Agile Software Development Life Cycle Model for Internet of Things

Goran Filinić
University of Zagreb
Faculty of Organization and Informatics
Pavlinska 2, 42000 Varaždin, Hrvatska
gfilinic22@student.foi.hr

Ivan Magdalenić
University of Zagreb
Faculty of Organization and Informatics
Pavlinska 2, 42000 Varaždin, Hrvatska
ivan.magdalenić@foi.unizg.hr

Abstract. *In the fast developing landscape of IoT technology, the development of effective software solutions faces unique challenges due to the integration of diverse hardware components and the need for continuous adaptation to evolving requirements. Traditional software development lifecycle (SDLC) models often struggle to accommodate the iterative nature and rapid pace of IoT projects. This paper proposes an Agile SDLC model tailored specifically for IoT systems, integrating principles from Agile methodologies, Rapid Prototyping, and Model-Driven Development. The model emphasizes continuous iteration, comprehensive documentation, and adaptive design, aiming to enhance development efficiency, mitigate risks, and accelerate time-to-market for IoT solutions. By addressing the complexities of IoT development through structured yet flexible phases—ranging from requirement gathering and design to deployment and maintenance—the proposed model seeks to provide a robust framework capable of meeting the dynamic demands of IoT environments. This model aims to contribute to the advancement and adoption of effective SDLC practices in IoT development.*

Keywords. Agile SDLC, IoT systems, Rapid Prototyping, Model-Driven Development, software development lifecycle, embedded systems, embedded programming

1 Introduction

The accelerated advancement of the Internet of Things (IoT) technology has brought about significant transformations in various industries, enhancing efficiency, connectivity, and automation. Despite these advancements, there remains a notable gap in structured guidelines for the development of IoT projects (Fahmideh et al., 2022). This absence of a comprehensive framework poses challenges for developers and organizations aiming to implement IoT solutions efficiently and effectively.

Because IoT projects inherently involve unique challenges with the perspective of integrating physical devices with software components, they introduce a layer of complexity not present in traditional software development. Each IoT project is unique due to the specific hardware requirements and the need for seamless interaction between devices and software Singh et al. (2014), Guerrero-Ulloa et al. (2019), Dado et al. (2015), Madakam et al. (2015). This added complexity necessitates a more flexible and adaptive approach to the SDLC, especially when considering the integration of embedded systems which play a critical role in IoT infrastructure.

To bridge this gap, an enhancement of the classic SDLC model is proposed by integrating present development practices such as Agile methodologies Srivastava et al. (2017), Schwaber and Sutherland (2011), Salo and Abrahamsson (2008), Rapid Prototyping Lantz (1986), Fern and Donaldson (2003), and Model-Driven Development Cabot (2012). These modern approaches not only streamline the development process but also foster greater flexibility, iterative improvements, and a higher degree of customization, which are critical in IoT project management.

The goal of this research is to synthesize these insights and propose a refined SDLC model tailored specifically for IoT projects. By leveraging modern practices, this model aims to optimize the development process, ensuring that IoT solutions are not only innovative but also robust, scalable, and aligned with the fast-paced evolution of technology. Through this endeavor, we hope to contribute to the establishment of a more structured and efficient framework for IoT development, ultimately driving further advancements in this transformative field.

The rest of the paper is structured as follows. Section 2 presents a review of relevant related work in IoT development methodologies. Section 3 introduces the proposed Agile SDLC model for IoT, highlighting its fundamental components and principles. Section 4 provides a detailed discussion on the potential benefits and challenges of the model. Finally, Section 5 concludes the paper by summarizing key findings, discussing implications for future research, and outlining potential avenues for further exploration.

2 Related Work

The current landscape of IoT systems development has been extensively explored by Fahmideh et al. (2022). Their mixed-methods study identified a significant knowledge gap in the development lifecycle for IoT projects. By synthesizing existing literature, they derived and validated a conceptual framework outlining 27 crucial tasks for integrating IoT systems into development processes Fahmideh et al. (2022). This framework, validated through a global survey of 127 IoT practitioners, forms the cornerstone of this paper's proposed model, which aims to establish effective processes documented in the literature.

Gupta and Gayathri (2022) closely examined the Software Development Lifecycle (SDLC), as well as emphasizing the critical role of early testing Jat and Sharma (2017). Their studies highlighted that early initiation of testing prevents bugs, reduces costs, and accurately determines requirements, ultimately improving software reliability and performance (Gupta and Gayathri (2022), Jat and Sharma (2017)). This underscores the importance of testing in the SDLC and its role in enhancing software development outcomes.

In safety-critical domains like medical device software development, Klespitz et al. (2015) , Dayo et al. (2022) emphasized the necessity of robust lifecycle management systems. Their case study highlighted the need for stringent documentation and objective measures to ensure effective and mature product development. This is particularly crucial for meeting safety standards and accelerating system implementation in medical software development.

Agile methodologies applied to IoT systems (IoTS) development were reviewed by Guerrero-Ulloa et al. (2023). Analyzing 60 documents from an initial 4,303, they found that 42.1% of IoTS developments used Scrum exclusively, with others combining it with methodologies like XP, Kanban, and Rapid Prototyping. The study highlighted the prominence of model-based approaches such as Model-Driven Engineering (MDE) and Model-Driven Development (MDD) in addressing IoTS technology heterogeneity Cabot (2012). However, gaps in requirement elicitation, maintenance, and withdrawal phases were noted, emphasizing the need for comprehensive methodologies.

Alfawair (2022) proposed the IoTSDLC model, which adapts the traditional SDLC to address IoT-specific challenges like heterogeneity and complex integration requirements. Despite its emphasis on detailed documentation and system specification, the model's linear approach lacks flexibility for agile development. Another example can be looked at Yang and Bi (2014). This rigidity and absence of agile integration guidelines limit its applicability in dynamic IoT environments where rapid prototyping and continuous iteration are essential.

These studies underscore the diverse challenges and methodologies in IoT systems development. They highlight the need for structured SDLC tailored to IoT projects, the critical role of early testing, robust management systems, and the application of agile methodologies. Integrating these insights into a unified SDLC model is imperative to meet the dynamic demands of current IoT environments, ensuring comprehensive support for developers and stakeholders. The proposed unified agile SDLC model integrates these insights to meet the IoT environments, ensuring comprehensive support for developers and stakeholders, and enhancing the overall effectiveness and adaptability of IoT system development.

3 Agile SDLC Model for IoT

Figure 1 illustrates the proposed Agile SDLC model tailored specifically for IoT projects. Unlike traditional Agile SDLC, which typically focuses on rapid iterative cycles for software delivery Yang and Bi (2014), this model is designed to address the extended lifecycle and complex integration challenges characteristic of IoT systems. This include device heterogeneity and system reliability Alberternst et al. (2021), Lakhan et al. (2022). Primary divergences from traditional Agile SDLC include:

1. **Long-term lifecycle management**: The model considers the extended lifecycle typical of IoT projects, which require careful planning not only for development but also for ongoing maintenance and adaptability. It incorporates continuous optimization and grooming to minimize future interventions, ensuring sustained performance over time.

2. **Grooming phase**: A dedicated phase called grooming is included to refine and optimize the system's architecture and components, ensuring robustness and scalability. This phase addresses the specific needs of IoT projects, where ongoing system refinement is critical.

3. **Maintenance and Support**: This phase introduces mechanisms for both small updates and major changes, providing structured support for the long-term maintenance of IoT systems:

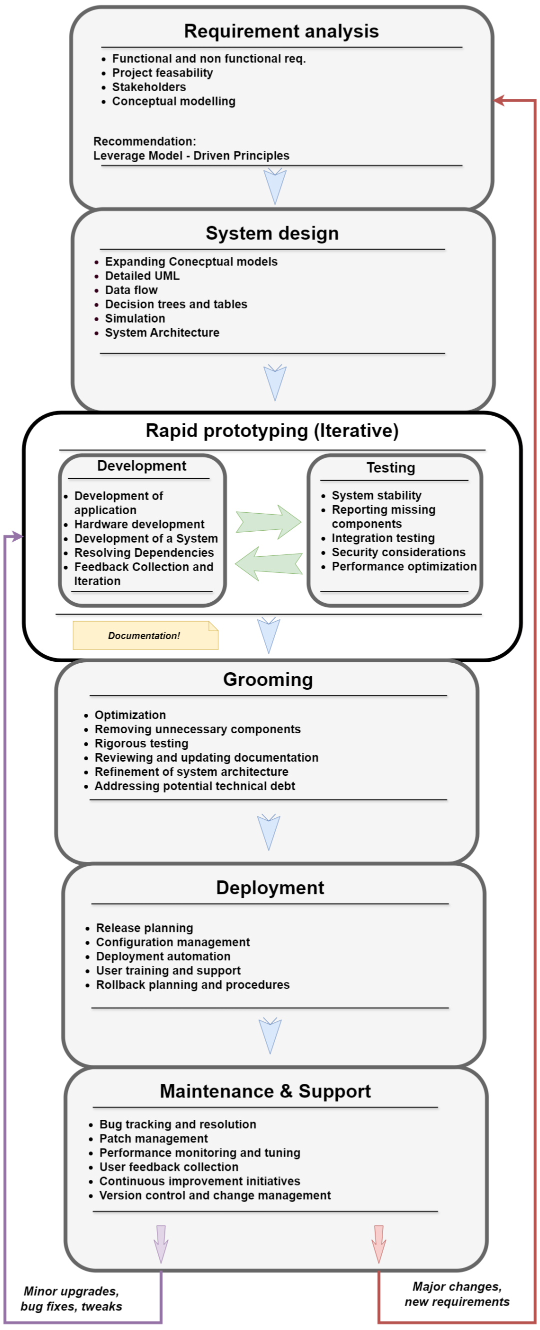
   - **Small updates**: Incremental updates are applied to address minor bugs and optimize performance, ensuring compatibility with evolving technologies without disrupting stability.
   - **Major changes**: Significant upgrades or modifications are planned, tested, and deployed in a structured manner to achieve substantial improvements while minimizing disruption.

The model is structured into several distinct phases, each crucial for IoT project success. The **Requirement Phase** gathers specific requirements, considering the diverse interactions within IoT systems. The **Design Phase** translates these requirements into a flexible design adaptable to iterative development. The **Rapid Prototyping Phase** involves continuous cycles of development and testing, particularly suitable for addressing hardware-related challenges. The **Grooming Phase** refines and optimizes the system, ensuring robustness before deployment. The **Deployment Phase** ensures seamless integration and operation of IoT components. Finally, the **Maintenance and Support Phase** manages both minor updates and major changes, ensuring the system remains adaptable and responsive to new requirements and technological advancements.

The proposed model serves as a flexible guideline rather than a rigid framework, accommodating the varied requirements of IoT projects. Iteration lengths and phase transitions are intentionally not fixed, as they depend on the project's scope, the involved stakeholders, and specific planning needs. In early phases, the process can either progress quickly or extend over several years. During the Rapid Prototyping phase, the development approach can vary—from Scrum to rapid prototyping or custom methodologies—allowing teams to adapt based on their preferences. Similarly, the Grooming and Maintenance phases are highly adaptable, recognizing that the timeline for product retirement is unpredictable Yang and Bi (2014).

The **Requirement phase** is essential for gathering detailed and specific requirements that account for the diverse components and interactions within IoT systems. This phase involves collecting functional and non-functional requirements, assessing project feasibility, identifying relevant stakeholders, and developing conceptual models International Organization for Standardization (2019). Leveraging Model-Driven Principles enhances the clarity and precision of requirements, ensuring that all system aspects are thoroughly considered. MDD can be particularly useful in IoT projects by enabling systematic transformations from high-level conceptual models to more detailed designs, facilitating ongoing refinement throughout the development process. This approach ensures that the system architecture remains adaptable to evolving requirements Cabot (2012).

In contrast, Model-Driven Architecture (MDA), a specific framework within MDD, is applied when there is a need to separate platform-independent models (PIMs) from platform-specific models (PSMs) Group) (2024). MDA is especially relevant in IoT environments where diverse hardware and software platforms must be supported. By using MDA, IoT systems can be designed in a way that allows for seamless deployment across various platforms, while ensuring the integrity and functionality of the system are preserved Kautz et al. (2018). Thus, while MDD provides the overall methodology for refining and evolving models, MDA should be employed in scenarios where cross-



Requirement analysis
Functional and non functional req.
Project feasability
Stakeholders
Conceptual modelling
Recommendation:
Leverage Model - Driven Principles
System design
Expanding Conceptual models
Detailed UML
Data flow
Decision trees and tables
Simulation
System Architecture
Rapid prototyping (Iterative)
Development
Development of application
Hardware development
Development of a System
Resolving Dependencies
Feedback Collection and Iteration
Testing
System stability
Reporting missing components
Integration testing
Security considerations
Performance optimization
Documentation!
Grooming
Optimization
Removing unnecessary components
Rigorous testing
Reviewing and updating documentation
Refinement of system architecture
Addressing potential technical debt
Deployment
Release planning
Configuration management
Deployment automation
User training and support
Rollback planning and procedures
Maintenance & Support
Bug tracking and resolution
Patch management
Performance monitoring and tuning
User feedback collection
Continuous improvement initiatives
Version control and change management
Minor upgrades, bug fixes, tweaks
Major changes, new requirements

Figure 1: Proposed new model for agile IoT SDLC

platform deployment and compatibility are major considerations.

In the **Design phase**, the focus shifts to translating the gathered requirements into a robust and flexible design. This phase includes expanding conceptual models into detailed UML diagrams, mapping data flows, constructing decision trees, and conducting simulations Brambilla et al. (2017). The objective is to develop a comprehensive system architecture that addresses all identified requirements while remaining adaptable to iterative enhancements.

The **Rapid prototyping phase** embodies the core principles of agile development, with continuous cycles of development, testing, and coding. This iterative approach allows for rapid prototyping and immediate resolution of issues, enabling the system to evolve in alignment with user needs and technological advancements Guerrero-Ulloa et al. (2023), Zelfia et al. (2022).

The **Grooming phase** focuses on refining and optimizing the system, essential for IoT projects. This phase involves rigorous testing, optimizing performance, and updating documentation to reflect the latest system state. The Grooming Phase enhances overall system stability and performance, aligning it with the dynamic requirements of IoT environments.

The **Deployment phase** ensures smooth integration and deployment of IoT components, facilitating a seamless transition from development to operation. Deployment strategies are designed to minimize disruption and ensure the system's high performance and reliability Yang and Bi (2014).

Finally, the **Maintenance and support phase** provides mechanisms for both minor upgrades and major changes, ensuring the system remains adaptable and responsive to new requirements and technological advancements. Minor upgrades are handled through rapid development and testing, while significant changes trigger a comprehensive reassessment and planning process. This phase includes bug tracking, patch management, performance monitoring, and continuous improvement initiatives, ensuring the system remains robust throughout its lifecycle.

By integrating these phases, the proposed Agile SDLC model for IoT offers a structured yet flexible framework tailored to the unique challenges of IoT system development, emphasizing continuous iteration, comprehensive documentation and adaptive design.

4 Discussion

The proposed Agile SDLC model for IoT systems represents a framework developed from practical experience in IoT project environments, particularly in smart home and smart city domains where embedded systems play an important role. It requires thorough review and validation by peers and industry experts. One of the challenges in evaluating the effectiveness of this model lies in the difficulty of quantifying its benefits. While a comparative analysis using time as a metric—measuring the duration of similar projects with and without this model—could offer insights, it may not fully capture the model's advantages in adaptability and responsiveness.

Traditional SDLC models often struggle with the dynamic and iterative nature of IoT development. By integrating Agile methodologies, this model aims to enhance the ability to pivot, make changes, and adapt to evolving project requirements, which is a vital requirement in IoT-related projects. This approach seeks to offer a more flexible and responsive framework.

One significant potential benefit of this model is its focus on maintenance. It is designed to reduce the frequency and severity of interventions required post-deployment. Ideally, this model will lead to fewer maintenance incidents, and when issues do arise, they will be quicker and easier to address, particularly concerning embedded contexts. This could result in a more stable and reliable system over the long term.

Moreover, the proposed model integrates modern practices from IoT development, such as Model-Driven Architecture (MDA) and Agile methodologies, into a streamlined SDLC framework. This hybrid approach is intended to provide a balance between the structured processes of traditional SDLC and the flexibility needed for IoT projects. By doing so, it aims to be adaptable across various project types and sizes, offering a versatile tool for developers.

However, it is important to note that this model is likely to work most effectively with small development teams, as it aligns with the principles of Agile and Scrum practices. Agile methodologies are typically best suited for smaller teams due to their emphasis on close collaboration, rapid iteration, and flexibility. The model's effectiveness and scalability for larger teams remain uncertain, as it was not specifically designed for that context.

While the model shows promise, it is currently conceptual and has yet to be empirically validated through extensive real-world applications. The model's success hinges on its application in pilot projects, which will be crucial in identifying its strengths and weaknesses. These pilot implementations will provide valuable feedback, helping to refine the model and ensure it can address practical challenges and deliver tangible benefits.

Additionally, the model's effectiveness in different industry contexts needs to be assessed. IoT projects vary widely in scope and complexity, from small-scale consumer devices to large-scale industrial systems. The model's adaptability and scalability across these diverse contexts will be a main factor in its broader adoption. Future research should focus on empirical studies and case studies to evaluate how well the model performs in these diverse settings, thereby solidifying its practical applicability.

5 Conclussion

The proposed Agile SDLC model for IoT systems represents a significant advancement in addressing the complexities of IoT development. By integrating Agile methodologies, Rapid Prototyping, and Model-Driven Development, the model offers a flexible and iterative approach essential for managing diverse hardware integrations and complex software interactions inherent in IoT projects. Its emphasis on continuous iteration and collaboration enhances development efficiency, minimizes risks, and accelerates time-to-market for IoT solutions. Moreover, the model's structured yet adaptable framework ensures responsiveness to changing requirements, technological advancements, and user feedback throughout the development lifecycle. By focusing on maintenance and support phases, it aims to reduce post-deployment issues and enhance system reliability, necessary for sustaining IoT solutions over the long term. The iterative development cycles facilitate rapid prototyping and validation, enabling early detection and resolution of issues while allowing for incremental enhancements based on real-world performance. Looking forward, further validation through empirical studies and case studies across diverse IoT applications will be essential to refine and scale the model for broader adoption in different industry contexts where embedded systems play a pivotal role. The proposed Agile SDLC model stands poised to drive innovation and efficiency in IoT development, contributing to the advancement of transformative IoT solutions in a rapidly evolving technological landscape.

References

Alberternst, S., Anisimov, A., Antakli, A., Duppe, B., Hoffmann, H., Meiser, M., Muaz, M., and Spieldenner, D. (2021). Zinnikus, i. orchestrating heterogeneous devices and AI services as virtual sensors for secure Cloud-Based IoT applications. *Sensors*, 21.

Alfawair, M. (2022). Internet-of-things: A system development life cycle (sdlc). *J. Theoretical Appl. Inform. Technol.*, 100(5):1643–1653.

Brambilla, M., Umuhoza, E., and Acerbis, R. (2017). Model-driven development of user interfaces for IoT systems via domain-specific components and patterns. *Journal of Internet Services and Applications*, 8(1):14.

Cabot, J. (2012). Clarifying concepts: Mbe vs mde vs mdd vs mda. *Post at Modeling Languages, http://modeling-languages.com/clarifying-concepts-mbe-vs-mde-vs-mdd-vs-mda/*, 86. Accessed: 2024-8-12.

Dado, M., Janota, A., and Spalek, J. (2015). Challenges and unwanted features of the smarter cities development. In *Internet of Things. IoT Infrastructures*, Lecture notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, pages 3–8. Springer International Publishing, Cham.

Dayo, Z. A., Aamir, M., Dayo, S. A., Khoso, I. A., Soothar, P., Sahito, F., Zheng, T., Hu, Z., and Guan, Y. (2022). A novel compact broadband and radiation efficient antenna design for medical IoT healthcare system. *Math. Biosci. Eng.*, 19(4):3909–3927.

Fahmideh, M., Ahmad, A., Behnaz, A., Grundy, J., and Susilo, W. (2022). Software engineering for internet of things: The practitioners' perspective. *IEEE Transactions on Software Engineering*, 48(8):2857–2878.

Fern, D. A. and Donaldson, S. E. (2003). Tri-Cycle: a prototype methodology for advanced software development. In *[1989] Proceedings of the Twenty-Second Annual Hawaii International Conference on System Sciences. Volume II: Software Track*. IEEE Comput. Soc. Press.

Group), O. O. M. (2024). MDA specifications. `http://www.omg.org/mda/specs.htm`. Accessed: 2024-8-12.

Guerrero-Ulloa, G., Rodríguez-Domínguez, C., and Hornos, M. J. (2019). IoT-based system to help care for dependent elderly. In *Communications in Computer and Information Science*, Communications in computer and information science, pages 41–55. Springer International Publishing, Cham.

Guerrero-Ulloa, G., Rodríguez-Domínguez, C., and Hornos, M. J. (2023). Agile methodologies applied to the development of internet of things (iot)-based systems: A review. *Sensors*, 23(2).

Gupta, S. and Gayathri, N. (2022). Study of the software development life cycle and the function of testing. In *2022 International Interdisciplinary Humanitarian Conference for Sustainability (IIHC)*, pages 1270–1275.

International Organization for Standardization (2019). Iso/iec/ieee international standard – systems and software engineering - content of life-cycle information items (documentation). *ISO/IEC/IEEE 15289:2019(E)*, pages 1–86.

Jat, S. and Sharma, P. (2017). Analysis of different software testing techniques. *International Journal of Scientific Research in Computer Science and Engineering*, 5(2):77–80.

Kautz, O., Roth, A., and Rumpe, B. (2018). Achievements, failures, and the future of model-based software engineering.

Klespitz, J., Bíró, M., and Kovács, L. (2015). Aspects of improvement of software development lifecycle management. In *2015 16th IEEE International Symposium on Computational Intelligence and Informatics (CINTI)*, pages 323–327.

Lakhan, A., Mohammed, M. A., Abdulkareem, K. H., Jaber, M. M., Nedoma, J., Martinek, R., and Zmij, P. (2022). Delay optimal schemes for internet of things applications in heterogeneous edge cloud computing networks. *Sensors (Basel)*, 22(16):5937.

Lantz, K. E. (1986). *The prototyping methodology*. Prentice-Hall, Inc., USA.

Madakam, S., Ramaswamy, R., and Tripathi, S. (2015). Internet of things (IoT): A literature review. *J. Comput. Commun.*, 03(05):164–173.

Salo, O. and Abrahamsson, P. (2008). Agile methods in european embedded software development organisations: a survey on the actual use and usefulness of extreme programming and scrum. *IET Softw.*, 2(1):58.

Schwaber, K. and Sutherland, J. (2011). The scrum guide. *Acrum Allience*, 21.

Singh, D., Tripathi, G., and Jara, A. J. (2014). A survey of Internet-of-Things: Future vision, architecture, challenges and services. In *2014 IEEE World Forum on Internet of Things (WF-IoT)*. IEEE.

Srivastava, A., Bhardwaj, S., and Saraswat, S. (2017). SCRUM model for agile methodology. In *2017 International Conference on Computing, Communication and Automation (ICCCA)*. IEEE.

Yang, L. Q. and Bi, Y. Y. (2014). Internet of things technology implementation by applying SDLC model: The intelligent storage management system. *Appl. Mech. Mater.*, 556-562:5385–5390.

Zelfia, H., Simanungkalit, T., and Raharjo, T. (2022). Comparison of scrum maturity between internal and external software development: A case study at one of the state-owned banks in indonesia. In *2022 1st International Conference on Information System Information Technology (ICISIT)*.