555C - Agile Methods for Software Development

Project Group 4: Yash Deshpande, Akshat Sahu, Shreyas Desai, Prasanna Limaye
Instructor : Harun Gultekin

Stevens Institute of Technology

Abstract

Agile has become a revolutionary approach to software development, emphasizing flexibility, continuous interaction with clients, and teamwork harmony. In an industry where rapid change is the rule rather than the exception, Agile encourages iteration, continuous feedback, and ability to pivot quickly; hence, it is very different from step-by-step traditional models. This article looks into how Agile promotes collaboration and software quality, with the main focus on some of the prominent frameworks: Scrum, Kanban, and Extreme Programming, also known as XP. Agile, by addressing the common issues with traditional models such as limited feedback opportunity and inflexible project planning, creates an environment to involve the client into the process of welcoming changing requirements. Also, the practices of Agile contribute to an enhancement in workflow and quality of a product by facilitating free interaction and active project management. The study herein looks, through a literature review, into recent research regarding the foundational principles of Agile, the advantages compared with traditional methodologies, the challenges, and the tools that support Agile's successful implementation for enhanced collaboration and quality.

Introduction

Within the last years, the Agile methodology has turned crucial in the software development industry, turning linear models of fixed nature into an adaptive approach, customer-oriented. Traditional development models, such as the Waterfall model, cannot often keep pace with the evolution of requirements and user needs; it results in gaps in communication, long cycles of development, and a lack of correspondence between developers and their clients. These are the reasons that have pushed the industry towards Agile frameworks, which focus on iterative development, continuous feedback, and regular stakeholder collaboration. In fact, the Agile Manifesto captured this need for change quite succinctly when, in 2001, it professed four core values: individuals and interactions, working software, customer collaboration, and responsiveness to change. These form the bedrock for Agile's adaptive way of collaboration, thereby laying the foundation for flexibility in which the teams can operate to overcome complex project requirements and emerging needs.

The article discusses how Agile improves collaboration and software quality, focusing on the core enabling frameworks: Scrum, Kanban, and Extreme Programming. Each of these brings something unique to the table as one reason for Agile's pliability: Scrum provides structured sprints and roles; Kanban touts its visual task management; and XP emphasizes high-quality engineering practices. Agile ensures timely and qualitative results, closer to the customer needs through these frameworks that will further ensure satisfaction without the need for extended rework. The literature reviewed in this section discusses in detail the Agile impact on software development with regard to its main principles, frameworks, and benefits compared to traditional methodologies.

Literature Review

Agile Methodology: Origins and Principles

Agile methodology was thus born out of inefficiencies related to traditional, sequential development models such as Waterfall. Many times, the latter resulted in ineptness regarding adaptation to new requirements or customer needs during the course of development. The bedrock of Agile practices was laid by Beck et al. (2001) in the form of the Agile Manifesto, which introduced the four main values: "Individuals and interactions", "Working software", "Customer collaboration", and "Responding to change". Further research on Agile development emphasizes an iterative approach in project segmentation into cycles or sprints, each consisting of the phases of planning, developing, testing, and reviewing to ensure continuous feedback from the client and rapid iteration.

Agility encourages intimate collaboration among team members and frequent interactions with stakeholders; thus, it is effective in high-flexibility environments. Frameworks such as Scrum and Kanban also support agile principles with tangible practices that foster adaptability and communication. Scrum would implement predefined roles of Scrum Master, Product Owner, Development Team, and regular ceremonies of stand-ups and retrospectives to promote accountability and continuous improvement. Kanban, in turn, offers a more visual representation of tasks that supports WIP limits and bottleneck identification by teams.

Core Frameworks: Scrum, Kanban, and Extreme Programming (XP)

Scrum and Kanban are the most common Agile frameworks; each has a different way of dealing with structures in software projects: Scrum organizes work in sprints, generally two weeks in length, where certain project goals are set, completed, and then reviewed with stakeholders. This framework very much believes in team accountability for transparency so that the continuous progress track and iterative improvements will take place based on customers' feedback. Kanban's visual task board is targeted at teams to handle work in a flexible manner without time-based constraints and is thus ideal for projects with fluctuating requirements.

Extreme Programming gives emphasis to the engineering practices that are designed to emphasize code quality through test-driven development, continuous integration, and pair programming. XP reduces the level of technical debt through heavy testing and frequent releases of the product so the code doesn't lose reliability and never lag on account of bugs in the code. Together, these have enabled the development of various tools and practices that enable Agile teams to consistently deliver high-quality software with effective responses to changes in project requirements.

Comparative Advantages of Agile Over Traditional Models

Agile represents the complete opposite of traditional models, like Waterfall, with layers of rigidity, postponed feedback, and extended cycles of development. Indeed, literature suggests that the iterative nature of Agile facilitates the potential of a team to act on emerging needs with rapidity and avoid costly adjustments much later in the product's life cycle. Issues like misalignment between development and the expectations of clients occur very frequently in traditional models since customers are less involved. Hence, Agile methodologies involve customers in periodic reviews and ensure customer inputs at every stage of the process. Moreover, frequent test cycles that Agile goes through help in the early detection of defects occurring during the development process, and resolution therefore reduces technical debts, enhancing the quality of the software over time.

Key Benefits and Challenges of Agile Implementation

In fact, collaborative Agile better aligns the teams, frequently interactively develops accountability among team members, and enhances communications and engagement significantly. Moreover, the flexibility of Agile speeds up time-to-market since shorter cycles of development can also be updated incrementally for quick responses to changes in the market. However, the implementation of Agile does not come devoid of challenges; it is mostly difficult to have consistent documentation and avoid scope creep, where the scope of the project grows from its limited initial goal. In as much as Agile places a greater emphasis on working software over documentation, such a concentration risks overlooking crucial project documentation if not appropriately checked.

Other challenges involve adaptation to the Agile mindset: teams working with traditional models often find it rather difficult to adapt to the dynamic related to continuous feedback that Agile offers. This can be made easy for the organizations by implementing Agile practices step by step, thereby offering training programs, and using collaborative communication tools like Slack, Jira, Zoom to help in effective communication, especially for distributed teams. The ultimate advantages of Agile as a development methodology concern encouragement of teamwork, rise in the quality of software produced, and satisfaction of end-users' needs.

Agile Practices that enhance Communication

Organizations are adopting Agile approaches more and more in the fast-paced business world of today in order to stay competitive and adapt to changing consumer needs. Agile methods have gained popularity because of their capacity to enhance customer satisfaction, spur innovation, and improve project delivery. Effective communication is fundamental to Agile since it promotes teamwork, guarantees openness, and produces positive project outcomes.

Iterative and incremental development, quick feedback loops, and flexible planning are key components of methodologies like Scrum, Kanban, and Lean. By fostering a cooperative, customer-focused mentality, these strategies enable teams to react quickly to changing needs and provide value more successfully. In order to support Agile concepts and practices, lead teams through obstacles, and assist them in achieving their goals, clear and effective communication is essential.

A hierarchical structure for information sharing and thorough documentation are the main components of traditional project communication. Conventional communication plans aim to reduce or do away with the necessity of face-to-face meetings entirely and set expectations for when stakeholders will receive updates from developers and other team members. Instead, agile communication places a higher priority on increased stakeholder participation and information exchange to guarantee alignment, facilitate prompt decision-making, and enable process pivots.

Agile places a strong emphasis on team members, stakeholders, and customers interacting continuously and in real time rather than relying on extensive documentation. Some different kind of methods adopted by agile which was not done previously:

Standup Calls: In a standup call, one by one team members share their updates on what they are working on and any issues if they are facing. This is a great practice as if some other member has faced similar issue, then he/she could help on it. For Example, Jack is facing issue while pushing his code, he can quickly share his screen and other experienced people can help resolve. These sync up calls ensures everyone is aligned with their tasks and there is no communication gaps. This definitely leads to better work culture and continuous improvement. Also, it helps for team lead or whoever in charge to prepare
daily reports.

Face-to-Face Conversations: Agile unlike traditional methods want to see more face to face meetings. As mentioned earlier, traditional system wants to eliminate face to face meetings. The Agile approach is better as it promotes an environment of trust, bonding among team members. During Covid19, for a year or two everything was remote, but after that employees started returning to office at least in Hybrid mode so that they at least have some face to face interaction. Many workers reported mental health issues as completely getting remote may not be a great idea for long term.

Cross Functional Teams

Cross functional teams are small group of people from different internal departments in the organization. For example it can be IT, Marketing, Sales, Finance who colaborate and work on a similar task or goal. This ensures good collaboration and gaps are filled. Sometimes a finance guy might need some insights related to IT , this can work if there are cross functional teams. Some scenarios related to cross functional teams:

1) Amazon Mega Day sale is here and developers need proper opinion of Marketing guys so that proper stuff is displayed. 'What to display' will be taken care by Marketing guys and how properly it will be displayed without bugs has to be ensured by Developers.
2) There is a new product launched and its getting a lot of complaints from customers.Here customer support can work with Product Managers to resolve the issues efficiently.

Advantages of Cross Functional Teams:-

High Innovation - In any organization at any point of time, every department and entity is responsible for the success. Hence, it is crucial to see a bigger picture and not just stick to your own task. Cross functional teams provide that perspective and enables organizations to think broader and boost innovation. It brings all the expertise together. Because they can see the viewpoints of other capabilities, they are able to come up with comprehensive solutions to fulfill the needs of the organization.

High Efficiency - The efficiency of project completion is increased by cross-functional teams rather than a project passing via one department before being transferred to the next. Working with staff from other departments allows the team to address possible issues before the process progresses too far.

Some Downsides of Cross Functional Teams

1) Context Understanding Issues - Since the access to any kind of information is uneven among the departments, it becomes diffuclt for everyone to have a good understanding of content. Some teams may have a better knowledge, some may not understand or not even have access to stuff. This is an obstacle whoch needs to be resolved. It can be challenging for heterogeneous teams to collaborate effectively when there is a lack of common understanding since nobody is in agreement.

2) Communication Issues - People from several backgrounds and departments are together, it can create difference and gaps in viewpoint, communication styles, expertise. Also, method of documentation can be different. Definiely the diversity helps in getting creative solutions to problems, but there are some downsides and disadvantages which cannot be ruled out. It is important for organizations to create a streamlined process which will reduce the communication issues between multiple departments.

Ways to Improve

1) Align Project Goals
   Aligning project goals is crucial for ensuring that all team members understand the shared vision and objectives. By clearly articulating these goals and distributing them to

everyone, teams can foster a sense of ownership and accountability. Regularly revisiting and reinforcing these goals during meetings helps maintain focus and encourages collaboration across different functions.

2) Set a Clear Timeline
   Aligning project goals and sharing them with everyone improves cross-collaboration by ensuring that all team members understand the project's objectives and their roles. This clarity fosters a sense of shared purpose, encourages accountability, and facilitates communication among diverse functions.

Agile Ceremonies and their Impact on Collaboration

Agile ceremonies are integral to fostering a collaborative and high-performing Agile team environment. These structured meetings, which include daily standups, sprint planning, sprint reviews, and retrospectives, encourage communication, alignment, and shared accountability within the team.

Daily Standups are short, daily gatherings where team members quickly update each other on their progress, discuss any obstacles, and align on priorities for the day. By meeting consistently, the team remains informed about each member's tasks, promoting a proactive approach to problem-solving and helping identify issues early.

Sprint Planning is an important ceremony that takes place at the beginning of each sprint. A sprint is a brief, time-boxed period of time (usually 1-4 weeks) in which a team works to finish a particular set of features or tasks. This planning meet allows team to collaboratively define tasks. The team gets clarification about the goals and responsibilities. It minimizes confusion and creates a good foundation for team work.

Sprint Reviews - At the conclusion of each sprint, Sprint Reviews offer an opportunity for the team to present their completed work and gather essential feedback from stakeholders. This session keeps the team in sync with stakeholder expectations, enabling any necessary changes to keep the project on track with its goals. The atmosphere of open, constructive feedback promotes accountability and strengthens a collective commitment to ongoing improvement.

Sprint Retrospectives takes place at the end of each sprint. Here, the team discusses improvements in perofrmance, any other issues encountered. It helps team members to identify ways to work better in future sprints.Retrospectives support a culture of honesty, trust, and continuous improvement, enhancing both individual and team growth over time.

Together, these Agile ceremonies establish a consistent cadence for collaboration, enabling teams to communicate effectively, adapt quickly to change, and collectively pursue project goals. By

creating spaces for regular alignment and reflection, Agile ceremonies help teams stay connected, foster a shared purpose, and enhance the overall quality of project outcomes.

Fostering a Collaborative Culture within Agile Teams

In order to foster the flexibility, inventiveness, and responsiveness that Agile approaches require, Agile teams must cultivate a collaborative culture. Important principles like mutual respect, trust, openness, and agreement on common objectives enable team members to collaborate effectively and value each other's efforts. This creates a non toxic work environment and will eventually lead to better results.

Fostering Open Communication - There should always be an open communication, it is an essential step always. Its the responsibility of Agile leaders to promote this culture and ensure that team members are pretty comfortable with each other sharing ideas, asking doubts without fear of criticism. This enables team to be more innovative and better equipped to address obstacles.By promoting frequent reports on both successes and setbacks, this openness lays the groundwork for accountability, assisting each team member in accepting responsibility for their part in the project.

Defining Clear, Shared Goals - It is very crucial to have defined and clear goals. It reduces miscommunication among the team members and ensures that everyone is going the right way.Knowing that all are moving toward a common objective helps team members support one another, creating a stronger sense of unity and purpose.

Facilitating Cross-Functional Collaboration - It is key to strengthening team dynamics. Involving team members with diverse skills—such as developers, designers, and testers—enables Agile teams to tackle problems from multiple perspectives, promoting both creative thinking and efficient problem-solving. Cross-functional teamwork helps eliminate silos, as team members rely on each other’s unique skills to deliver a successful product. This interdependence builds trust and enhances the team’s ability to take on complex tasks collectively.

Rewarding Collaborative Efforts - It reinforces a team-focused mindset. Acknowledgint the team efforts and successes while recognizing individual contributions that benefit the group helps to promote collaboration and creates an apperciative environment where team members feel more valued about their effort. Even smallest of achievements should be counted as it is equally accountable for the success of project

Supporting Continuous Learning - Continuous learning is anyway becoming a norm in today's competitive world. Especially in agile it is even more important. Agile leaders should promote

more learning by providing resources for training and other opportunities. Sharing knowledge with each other from the team, learning from one another, solving issues together in a team leads to continuous learning.

Creating a collaborative culture requires deliberate actions, leadership support, and a commitment to Agile values. By fostering open communication, aligning on shared goals, encouraging cross-functional teamwork, and valuing continuous improvement, Agile teams can establish a collaborative environment that boosts both team cohesion and project success.

Quality Assurance in Agile Development

Quality Assurance (QA) in agile development is fundamental to ensuring software quality, reliability, performance, and overall user satisfaction. Rather than following the traditional model where quality checks are performed near the end of the project, agile QA is a fully integrated process throughout the development lifecycle. This approach promotes ongoing collaboration between developers, testers, and stakeholders, making QA a shared responsibility. Agile teams employ methods like Continuous Integration and Continuous Deployment (CI/CD), Test-Driven Development (TDD), and agile testing strategies to promptly identify and address issues, provide rapid feedback, and adapt quickly to changing requirements.

Key metrics like defect density and code coverage are invaluable for guiding agile teams. They allow teams to monitor software quality continually and refine processes where necessary. Through these practices and metrics, agile QA contributes to delivering resilient, user-centered products that uphold high standards. The essential elements of Quality Assurance in Agile Development include:

1) **Continuous Integration and Continuous Deployment (CI/CD)**

   CI and CD are foundational QA practices within agile frameworks, streamlining testing and deployment while supporting high software quality. Continuous Integration (CI) emphasizes regularly integrating code changes into a shared repository, often several times daily. Each integration triggers an automated sequence of builds and tests, allowing rapid detection and resolution of issues. This approach is particularly valuable in agile settings, where frequent, incremental updates are the norm, enabling early bug detection and fast feedback loops.

Continuous Deployment (CD) takes CI further by automating the release of code into production, once all required tests have passed. This approach accelerates the release of new features and updates by pushing them live as soon as they meet quality standards, minimizing delays and reducing manual intervention. Integrating CI/CD into the development workflow creates a seamless delivery pipeline, improves software reliability, and minimizes the risk of issues accumulating over time. CI/CD also fosters strong cross-functional collaboration across development, testing, and operations teams by promoting shared responsibility for quality and timely delivery. The transparency and consistent feedback inherent in CI/CD reinforce a culture of continuous improvement, vital for agile projects that demand high-quality, responsive software in dynamic environments (Popoola et al., 2024).

2) **Test-Driven Development (TDD)**

TDD is an agile practice that emphasizes writing tests before actual code, ensuring that new features or modules are aligned with pre-defined test criteria. By focusing on functionality and constraints before implementation, TDD reduces defect rates and enhances code quality from the start. This approach typically involves three steps: writing a failing test, coding to pass the test, and refactoring as needed. This test-first method embeds quality directly into the development process, making tests an integral part of adding new features.

TDD also simplifies refactoring, with each test serving as a checkpoint to verify that code changes do not introduce regressions. Additionally, the cumulative test suite becomes a valuable resource, documenting the system's behavior and supporting maintenance and updates. TDD promotes modular design by encouraging developers to break down functionalities into smaller, manageable components—a critical advantage in agile projects where code evolves quickly to meet new requirements (Ibeh et al., 2024).

3) **Agile Testing Strategies**

Agile testing integrates testing activities into every phase of development, diverging from traditional methods where testing occurs at the end of the cycle. In agile, testing is continuous and iterative, aligning with the rhythm of development sprints to support quick feedback and adaptability. Testing is not solely a QA team responsibility; it's a collaborative effort shared across developers, testers, and business analysts. Techniques like exploratory testing allow testers to engage directly with the application, identifying usability and functionality issues beyond those detected by automated tests.

User stories and acceptance criteria are fundamental in defining test cases in agile environments, aligning testing with user needs and business goals. Automated testing frameworks ensure consistent testing across different builds, while manual testing

techniques address areas that benefit from human judgment, such as user experience assessments. These agile testing strategies enhance responsiveness to change, reduce feedback loops, and uphold quality standards throughout the development process (Hess et al., 2019).

4) **Emphasizing Quality Metrics**

In agile, quality metrics are essential for tracking software health, supporting decision-making, and aligning development with quality objectives. Common metrics in agile include defect density, code coverage, and build success rate. Defect density, which counts defects per unit of code, provides insights into code reliability and highlights areas needing attention. High defect density suggests potential quality issues, prompting teams to allocate resources toward debugging or refactoring.

Code coverage tracks the percentage of code executed by tests, indicating test completeness. While high coverage doesn't eliminate all bugs, it does mean that most of the codebase undergoes regular verification. Build success rate measures the frequency of successful builds compared to failures, offering a quick health assessment of code stability. Regular tracking of these metrics helps agile teams maintain a strong quality focus while adapting to evolving project demands.

These metrics foster a culture of accountability and continuous improvement, essential for achieving and maintaining high-quality standards in agile projects. By relying on these data-driven insights, teams can make informed decisions that enhance software quality and optimize development practices over time (Pikkarainen et al., 2008).

Challenges and Solutions in Agile Implementation

In developing Agile software, achieving successful execution requires managing multiple complexities and challenges, mainly when striving to build a bond among teams and uphold quality standards. Below, we investigate key obstacles and active solutions informed by recent research.

1. Challenges in Embracing an Agile Culture

Transitioning to an Agile framework often involves moving farther from conventional hierarchical structures toward a collaborative, cross-functional environment. Organizations frequently face challenges here, as Agile disrupts established practices by emphasizing autonomy, open communication, and iterative work. Research by Gandomani and Nafchi highlights that resistance from employees accustomed to fixed roles and routines can impede Agile adoption (Gandomani & Nafchi, 2016).

Solution: Leadership can ease this transition by providing focused training on Agile principles and implementing changes incrementally, allowing team members to adapt gradually. Leaders should actively promote Agile principles to cultivate an environment that supports these values. Activities like role-swapping exercises can help staff embrace flexibility, while pairing less experienced team members with Agile veterans fosters trust and understanding. A culture of constant improvement, reinforced by timely feedback, embeds Agile principles into daily work and shifts Agile from being perceived as a new process to an organization-wide standard (Ambler Report, 2016).

2. Overcoming Communication and Collaboration Barriers

Strong communication is foundational to Agile's collaborative structure, but large or widely dispersed teams may encounter coordination challenges, leading to delays or misunderstandings. This is especially common in interactions between development and QA teams, where miscommunication can significantly affect outcomes. Studies in technology and healthcare reveal that more than half of respondents view inadequate communication as a primary challenge in Agile settings (IJACSA, 2024).

Solution: To enhance communication, teams need dependable channels and regular opportunities to check in, such as through daily stand-ups and retrospectives. Tools like Slack, Trello, and JIRA improve communication flow and accountability for remote or dispersed teams. Scrum's short cycles and feedback-focused approach help prevent misalignment and keep the team synchronized through frequent, structured check-ins (ScholarsBank, 2016).

3. Navigating Limited Agile Expertise

Teams without a deep background in Agile practices, like Scrum or Kanban, may struggle to implement the approach fully, often resulting in partial or surface-level adoption. Without thorough training, teams may bypass critical processes, such as retrospectives, leading to missed feedback and improvement opportunities. Studies indicate that limited Agile training is a major factor contributing to inefficiencies, as teams may revert to traditional practices or only superficially adopt Agile (PMI, 2016).

Solution: Bridging the skills gap can be achieved by investing in Agile-focused training, such as Certified ScrumMaster (CSM) programs or in-house workshops. Pairing newer team members with experienced Agile professionals also helps build essential skills. Introducing Agile principles gradually, rather than through a single intensive training session, often results in more effective and lasting adoption (Hohl et al., 2016).

4. **Scaling Agile for Large and Complex Projects**

   Agile works best with small, flexible teams, but scaling it to larger projects or organizations presents unique coordination challenges, often due to overlapping roles and objectives. Frameworks like SAFe (Scaled Agile Framework) adapt Agile practices for large-scale use but can introduce additional management layers, increasing complexity. Studies show that organizations frequently face difficulties when applying Agile across departments or in projects requiring coordination with external vendors (Ambler Report, 2016).

   Solution: A hybrid approach that combines Agile with traditional project management techniques, such as using PERT for tracking key milestones, can offer a balanced solution. Scaling frameworks like SAFe or LeSS (Large Scale Scrum) are designed to support larger teams, while a "team of teams" model—where each team operates semi-independently under a shared Agile framework—can streamline coordination and promote alignment across departments (PMI, 2016).

5. **Challenges in Ensuring Quality Assurance and Continuous Testing**

   Agile's focus on constant testing throughout development can be challenging for teams without access to automated tools or established testing protocols. Studies indicate that insufficient resources and inconsistent testing practices can negatively impact quality, as Agile often demands frequent releases (Ambler Report, 2016).

   Solution: Continuous integration (CI) and continuous delivery (CD) pipelines are essential for maintaining code quality and providing prompt feedback. Automated testing tools like Selenium for UI tests or JUnit for unit testing in Java streamline iterations and track issues, ensuring quality without slowing development. Adopting test-driven development (TDD), where tests are created before the code itself, helps maintain consistent quality across releases (PMI, 2016).

Conclusion

Agile has the potential to revolutionize software development by fostering collaboration and improving product quality. However, its success depends on careful planning to navigate inherent challenges. Organizations that promote Agile with strong leadership, an adaptable culture, and ongoing skill-building initiatives are better positioned for success. By prioritizing clear communication, iterative learning, and thoughtful scaling, teams can make the most of their Agile approach, fostering continuous improvement and achieving successful project outcomes.

References

1. https://www.sciencedirect.com/science/article/abs/pii/S0950584909002043
2. https://ieeexplore.ieee.org/document/854065/authors#authors
3. https://link.springer.com/chapter/10.1007/978-1-4471-0947-1_11
4. https://www.researchgate.net/publication/3247627_The_Scrum_Software_Development_Process_for_Small_Teams
5. https://www.sciencedirect.com/science/article/abs/pii/S1383762106000671
6. https://www.researchgate.net/publication/221473880_On_the_Impact_of_Kanban_on_Software_Project_Work_An_Empirical_Case_Study_Investigation
7. Popoola, O. A., Adama, H. E., Okeke, C. D., & Akinoso, A. E. (2024). Conceptualizing Agile Development in Digital Transformations: Theoretical Foundations and Practical Applications. *Engineering Science & Technology Journal*, 5(4), 1524-1541.
8. Ibeh, C. V., Awonuga, K. F., Okoli, U. I., Ike, C. U., Ndubuisi, L. N., & Obaigbena, A. (2024). A Review of Agile Methodologies in Product Lifecycle Management: Bridging Theory and Practice for Enhanced Digital Technology Integration. *Engineering Science & Technology Journal*, 5(2), 448-459.
9. Hess, A., Diebold, P., & Seyff, N. (2019). Understanding Information Needs of Agile Teams to Improve Requirements Communication. *Journal of Industrial Information Integration*, 14, 3-15.
10. , M., Haikara, J., Salo, O., Abrahamsson, P., & Still, J. (2008). The impact of agile practices on communication in software development. *Empirical Software Engineering*, 13(3), 303-337. DOI: 10.1007/s10664-008-9065-9.
11. Gandomani, T. J., & Nafchi, M. Z. (2016). An Empirical Study of Challenges Faced by Transitioning from Traditional to Agile Software Development. Springer.
12. Hohl, P., Münch, J., Schneider, K., & Stupperich, M. (2016). Forces that Prevent Agile Adoption in the Industry. In Proceedings of the International Conference on Software and System Processes.
13. Project Management Institute (PMI) (2016). Agile Problems, Challenges, and Failures
14. Ambler, S. (2016). Challenges in Agile Software Development Projects. Agile Modeling and Methodology Reports.
15. ScholarsBank (2016). Implementing Agile Methodology: Challenges and Best Practices. University of Oregon Library System
16. Boehm, B. and Turner, R. (2004) Balancing Agility and Discipline: Evaluating and Integrating Agile and Plan-Driven Methods. In Proceedings of 26th International Conference on Software Engineering, 718-719.