Combining Model Driven Development and Agile Software Development

Himesha Wijekoon, Vojtěch Merunka
Department of Information Engineering
Faculty of Economics and Management
Czech University of Life Sciences Prague
Prague, Czechia
email: wijekoon@pef.czu.cz, merunka@pef.czu.cz

Vojtěch Merunka
Department of Software Engineering
Faculty of Nuclear Sciences and Engineering
Czech Technical University in Prague
Prague, Czechia
email: vojtech.merunka@fjfi.cvut.cz

***Abstract*— Agile Software Development (ASD) is certainly the mostly used software development methodology now. ASD is lightweight and it provides faster development of software in an agile manner without unnecessary time-consuming activities. In contrast, Model Driven Development (MDD) is a heavyweight methodology which considers models as the basis of software development. MDD relies on extensive modelling which ensures consistent models. But this requires extensive tool support and automation. In this study, we review state of art of merging these two approaches. We found that most of the related research focus on coming up with new methodologies which apparently violate lightweight agile principles. However, it seems beneficial to keep ASD as lightweight as it is and try to assist it with components from MDD in problematic areas such as agile requirements engineering. In the end, we propose future research to investigate how to use components of MDD to improve the agile requirements engineering.**

Keywords- Model Driven Development; Agile Software Development; Requirements Engineering; Software Engineering.

I. INTRODUCTION

Agile Software Development (ASD) has become a major software engineering discipline and the most popular software development methodology [1]. ASD treats the customer in the center of the development process and aims to develop software fast as possible by cutting down unnecessary activities.

On the other hand, Model Driven Development (MDD) is based on the use of models through the development life cycle of a system. MDD uses methods and tools to provide this involving automation and semi-automation such as model generation and model transformation. MDD focuses to ensure traceability and completeness between the different levels of software design. Yet, MDD is not as popular as ASD due to its complexity, time consumption and the requirement for extensive tools.

In this paper, we review the work related to combine MDD and ASD to identify potential future research. We believe combining of these two approaches will bring huge benefits in the software development.

This paper is organized as follows. In Section II a brief description about MDD is provided. Then ASD is briefly described in Section III. Section IV provides a review of previous work combining MDD and ASD. In Section V, we describe a potential future research opportunity. Finally, Section VI provides the conclusion of the study.

II. MODEL DRIVEN DEVELOPMENT (MDD)

MDD is a software development methodology which uses models to drive the development. In the ideal scenario, MDD uses and manipulates domain models instead of code when developing software. Therefore, in MDD models are created before source code [2]. These models are used to generate code to produce software. Thus, MDD facilitates Domain Driven Design (DDD) as well. MDD and Model Driven Architecture (MDA) are subsections of Model Driven Engineering (MDE) [3]. The difference between MDD and MDA is that MDA follows standards of Object Management Group (OMG). MDA promotes the detachment of Platform-Independent Models (PIM) from Platform-Specific Models (PSM). Even though the MDD methods approach to reduce development time, they do not allow sufficient client involvement [4]. Therefore, MDD methodologies are not agile as they highly focus on modelling activities.

III. AGILE SOFTWARE DEVELOPMENT (ASD)

The "Agile Manifesto", which was published in 2001 aims to provide best practices for software development [5]. The agile manifesto values following principles.

- Individuals and interactions over processes and tools.
- Working software over comprehensive documents.
- Customer collaboration over contract negotiation.
- Responding to change over following plans.

ASD is a collection of software development methods based on an iterative development. ASD advocates the incremental development of software based on continuous interaction with the client and implementation starts much earlier (e.g., prototypes) in the life cycle rather than detailed specifications and documents [6]. Agile methodology has shown advantages over traditional software development by focusing on fast delivery of business value and assisting teams to continuously evolve and change. This minimizes the overall risk connected to software development. In ASD the software development team works with short iterations which includes development of features [7]. Yet ASD put less emphasis on analysis and design to speed up the making of working software and can create difficulties in large-scale projects [8].

 ISBN: 978-1-61208-981-2

IV. COMBINING MDD AND ASD

There have been numerous attempts for combining MDD and ASD. The idea behind combining these approaches is to overcome the shortcomings of each approach and to create a superior methodology. For an example ASD has difficulties with larger projects with the need of a high-level design and MDD practices don't have the full support of stakeholders which decreases the chances of a desirable application.

Matinnejad defines that the integration of Agile in MDD process can be developed by [9]:

- MDD-based: introducing Agile method to an MDD process,
- Agile-based: applying MDD process to an agile method and
- Assembly-based: integrating some fragments from Agile and others from MDD to develop the process.

Until now there is only one related survey [9] and two Systematic Literature Reviews (SLR) [10][11] in this area. All three surveys conclude that Agile MDD is still in its early stages. It is also mentioned that most of the attempts of combining MDD and ASD have not clearly mentioned the comprehensive details about the integration, benefits, and challenges. Therefore, it is suggested that more experience reports and evaluations are required to advance the area of Agile MDD. Agile MDD approaches have reported different positive impacts of such as improvement in productivity and quality, faster development rate and better customer satisfaction. Most often reported problems are lack of model management, lack of verification, and steep learning curve and start-up overheads.

Following are major Agile MDD approaches carried out by the researchers.

- Alfraihi & Lano have proposed a general and comprehensive process for integrating ASD and MDD [12]. It allows applications to be safely developed in an iterative and incremental manner.
- Romano and da Cunha have developed the Agile and Collaborative Model Driven Development (AC-MDD) [13]. A novel Unified Modeling Language (UML) profile named Web-AML was designed, allowing to represent agile models of web applications. To apply the proposed framework using these new models, a method was defined providing steps to transform agile models into web application source-codes.
- eXtreme Modeling is a model-based development analogue of eXtreme Programming [14]. It is an agile development approach based on the use of software models to specify and synthesize software systems.
- Essebaa and Chantit have combined MDA and Scrum agile methodology to improve sprints of scrum and benefit from MDA principles [15]. It is proposed to use V life cycle in each sprint of the project where they combine another variant of MDE, to generate automatically different tests applying Model Based Testing (MBT) principles.
- Agile Concern-Driven Development (Agile CDD) is a software development process that uses concerns as its primary artifact and applies agile practices [16]. Agile CDD is a reuse-focused development process in which an application is built incrementally by repeatedly reusing other existing concerns.
- ScrumDDM is a hybrid metaprocess which integrates MDD practices into the SCRUM method used in ASD. It is a as metaprocess, which can be used for multiple domains to support software maintenance and evolution [17].

V. RESEARCH OPPORTUNITY

As per the analysis, most of the Agile MDD work are towards specifying a new software development process or methodology by merging ASD and MDD. On the contrary the popularity of the ASD is due to its simplicity and faster feedback from customers. Therefore, it seems most of these proposed methodologies/processes break original intentions of Agile Manifesto. Further, when the processes are complex and difficult to follow, the chances are low for software development community to embrace them. On the other hand, very few research have been conducted to empower agile development with the MDD components preserving the simplicity of ASD. In this way, we could try to use fragments from MDD to overcome issues with ASD.

ASD has its own shortcomings. Especially agile requirement engineering process has its challenges and limitations. Rasheed et. al provides a comprehensive review about these issues [18]. Therefore, agile requirements engineering is a good place to involve the MDD practices as modelling seems to be lacking in ASD. However, the requirements modelling performed in agile software development methods is different from models developed in traditional software development methods [19]. Another difference in ASD is to use user stories as specifications of the customer requirements. Therefore, there is a need of novel techniques to automate/semi-automate model generation to assist agile requirements engineering.

There are only few research attempting to help ASD with automatic generation of models from user stories. Gupta et. al have developed a tool that automatically creates conceptual models from a given set of user stories [20]. The tool takes user stories with acceptance criteria as input and produces four conceptual models (Domain Model, Process Model, Use Case Model and Finite State Machines) as output. Robeer et. al have developed a Visual Narrator tool which automatically generates a conceptual model from a collection of agile requirements expressed as user stories [21]. Gilson et. al have developed a tool to create robustness diagrams, i.e., a form of semi-formal use case scenarios, from the automated analysis of user stories [22]. All these approaches use Natural Language Processing techniques to some extent to process the textual user stories.

As this is a quite new research area there is a need for proposing more useful techniques to generate or assist generating models in agile requirements engineering. The proven MDD practices such as automatic model generation

 ISBN: 978-1-61208-981-2

and model transformations can be utilized in this regard. Stable architecture is important in ASD to achieve a flat effort curve. Support modelling in the agile requirements engineering can help to achieve this.

VI. CONCLUSION

In this paper, we have first reviewed the work related to combining MDD and ASD. Then we have analyzed the findings to identify areas for future research in this regard. Most of the related work so far focus on creating a new software development methodology combining ASD and MDD. These approaches seem to be complex and break the simplicity of ASD which is the main software development methodology embraced by the software development community now.

Therefore, we believe it could be very useful to the industry if we bring good practices from MDD into ASD to improve it without making the process too complex. Hence, agile requirements engineering is identified as a problematic area which can be improved using modelling support from MDD components and tools.

REFERENCES

[1] R. Hoda, N. Salleh, and J. Grundy, "The rise and evolution of agile software development," IEEE software 35, no, pp. 58-63, 2018.

[2] Y.C. Huang, and C.P. Chu, "Legacy System User Interface Reengineering Based on the Agile Model Driven Approach," In Recent Advances in Computer Science and Information Engineering, pp. 309-314, Springer, Berlin, Heidelberg, 2012.

[3] T. Mens and P. Van Gorp, "A taxonomy of model transformation," Electronic notes in theoretical computer science, 152, pp. 125-142, 2006.

[4] J. Grigera, J.M. Rivero, E. Robles Luna, F. Giacosa, and G. Rossi, "From requirements to web applications in an agile model-driven approach," In International Conference on Web Engineering, pp. 200-214, Springer, Berlin, Heidelberg, 2012.

[5] K. Beck et al., "The agile manifesto," Agile Alliance, 2001.

[6] S. Urli, M. Blay-Fornarino, P. Collet, and S. Mosser, "Using composite feature models to support agile software product line evolution," In Proceedings of the 6th International Workshop on Models and Evolution, pp. 21-26. 2012.

[7] J. Highsmith and A. Cockburn, "Agile software development: The business of innovation," Computer 34, no. 9, pp. 120-127, 2001.

[8] V. Kulkarni, S. Barat, and U. Ramteerthkar, "Early experience with agile methodology in a model-driven approach," In International Conference on Model Driven Engineering Languages and Systems, pp. 578-590, Springer, Berlin, Heidelberg, 2011.

[9] R. Matinnejad, "Agile model driven development: An intelligent compromise," In 2011 Ninth International Conference on Software Engineering Research, Management and Applications, pp. 197-202. IEEE, 2011.

[10] H.A.A. Alfraihi and K.C. Lano, "The integration of agile development and model driven development: A systematic literature review," The 5th International Conference on Model-Driven Engineeing and Software Development, 2017.

[11] S. Hansson, Y. Zhao, and H. Burden, "How MAD are we? empirical evidence for model-driven agile development," In 3rd Workshop on Extreme Modeling, XM 2014, CEUR Workshop Proceedings, vol 1239, pp. 2-11, 2014.

[12] H.A.A. Alfraihi and K.C. Lano, "A process for integrating agile software development and model-driven development," In 3rd Flexible MDE workshop: ACM/IEEE 20th International Conference on Model Driven Engineering Languages and Systems (MODELS 2017), pp. 412-417. 2017.

[13] B.L. Romano and A.M.D. Cunha, "An Agile and Collaborative Model-Driven Development Framework for Web Applications," In Information Technology-New Generations, pp. 383-394. Springer, Cham, 2018.

[14] S. Kolahdouz Rahimi, K. Lano, H. Alfraihi, and H. P Haughton, "eXtreme Modeling: an approach to agile model-based development," Journal of Computing and Security, 6, no. 2, pp. 43-52, 2019.

[15] I. Essebaa and S. Chantit, "Model Driven Architecture and Agile Methodologies: Reflexion and discussion of their combination," In 2018 Federated Conference on Computer Science and Information Systems (FedCSIS), pp. 939-948, IEEE, 2018.

[16] O. Alam, "Towards an agile concern-driven development process," In 2019 IEEE/ACM International Conference on Software and System Processes (ICSSP), pp. 155-159. IEEE, 2019.

[17] E.F. da Silva, R.S.P. Maciel, and A.P.F. Magalhães, "Integrating Model-Driven Development Practices into Agile Process: Analyzing and Evaluating Software Evolution Aspects," In ICEIS (2), pp. 101-110, 2020.

[18] A. Rasheed, et al., "Requirement Engineering Challenges in Agile Software Development," Mathematical Problems in Engineering, 2021.

[19] I. Inayat, S.S. Salim., S. Marczak, M. Daneva, and S. Shamshirband, "A systematic literature review on agile requirements engineering practices and challenges," Computers in human behavior, 51, pp. 915-929, 2015.

[20] A. Gupta, G. Poels, and P. Bera, "Creation of Multiple Conceptual Models from User Stories–A Natural Language Processing Approach," In International Conference on Conceptual Modeling, pp. 47-57. Springer, Cham, 2019.

[21] M. Robeer, G. Lucassen, J.M.E. Van Der Werf, F. Dalpiaz, and S. Brinkkemper, "Automated extraction of conceptual models from user stories via NLP," In 2016 IEEE 24th international requirements engineering conference (RE), pp. 196-205. IEEE, 2016.

[22] F. Gilson, G. Matthias, and G. François, "Generating use case scenarios from user stories," In Proceedings of the International Conference on Software and System Processes, pp. 31-40. 2020.

 ISBN: 978-1-61208-981-2