Towards Improving Agility in Model-driven Development ⋆ ⋆⋆

Hessa Alfraihi

Dept of Informatics, King's College London, Strand, London, WC2R 2LS, UK

Abstract. Agile Model Driven Development (Agile MDD) is an approach that aims to combine Agile development and Model Driven Development (MDD). It is the intention of our research to analyse the impact and the challenges of combining Agile and MDD and try to improve this approach by increasing its adaptability by proposing a framework that facilitate Agile and MDD. This includes essential procedures, recommendations, and guidelines. The research is based upon extensive empirical work. At least two case studies will be carried out addressing a wide set of aspects and challenges of Agile MDD. The results of this research should enable developers to successfully adopt Agile MDD in practice.

1 Introduction and Problem Statement

The constant evolution of software development led to the emergence of new approaches and methods to cope with the continuously growing complexity of systems. Model-Driven Development (MDD)[18] has emerged as a new paradigm where models are the main artifacts in developing software. Essentially, MDD has changed the view of software development; instead of concentrating on writing code, MDD puts emphasis on the early stages of software development especially during the analysis stage where models are created [15]. After creating models, code is generated from them with the aid of some transformation tools. Using models can alleviate complexity by enabling developers to work at a higher level of abstraction and with the intrinsic separation of concerns. The ultimate objective of MDD is to automate the development process and to raise the level of abstraction [19].

Agile development [1] is another approach which imposes a disciplined project management upon software development. Basically, the software is developed based on iterative and incremental activity with direct customer involvement. The main goal of Agile development is to deliver a software that meets customer's needs in the shortest possible time considering rapid response to requirements' changes.

Despite the fact that Agile development and MDD paradigm seem to share the same motivation, accelerating the development process, they have conflicting

⋆ Initial stage of research

⋆⋆ Supervisor: Kevin Lano

viewpoints. Contrasts are based on the fact that MDD emphasises the importance of high-level models while Agile is heavily code-centric; it works at a low level of abstraction (programming languages). Another distinction is visible in the fact that agile addresses the methodological aspects, whereas MDD is more concerned with the architectural-aspects [21]. Agile claims that simplicity is essential whereas MDD can be viewed as a heavyweight process; many steps have to be taken before any software is developed. Another difference is visible in the fact that people represent an essential factor with the highest priority in Agile methods, whereas MDD relies on tools and technologies to develop a system and people are not an explicit feature [16].

On the other hand, they complement each other in some ways. Both of them aim to reduce the gap between requirements analysis and implementation and hence the errors that arise from incorrect formulation. Agile approaches achieve this by using short incremental iteration for development with direct customer collaboration while MDD do so by automating the development process. In addition, the automation in code generation in MDD implies faster development which is the promise of agility [10]. Moreover, the degree of abstraction in design, in MDD, consorts with what Agile tries to achieve: eliminate the gap between customer and developers, and provide rapid feedback to validate the system [13]. Besides, executable models - which are models that can be run- in MDD could serve as a communication medium among developers and customers, supporting collaboration which is the key element of Agile practices. Embracing changes is one of the key elements of Agile in which MDD can provide strong support for this fact in many-fold ways. For instance, traceability among artifacts, which helps to determine which parts of the system are affected by requirement change, facilitates quick response to changes. Likewise, any change in requirements can be easily reflected in code since it is generated automatically from models. Also, the separation of concerns concept makes the system less sensitive to requirement changes.

Agile Model Driven Development (Agile MDD) is an attempt to effectively combine these two approaches to gain the benefits of the two worlds while mitigating their drawbacks. Early work of combining Agile and MDD seems promising [20]. However, the field of Agile MDD is still immature and there is a need for further investigation before claiming its benefits and drawbacks on software development process [12, 3]. Currently, a number of Agile MDD approaches/tools have emerged, each of which identifies different practices and deals with different concerns, yet these approaches have not been objectively analysed. Therefore, there is a need for a critical assessment of these approaches, fundamentally aimed at identifying their benefits, and the problems that should be tackled. Moreover, there exists no standard practices or guidance to adapting Agile techniques and MDD paradigms, and there are only a few published case studies with which to examine different Agile MDD approaches. This research will study the development approaches that employ the practices of Agile development and the

principles of MDD in order to build software systems. To this aim, we will analyse and evaluate the features of those development methodologies as well as identify the major drawbacks and deficiencies of the related works. Based on this, a framework will be proposed for more effective Agile MDD includes solutions and improved techniques to overcome these challenges and provide guidance for the adoption of Agile MDD. More precisely, the research will answer the following research questions:
RQ1: What are the benefits of combining Agile methods and MDD?
RQ2: What are the main obstacles and challenges of the adoption of Agile MDD in practice?
RQ3: Are there solutions to addressing these challenges? What are the solutions?
RQ4: How can the process of Agile MDD in practice be improved?

2 Related Work

In the last years, a few studies have been published that investigate the combination of Agile and MDD. Hansson and Zhao [3] have conducted a substantial systematic literature review for the experiences of Agile MDD approaches from an empirical point of view. In their results section, they present some strategies of achieving Agile MDD and they briefly highlight the impact and the challenges of the current approaches. However, their focus was on the strategies of achieving Agile MDD not its benefits and challenges. Matinnejad [12] has proposed a criteria-based evaluation framework to review and compare some Agile MDD approaches. Based on the evaluation results, an empirical analysis was performed. In addition, Agile MDD approaches were characterised into three different categories: an Agile-based approach where MDD process is introduced into agile software project, an MDD-based approach where Agile methods are applied to existing MDD process, and an Assembly-based approach which has some elements from agile methods and other elements from the MDD process. Although this work represents a significant attempt to examine Agile MDD approaches, it was limited to a narrow scope. Stavru et al. in [20], have analysed the challenges of MDD and then they evaluated the potential of Agile practices and techniques to address these challenges. Yet, this evaluation is derived from subjective opinions of experts and there is no proof of validation. Mahé et al. in [11], have examined the possible joins between Agile and MDD approaches alongside the expected benefits and challenges of combining them. However, there is no evidence to prove their results since it is based on a theoretical perspective. With regards to MDD, Whittle et al. [22] have conducted a substantial empirical research on MDD where they identified the success and failure factors of MDD use. But, their focus was on MDD, not Agile perspectives.

To summarise, there is a lack of empirical research examining the impact and challenges of Agile MDD and how to maximise its effectiveness. In particular,

it appears there are three gaps in: understanding the state of practice of Agile MDD, examining the benefits that Agile MDD is supposed to provide, and identifying the challenges of Agile MDD's adoption. Therefore, this motivates us to conduct this research to fill existing gaps in the area.

3 Proposed Solution and Contributions

This section outlines the methods and solutions selected to investigate the gaps as well as the expected contributions.

3.1 Research Design

3.1.1 Systematic Literature Review (SLR) A Systematic Literature Review will be performed in order to examine the practices of Agile MDD in software development. The key aim of this review is to find the answer to the following research questions:

1. What is the current state of practice in Agile MDD approaches?
2. What Agile and MDD practices and techniques are being used in Agile MDD approaches?
3. What are the impact and the challenges of adopting Agile MDD?

3.1.2 Interview-based Survey At first glance, Agile MDD seems a straightforward summation of the two approaches, however it is quite challenging to obtain a coherent Agile MDD that can inherit the advantages of both approaches while avoiding their disadvantages [24]. To identify the impacts and the challenges of Agile MDD approaches from a practical point of view, we will conduct an interview. The interview has an exploratory and descriptive purpose. Specifically, the objectives of this interview are three fold: firstly, to gain insight into development of Agile MDD and how it has been achieved; secondly, to find out the benefits that Agile MDD has brought to the software development process; and thirdly, to investigate the main challenges and obstacles practitioners have encountered during the Agile MDD process. The interview will be conducted with industry practitioners who have worked in Agile and MDD approaches altogether. The interview is based on a semi-structured, open-ended approach [17] where the participants will be given the same questions in the same order while allowing for limited probing for further information. This kind of approach will make the analysis easier and comparison can be made.

3.1.3 Developing Agile MDD Framework Upon the completion of the analysis of the results of the literature reviews and interview findings, a framework will be developed considering the limitations and benefits. In addition, this framework will combine the best practices of Agile and MDD alongside recommendations and guidelines to developers to more effectively adopt Agile MDD.

3.1.4 The Case Study To evaluate the usability and the effectiveness of the proposed framework, at least two case studies will be developed. We have already planned one of these case studies and its requirements and specification are described thereafter. This case study will focus on transformation domain. In this case study we will translate UML models to ANSI C language using UML-RSDS [9]. The main functional requirement is *(F1): Translate UMLRSDS designs (UML class diagram, use cases, OCL, activities) into ANSI C code*, Figure 1 depicts the architecture of this code generator .
This functionality in turn is decomposed into five sub-goals in which each sub-goal represents a separate sub-transformation:

- F1.1: Translation of types
- F1.2: Translation of class diagrams
- F1.3: Translation of OCL expressions
- F1.4: Translation of activities
- F1.5: Translation of use cases

Each translation depends on all the previous translations. Therefore, the development of these translations is organised into five iterations, one for each translations. The stakeholders of this case study are: customers who needed to develop this system, the UML-RSDS development team, and end users who use such a system. Direct access is only available to the development team. Therefore, access to other stakeholders is substituted by the research team due to the need for those stakeholders. Besides the above functional requirement, a number

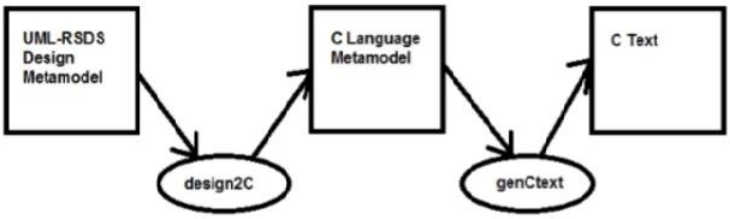

Fig. 1: C Code Generator Architecture

of non-functional requirements are considered, which includes: termination, syntactic correctness, model-level semantic preservation, traceability,and efficiency. The Agile MDD approach will be compared with a non-agile, non-MDD approach with manual coding.

The other case study concerns re-engineering financial optimisation algorithm from Matlab to `C#` platform. This case study will be developed for a public

financial management company, which serves local and national governments with a broad range of services, including investment management, pricing, risk evaluation, and other financial services. A representative of this company will be the customer of the development. Initially, we will incorporate the basic of MDD concepts such as models, transformations, and code generator with some agile principles such as re-factoring, direct collaboration with the customer, pair development, and Test Driven Development. The main aim of this case study is to find an effective way to combine agile development and MDD and to examine their benefits and shortcomings. To evaluate the use of Agile MDD process, the results will be compared against the current application in the company which has been developed in a traditional approach (neither MDD nor Agile development were used).

3.2 The Expected Contributions

The expected contributions of this PhD thesis research is as the following:

1. A Systematic Literature Review in order to give a comprehensive overview of the practical practices of Agile MDD approaches, their impact on development process, and the challenges/issues of Adopting Agile MDD.
2. A framework that comprises the best practices for adopting more effective Agile MDD.
3. Recommendations, techniques and guidelines to developers who need to cope with the combination of Agile and MDD in practice.
4. A set of case studies which implement the proposed framework and recommendations.

This research will help to create more productive and higher quality Agile MDD as it helps developers to be aware of and address potential challenges early via an Agile MDD framework. Also, it will help developers to find validated solutions to address the challenges in their practices. The SLR has relevance in the software engineering community as it should improve the understanding of the mechanisms behind the influence of Agile MDD on software development process.

4 Current State and Outlook

This research effort is still at an early stage. What we have carried out until now is:

4.1 The Systematic Literature Review

The main source we used to search for primary papers are ACM and IEEE Explore digital libraries by applying the search string (**AGILE** AND **MODEL-DRIVEN**). We experimented with many search strings and this one returned the largest number of publications. Further relevant publications have been found

at Google Scholar. Only publications written in English and published after 2001 were included. To select the eligible studies, we have performed two activities. Firstly, the title, the abstract, and the conclusion of the publications were analysed to check their eligibility. Secondly, the full text of the study is analysed for further refinement of the results. After applying our criteria of inclusion and exclusion, the number of relevant publications was only 10 papers. This relatively low number of publications indicates that the area of Agile MDD is still immature. However, a manual search was performed on the references of the eligible publications to make sure we have not missed any relevant study.
The initial results of the literature review revealed that there are promising benefits for combining Agile practices and MDD ,which include: increased productivity, improved product quality, the ease to respond to requirements changes, and a faster development process. While the most frequently reported challenges are: steep learning curve, insufficient tool-support, insufficient agility, lack of model and transformation management, and lack of systematic process. In order to make Agile MDD more efficient, these problems need to be addressed. Table 1 presents a comparison between different Agile MDD approaches.

Publication ID	Reference	Approach-type	Based-upon	Aim	Domain	Models used	Verification	Round-trip Engineering	Automatic Model Integration
P1	[24]	MDD-Based	Scrum, XP, MDD	Shorten delivery cycle time, improve quality	Telecommunication	Class diagram, state-machine,sequence diagram	High	×	×
P2	[6]	Assembly-Based	Parallel Agile, MDD	Apply MDD approach to small-sized projects	Web application	Domain model, test model, templates	Low	×	×
P3	[5]	MDD-Based	Mockup Models, Scrum-like	Involve customers, create high level designs	Web application	Mock-up models, use case	Low	×	×
P4	[4]	MDD-Based	General Agile process, MDE	Shorten development cycle, get early feedback	Mechatronic systems	plant models	High	×	×
P5	[23]	MDD-Based	XP, MDD	Enhance MDD for agility and quality	Telecommunication	Class diagram, use case, state transition	High	×	×
P6	[7]	MDD-Based	Reactive System Modelling	Get early feedback	Reactive multi-agent systems	Environment, behaviour, design, run-time models	Low	×	×
P7	[14]	MDD-Based	Scrum, MDA	Shorten development cycle	Financial systems	XML model	Medium	✓	✓
P8	[8]	MDD-Based	General Agile process, MDD, Product-line	Shorten development time	Business systems	Feature models	Low	×	×
P9	[2]	Agile-Based	Scrum, MDE, Rapid Application Prototype	Involve customers, quick designs	Web applications	Mock-up models	Low	×	×
P10	[9]	MDD-based	Scrum, MDD	Improve agility in MDD	Transformation	Class diagram, use case	Medium	×	×

Table 1: Comparison of Agile MDD Approaches

4.2 The Interview-based Study

So far, we have performed four interviews with industrial practitioners from different domains in order gain insight on the practical aspects of Agile MDD. At least two more interviews are required to have a sufficient understanding of the area. The initial result of the interviews show significant benefits of adopting

Agile MDD such as a higher level of abstraction and shorter development time. However, all the participants encountered some challenges and obstacles. The most frequently stated issues are: a sharp learning curve and lack of tool support. In-depth analysis of the interviews will be conducted to answer our research questions.

5 Conclusion and Future Work

The proliferation of Agile development and MDD over recent years make it an important research field. Yet, we have limited understanding of the benefits and challenges of combining them and how they can effectively be adopted. The specific characteristics of Agile MDD impose many challenges that need to be properly addressed in the adoption of Agile MDD. An important contribution of our work is to obtain a comprehensive understanding of the current state of Agile MDD, to analyse their impact and issues in more detail and what can be done to improve the situation in practice. As future work, we are aiming for a paper summarising our findings from the SLR and the interviews (once the interviews are completed). Based on the results of both SLR and the interview, a framework will be developed for adopting Agile MDD in practice alongside formulating recommendations and guidelines for an improved approach. The framework should propose the best techniques, processes, and recommendation to better achieve Agile MDD in practice. To evaluate and validate the framework and the recommendations, a set of case studies will be conducted in some domains and the framework will be discussed with experts in industry to validate it.

References

1. Manifesto for agile software development. *Available at* `http://www.agilemanifesto.org`, march 2015.
2. Fábio Paulo Basso, Raquel Mainardi Pillat, Fabricia Roos-Frantz, and Rafael Z Frantz. Combining mde and scrum on the rapid prototyping of web information systems. *International Journal of Web Engineering and Technology*, 10(3):214–244, 2015.
3. Håkan Burden, Sebastian Hansson, and Yu Zhao. How MAD are we? Empirical Evidence for Model-driven Agile Development. In *Proceedings of XM 2014, 3rd Extreme Modeling Workshop*, volume 1239, pages 2–11, Valencia, SPain, September 2014. CEUR.
4. Ulf Eliasson, Rogardt Heldal, Jonn Lantz, and Christian Berger. Agile model-driven engineering in mechatronic systems-an industrial case study. In *Model-Driven Engineering Languages and Systems*, pages 433–449. Springer, 2014.
5. Julián Grigera, José Matías Rivero, Esteban Robles Luna, Franco Giacosa, and Gustavo Rossi. From requirements to web applications in an agile model-driven approach. In *International Conference on Web Engineering*, pages 200–214. Springer, 2012.
6. Gábor Guta, Wolfgang Schreiner, and Dirk Draheim. A lightweight mdsd process applied in small projects. In *Software Engineering and Advanced Applications, 2009. SEAA'09. 35th Euromicro Conference on*, pages 255–258. IEEE, 2009.

7. James Kirby Jr. Model-driven agile development of reactive multi-agent systems. Technical report, DTIC Document, 2006.
8. Vinay Kulkarni, Souvik Barat, and Uday Ramteerthkar. Early experience with agile methodology in a model-driven approach. In *International Conference on Model Driven Engineering Languages and Systems*, pages 578–590. Springer, 2011.
9. Kevin Lano. Uml-reactive system design support. Technical report, King's College London, 2012.
10. Kevin Lano, Hessa Alfraihi, Sobhan Yassipour Tehrani, and Howard Haughton. Improving the application of agile model-based development: Experiences from case studies. In *The Tenth International Conference on Software Engineering Advances*, pages 213–219. International Academy, Research, and Industry Association (IARIA), November 2015.
11. Vincent Mahé, Benoit Combemale, and Juan Cadavid. Crossing model driven engineering and agility: Preliminary thought on benefits and challenges. In *Proceedings of the 3rd Workshop on Model-Driven Tool & Process Integration, in conjunction with Sixth European Conference on Modelling Foundations and Applications*, pages 97–108, Paris, France, 2010.
12. Reza Matinnejad. Agile model driven development: An intelligent compromise. In *Software Engineering Research, Management and Applications (SERA), 2011 9th International Conference on*, pages 197–202. IEEE, 2011.
13. Stephen J Mellor. *MDA distilled: principles of model-driven architecture*. Addison-Wesley Professional, 2004.
14. Mina Boström Nakićenović. An agile driven architecture modernization to a model-driven development solution. *International Journal on Advances in Software Volume 5, Number 3 & 4, 2012*, 2012.
15. Ruben Picek. Suitability of modern software development methodologies for model driven development. *Journal of Information and Organizational Sciences*, 33(2):285–295, 2009.
16. Ruben Picek. Suitability of modern software development methodologies for model driven development. *Journal of Information and Organizational Sciences*, 33(2):285–295, 2009.
17. Jane Ritchie, Jane Lewis, Carol McNaughton Nicholls, Rachel Ormston, et al. *Qualitative research practice: A guide for social science students and researchers*. Sage, 2013.
18. Bran Selic. The pragmatics of model-driven development. *IEEE software*, 20(5):19, 2003.
19. Bran Selic. Model-driven development: Its essence and opportunities. In *Object and Component-Oriented Real-Time Distributed Computing, 2006. ISORC 2006. Ninth IEEE International Symposium on*, pages 7–pp. IEEE, 2006.
20. Stavros Stavru, Iva Krasteva, and Sylvia Ilieva. Challenges of model-driven modernization-an agile perspective. In *MODELSWARD*, pages 219–230, 2013.
21. Hans Wegener. Agility in model-driven software development? implications for organization, process, and architecture. In *OOPSLA 2002 Workshop on Generative Techniques in the Context of Model Driven Architecture*, volume 23, 2002.
22. Jon Whittle, John Hutchinson, and Mark Rouncefield. The state of practice in model-driven engineering. *Software, IEEE*, 31(3):79–85, 2014.
23. Yuefeng Zhang. Test-driven modeling for model-driven development. *Software, IEEE*, 21(5):80–86, 2004.
24. Yuefeng Zhang and Shailesh Patel. Agile model-driven development in practice. *IEEE software*, 28(2):84, 2011.