Intelligent Multimedia Processing & Communication Systems Journal

IMPCSJ

J IMPCS (2025) 19: 1-7
DOI 10.71856/IMPCS.2025.1200970

Research Paper

A Reliability-Focused Adaptation of Scrum: An Empirical Study on Software Reliability in Agile Teams

Taghi Javdani Gandomani[*]

Associate Professor, Department of Computer Science, Faculty of Mathematical Sciences, Shahrekord University, Shahrekord, Iran. **Corresponding Author* javdani@sku.ac.ir

Article Info

Article history:
Received: 24 Dec 2024
Accepted: 3 Feb 2025

Keywords:
Agile Software Development,
Case Study,
Scrum,
Software Reliability.

ABSTRACT

Software products should be reliable enough to gain customer satisfaction. Paying more attention to the development of highly reliable software products requires additional time and effort in software projects, often leading to increased development overhead. This issue presents a challenge when employing Agile methods, which prioritize flexibility and rapid iterations. While Agile methodologies emphasize adaptability, integrating extra practices to ensure objective reliability remains a critical concern. In recent years, the demand for high-quality and failure-resistant software has significantly increased, necessitating new strategies to incorporate reliability engineering into Agile frameworks. Given the growing complexity of modern software systems, achieving a balance between agility and reliability is crucial for software development teams.
This paper investigates the combination of Scrum, the most widely adopted Agile method, with software reliability engineering (SRE) practices. The study evaluates the additional cost of implementing reliability-focused practices in a Scrum-based project through a case study. A comprehensive analysis was conducted to assess the impact of these practices on project timelines, software defects, and overall team performance. The study also examined the extent to which reliability-focused modifications influence team productivity and customer satisfaction. The results indicate that the modifications made to Scrum led to approximately 3.9% higher human effort. However, the integration of SRE practices significantly reduced the number of failures and rework occurrences, demonstrating the effectiveness of the proposed approach. Moreover, the study highlights the importance of structured reliability assessment techniques, which can aid teams in proactively identifying and mitigating potential software failures before product deployment.
These findings suggest that, despite a marginal increase in project cost, the enhanced reliability justifies the investment. The proposed methodology can serve as a model for Agile teams seeking to improve software quality while maintaining development speed and flexibility. This study highlights the potential of balancing agility and reliability in software development, offering valuable insights for Agile teams aiming to improve software quality without compromising efficiency.

Copyright © Author(s).
Publisher: Islamic Azad University – Zanjan Branch

I. Introduction

Software teams and companies continually are working on employing new methods, techniques, and processes to increase customer satisfaction and subsequently take more market share in the software industry. Employing more flexible and productive software development methodologies, development frameworks, and best development practices are parts of their strategies.

Scrum is the most popular Agile software methodology, recently employed by software teams [1]. This method, which is mainly known as a development framework, is often used to manage software projects because of its focus on project management. Scrum defines a small set of roles, activities, and artifacts to simplify software development. However, adapting to such an Agile methodology is not as easy as expected [2-4]. Meanwhile, since the Agile approach generally, and Scrum particularly promise higher customer satisfaction through faster return on investment, embracing the requested changes, and such tempting values, software teams are interested in adapting with Scrum despite its adoption challenges [5, 6].

The reliability of a software product is a quality factor that many software teams pay great attention to it [7-9]. Also, a reliable software product gets more customer satisfaction. Indeed, software teams try to get particular practices to ensure that their artifacts are reliable enough before their final release. Thus, they need to adapt to reliability focused practices or processes to get a higher degree of reliability in their development methodologies [10]. Software Reliability Engineering (SRE) is a part of Software Engineering, which has mainly focused on reliability-related issues [11, 12].

A big issue is employing non-agile practices while using an Agile methodology in software development. Since now, no Agile specific reliability process has been introduced [13-16]. So, applying any SRE processes within any Agile method may lead to less agility, which can be contradictory with Agile values promised in the Agile manifesto [17, 18]. This study mainly aimed to show to what extend the project metrics can be affected by employing SRE activities in a Scrum project.

The rest of the paper is organized as follows: Section 2 briefly explains the Scrum framework. Section 3 introduces the SRE process, followed by Section 4, which shows Scrum's combination and the SRE process. Section 5 deals with the results of the application of the proposed framework in a Case Study. Finally, Section 6 concludes the paper and gives some research insights for future work.

II. Scrum

Scrum was addressed by Ken Schwaber and Jeff Sutherland in the early 1990s and formally introduced in 1995 [19]. After that, by the creation of Agile manifesto, Scrum has been considered as an Agile methodology. Scrum emphasizes flexibility, openness, courage, trust, focus, collaboration, and empowerment [20, 21]. These values, as Scrum developers claim, can facilitate project management and decrease the risks that a software project may encounter. Indeed, Scrum acts more like a project management framework rather than a software development process. This is the main reason why software teams employ Scrum together with other Agile methodologies.

Scrum defines a simple software development framework by considering three artifacts, including Product Backlog, Sprint Backlog, and Sprint Burndown chart. Product Backlog contains customer requirements list, known as User Stories, which are sorted in based on their priority from the customer perspective. Sprint Backlog is a subset of Product Backlog picked up by the Scrum team to develop iteration, known as Sprint. Sprint Burndown chart is the Scrum tool to show the project's progress based on the remaining work.

Scrum provides three roles, only including Scrum Master, Product Owner, and Development team members. Scrum Master helps Scrum team members to do their tasks and avoids them from unpredictable risks and challenges. He/she acts as a servant leader and promotes self-organization. The Product Owner is a customer representative who creates and manages the Product Backlog and defines software project direction. All technical developers are known as Scrum team members or simply Scrum developers. They are responsible for software development activities.

Scrum also has a few main activities, including Sprint Planning, Sprint execution, Sprint Review, and Retrospective meeting. In Sprint Planning, Scrum team members select some of the User Stories from Product Backlog and create Sprint Backlog. They also break the selected User Stories into several tasks to be done in the comping Sprint. In Sprint execution, Scrum team members try to realize the content of the Sprint Backlog. Each Sprint is a time-box cycle, often 2 to 4 weeks. Each day starts with a short meeting called "Scrum daily meeting" to review the challenges faced with the past day and talk about their today's work. At the end of each Sprint, the Scrum team sits with the customer to review the newly developed product increment in a meeting called Sprint Review meeting. Finally, Scrum team members review the past Sprint and adjust their development process if necessary. The scrum framework is shown in Figure 1.

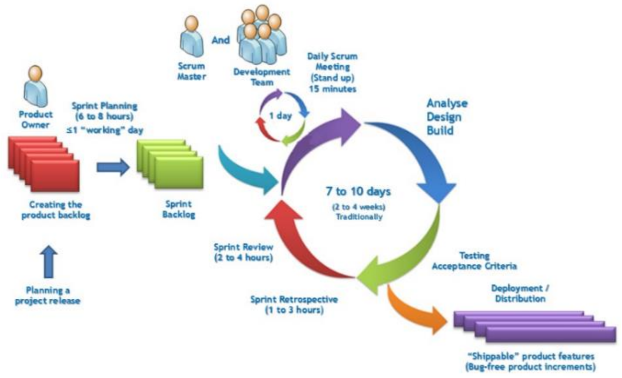
Scrum Master
And
Development Team
Daily Scrum Meeting (Stand up) 15 minutes
Product Owner
Sprint Planning (6 to 8 hours) ≤1 "working" day
1 day
Analyse Design Build
Creating the product backlog
Sprint Backlog
7 to 10 days
(2 to 4 weeks) Traditionally
Sprint Review (2 to 4 hours)
Testing Acceptance Criteria
Planning a project release
Sprint Retrospective (1 to 3 hours)
Deployment / Distribution
"Shippable" product features (Bug-free product increments)

Fig. 1. Scrum Framework

Scrum does not force the team to employ specific activities, methods, and practices besides its defined ones. But usually, software teams need to use particular practices to achieve other development goals such as the development of a defect-free product, reducing the development risks, and so on. The development of highly reliable software products is another goal of Scrum development. However, such practices may reduce the agility promised by Scrum.

III. Software Reliability Engineering

Reliability is a key factor in the success of a software product mainly because it is focused on by its users. The focus of SRE is primarily on reliability, not defects. SRE deals with the measurement and improvement of the reliability of software products. Therefore, SRE needs a quantitative approach to measure the reliability of each software product in real environments. To do this, various practices can be defined and employed while the software is under construction.

ANSI/IEEE defined software reliability as "the probability of failure-free software operation for a specified period in a specified environment" [22]. The roots of software failures are errors and faults during software development phases and activities, including requirements definition, analysis, design, implementation, test, and deployment [11]. However, most often, these errors and faults remain hidden until a failure occurs.

Software reliability is a concern in the field of "software quality." Software engineers need to use various quantitative data and information to select the most suitable strategies to deal with reliability engineering in their software projects. Many practices are defined in SRE to handle reliability-related issues in a software project. Some of these practices are the definition of reliability objectives, using operational profiles to manage and guide the test process, failure tracking, using reliability growth strategies, and releasing the product only when meeting reliability objectives [11, 23].

John Musa [11] defined a particular process to organize SRE, as shown in Figure 2.

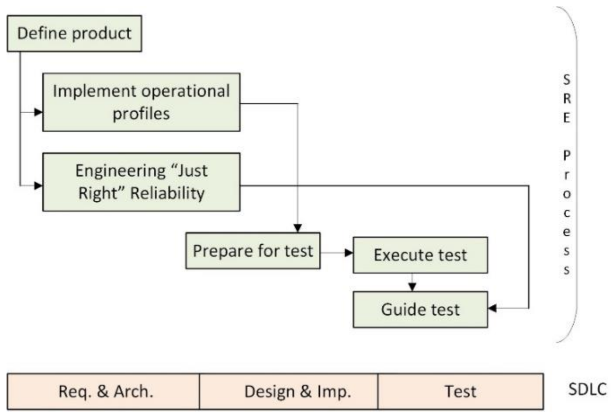

Fig. 2. The general process of Software Reliability Engineering [11]

The definition of the expected product and highlighting its specific characteristics is the first phase of SRE. To measure SRE metrics, this process suggests using operational profiles that contain a list of operations and their occurrence frequency. At the same time, particular metrics and measures need to be defined and used to ensure providing a quantitative method to assess the reliability of the product. After that, the team should be prepared for testing by precise planning and estimation. Then, the required and possible tests should be run. Simultaneously, the test should be directed so that risk-prone areas should be tested carefully and ensure that all paths are covered by at least one test case [11, 14, 24, 25].

The application of this process needs proper human resources and may affect the cost and time of the project [26]. As mentioned earlier, this can be a severe threat to the project cost. Particularly, for lightweight software development methodologies, like Scrum, it would also be a threat to the agility of the development framework [27, 28]. However, paying the objective reliability is so important that it is necessary to make a clear decision about it. Agile teams often worry about additional costs they need to pay for reliability [13, 16, 26, 29]. This is the main reason for conducting this study.

IV. The Proposed Reliable Scrum

In this study, a customized version of the Scrum framework has been proposed and then employed in a Case Study. Figure 3 shows the proposed framework.

A. Changes made to Scrum framework

As shown in the Figure 3, three changes, including one role, one artifact, one activity, have been made to the Scrum framework to focus on the SRE. The Reliability Engineer is responsible for directing the Scrum team to establish the SRE process. This role works with the Product Owner for managing Product Backlog and adding reliability-related items, if necessary. Also, he/she works together with Scrum developers to clarify those items during each Sprint. Also, the Reliability Engineer is directly responsible for creating and managing the Operational profiles document as an extra added artifact. This artifact is the same as defined in the SRE process [11]. Finally, at the end of each Sprint and before starting Sprint Review, Software Reliability Assessment should be performed by Reliability Engineer, Product Owner, and Scrum Team Members to ensure they have committed to the expected reliability.

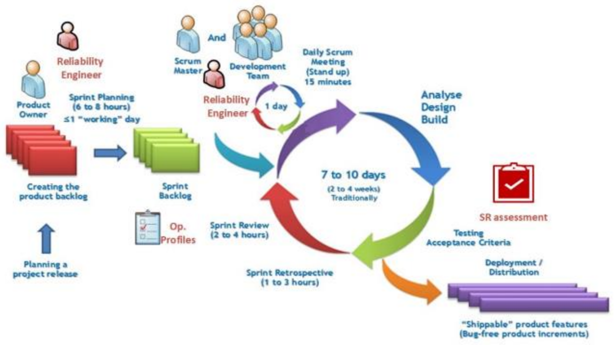
Reliability Engineer
Product Owner
Sprint Planning (6 to 8 hours) ≤1 "working" day
Scrum Master
And
Development Team
Reliability Engineer
1 day
Daily Scrum Meeting (Stand up) 15 minutes
Analyse Design Build
Creating the product backlog
Sprint Backlog
7 to 10 days
(2 to 4 weeks) Traditionally
SR assessment
Op. Profiles
Sprint Review (2 to 4 hours)
Testing Acceptance Criteria
Planning a project release
Sprint Retrospective (1 to 3 hours)
Deployment / Distribution
"Shippable" product features (Bug-free product increments)

Fig. 3. The reliability-focused version of Scrum

B. Case Study

This study has been conducted in a software company that had more than 30 software developers. They were well-adopted Scrum framework for about three years. This company was working in core banking systems for more than one year. To conduct this study, a particular sub-system was selected. This sub-system was a part of a banking mortgage system that was estimated to be implemented in 7 two-week Sprints by about 3000 person-hours. In this study, two teams were assigned voluntarily; each consists of 5 developers. Team 1, Control Team (CT), worked on the project without considering SRE and its related practices. Team 2, Study Team (ST), started the project using the customized Scrum framework, as depicted in Figure 3. It should be noted that all ST members participated in a one-day workshop to be familiar with the proposed framework and SRE related aspects in the coming project. Fortunately, senior management agreed that both teams work simultaneously on the same sub-system. Also, both teams had almost the same technical skills and experiences and well-familiar with the Scrum framework.

C. Metrics

Although both teams had five developers, one part-time expert as Reliability Engineer has been added to ST. At first glance, it seemed that the addition of this expert would have led to an increase in the human effort in ST, but, the researchers hoped that by focusing on SRE and in turn, proper test management and distribution, the reduction of human effort and cost is not far off. To measure the effectiveness of the framework in its defined goal, some metrics have been considered, and their values have been collected manually during each Sprint. These metrics are as follows:

- Person-hours in each Sprint (PH): this metric shows the real amount of human effort in each Sprint.
- The number of failures (NF): This metric shows the number of failures reported by customers after delivery of each Product Increment. Indeed, the customer worked with the delivered Product Increment and listed the occurred failures.
- Number of Re-works (NR): This factor indicates the number of re-works caused by failure (after delivery).

Besides these criteria, some other metrics have been collected, but since they are not directly related to SRE, they are ignored in this article

V. Results and Discussion

As mentioned in the previous section, three main metrics were focused in the Case Study. This section shows the results of each metrics during the completed project.

A. Person-hours/ Project cost

This factor was used to compare the labor cost in both teams, CT and ST. Table 1 shows the value of this metric in the completed Sprints in both teams.

TABLE I Person-hours in the Case Study

Sprints	SP1	SP2	SP3	SP4	SP5	SP6	SP7	ALL
PH in CT	380	390	420	400	410	420	410	2830
PH in ST	410	420	430	420	410	430	430	2940

Figure 4 shows person-hours in all Sprints during the Case Study. As shown in the figure, it seems that in all the Sprints, ST paid more person-hour compared to the CT team. In CT, normal teamwork time could be about 400 hours in each Sprint. This number could be about 440 for ST because they hired a half-time Reliable Engineer. However, in 5 Sprints (SP3 to Sp7), CT had extra work, more than the maximum number of working hours allowed in each Sprint. But it seems that ST usually works without additional work. This is known as a good sign for the team's workflow.

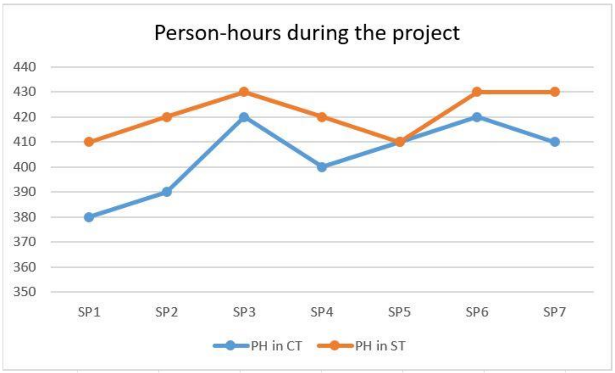

Fig. 4. Number of staff-hours spent during the project

Comparing total human effort in both teams shows that 2830 hours have been spent in the CT. This number has increased to 2940 hours in the ST team. Indeed, the cost of the project by employing SRE has increased by 3.9%.

B. Number of failures

The second metric was the number of failures reported by customers after each small release, i.e., each Sprint. The data related to this metric were collected from the end of the second Sprint to project completion, Sprint by Sprint. Table 2 shows these data.

TABLE II Number of failures reported by the customer

Sprints	SP1	SP2	SP3	SP4	SP5	SP6	SP7	ALL
NF in CT	-	4	4	5	5	6	7	31
NF in ST	-	4	4	4	3	3	3	21

As shown in Table 2, it seems that employing the proposed framework led to a notable decrease in the number of detected failures. Although this result was expected, the reduction rate was amazing. Indeed, ST experienced about 30% less failure compared to CT. Figure 5 shows the data regarding the number of reported failures.

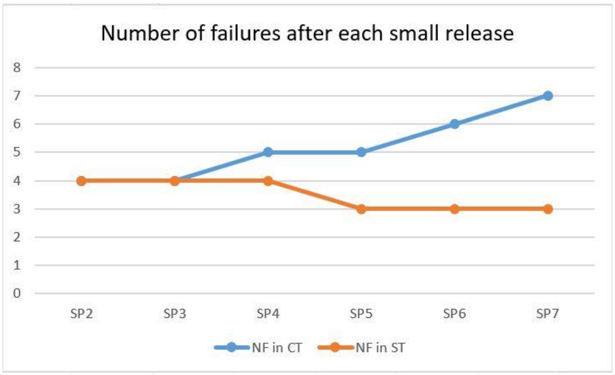

Fig. 5. Number of failures reported by the customer

Figure 5 shows that detected failure was increasing in CT, while this metric was declining in ST. This indicated the product developed by CT would encounter more risk of failure in the future. This fact the reality that managers need to consider in their marketing decisions.

C. Number of re-works

This metric was used to count the number of re-works related to the failures. It should be said that most often, failures can be repaired without re-work. But, sometimes, a failure leads to re-work. This increases the cost and risk of the development process as well as customer dissatisfaction. Table 3 shows the data related to this metric.

TABLE III The number of re-works rooted in the occurred failures

Sprints	SP1	SP2	SP3	SP4	SP5	SP6	SP7	ALL
PH in CT	380	390	420	400	410	420	410	2830
PH in ST	410	420	430	420	410	430	430	2940

Table 3 shows a substantial decrease in the number of re-works in ST. Indeed, ST experienced less than about 50% regarding the number of re-works compared to CT. Figure 6 also shows these data. Figure 6 shows that CT encountered more re-works in all Sprints. This is the main reason that CT should spend extra effort in each Sprint.

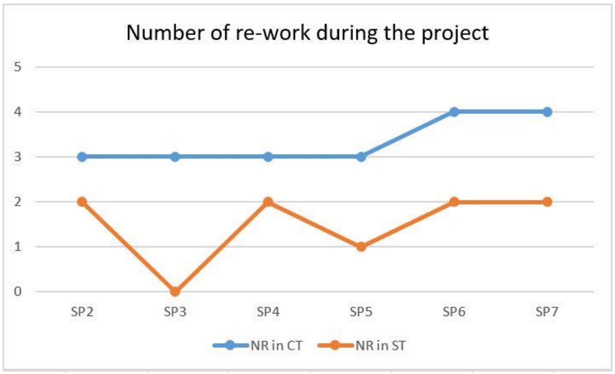

Fig. 6. Number of necessary re-works reported by the customer

Reviewing the results shows that employing the proposed framework had led to significant findings. However, the project cost increased a little bit; the number of occurred failures and re-works were decreased dramatically.

VI. Discussion

As mentioned in the previous sections, Agile software development and its methodologies have left several development issues untouched. This is mainly due to maintaining the flexible nature of the promised Agile methods. Software reliability and its required process are one of such issues. However, SRE and its process play a vital role in software engineering and software development methodologies, and neglecting such a process can cause a lot of damage to the software products [18, 24].

As explained in this study, the proposed version of Scrum tries to handle the potential reliability risks in Scrum projects. The changes made to the Scrum framework are to manage the SRE process within the Scrum framework. However, no change has been made on each Sprint structure, as addressed by other researchers [30]. The new role, Reliability Engineer, is responsible for handling SRE issues during the Scrum project. Adding new roles is a common strategy for managing specific concerns in Agile methodologies [31-33]. This is the same for new artifacts and activities too.

The revised version of Scrum results in some improvements and advantages in software development. The most significant improvement was on the number of failures reported by the customer. A 30% reduction in this item is amazing and heralds the efficiency of the proposed framework. However, since the related works have not reported any quantitative improvement, this study cannot compare this achievement to the previous studies.

Another achievement was a severe reduction in the number of re-works rooted in the occurred failures. This benefit leads to a reduction in the project cost directly. However, in the Case Study reported in this research, the project's person-hour increased by 3.9%. Adding one person, i.e., a Reliable Engineer, obviously increases the project cost, but applying the SRE process reduces re-works and project cost. Nonetheless, increasing the project's cost in this situation seems to be normal, as addressed in the literature [9, 29, 30].

VII. CONCLUSION

Considering software development reliability is an important factor that software teams and companies pay considerable attention to it. However, focusing on reliability results in more development time and cost. In Agile methodologies, due to their processes' weight-light nature, they paid less attention to extra tasks regarding SRE. Conducting a Case Study research, the authors tried to investigate considering SRE related practices while using Scrum. To do this, a customized version of Scrum was developed in which one new role, one new artifact, and one new activity were added to the Scrum framework. Employing this framework in a Case Study showed 3.9% extra human effort in a software project. However, the results showed a dramatic decrease in the number of failures that occurred as well as a number of re-works. This showed that it seems that it worth employing software reliability practices together with Scrum.

This framework needs to be applied to other projects to assess the results. Also, the agility degree of the customized Scrum would be evaluated quantitatively in the future study.

ACKNOWLEDGEMENTS

The author would like to express his gratitude to all the developers and managers who participated in this study. Without their support, this study could not have been completed.

REFERENCES

[1] VersionOne, "14 annual state of Agile report," 2020. Accessed: June 2020. [Online]. Available: https://stateofagile.com/#ufh-c-7027494-state-of-agile

[2] T. J. Gandomani and M. Z. Nafchi, "Agile transition and adoption human-related challenges and issues: A Grounded Theory approach," *Computers in Human Behavior,* vol. 62, pp. 257-266, 2016.

[3] T. J. Gandomani, H. Zulzalil, A. A. A. Ghani, A. M. Sultan, and M. Z. Nafchi, "Obstacles to moving to agile software development; at a glance," *Journal of Computer Science,* vol. 9, no. 5, pp. 620-625, 2013, doi: 10.3844/jcssp.2013.620.625.

[4] D. A. Lawong and O. Akanfe, "Overcoming team challenges in project management: The scrum framework," *Organizational Dynamics,* p. 101073, 2024.

[5] T. Javdani Gandomani and M. Ziaei Nafchi, "An empirically-developed framework for Agile transition and adoption: A Grounded Theory approach," *Journal of Systems and Software,* vol. 107, pp. 204-219, 2015, doi: 10.1016/j.jss.2015.06.006.

[6] B. A. A. Ammourah and S. A. Pitchay, "Challenges of Applying Scrum Model and Knowledge Management for Software Product Management," in *RITA 2018*: Springer, 2020, pp. 123-130.

[7] Q. Li and H. Pham, "A generalized software reliability growth model with consideration of the uncertainty of operating environments," *IEEE Access,* vol. 7, pp. 84253-84267, 2019.

[8] P. Roy, G. S. Mahapatra, and K. N. Dey, "Forecasting of software reliability using neighborhood fuzzy particle swarm optimization based novel neural network," *IEEE/CAA Journal of Automatica Sinica,* vol. 6, no. 6, pp. 1365-1383, 2019.

[9] U. Samal and A. Kumar, "Metrics and trends: a bibliometric approach to software reliability growth models," *Total Quality Management & Business Excellence,* vol. 35, no. 11-12, pp. 1274-1295, 2024.

[10] Y.-S. Huang, K.-C. Chiu, and W.-M. Chen, "A software reliability growth model for imperfect debugging," *Journal of Systems and Software,* vol. 188, p. 111267, 2022.

[11] J. D. Musa, *Software reliability engineering: more reliable software, faster and cheaper*. Tata McGraw-Hill Education, 2004.

[12] S. Khurshid, A. Shrivastava, and J. Iqbal, "Effort based software reliability model with fault reduction factor, change point and imperfect debugging," *International Journal of Information Technology,* pp. 1-10, 2019.

[13] G. Islam and T. Storer, "A case study of agile software development for safety-Critical systems projects," *Reliability Engineering & System Safety,* p. 106954, 2020.

[14] P. Sharma and A. L. Sangal, "Soft Computing Approaches to Investigate Software Fault Proneness in Agile Software Development Environment," in *Applications of Machine Learning*: Springer, 2020, pp. 217-233.

[15] M. F. M. Fudzee and J. Wadata, "A Mechanism to Support Agile Frameworks Enhancing Reliability Assessment for SCS Development: A Case Study of Medical Surgery Departments," in *Recent Advances on Soft Computing and Data Mining: Proceedings of the Fourth International Conference on Soft Computing and Data Mining (SCDM 2020), Melaka, Malaysia, January 22– 23, 2020*, 2019, vol. 978: Springer Nature, p. 66.

[16] S. Ali, Y. Hafeez, S. Hussain, and S. Yang, "Enhanced regression testing technique for agile software development and continuous integration strategies," *Software Quality Journal,* pp. 1-27, 2019.

[17] K. Beck, A. Cockburn, R. Jeffries, and J. Highsmith. "Agile manifesto." http://www.agilemanifesto.org (accessed Feb. 2020.

[18] S. Dwivedi and N. K. Goyal, "Effect of Fault Correction Delay on Software Reliability Modelling in Agile Software Development," in *International Conference on Reliability, Safety, and Hazard*, 2024: Springer, pp. 795-802.

[19] K. Schwaber, "Scrum development process," in *Business object design and implementation*: Springer, 1997, pp. 117-134.

[20] M. Cohn, *Succeeding with Agile: Software Development Using Scrum*. Boston, MA: Addison-Wesley Professional, 2009.

[21] M. Hron and N. Obwegeser, "Why and how is Scrum being adapted in practice: A systematic review," *Journal of Systems and Software,* vol. 183, p. 111110, 2022.

[22] ANSI/IEEE, "Standard Glossary of Software Engineering Terminology," 1991.

[23] M. R. Lyu, *Handbook of software reliability engineering*. IEEE computer society press CA, 1996.

[24] K. Sahu and R. Srivastava, "Revisiting software reliability," in *Data Management, Analytics and Innovation*: Springer, 2019, pp. 221-235.

[25] U. Samal and A. Kumar, "Empowering software reliability: Leveraging efficient fault detection and removal efficiency," *Quality Engineering,* vol. 37, no. 1, pp. 118-129, 2025.

[26] W. D. Van Driel, J. W. Bikker, M. Tijink, and A. Di Bucchianico, "Software reliability for agile testing," *Mathematics,* vol. 8, no. 5, p. 791, 2020.

[27] N. Barraza, "Software Reliability Modeling for Dynamic Development Environments," in *Recent Advancements in Software Reliability Assurance*: CRC Press, 2019, pp. 29-37.

[28] P. Jain, A. Sharma, and L. Ahuja, "A customized quality model for software quality assurance in agile environment," *International Journal of Information Technology and Web Engineering (IJITWE),* vol. 14, no. 3, pp. 64-77, 2019.

[29] S. Rawat, N. Goyal, and M. Ram, "Software reliability growth modeling for agile software development," *International Journal of Applied Mathematics and Computer Science,* vol. 27, no. 4, pp. 777-783, 2017.

[30] S. Munawar, R. Yousaf, and M. Hamid, "Extended Scrum Process Model Using Software Reliability Engineering Concerns," *Journal of Information Communication Technologies and Robotic Applications,* pp. 1-10, 2018.

[31] M. Esteki, T. Javdani Gandomani, and H. Khosravi Farsani, "A Risk Management Framework for Distributed Scrum using PRINCE2 Methodology," *Bulletin of Electrical Engineering and Informatics,* vol. 9, no. 3, 2020.

[32] M. Mousaei and T. Javdani Gandomani, "A new project risk management model based on scrum framework and Prince2 methodology," *International Journal of Advanced Computer Science and Applications,* vol. 9, no. 4, pp. 442-449, 2018, doi: 10.14569/IJACSA.2018.090461.

[33] M. Afshari and T. J. Gandomani, "A novel risk management model in the Scrum and extreme programming hybrid methodology," *International Journal of Electrical & Computer Engineering (2088-8708),* vol. 12, no. 3, pp. 2911-2921, 2022, doi: 10.11591/ijece.v12i3.