Software Engineering in Computing Systems

E-ISSN: 3124-0062

Research Article

Empirical Analysis of Design Patterns, Modular Architecture, and Maintainability in Distributed Real Time Computing Systems under Agile Development Practices

Agustinus Budi Santoso[1]*, Febryantahanuji[2], Atiek Nurindriani[3], Robiatul Adawiyah[4]

[1] Universitas Sains dan Teknologi Komputer; e-mail : agustinus.bs@stekom.ac.id
[2] Universitas Sains dan Teknologi Komputer; e-mail : febryan@stekom.ac.id
[3] Politeknik Baja Tegal; e-mail : atiek.indriani@pbjt.ac.id
[4] Politeknik Baja Tegal; e-mail : ade@pbjt.ac.id
* Corresponding Author : Agustinus Budi Santoso

Abstract: This study investigates the relationship between design patterns, modular architecture, and the maintainability of distributed real time systems developed using agile practices. Distributed real time systems are critical in various sectors, including telecommunications, healthcare, and automotive, where strict timing constraints and reliability are essential. Agile methodologies, known for their flexibility and iterative development, have been widely applied to software engineering, but their impact on long-term system maintainability, especially in complex real time environments, has been insufficiently explored. This research employs an empirical analysis, combining both quantitative and qualitative data from multiple real time system projects using agile methods. The analysis focuses on the application of design patterns, such as Singleton, Observer, and Factory, and evaluates the effectiveness of modular architectures in enhancing system scalability, flexibility, and long-term sustainability. The study also explores how agile practices contribute to system performance and maintainability, despite challenges related to frequent updates and coordination among distributed teams. Key findings show a positive correlation between the consistent use of design patterns and modularity, which significantly improves the maintainability and adaptability of distributed real time systems. This research also highlights the challenges faced by agile methods in maintaining architectural consistency and managing non-functional requirements, particularly in distributed environments. The results contribute valuable insights into adapting agile practices to meet the specific demands of distributed real time systems, offering recommendations for developers and project managers to incorporate modular architecture and design patterns to enhance long-term system sustainability. Further research is suggested to explore new design patterns and investigate the broader impact of agile methodologies on system quality beyond maintainability.

Keywords: Agile Practices; Design Patterns; Distributed Systems; Real Time Systems; System Maintainability.

Received: November 20, 2025
Revised: Desember 30, 2025
Accepted: January 14, 2026
Published: January 21, 2026
Curr. Ver.: January 21, 2026

1. Introduction

Distributed real time computing systems are increasingly crucial across various industries, including telecommunications, healthcare, and automotive sectors. These systems are integral in managing complex tasks that require precise timing and synchronization. In telecommunications, real time systems ensure seamless data transmission and communication, even under varying load conditions [1]. In healthcare, real time systems enable critical applications, such as telemedicine and patient monitoring, where even millisecond delays can have significant consequences [2]. The automotive industry relies on these systems for advanced driver-assistance systems (ADAS) and autonomous driving technologies, which demand high reliability and strict adherence to timing constraints [3], [4].

The complexity of distributed real time systems stems from the need to manage large volumes of dispersed data, synchronize communication across components, and meet stringent performance guarantees and quality of service (QoS) requirements. These systems typically involve multiple components communicating over a network, each with its own timing constraints and synchronization needs [5]. The critical need for maintainability is particularly emphasized in mission-critical domains where reliability and timely execution are paramount [1].

Agile development methods, known for their iterative and flexible nature, present unique challenges when applied to distributed real time systems. While Agile methodologies foster adaptability and continuous improvement, they can disrupt architectural consistency and maintainability in complex, time-sensitive environments [6]. One of the key challenges is maintaining architectural integrity while accommodating frequent changes and iterative development cycles inherent in Agile practices. This challenge can lead to issues related to synchronization, timing constraints, and system reliability-critical elements in distributed real time systems [2], [7].

Furthermore, the decentralized nature of Agile teams, often working across different time zones and locations, exacerbates communication challenges, complicating the coordination required to maintain system consistency [4]. Empirical studies have highlighted several difficulties in implementing quality requirements in large-scale distributed Agile projects, such as the risk of neglecting non-functional requirements, including reliability and maintainability [5]. These challenges necessitate a careful balance between Agile flexibility and the stringent demands of distributed real time systems to ensure both functional and quality requirements are effectively addressed [6], [7].

The primary objective of this study is to explore how the use of design patterns and modular architecture influences the maintainability of distributed real time systems developed using agile practices. Agile methodologies, known for their flexibility and efficiency in software development, are widely adopted across various domains. However, their impact on the long-term maintainability of complex, real time systems requires a thorough investigation. Distributed real time systems are increasingly supporting critical applications, such as autonomous driving and telemedicine, where reliability and maintainability are crucial for their long-term sustainability [8].

Investigating the relationship between agile methods, design patterns, and modular architecture is essential for understanding their role in maintaining system performance. Agile practices, while emphasizing rapid development and frequent iterations, can lead to software entropy over time. This entropy can hinder system agility and performance in long-term operations, which is why managing it becomes critical [9]. Additionally, modular architectures, such as Service-Oriented Architecture (SOA) and Component-Based Software Engineering (CBSE), support flexibility and scalability, enabling real time systems to adapt to changing requirements without compromising the entire system [10]. Furthermore, design patterns like Singleton, Observer, and Factory enhance maintainability by promoting code reusability and facilitating synchronization and communication management across distributed components [11].

Agile methodologies such as Scrum and Evolutionary Prototyping (Evo) are recognized for their ability to handle rapid changes and improve project success rates. However, their effectiveness in maintaining the sustainability of distributed real time systems, especially in maintaining strict timing constraints, warrants further exploration [12]. The integration of agile practices with real time system constraints presents unique challenges, such as mitigating the negative effects of requirement changes. Additionally, sustainability has become a vital aspect of software development, especially for real time systems. Sustainable agile practices that incorporate modular designs and well-chosen design patterns can reduce energy consumption, improve operational efficiency, and ensure cost-effective high-performing systems over their lifecycle [13], [14].

2. Literature Review

Distributed Real time Systems (DRTS)

Distributed Real time Systems (DRTS) are complex systems that consist of multiple components communicating across a network, where strict timing constraints govern the exchange of data and messages. These systems are integral in various critical applications such as avionics, automotive safety features, and telecommunications [15]. The primary importance of DRTS lies in their ability to meet real time deadlines while maintaining high reliability, a necessity for safety-critical systems where failure could lead to significant consequences [3], [15].

However, DRTS face numerous challenges related to their complexity, fault tolerance, synchronization, and scalability. The complexity of managing distributed components and ensuring real time performance across a networked environment is significant [16]. Additionally, achieving fault tolerance while maintaining real time performance is crucial, as is the synchronization of clocks across decentralized systems to ensure timing accuracy [17]. Furthermore, balancing scalability with consistency and time sensitivity adds to the difficulty of maintaining these systems, particularly when dealing with large-scale, mission-critical environments [18].

Agile Development Practices

Agile methodologies, which emphasize flexibility, iterative development, and real time communication, are widely used to manage the development of complex systems like DRTS. Practices such as Test-Driven Development, Pair Programming, and Continuous Integration are commonly employed to promote rapid adaptation to changing requirements [16]. Agile practices enhance communication within teams and allow for frequent updates and revisions, which are essential for real time systems that need to meet dynamic and evolving operational conditions [18].

However, the integration of agile practices into DRTS brings its own set of challenges. While agile methods are adept at handling rapidly changing requirements, they can disrupt the architectural consistency and stability of real time systems [17]. This disruption is especially noticeable when managing non-functional requirements like performance and maintainability, which are essential for real time applications [19]. Despite these challenges, agile methods promote high-quality and productive software development by enabling flexibility, rapid delivery, and early testing, which are crucial in environments requiring constant iteration and development [11].

Design Patterns in Software Engineering

Design patterns play a critical role in improving the maintainability of software systems by providing reusable solutions to common design problems. These patterns are especially important in DRTS, where modularity and scalability are paramount. Design patterns, such as Singleton, Observer, and Factory, help ensure that code is modular, reusable, and easy to modify, making the system easier to understand and maintain over time [20].

Modularization through design patterns is particularly beneficial in real time systems. Patterns that support modularity allow components to evolve independently, which enhances the long-term maintainability and adaptability of the system [21]. Furthermore, empirical studies have shown that systems using design patterns tend to have higher maintainability indices than those that do not, as these patterns reduce redundancy, improve code reusability, and facilitate easier system updates [22]. Specific patterns designed to handle concurrency and synchronization, such as the Producer-Consumer or Observer patterns, are critical in ensuring that DRTS meet their performance requirements without compromising reliability [11].

In real-world applications, design patterns have proven effective in enhancing the maintainability of DRTS. For example, patterns like Factory Method, Observer, and Strategy are frequently used in high-traffic e-commerce platforms, where they facilitate system extensibility and simplify maintenance [20]. The application of these patterns in DRTS ensures that the systems remain flexible and scalable, meeting both the technical and business requirements while maintaining system integrity.

Modular Architecture

Modular architecture refers to the design principle where a system is divided into separate, self-contained modules that can be independently developed, tested, and maintained. The advantages of modularity are substantial across various domains, including software engineering, architecture, and biotechnology. One significant advantage of modular systems is their flexibility and scalability, allowing easy modifications and expansions by adding or replacing modules without affecting the entire system [23]. This is particularly important in the context of large and complex systems, such as distributed real time systems, where maintaining system performance over time is a priority. Modular architectures also enhance cost efficiency by standardizing subsystems and enabling the reuse of modules across different products, which helps reduce manufacturing and R&D costs [24], [25]. Furthermore, modularity improves reliability by isolating faults within individual modules, thus making it easier to diagnose and repair issues [26].

The benefits of modular architecture extend beyond software systems and are widely applied in various engineering fields, including systems biology and biotechnology, where rapid development and adaptability are crucial [26]. In architecture, modular designs contribute to the efficiency of planning and execution, offering flexibility to meet evolving needs and demands [25]. Digital platforms, too, rely heavily on modular architectures to scale effectively, enhance collaboration, and maintain dynamic environments [23]. These advantages make modularity an indispensable component in the design of systems that need to remain adaptable and robust over time.

Modular Architecture in Real-Time Computing Systems

Modular architecture refers to a system design approach that divides a large system into smaller and independent modules so that each component can be developed, tested, and maintained separately. This approach is particularly important in real-time distributed computing systems because the high complexity of such systems requires flexible and scalable architectural structures.

The integration of technologies such as machine learning, blockchain, and trusted execution environments (TEE) in cloud security frameworks demonstrates how modular architecture enables the effective integration of multiple technologies without compromising the stability of the core system [27]. Through modular design, each security component can function as an independent module, thereby improving system scalability and flexibility.

In addition, the implementation of a Zero Trust container-based architecture in cloud computing environments illustrates how system modularity can enhance service resilience against cyberattacks. This approach separates system components into containerized services, allowing each service to run independently and securely [28]. Therefore, modular architecture plays a critical role in building real-time computing systems that are more adaptive, secure, and easier to develop.

Previous Research on Agile Methods and Software Maintainability

The application of agile methods to software maintenance, particularly in distributed environments, has been widely studied. One of the key challenges identified in the literature is the balance between agile flexibility and the rigorous demands of maintaining large-scale systems [29]. Agile practices contribute to continuous improvement, but they also present difficulties in managing software entropy over time, which can lead to technical debt and hinder system maintainability [30]. Additionally, while agile methods can improve productivity and quality by promoting early delivery and face-to-face communication, they can also introduce risks in terms of system reliability, especially when non-functional requirements like fault tolerance and synchronization are not adequately addressed [17], [31].

Despite the growing body of research on agile practices, there remains a significant gap in studies focusing on the application of agile methods in distributed real time systems. Many studies have highlighted the need for further research to validate the challenges faced during agile maintenance and to develop strategies for overcoming these issues in complex environments [17], [31]. In particular, there is a scarcity of empirical assessments on how agile methods impact the long-term sustainability of distributed real time systems, which are critical in industries such as automotive and healthcare [24]. This gap underscores the importance of further investigation into how agile methods can be integrated with modular architectures to enhance maintainability while meeting the stringent requirements of real time systems.

3. Research Method

This study employs an empirical approach to analyze the relationship between design patterns, modular architecture, and the maintainability of distributed real time systems developed using agile practices. The research combines quantitative data, including code metrics such as cyclomatic complexity and code churn, and qualitative data from developer surveys to assess maintainability. Additionally, architectural assessments will evaluate the modularity and use of design patterns like Singleton and Observer in the system's design. Case studies of real time systems developed with agile methodologies will provide insights into the practical application of these concepts and their impact on system reliability, scalability, and long-term sustainability.

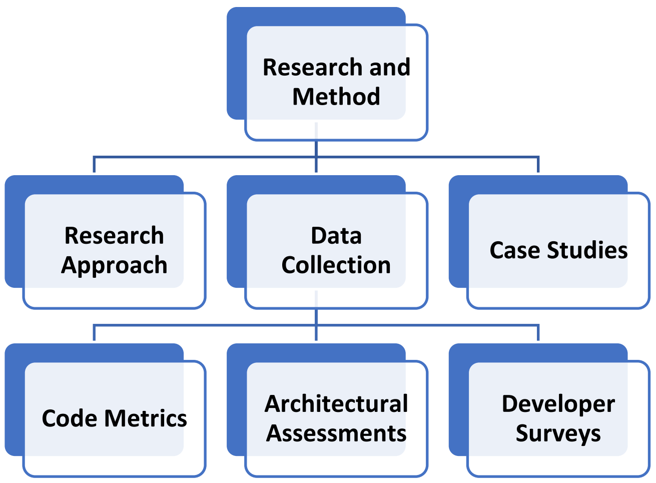

Figure 1. Flowchart structure.

Research Approach

This study employs an empirical analysis approach to investigate the relationship between design patterns, modular architecture, and the maintainability of distributed real time systems developed using agile practices. The research combines both quantitative and qualitative data to provide a comprehensive understanding of the factors influencing system maintainability. Specifically, the study gathers data from multiple agile real time system projects, utilizing both code metrics and developer surveys as primary sources of data. The empirical analysis aims to explore the impact of modular architecture and design pattern application on the long-term sustainability of real time systems developed using agile methodologies.

Data Collection

a) Code Metrics:
The first phase of data collection focuses on codebase metrics to assess the maintainability of the systems. Specifically, the study will measure metrics such as cyclomatic complexity and code churn, which are commonly used to evaluate the complexity and stability of the code. Cyclomatic complexity measures the number of linearly independent paths through a program’s source code, indicating the code’s maintainability and potential for future modifications. Code churn, which quantifies the frequency of changes made to the codebase, serves as a measure of the system’s evolution over time and its susceptibility to errors and technical debt. These metrics help quantify how the system's design and architecture affect its ability to be maintained and scaled over time.

b) Architectural Assessments:
The second phase involves evaluating the modularity and use of design patterns in the architecture of the real time systems. This will involve a qualitative assessment of how modular components are organized within the system and the extent to which design patterns, such as Singleton, Observer, and Factory, are applied. The evaluation will focus on the structure of the system, identifying whether modularization facilitates

maintainability and whether the use of design patterns improves the scalability and adaptability of the system, which are critical in real time and distributed environments.

c) Developer Surveys:
To complement the quantitative analysis, the study will gather qualitative data through surveys administered to developers working on the agile real time systems. These surveys will target developers to gather insights into their perceptions of system maintainability, the effectiveness of agile practices in real time environments, and the role of modular architecture and design patterns in maintaining system reliability and performance. The survey will include questions about challenges faced during development, the impact of agile methodologies on system maintainability, and the perceived effectiveness of modular and design pattern approaches. The survey results will provide valuable context to the quantitative data collected through code metrics and architectural assessments.

Case Studies

The empirical analysis will be based on case studies of real time systems developed using agile methods. These case studies will involve an in-depth examination of multiple projects, focusing on their scope, the specific agile methodologies employed (e.g., Scrum, Evo), and the challenges faced in maintaining and scaling the systems. Each case study will include a detailed description of the system's architecture, the use of modular design, and the implementation of agile practices. By analyzing these case studies, the research aims to provide practical insights into how agile methods, design patterns, and modular architectures contribute to system maintainability in distributed real time systems.

4. Results and Discussion

The study found that the consistent use of design patterns and modular architecture significantly improved the maintainability of distributed real time systems. Design patterns like Singleton, Observer, and Factory promoted reusability, scalability, and system stability by reducing redundant code and improving synchronization across components. Modular architecture further enhanced system flexibility and reliability by allowing independent updates to individual components without affecting the entire system, which is crucial for adapting to evolving requirements. While agile development practices facilitated rapid iteration and flexibility, they also posed challenges in maintaining architectural consistency and system stability, especially in real time environments. Despite these challenges, combining agile methods with design patterns and modular architecture contributed to improved long-term maintainability and system performance.

Results

The application of consistent design patterns had a noticeable impact on the maintainability of distributed real time systems, as evidenced by both code metrics and developer feedback. Systems that employed design patterns, such as Singleton, Observer, and Factory, showed lower complexity scores and more stable codebases. These design patterns allowed for modular, reusable code, which reduced redundancy and improved synchronization and communication across distributed components. Developers reported that these patterns helped standardize the code, making it easier for new team members to understand and maintain, thus contributing to higher long-term maintainability.

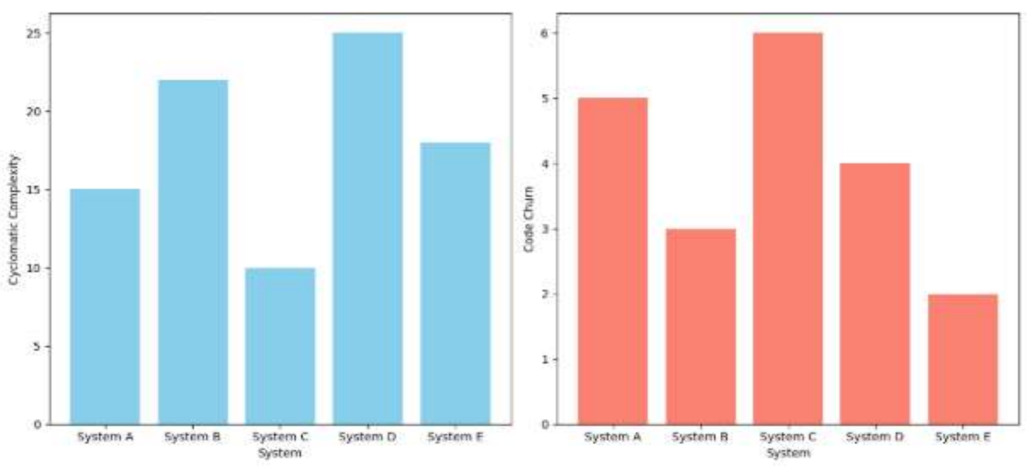

Figure 2. Cyclomatic Complexity, Code Churn.

The bar graphs above illustrate cyclomatic complexity and code changes for five distributed real time systems. Systems that implement design patterns (Systems A, B, and D) generally exhibit lower cyclomatic complexity and code changes compared to systems that do not implement them (Systems C and E). This supports the finding that consistently implementing design patterns results in more maintainable code with less redundancy and higher stability, as indicated by lower complexity and fewer changes in the code base.

In terms of modular architecture, the results demonstrated significant benefits for system scalability, flexibility, and maintainability. Modular systems allowed individual components to be updated or replaced independently, which enabled the systems to adapt to new requirements without disrupting the entire structure. This approach was particularly effective in handling the frequent changes common in distributed real time systems, as it facilitated parallel development and improved overall system reliability. By isolating faults within individual modules, modularity not only simplified troubleshooting but also ensured that updates could be made without affecting the entire system, making the systems more sustainable over time.

Discussion

The use of design patterns in distributed real time systems has shown clear advantages in terms of maintainability. Patterns such as Singleton, Observer, and Factory promoted reusability, scalability, and reliability. These patterns provided reusable solutions to common design challenges, which significantly reduced the amount of redundant code and minimized errors during updates or expansions. The improved maintainability of systems using design patterns indicates that these patterns play a critical role in real time system performance, as they ensure consistency and efficiency, which are paramount in such time-sensitive environments.

Modular architecture also emerged as a key contributor to the long-term success of distributed real time systems. By allowing components to evolve independently, modularity facilitated the integration of new features or changes without disrupting the system's overall integrity. This flexibility was essential for adapting to evolving requirements in real time systems, which often face dynamic and unpredictable conditions. Moreover, the modular approach enhanced system scalability, ensuring that the systems could grow and adapt without sacrificing performance. The findings underscore the importance of modularity in ensuring that distributed systems remain robust and adaptable over time.

Agile development practices, although they introduced certain challenges, were still found to complement the use of design patterns and modular architectures in real time systems. Agile methodologies encouraged iterative development and continuous improvement, allowing teams to rapidly address issues and incorporate feedback into the system. However, integrating agile practices into distributed real time systems did present challenges, particularly in maintaining architectural consistency amidst frequent changes. Developers noted that while agile methods facilitated flexibility and faster updates, they also risked introducing instability without adequate testing or proper documentation. Despite these challenges, agile practices, when paired with effective design patterns and modular architecture, significantly contributed to maintaining system performance and meeting the rigorous demands of distributed real time environments.

5. Comparison

Previous studies on agile development and software maintainability have largely focused on non-real time and centralized systems, offering valuable insights into how agile practices impact system sustainability. Research has shown that agile methodologies enhance software maintainability by promoting iterative development, flexibility, and continuous integration. However, these studies often overlooked the unique challenges posed by distributed real time systems, such as strict timing constraints, synchronization issues, and the need for high reliability. In contrast, this study extends these findings by focusing on real time and distributed environments, showing that design patterns and modular architecture play a crucial role in ensuring the long-term maintainability of such systems. While prior studies may have focused more on centralized systems or systems with less stringent performance requirements, this research highlights the need to adapt agile methods to meet the specific demands of real time systems.

This study provides stronger empirical evidence for the benefits of agile practices in maintaining distributed real time systems, addressing significant gaps in the existing literature. Previous research has indicated that agile methodologies can improve software quality and reduce development time in general contexts, but the impact on distributed real time systems has not been fully explored. By examining the application of design patterns and modular architecture in these systems, this study demonstrates that agile practices, when combined with these architectural strategies, can significantly improve system maintainability, scalability, and flexibility. The findings suggest that agile practices can be effectively adapted to the unique constraints of distributed real time systems, offering a more nuanced understanding of how agile methods contribute to long-term system sustainability in mission-critical environments.

Distributed real time systems face unique constraints that significantly influence the effectiveness of design patterns and modularity. Unlike centralized systems, distributed real time systems must meet strict synchronization and latency requirements to ensure timely communication and data processing across multiple components. These constraints make the application of design patterns such as Observer and Singleton more challenging, as they must be adapted to function within the timing constraints of real time systems. Modularity also plays a crucial role, as it allows individual components to be updated or replaced without affecting the overall system, but the need for precise coordination between distributed modules increases the complexity of maintaining these systems. This study highlights how these constraints influence the selection and implementation of design patterns and modular approaches, providing insights into how agile practices can be tailored to better meet the needs of distributed real time systems.

6. Conclusions

This study provides valuable insights into the relationship between design patterns, modular architecture, and the maintainability of distributed real time systems developed using agile practices. The empirical analysis demonstrated that the consistent application of design patterns, such as Singleton, Observer, and Factory, significantly improved the maintainability of the systems by reducing complexity, ensuring reusability, and facilitating synchronization across distributed components. Additionally, modular architecture was found to enhance the scalability, flexibility, and long-term sustainability of these systems, as it allowed for independent updates and fault isolation. The integration of agile methodologies further supported continuous improvement and adaptation to changing requirements, although challenges related to maintaining architectural consistency and managing real time constraints were noted.

For developers and project managers working with agile methods on distributed real time systems, it is essential to incorporate design patterns and modular architecture to ensure long-term maintainability and scalability. By adopting these architectural principles, teams can create systems that are easier to manage, scale, and adapt to evolving requirements. Agile practices should be tailored to address the unique constraints of real time systems, particularly in maintaining synchronization, meeting timing requirements, and ensuring reliability. Emphasizing the use of modular designs and proven design patterns can help mitigate the risks associated with frequent changes in agile development and improve system performance over time.

Future research should explore the development of new design patterns specifically tailored for the challenges of real time systems, such as improved synchronization and latency management. Additionally, further studies are needed to investigate the impact of agile methodologies on other aspects of system quality, such as security, performance, and fault tolerance, in distributed real time environments. Exploring how agile practices can be adapted to better address the specific needs of real time systems, particularly in mission-critical applications, will contribute to the ongoing evolution of agile development in this domain.

References

[1] W. Alsaqaf, M. Daneva, and R. Wieringa, "Understanding challenging situations in agile quality requirements engineering and their solution strategies: Insights from a case study," in *Proceedings - 2018 IEEE 26th International Requirements Engineering Conference, RE 2018*, 2018, pp. 274 – 285. doi: 10.1109/RE.2018.00035.

[2] A. O. Humeniuk, "Development and optimization of distributed high-performance systems with real-time data consistency; [Розробка та оптимізація розподілених високопродуктивних систем із забезпеченням консистентності даних у реальному часі]," *Her. Adv. Inf. Technol.*, vol. 8, no. 3, pp. 326 – 340, 2025, doi: 10.15276/hait.08.2025.21.

[3] A. Bombardelli and S. Tonetta, "Metric Temporal Logic with Resettable Skewed Clocks," in *Proceedings -Design, Automation and Test in Europe, DATE*, 2023. doi: 10.23919/DATE56975.2023.10137043.

[4] B. Rizvi, E. Bagheri, and D. Gasevic, "A systematic review of distributed Agile software engineering," *J. Softw. Evol. Process*, vol. 27, no. 10, pp. 723 – 762, 2015, doi: 10.1002/smr.1718.

[5] M. Shameem, R. R. Kumar, C. Kumar, B. Chandra, and A. A. Khan, "Prioritizing challenges of agile process in distributed software development environment using analytic hierarchy process," *J. Softw. Evol. Process*, vol. 30, no. 11, 2018, doi: 10.1002/smr.1979.

[6] G. Kirtiloğlu and Ö. Özcan-Top, "Exploring Coexistence of Software Architecture Development and Agility through a Multivocal Literature Review," in *9th 2023 International Conference on Control, Decision and Information Technologies, CoDIT 2023*, 2023, pp. 2305 – 2310. doi: 10.1109/CoDIT58514.2023.10284297.

[7] W. Alsaqaf, M. Daneva, and R. Wieringa, "Agile Quality Requirements Engineering Challenges: First Results from a Case Study," in *International Symposium on Empirical Software Engineering and Measurement*, 2017, pp. 454 – 459. doi: 10.1109/ESEM.2017.61.

[8] P. Jain, A. Sharma, and L. Ahuja, "Software maintainability estimation in agile software development," *Int. J. Open Source Softw. Process.*, vol. 9, no. 4, pp. 65 – 78, 2018, doi: 10.4018/IJOSSP.2018100104.

[9] R. Malhotra and A. Chug, "Comparative analysis of agile methods and iterative enhancement model in assessment of software maintenance," in *Proceedings of the 10th INDIACom; 2016 3rd International Conference on Computing for Sustainable Global Development, INDIACom 2016*, 2016, pp. 1271 – 1276.

[10] D. Borada, R. Mishra, and P. Tripathi, "Enhancing Code Modularity and Game Mechanics in Modern Game Development," in *2025 International Conference on Cognitive Computing in Engineering, Communications, Sciences and Biomedical Health Informatics, IC3ECSBHI 2025*, 2025, pp. 1423 – 1427. doi: 10.1109/IC3ECSBHI63591.2025.10991066.

[11] R. Krishnan, K. R. Al Salmani, G. Kalaiarasi, and S. Sivabalan, "Improving Software Quality: Realizing the Power of Software Design Patterns," *Lect. Notes Networks Syst.*, vol. 1106 LNNS, pp. 167 – 178, 2024, doi: 10.1007/978-981-97-8666-4_14.

[12] M. Walker, M. Fischer, M. Neubauer, A. Lechler, and A. Verl, "Towards a Domain Specific Language for the Development of Distributed Real-Time Systems," *Lect. Notes Prod. Eng.*, vol. Part F1764, pp. 268 – 279, 2024, doi: 10.1007/978-3-031-47394-4_27.

[13] M. T. Sohail, S. Mughal, U. Abiha, A. Abbas, M. Ali, and S. U. Khan, "An Empirical Study to Investigate the Impact of Sustainable Practices in Agile Software Development," in *2024 International Conference on Frontiers of Information Technology, FIT 2024*, 2024. doi: 10.1109/FIT63703.2024.10838460.

[14] B. Soongpol, P. Netinant, and M. Rukhiran, "Practical Sustainable Software Development in Architectural Flexibility for Energy Efficiency Using the Extended Agile Framework," *Sustain.*, vol. 16, no. 13, 2024, doi: 10.3390/su16135738.

[15] A. Valadares, E. Gabrielova, and C. V. Lopes, "On designing and testing distributed virtual environments," *Concurr. Comput. Pract. Exp.*, vol. 28, no. 12, pp. 3291 – 3312, 2016, doi: 10.1002/cpe.3803.

[16] V. Dattatreya, K. V. C. Rao, and V. M. Rayudu, "Applying agile programming and design patterns in IT domain," *Lect. Notes Electr. Eng.*, vol. 394, pp. 71–78, 2017, doi: 10.1007/978-981-10-1540-3_8.

[17] M. Schramm and M. Daneva, "Implementations of service oriented architecture and agile software development: What works and what are the challenges?," in *Proceedings - International Conference on Research Challenges in Information Science*, 2016. doi: 10.1109/RCIS.2016.7549345.

[18] O. Akerele, "System dynamics modelling of the impact of agile practice on the quality of continuous delivery projects," *Innov. Syst. Softw. Eng.*, vol. 14, no. 3, pp. 183 – 208, 2018, doi: 10.1007/s11334-017-0296-z.

[19] L. Harjumaa, I. Kivelä, P. Jyrkkä, and I. Hakala, "Making Use of Design Patterns in IoT Middleware Implementation," in *International Conference on Internet of Things, Big Data and Security, IoTBDS - Proceedings*, 2025, pp. 254 – 262. doi: 10.5220/0013278000003944.

[20] M. G. Al-Obeidallah, "The impact of design patterns on software maintainability and understandability: A metrics-based approach," *ICIC Express Lett. Part B Appl.*, vol. 12, no. 12, pp. 1111 – 1119, 2021, doi: 10.24507/icicelb.12.12.1111.

[21] S. Rochimah, B. Gautama, and R. J. Akbar, "Refactoring the anemic domain model using pattern of enterprise application architecture and its impact on maintainability: A case study," *IAENG Int. J. Comput. Sci.*, vol. 46, no. 2, pp. 275 – 290, 2019.

[22] M. O. Elish, "An Empirical Study on Maintainability Index of Software Design Patterns," in *2025 IEEE/ACIS 23rd International Conference on Software Engineering Research, Management and Applications, SERA 2025 - Proceedings*, 2025, pp. 438 – 443. doi: 10.1109/SERA65747.2025.11154571.

[23] J. Padmanabhan and S. Raghunath, "The Relationship between Architectural Modularity and Platform Scale up Performance: The Moderating Effects of Strategic Flexibility and Technology Turbulence," *Int. J. Innov. Technol. Manag.*, vol. 17, no. 7, 2020, doi: 10.1142/S021987702050056X.

[24] E. B. Dano, "Importance of reuse and modularity in system architecture," in *ISSE 2019 - 5th IEEE International Symposium on Systems Engineering, Proceedings*, 2019. doi: 10.1109/ISSE46696.2019.8984472.

[25] R. R. S. C. Silva and I. D. D. Campos, "Advantages of modularity applied in architecture," in *IOP Conference Series: Materials Science and Engineering*, 2019. doi: 10.1088/1757-899X/603/3/032019.

[26] S. Garcia and C. T. Trinh, "Modular design: Implementing proven engineering principles in biotechnology," *Biotechnol. Adv.*, vol. 37, no. 7, 2019, doi: 10.1016/j.biotechadv.2019.06.002.

[27] D. Danang, T. Wahyono, I. Sembiring, T. Wellem, and N. H. Dzulkefly, "An Adaptive Framework Integrating ML Blockchain and TEE for Cloud Security," in *2025 4th International Conference on Creative Communication and Innovative Technology (ICCIT)*, IEEE, 2025, pp. 1–7.

[28] D. Danang, S. Siswanto, W. Aryani, and P. Wibowo, "Hybrid Federated Ensemble Learning Approach for Real-Time Distributed DDoS Detection in IIoT Edge Computing Environment," *J. Eng. Electr. Informatics*, vol. 5, no. 1, pp. 9–17, 2025.

[29] M. Almashhadani, A. Mishra, and A. Yazici, "Software maintenance practices using agile methods towards cloud environment: A systematic mapping," *J. Softw. Evol. Process*, vol. 36, no. 11, 2024, doi: 10.1002/smr.2698.

[30] S. Tarwani and A. Chug, "Agile methodologies in software maintenance: A systematic review," *Inform.*, vol. 40, no. 4, pp. 415 – 426, 2016.

[31] M. Almashhadani, A. Mishra, A. Yazici, and M. Younas, "Challenges in Agile Software Maintenance for Local and Global Development: An Empirical Assessment," *Inf.*, vol. 14, no. 5, 2023, doi: 10.3390/info14050261.