The Impact of Model-Driven Development on Agile Practices within Knowledge-Intensive Systems Engineering*

Ghazaleh Aghakhani[1,*], Konstantinos Tsilionis[2] and Sara Shafiee[3]

[1]*LouRIM, UCLouvain, Louvain-la-Neuve, Belgium*
[3]*Eindhoven University of Technology, Eindhoven, The Netherlands*
[4]*Technical University of Denmark, Lyngby, Denmark*

Abstract
Agile development cannot be applied in the same way in each domain of software engineering. Some type of software products do indeed have more inter-dependencies between features leading to a constrained planning game. This is notably the case for Product Configuration Systems (PCS), a domain where lots of knowledge on the software product needs to be explicit beforehand to come to the development of an accurate solution. To this end, software modeling is a must have. This paper explores the intersection of Model-Driven Development (MDD) and Agile methodologies within the context of developing knowledge-intensive systems like PCS. Through a literature review of three pivotal studies, we examine the consequences of this integration across project management, teaching, and lifecycle processes. Our findings highlight both the challenges and opportunities presented by combining MDD with Agile practices, offering insights into best practices and areas for future research.

Keywords
Agile development, Knowledge-Intensive Systems, Product Configuration Systems, Model-driven Development, Scrum, Agil-ISE

1. Introduction

Product Configuration Systems (PCS) refer to software tools or platforms that allow users to customize and configure complex products or systems according to their specific requirements [1]. These systems are often used in industries such as manufacturing, engineering, and retail, where products may have numerous customizable features, options, and variations [2]. Users typically interact with the PCS through an intuitive interface, where they can select from available options, specify parameters, and visualize the configured product in real-time. The system is then able to generate detailed specifications, sketches, or models based on the user's selections, which can be used for manufacturing, ordering, or further customization.

Model-Driven Development (MDD) is an approach to software development that focuses on creating and using models as primary artifacts throughout the software development lifecycle

Agil-ISE'24: 3rd International Workshop on Agile Methods for Information Systems Engineering, June 3rd, 2024, Limassol, Cyprus
ghazaleh.aghakhani@uclouvain.be (G. Aghakhani); konstantinos.tsilionis@kuleuven.be (K. Tsilionis); sashaf@dtu.dk (S. Shafiee)
0000-0001-6734-632X (G. Aghakhani); 0000-0001-9702-6941 (K. Tsilionis); 0000-0001-9433-5060 (S. Shafiee)

[3]. The goal of MDD is to create abstract conceptualizations that would drive the design of a solution of a specific software problem without actually relying on a specific platform.

The integration of MDD with agile methodologies within the context of PCS development is gaining traction due to its potential benefits for software development [7]. For example, in 2017, a Danish firm specializing in PCS transitioned from the Rational Unified Process (RUP), known for its comprehensive documentation and structured approach, to Scrum, marking a significant shift towards agility. The company is a global engineer-to-order organization specializing in chemical material production and processes. It heavily relies on user-intensive software systems like PCS and its continual efforts to enhance management and engineering practices. Over the past decade, the PCS development team has exclusively worked with two approaches: RUP and Scrum. The transition from RUP to Scrum occurred in a significant manner, with Scrum adopted as the sole method after the last RUP project was completed. The PCS team consists of various roles, including business analysts, configuration engineers, developers, testers, and Scrum masters. Despite the complexity of PCS development, Scrum principles and values remain intact. The company's transition to Scrum is considered a significant shift due to its agile nature compared to the plan-driven RUP method previously used. The research is particularly relevant as the company sought to understand how Scrum could be applied in PCS development. Additionally, the company already had experience with RUP and was familiar with the models used in Scrum-based PCS development.

The company invested in various aspects of this transition, spanning from the training phase to implementation. Despite this shift towards agile methodologies like Scrum, research indicates that there is still potential for integrating with MDD practices. However, this integration is in its early stages and requires further exploration. Challenges include the absence of a well-defined process, suitable tools, and the need for overcoming a steep learning curve. Numerous studies underscore the necessity for additional empirical evidence and industrial reports to advance the understanding of Agile MDD integration (for example, see [4, 5, 6, 7, 8]), with the ultimate goal of enhancing software quality and efficiency in this domain.

The focus of this research is to examine the integration of MDD with Agile methodologies in the PCS domain – a complex, knowledge-intensive field. Through analyzing three key researches around the same case company [9, 10, 11], we explore the impact of this integration on project management, education, and lifecycle processes, revealing both the challenges and benefits of merging MDD with Agile practices to enhance PCS development. Thereby, the research question driving the study is "*What are the consequences of agile life cycle adoption onto the development of knowledge intensive systems requiring an intensive use of models and model driven development*?" To such an end, we first start with discussing the evolution from RUP to Scrum in the case company, then we discuss the key focus points of an agile process fragment to be included in standard Scrum to use models into Scrum and combine them with user stories. Finally we highlight the consequences of the need for a customized Scrum life cycle for PCS development onto the teaching of Scrum for novice practitioners.

2. From RUP to Scrum: An Evolution in Software Development Life Cycles

Shafiee et al. [9] delves into the agile lifecycle adoption for PCS, contrasting RUP and Scrum methodologies in their PCS project activities and management. It distinguishes the distinct approaches of RUP and Scrum towards organizational strategies, stakeholder requirements, product knowledge, configuration processes, goals, software development, testing, deployment, maintenance, updates, and overall performance in PCS development. The study, through empirical evidence, underscores the differential impacts of these methodologies on addressing PCS-specific challenges, including organizational, knowledge acquisition, product modeling, IT technical, and resource constraints. On what follows below, different perspectives are given in terms of the Agile lifecycle adoption for PCS though a comparative analysis of RUP and Scrum:

- Organizational Approach and Project Activities. RUP employs a systematic, risk-reducing, specialization-sequential methodology with detailed initial analysis and planning, releasing PCS in large segments. In this context, moderate stakeholder engagement was noted in the case company. Scrum emphasizes rapid value delivery, engaging more IT and domain experts working parallel and interchangeably. It features quick, high-level initial planning, with detailed planning and daily micro-planning every three weeks, promoting fast and iterative PCS parts development. This approach fosters a higher stakeholder involvement.
- Stakeholder Requirements and Product Knowledge. RUP defines requirements through use-case diagrams and Product Variant Master (PVM), and prioritizes them early, aiming for versioned PCS delivery. Scrum uses user stories [12, 13, 14, 15, 16, 17] and acceptance criteria [18], requirements. This point was particularly tricky in the context of the case company. Indeed, user stories are artifacts particularly well suited for non-technical employees expressing their system expectations but the need is in this context rather for technical artifacts allowing the correct and consistent edition of knowledge. Indeed, users and stakeholders are here by nature rather technically oriented (e.g. engineers) leading to a tricky situation where both user stories and requirements models needed to be combined.
- Configuration Process and Goals. RUP approaches the configuration process and goals with detailed upfront documentation, utilizing in the context of the case company PVMs, Class-Responsibility-Collaboration (CRC) cards, in addition to classical UML diagrams like class diagrams for comprehensive understanding and visualization. Scrum documents goals and the configuration process in user stories and possibly case tools like Jira, opting for an overall analysis at project start and detailed sprint-based analyses without visual representation. Here again a cultural mismatch did appear and had to be tackled.
- Software Working, Testing, and Deployment. Both methodologies apply similar software development tasks but differ in organization and timelines. Scrum outperformed RUP in testing frequency and stakeholder engagement, offering a more responsive feedback mechanism. Deployment in Scrum is immediate upon stakeholder approval, contrasting RUP's project-end or at least phased deployment strategy.

All in all, Scrum has demonstrated faster development cycles, higher user satisfaction, and less need for maintenance, attributed to its iterative releases and flexible role management. Maintenance in Scrum, however, is challenged by minimal documentation, primarily consisting of user stories. RUP provides thorough documentation aiding maintenance but at the cost of slower development and resource consumption. The study reveals Scrum's superiority in tackling organizational, IT, and resource constraint challenges through its agile and iterative nature, fostering better stakeholder involvement and efficient task management. Conversely, RUP excels in knowledge acquisition, benefiting from its structured and comprehensive documentation approach. RUP relies on detailed product and process documentation, facilitating knowledge modeling and visualization. Scrum emphasizes user stories for knowledge documentation, trading off detailed visual knowledge representation for agility and scope flexibility.

PCS projects come with challenges such as resource constraints, product-related complexities, technical hurdles, knowledge acquisition difficulties, product modeling challenges, and organizational obstacles. The method used for planning, developing, and maintaining PCS projects can influence these challenges. Agile methods like Scrum are known for addressing shortcomings in traditional software development methods but may face differences when applied to PCS projects due to unique aspects like knowledge complexity, maintenance needs, and knowledge modeling techniques. While both RUP and Scrum can be effective in developing PCS projects, they differ significantly in their approaches. Scrum excels in handling organizational and IT challenges but may lack in knowledge acquisition aspects compared to RUP. The study emphasizes the importance of finding the right balance in PCS requirements representation and design integration to leverage the advantages of Scrum while addressing documentation and visualization needs specific to PCS projects.

The findings advocate for further research in diverse organizational contexts to validate these results and explore the integration of MDD in Agile PCS development, aiming to refine and adapt methodologies for improved PCS project outcomes.

3. On the Use of Models in Agile Development

MDD can play a critical role in enhancing agile project success. This is achieved through better communication, increased adaptability, and greater precision in both design and development stages [19, 20]. However, significant challenges have also been identified related to Sprint planning when having to combine models with value-driven development [21].

Wautelet et al. [10] introduces a process fragment, derived from extensive PCS development experience, to guide PCS development teams in adopting Scrum. The fragment, designed for flexibility, serves as a reference for tailoring to specific project needs rather than a prescriptive model and it has been built out of the experience of the Danish company. In other words, the fragment gives an additional set of tasks in order to integrate the models required for PCS development into Scrum. The fragment underscores the importance of feedback and validation in PCS, highlighting how to apply sprint-based development while managing dependencies and technical constraints.

Integrating models within the agile development of PCS projects necessitates a careful balance but MDD's focus on abstract modeling and automated code generation can complement

Agile's iterative and incremental nature. The combination aims to leverage MDD's strengths in managing complex domain knowledge and automation to support Agile's adaptability and customer-centric focus. This synergistic approach seeks to mitigate PCS-specific challenges through a harmonized methodology that respects the needs of PCS projects.

As said, PCS projects essentially use the PVMs, CRC cards and class diagrams as primary artifacts of the software development process [22]. Agile methods depict requirements on the basis of user stories; the latter also play a crucial role in the management of the agile life cycle. Indeed, sprints (the iterations of agile projects) have their content determined on the basis of the user stories set and effectively consist in fulfilling software scenarios associated with them. Traditionally, Scrum organizes its backlog using the user stories, which for PCS development, need supplementing with detailed product structures to define the configuration space comprehensively. For a complete understanding of product architecture, including the sequence and significance of selections and constraints on components, user stories often include a series of constraints. For instance, user stories specifying engine size choices are linked with scenarios outlining valid selections and their conditions. Due to the complexity of fully and efficiently expressing these constraints in natural language, PVM and CRC cards provide essential documentation support. These tools are indispensable for any complex PCS project, ensuring sprint backlogs respect all necessary constraints and dependencies. While user stories offer a quick reference, PVM provides formal documentation support. User stories categorize the sprint backlog by value, but precedence constraints and dependencies necessitate grouping them for implementation within the same sprint/release. The prioritization reflects the product's structure and technical constraints, requiring major components to be identified early in the project. This precedence significantly influences the software process management in every Scrum-based PCS project. In the Danish company under study, precedence and dependency constraints are manually established, with no tool support for ensuring traceability between user stories and PVM, necessitating deep domain knowledge for accurate backlog management. The development of a CASE tool aims to automate ensuring consistency and traceability between user stories and PVM, validating all constraints during sprint backlog preparation [23].

4. Challenges in Implementation and Teaching Agile Methods in Knowledge Intensive Domains

After examining the agile transition and the development of a process fragment within the case company, we now turn our attention to the impact that domain-specific customizations have on mastering agile development practices. As highlighted by Shafiee et al. [11], the adoption of new methodologies like Scrum in complex software system development is critically dependent on thorough and pertinent training. The research indicates that organizations often encounter difficulties in pinpointing the subtle yet crucial aspects required for effective Scrum training, especially in environments characterized by high complexity and interdependent features, such as PCS with the special focus on highly engineered manufacturing.

The study further elaborates on the challenges posed by Scrum's conventional use of user stories for capturing requirements within the PCS context. Given PCS's inherent complexity, there is a pressing need for conceptual models to accurately delineate and construct the system. This necessitates a significant shift from traditional agile methodologies towards a training regimen that explicitly addresses these challenges for teams engaged with knowledge-intensive systems.

A notable discrepancy exists between the language and structure of user stories and the conceptual models like the PVM that are crucial for encapsulating PCS's business logic. Consequently, the division and orchestration of development efforts based on user stories are problematic due to the challenges in encapsulating business logic and navigating PCS's intricate dependencies within them. This scenario demands a departure from evaluating developments merely on business value, advocating for a refined approach in teaching prioritization and task allocation that accommodates PCS features' interdependencies.

Training initiatives should therefore underscore the synergy between conceptual models such as PVMs and user stories, elucidating their critical role in unraveling PCS complexity and steering sprint planning and estimation. It is imperative for novice practitioners to be educated on leveraging conceptual models as the cornerstone for knowledge representation in PCS development, relegating user stories to a secondary role focused on communication and requirement elicitation. Moreover, Scrum training must encompass strategies for sprint management and task prioritization when facing knowledge-dense systems, tackling the hurdles posed by dense dependencies and the imperative for elaborate business logic articulation.

In addition, the development of CASE tools that facilitate the crafting of PVMs and other PCS-specific conceptual models is necessary to bridge the methodological divide between MDD and Agile practices, ensuring a harmonious integration of user stories and conceptual frameworks.

5. Conclusion

In this study, we looked at adopting an agile development life cycle for PCS for highly engineered manufacturing which are knowledge intensive software systems. Eventually, the strongest issue in this lead to study how to combine the use of models and MDD with Agile methods; the latter presents both challenges and opportunities. By focusing on a Danish company's shift from traditional to Agile methods, we have shown the complex yet beneficial nature of using MDD within Agile frameworks for PCS. This journey highlighted the need for specific changes and training to make Agile work well in complex system development. We also discussed the importance of creating new tools to help merge MDD and Agile more smoothly; generating the PVM and CRC automatically through CASE tools can be an efficient solution. Looking forward, we suggest more research to improve these methods, making it easier for companies to develop complex systems like PCS. Future work should aim at providing more evidence on the best ways to integrate MDD with Agile, and developing tools that support these practices more efficiently.

References

[1] C. Forza, F. Salvador, Managing for variety in the order acquisition and fulfilment process: The contribution of product configuration systems, International journal of production economics 76 (2002) 87–98.

[2] C.-H. Lee, C.-H. Chen, C. Lin, F. Li, X. Zhao, Developing a quick response product configuration system under industry 4.0 based on customer requirement modelling and optimization method, Applied Sciences 9 (2019) 5004.

[3] D. C. Schmidt, et al., Model-driven engineering, Computer-IEEE Computer Society- 39 (2006) 25.

[4] K. Tsilionis, Y. Wautelet, A model-driven framework to support strategic agility: Value-added perspective, Information and Software Technology 141 (2022) 106734.

[5] K. Tsilionis, Y. Wautelet, From service-orientation to agile development by conceptually linking business it services and user stories: A meta-model and a process fragment, in: 2021 IEEE 23rd Conference on Business Informatics (CBI), volume 2, IEEE, 2021, pp. 153–162.

[6] Y. Zhang, S. Patel, Agile model-driven development in practice, IEEE software 28 (2010) 84–91.

[7] T. T. V. Muthumanikandan, George, R. K. Harendra, et al., Enhanced model-driven web application development with code generation using deep learning technique, Intelligent Decision Technologies (2024) 1–16.

[8] S. Kiv, S. Heng, Y. Wautelet, S. Poelmans, M. Kolp, Using an ontology for systematic practice adoption in agile methods: Expert system and practitioners-based validation, Expert Syst. Appl. 195 (2022) 116520.

[9] S. Shafiee, Y. Wautelet, L. Hvam, E. Sandrin, C. Forza, Scrum versus rational unified process in facing the main challenges of product configuration systems development, Journal of Systems and Software 170 (2020) 110732.

[10] Y. Wautelet, S. Shafiee, S. Heng, Revisiting the product configuration systems development procedure for scrum compliance: An i* driven process fragment, in: International Conference on Product-Focused Software Process Improvement, Springer, 2019, pp. 433–451.

[11] S. Shafiee, Y. Wautelet, S. Poelmans, S. Heng, An empirical evaluation of scrum training's suitability for the model-driven development of knowledge-intensive software systems, Data & Knowledge Engineering 146 (2023) 102195.

[12] M. Cohn, User stories applied: For agile software development, Addison-Wesley Professional, 2004.

[13] M. Cohn, Advantages of user stories for requirements, InformIT Network (2004).

[14] Y. Wautelet, S. Heng, M. Kolp, I. Mirbel, Unifying and extending user story models, in: Advanced Information Systems Engineering: 26th International Conference, CAiSE 2014, Thessaloniki, Greece, June 16-20, 2014. Proceedings 26, Springer, 2014, pp. 211–225.

[15] G. Lucassen, F. Dalpiaz, J. M. E. M. van der Werf, S. Brinkkemper, Improving agile requirements: the quality user story framework and tool, Requir. Eng. 21 (2016) 383–403. URL: https://doi.org/10.1007/s00766-016-0250-x. doi:10.1007/S00766-016-0250-X.

[16] G. Lucassen, F. Dalpiaz, J. M. E. v. d. Werf, S. Brinkkemper, The use and effectiveness of user stories in practice, in: Requirements Engineering: Foundation for Software Quality: 22nd International Working Conference, REFSQ 2016, Gothenburg, Sweden, March 14-17, 2016, Proceedings 22, Springer, 2016, pp. 205–222.

[17] Y. Wautelet, S. Heng, S. Kiv, M. Kolp, User-story driven development of multi-agent systems: A process fragment for agile methods, Comput. Lang. Syst. Struct. 50 (2017) 159–176.

[18] K. Tsilionis, Y. Wautelet, C. Faut, S. Heng, Unifying behavior driven development templates,

in: 29th IEEE International Requirements Engineering Conference, RE 2021, Notre Dame, IN, USA, September 20-24, 2021, IEEE, 2021, pp. 454–455.

[19] H. Alfraihi, K. Lano, The integration of agile development and model driven development - A systematic literature review, in: L. F. Pires, S. Hammoudi, B. Selic (Eds.), Proceedings of the 5th International Conference on Model-Driven Engineering and Software Development, MODELSWARD 2017, Porto, Portugal, February 19-21, 2017, SciTePress, 2017, pp. 451–458.

[20] H. Alfraihi, K. Lano, Trends and insights into the use of model-driven engineering: A survey, in: ACM/IEEE International Conference on Model Driven Engineering Languages and Systems, MODELS 2023 Companion, Västerås, Sweden, October 1-6, 2023, IEEE, 2023, pp. 286–295.

[21] M. Snoeck, Y. Wautelet, Agile MERODE: a model-driven software engineering method for user-centric and value-based development, Softw. Syst. Model. 21 (2022) 1469–1494.

[22] S. Shafiee, L. Hvam, A. Haug, M. Dam, K. Kristjansdottir, The documentation of product configuration systems: A framework and an it solution, Advanced Engineering Informatics 32 (2017) 163–175.

[23] S. Shafiee, Y. Wautelet, S. C. Friis, L. Lis, U. Harlou, L. Hvam, Evaluating the benefits of a computer-aided software engineering tool to develop and document product configuration systems, Computers in Industry 128 (2021) 103432.