*International Journal of Engineering Research and Development*
*e- ISSN: 2278-067X, p-ISSN: 2278-800X, www.ijerd.com*
*Volume 22, Issue 3 (March 2026), PP 134-140*

# Agile Practices in-comparison to Six Sigma in Software Industries

Roopa B. Math[1], Prasadu Peddi[2]
[1]*Research Scholar, Department of Computer Science, Sunrise University, Alwar,*
[2]*Research Supervisor, Department of Computer Science, Sunrise University, Alwar*

***Abstract***
*Process improvement techniques are constantly sought after by the software business in an effort to increase delivery speed, lower faults, and improve quality. Two of the most well-known strategies are Agile, an adaptable framework that prioritizes flexibility and customer collaboration, and Six Sigma, a data-driven methodology centered on defect reduction. Despite having the same goal of enhancing software development results, their implementation, philosophy, and applicability for various project circumstances vary greatly. In this study, the principles, advantages, drawbacks, and applicability of Six Sigma and Agile techniques in the software business are compared. While Six Sigma is excellent at process stability and quality assurance, the study comes to the conclusion that Agile is more appropriate for software development settings that are dynamic and evolving quickly. For many firms, the best outcomes might be obtained with a mixed strategy.*

**Keywords:** *Agile Methodology, Software Industries, Six Sigma, Process Improvement*

---

Date of Submission: 08-03-2026 Date of acceptance: 20-03-2026

---

## I. INTRODUCTION

Software companies have been compelled to implement effective development and quality management techniques due to the quick growth of software technology and rising client expectations. Shorter development cycles and frequent requirement changes are typically difficult for traditional process improvement methods to adjust to. Methodologies like Agile and Six Sigma have therefore drawn a lot of interest. Six Sigma was first applied in the manufacturing sector before spreading to the software and service sectors as a way to reduce errors and boost productivity. In contrast, agile, which emphasizes flexibility, teamwork, and iterative delivery, arose in opposition to strict development methods. The purpose of this article is to examine the advantages, disadvantages, and application of Six Sigma and Agile methodologies in the context of the software industry. Iterative development, teamwork, and adaptability are the main focuses of agile software development approaches. They place a strong emphasis on fast releases, client feedback, and swift change adaptation. To continuously develop and provide functioning software, teams work in brief cycles known as sprints, which increases productivity and responsiveness to user requests. Agile software development (ASD) strategies are a collection of techniques that emphasize flexibility, teamwork, and iterative development. They place a strong emphasis on quick adaptation to changes, quick releases, and client feedback. ASD is an adaptable, iterative methodology created to handle intricate, quickly evolving projects and ASD approaches, in contrast to waterfall methodologies, divide a project into smaller, more manageable increments, allowing teams to continually deliver value and adjust to changes. The foundation of agile software development is the idea that teams should adapt to change rather than strictly adhere to a predetermined plan in order to provide the best outcomes. ASD is a well-liked method of developing software that places an emphasis on teamwork, adaptability, and producing functional software in brief iterations. Compared to conventional software development methods, it offers a number of benefits, such as lower risk, quicker time to market, and more customer satisfaction.

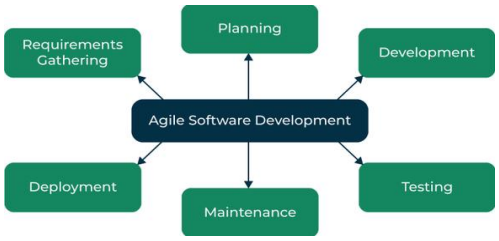



**Figure 1: Agile Software Development**
(Source: https://www.geeksforgeeks.org/software-engineering/software-engineering-agile-software-development/)

Agile prioritizes communication and teamwork to find innovative solutions to challenges, placing a higher priority on people and relationships than on procedures and equipment. Agile also encourages sustainable development and helps produce high-caliber software without causing fatigue. Fundamental principle of continuous improvement is that, teams should routinely evaluate their performance following each sprint and look for methods to increase their efficacy and efficiency in subsequent iterations.

## II. AGILE BEST PRACTICES

In Agile software development, it all adds to the software's durability, adaptability, and quality. Here is an example of some best practices for productive and successful collaboration in Agile teams to help you better grasp Agile methodology and practices. Self-organizing teams with the cross-disciplinary skill sets needed to create and test functional software are essential to the Agile methodology.

The following strategies form the foundation of the most widely used programming techniques:

1. Create the sprint backlog during the planning
2. Create initiatives centered on driven individuals
3. Develop a strategy to release planning meeting
4. Developing new solutions to lower hazards
5. Retrospective sprints to get insight from the previous sprint
6. Having the client available at all times
7. Cross-training
8. Send information in person
9. Development based on tests
10. Basic code design
11. One coding standard and a shared codebase
12. Promote self-organizing groups
13. Sprint evaluations to showcase work
14. Keep charts to track your development
15. Collaborate with the customer
16. Establishing the perfect Agile workspace where the group takes pleasure in working
17. Establishing a steady pace
18. Calculating the anticipated velocity
19 Refactoring code
20. Constant integration
21. Programming in pairs

-- -----

Organizations nowadays must quickly adjust to shifting consumer needs, technological advancements, design trends, the competitive environment, and a host of other constantly shifting factors. Agile methods of working have gained a lot of traction in a variety of industries because they foster adaptability, teamwork, and strategies for handling uncertainty. Teams and organizations must plan, deliver, and quickly adjust in response to feedback in today's VUCA reality. Teams that use agile methods are better able to prepare just enough, work together to deliver, and then evaluate and adjust in response to feedback. There is a plenty of information available online about agile, role-specific certificates, framework-specific certifications, and, of course, our own expertise derived from our prior enterprises. The entire goal of agile is undermined by the fact that there are many gaps in knowledge and practice, that organizations rarely use agile to influence their business goals, and that it has turned into a cult consisting of a number of roles, ceremonies, and practices. The product owner is in charge of establishing expectations on the sprint's objective and the order of importance of the work items. On the basis of capacity, the development team must commit to completing the work. The creation of a sprint backlog is the ultimate aim of sprint planning. This is the list of tasks that need to be completed during the sprint. However, here are some guidelines for developing a productive sprint backlog.

- Get an agreement on a SMART goal that the PO has set.
- Learn every story in depth, including the requirements for acceptance.
- Divide each narrative into manageable subtasks.
- Determine the tales' suitable estimation and delivery priority.
- As best you can, commit the stories and work items.

Setting expectations for the sprint's objective and the order of importance of the work items falls under the purview of the product owner. However, the development team is in charge of pledging to complete the work within their capacity. Developing a sprint backlog is the ultimate objective of sprint planning. The tasks that need to be completed during the sprint are listed here. However, these guidelines will help you create a productive sprint backlog.

- Have the PO create a SMART goal and reach an agreement
- Examine each story in depth using the acceptance criteria
- Divide each tale into manageable little chores
- Accurately estimate the tales and determine the delivery priority
- Depending on your capacity, commit the stories and work items

Cross functional teams are one of the main aspects to consider when you talk about forming agile teams. The team members possess all the skills required to turn a user story into functional software. One of the main best practices for the agile methodology is cross-training, in which team members learn and develop new abilities in

addition to their existing competencies. One or two team members with a particular set of skills can occasionally become a bottleneck.

A few tips that cross-training could facilitate are listed below:

- Reduces the likelihood of one individual acting as a bottleneck
- Provides more options and flexibility in planning
- Cuts down on waiting time waste and gated access
- Facilitates cooperation by making it simple to reach an agreement

## III. COMPARATIVE ANALYSIS: AGILE Vs SIX SIGMA

Six Sigma is ideal for organizations seeking process stability, quality improvement, and defect reduction, while Agile is better suited for environments requiring speed, flexibility, and frequent innovation. Many modern organizations combine both approaches to balance quality and adaptability.

Similarities between Six Sigma and Agile: 1) Both aim to improve organizational performance 2) Both rely on team collaboration 3) Both emphasize continuous improvement.

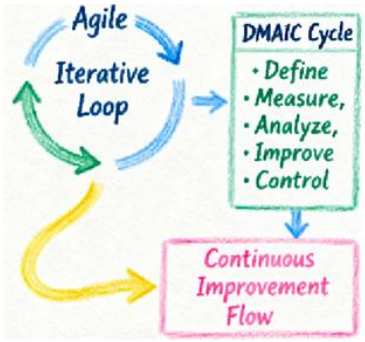
Table 1. Comparison of Six Sigma Vs Agile

| Aspect | Six Sigma | Agile |
|---|---|---|
| Definition | A data-driven methodology focused on improving quality by reducing defects and process variation. | An iterative project management approach focused on flexibility, collaboration, and rapid delivery of value. |
| Origin | Developed by Motorola in the 1980s and later popularized by General Electric. | Originated in software development with the Agile Manifesto created by a group of software experts in 2001. |
| Primary Goal | Improve process quality and reduce defects to near perfection (3.4 defects per million opportunities). | Deliver working products quickly and adapt to changing customer requirements. |
| Approach | Predictive and data-driven | Adaptive and iterative |
| | Uses a structured, statistical approach such as the DMAIC cycle. | Uses iterative cycles called sprints and continuous feedback loops. |
| Primary Focus | Defect reduction and quality | Flexibility and rapid delivery |
| | Process efficiency, defect reduction, and quality control. | Customer value, team collaboration, and rapid adaptation. |
| Flexibility | Less flexible due to structured methodology and heavy data analysis. | Highly flexible and adaptable to change. |
| Team Structure | Hierarchical roles such as Green Belt, Black Belt, and Master Black Belt. | Cross-functional teams with roles like Product Owner, Scrum Master, and Development Team. |
| Industries Used | Manufacturing, healthcare, finance, logistics. | Software development, IT, startups, and product development. |
| Tools & Techniques | Statistical tools, process mapping, root cause analysis, control charts. | Scrum boards, sprint planning, daily stand-ups, backlog prioritization. |
| Timeframe | Typically, long-term projects focusing on deep process improvement. | Short development cycles (Usually 1–4 weeks per sprint). |
| Speed | Slower implementation | Faster delivery cycles |
| Documentation | Extensive | Minimal but sufficient |
| Customer Involvement | Limited | Continuous |
| Project Size Suitability | Large, stable projects | Small to medium, dynamic projects |
| Change Handling | Resistant to frequent changes | Embraces change |

## IV. AGILE AND SIX SIGMA: A HYBRID APPROACH IN SOFTWARE INDUSTRIES

Organizations in today's software sectors frequently struggle to produce software fast, maintain high product quality, and adjust to shifting client needs. Both needs are successfully met by a hybrid strategy that blends Agile and Six Sigma.

1. **Need for a Hybrid Approach:** Software development environments are dynamic and customer-driven. Agile methods enable teams to deliver software rapidly through iterative development, while Six Sigma focuses on improving processes and reducing defects. Integrating both methodologies allow companies to achieve speed, flexibility, and quality simultaneously.
2. **Combining Strengths of Both Methods:** Agile offers regular releases, iterative development, and ongoing client feedback. Six Sigma offers statistical analysis, defect reduction methods, and systematic problem-solving. Organizations may guarantee the timely delivery of software products while retaining high performance and reliability by combining these capabilities.
3. **Improved Quality and Reduced Defects:** Organizations may guarantee the timely delivery of software products while retaining high performance and reliability by combining these capabilities.
4. **Faster and More Reliable Delivery:** Faster releases are made possible by Agile's sprint-based workflow, while Six Sigma maintains the effectiveness and consistency of the underlying development processes. This hybrid methodology helps businesses achieve deadlines without compromising quality by lowering rework and increasing efficiency.
5. **Better Decision Making Through Data:** Six Sigma places a strong emphasis on using measurements and analysis to make data-driven decisions. Teams may make better decisions when they use Agile methods like sprint reviews and retrospectives with quantitative data from process metrics and qualitative customer input.
6. **Industry Adoption:** In order to increase development speed and product reliability, many tech organizations, such as IBM, Microsoft, and Amazon, have implemented hybrid models that combine Agile processes with formal quality control procedures.




- Agile places a strong emphasis on quickness, early delivery, and adaptability. Agile teams quickly adjust to changing customer needs through brief iterations and ongoing feedback.
- Lean Six Sigma Offers a framework for methodically locating and eliminating waste and flaws, guaranteeing that the final product satisfies quality requirements and minimizes rework.
- The hybrid model combines discipline and speed. It enables businesses to deliver iteratively and continually optimize, resulting in scalable improvement that is quick and under control. Not only is delivery quicker, but it's also more intelligent and data-driven.

**Figure 2. The Hybrid Future: Scaling Agile with Lean Six Sigma Excellence** (Source: https://www.linkedin.com/pulse/hybrid-future-scaling-agile-lean-six-sigma-excellence-ashish-jain-shkgf/)

Software industry greatly benefit from a hybrid approach that combines Six Sigma and Agile. Six Sigma guarantees great quality and process control, while Agile guarantees flexibility and quick development. When combined, they provide a well-balanced framework that enables businesses to effectively produce high-quality software while responding to shifting consumer demands. The combined approach works particularly effectively in settings that prioritize continuous assessment, open communication, and customer-focused tactics. An IT support department may gain from applying Lean methodologies to map out its ticket resolution process, identify bottlenecks, and then use short Agile cycles to iterate on improvements. The team can respond to new issues almost instantly by looking at data on ticket traffic and response times. Lean Six Sigma's careful focus on defect reduction may be incorporated into Agile processes in a software development setting, guaranteeing that every new feature is carefully measured, examined, and optimized for efficiency and dependability.

## V. RISK INVOLVED IN AGILE PRACTICES

Agile approaches are popular in software development, but there are hazards associated with them that businesses need to be aware of.

- **Documentation:** Agile prioritizes functional software over copious documentation. Although this expedites development, it could cause issues later on when teams want comprehensive documentation for system updates, maintenance, or training.
- **Scope:** Project requirements may change often since Agile allows for ongoing modifications and user feedback. This may result in scope creep, which is the ongoing addition of new features that raises project costs and delays.
- **Dependence on Team Collaboration:** Agile greatly depends on team members' cooperation and communication. The team's productivity and project results may suffer if there is a lack of commitment, coordination, or experience.
- **Difficulty in Predicting Timelines and Costs:** Estimating the precise time, money, and resources needed for the complete project at the outset may be challenging because Agile projects are flexible and iterative.
- **Limited Suitability for Large Projects:** Agile is most effective when applied to small to medium-sized teams and projects. Managing Agile methods can be difficult in very big projects with several teams and intricate dependencies.
- **Customer Availability Risk:** Agile demands ongoing input from stakeholders or customers. Decision-making may be slowed down and the quality of the finished work may suffer if the client is unavailable for frequent evaluations or discussions.
- **Skill Requirements:** Agile teams require competent project managers, developers, and testers who can quickly adjust to changes. The project may incur delays and inefficiencies if the crew is unfamiliar with Agile methodologies.

In order to control the risks involved in a development project, a software solution to an issue must be taken into account in the domain's broader context. For instance, it makes no difference how much software has been built if delays force the team to miss the market "window" because the software is no longer useful to the user and customer. Risk management is the process of evaluating hazards and taking action to lessen them; it is a crucial aspect of software development. Any project will probably include a trade-off between the software system's capability and what can be produced within a particular time frame and budget. Choices must be made since a solution's desirable qualities frequently outweigh those that can be provided for a given cost and timeframe. Calculating the risk of not delivering each feature is one method for making such decisions. According to an agile methodology, requirements should be continuously reviewed because they will change as the project progresses. It is simpler to handle requirements changes and lowers the chance of making a poor choice when consumers are included at every stage of development.

Combining Agile with Lean Six Sigma provides a comprehensive strategy for ongoing development. Lean Six Sigma offers the rigorous problem-solving framework required to improve performance, whereas Agile alone occasionally finds it difficult to address deeper process differences. Lean Six Sigma programs are kept flexible and closely connected with changing requirements. Agile's emphasis on end-user feedback. Businesses that are successful in combining these two approaches should anticipate increased productivity, higher-quality results, and a culture that is always learning, assessing, and developing. Businesses can easily traverse complexity and innovate at a speed that keeps them one step ahead of the competition by utilizing the finest aspects of both worlds. Using these techniques frequently necessitates a change in corporate culture. Lean Six Sigma's openness to data sharing and group problem-solving complements Agile's emphasis on cooperation and feedback. In this setting, coaches should promote a culture of experimentation so that teams are empowered to make suggestions for enhancements and quickly fix any inefficiencies. Transparency is essential: disclosing KPIs, project schedules, and process results encourages accountability and keeps everyone focused on objectives.

## VI. DISCUSSION

A key conceptual distinction between Six Sigma and Agile approaches is shown by the comparative study of the two approaches. Agile places more emphasis on flexibility, quick feedback, and customer-centric development than Six Sigma, which is based on process control, statistical rigor, and defect elimination. The way each technique functions in software development environments is greatly influenced by these opposing foundations. Six Sigma shows distinct benefits in software businesses with repeatable processes and consistent requirements, like corporate system upgrades, maintenance initiatives, and regulated domains. Organizations may find inefficiencies, lower failure rates, and get predictable results with its DMAIC methodology. The focus

on measurement and control promotes adherence to industry standards and improves quality assurance. However, Six Sigma is less appropriate for software projects that are experimental or driven by innovation because of its strong emphasis on documentation, statistical analysis, and formal reviews, which can limit responsiveness.

On the other hand, agile processes are made to function well in unpredictable and changing environments. Agile enables software teams to react swiftly to changing needs by encouraging brief development cycles, continuous integration, and frequent client contact. Higher customer satisfaction and quicker delivery of company value are frequent outcomes of this flexibility. However, Agile's weaker focus on official documentation and long-term process metrics can cause problems with scalability, cost estimation, and consistency of quality, especially with big or dispersed teams. Organizational culture is another crucial factor. Six Sigma calls for specialized positions (like Black Belts and Green Belts), hierarchical decision-making, and a strong dedication to data-driven management. Conversely, agile works best in settings that encourage team autonomy, cooperation, and decentralized management. Businesses that try to apply either approach without coordinating structural and cultural components frequently have little success.

The software industry has recently shown a rising affinity for hybrid approaches that blend the flexibility of Agile with the analytical discipline of Six Sigma. Quality can be increased without sacrificing speed by incorporating Six Sigma tools into Agile iterations, including as performance data, control charts, and root cause analysis. Agile techniques can also make Six Sigma projects more customer-focused and responsive. The industry's realization that no one approach can adequately handle the intricate requirements of contemporary software development is reflected in this convergence. Six Sigma and Agile meet various organizational needs, as the comparison shows. Six Sigma works best in settings where stability is essential and procedures are clearly defined. Agile works best in situations that are unpredictable and where requirements change frequently. Software companies frequently encounter difficulties when implementing Agile without adequate quality controls or when applying Six Sigma rigorously to creative development methods. The acceptance of hybrid models, such Lean Six Sigma with Agile, which combine the flexibility of Agile methods with the quality focus of Six Sigma, is indicated by recent trends in the industry.

In order to increase flexibility, product quality, and software product delivery speed, agile approaches are widely used in contemporary software companies. Agile emphasizes teamwork, iterative development, ongoing feedback, and flexibility. Agile-based methods like Scrum, Kanban, and Extreme Programming have replaced traditional models like the Waterfall Model in many organizations. Agile methodologies have revolutionized the software sector by facilitating quick, adaptable, and customer-focused development. Software firms may increase efficiency and produce high-quality products by implementing iterative development, continuous testing, and good teamwork. However, organizational commitment, appropriate training, and efficient team communication are necessary for successful implementation.

## VII.CONCLUSION

Agile approaches place a strong emphasis on close communication between developers and stakeholders, iterative development, and ongoing feedback. This strategy increases customer satisfaction, improves product quality, and enables businesses to produce useful software in shorter development cycles. But there are drawbacks to Agile as well, such poor documentation, a reliance on teamwork, and trouble forecasting budgets and schedules. To guarantee successful execution, these difficulties demand competent teams, appropriate project management, and good communication. Agile methods, on the other hand, place more emphasis on incremental delivery, customer collaboration, and flexibility. Agile is especially well-suited for competitive, dynamic marketplaces where requirements change often because of these features. Agile's speedy delivery of functional software improves time-to-market and customer happiness, but it may also present issues with scalability, documentation, and quality standardization. According to the study's conclusions, project features, organizational maturity, and business objectives should all be taken into consideration when choosing a methodology.

Software companies can gain from combining Six Sigma and Agile's complementary strengths rather than seeing them as rival methodologies. Teams can uphold high standards of quality while staying adaptable with a hybrid model. With an emphasis on their guiding principles, benefits, drawbacks, and applicability, this study analyzed and contrasted Six Sigma and Agile methodologies in the context of the software industry. Although both approaches seek to enhance software development performance, the research shows that they do it from essentially different angles. Six Sigma provides a methodical, metrics-based methodology that is very successful in lowering errors, increasing process effectiveness, and guaranteeing consistency in environments that are stable and clearly defined. Long-term process improvement, predictability, and quality assurance are its strong points.

Because they encourage adaptability, teamwork, and quick software product delivery, agile methods have gained widespread acceptance in the software sector. Development teams may react swiftly to shifting customer needs and market expectations by adhering to the Agile Manifesto's tenets. All things considered, Agile methods are crucial to today's software industries because they help businesses stay creative, flexible, and competitive in a quickly changing technological landscape. Agile, when used properly, enables businesses to exceed customer expectations while producing high-quality software solutions quickly. However, in software projects that are developing quickly, its rigidity and large implementation overhead might limit adaptability and impede innovation.

## VIII. FUTURE SCOPE

The software industry benefits greatly from Six Sigma and Agile approaches. Agile encourages adaptability and customer-focused growth, whereas Six Sigma offers organized process improvement and defect reduction. Neither approach by itself is always the best. Business goals, corporate culture, and project type all influence the decision. A hybrid strategy that combines the two approaches can produce better outcomes by guaranteeing high-quality software and preserving adaptability. Future studies might concentrate on quantitative performance comparisons across various project kinds, industry-specific adoption techniques, and empirical assessments of hybrid frameworks. The ability to strike a balance between discipline and agility will continue to be a crucial component of software organizations' success as software systems continue to increase in complexity and scale.

## REFERENCES


[1] Gonen B. and Sawant D. (2020). Significance of Agile Software Development and SQA Powered by Automation. In 2020 3rd International Conference on Information and Computer Technologies – ICICT, pp. 7-11.

[2] Jain P., Sharma A., and Ahuja L. (2018). The Impact of Agile Software Development Process on the Quality of Software Product. In 2018 7th International Conference on Reliability, Infocom Technologies and Optimization: Trends and Future Directions – ICRITO, pp. 812-815.

[3] Saleh S., Huq S., and Rahman M. (2019). Comparative Study within Scrum, Kanban, XP Focused on Their Practices. In 2019 International Conference on Electrical, Computer and Communication Engineering – ECCE, pp. 1-6.

[4] Sharma P., and Hasteer N. (2016). Analysis of linear sequential and extreme programming development methodology for a gaming application. In 2016 International Conference on Communication and Signal Processing – ICCSP, pp. 1916-1920.

[5] H. Edison, X. Wang and K. Conboy, "Comparing Methods for Large-Scale Agile Software Development: A Systematic Literature Review," in IEEE Transactions on Software Engineering, vol. 48, no. 8, pp. 2709- 2731, 1 Aug. 2022, doi: 10.1109/TSE.2021.3069039.

[6] A. Agarwal, N. K. Garg and A. Jain (2014)., "Quality assurance for Product development using Agile," 2014 International Conference on Reliability Optimization and Information Technology (ICROIT), Faridabad, India, pp. 44-47, doi: 10.1109/ICROIT.2014.6798281.

[7] Felix Temole and Desislava Atanasova (2025). Agile Integration in Software Development: Principles, Practices, and Challenges, Software Engineering, Vol-11, Issue-1, pp: 18-29

[8] 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.

[9] 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.

[10] 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

[11] M. Kuhrmann et al. (2021), “What Makes Agile Software Development Agile?,” arXiv preprint arXiv: 2109. 11435, https://doi.org/10.48550/arXiv.2109.11435.

[12] K. P. Pham and M. Neumann (2025). "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.

[13] M. Huss, D. R. Herber and J. M. Borky (2023). "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.

[14] N. Yang, X. Wang, Z. Zhang, D. Siemon and S. Hyrynsalmi (2025). 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.