The Integration of Agile Development and Model Driven Development - A Systematic Literature Review

Conference Paper · January 2017

DOI: 10.5220/0006207004510458

2 authors:

Hessa Alfraihi
King's College London
8 PUBLICATIONS **21** CITATIONS
SEE PROFILE

Kevin Lano
King's College London
287 PUBLICATIONS **2,897** CITATIONS
SEE PROFILE

The Integration of Agile Development and Model Driven Development
A Systematic Literature Review

Hessa Alfraihi and Kevin Lano
Dept. of Informatics, King's College London, London, U.K.

Keywords: Model-Driven Development, Agile, Agile Model-Driven Development, Systematic Literature Review.

Abstract: In this paper, we present a Systematic Literature Review (SLR) on combining Agile development and Model-Driven Development (MDD). The objectives of this paper are to identify what are the main characteristics of current Agile Model-Driven Development (Agile MDD) approaches, as well as the benefits and the problems of adopting these approaches. Fifteen publications have been identified and selected as primary studies on which we conducted the analysis. The results show that Agile development and MDD can coexist and benefit from their integration. However, combining Agile and MDD is still in its early stages and more effort is required in research to advance this area. The main contributions of this paper are: detailed and condensed results in the context of current Agile MDD approaches, detailed results on the benefits of Agile MDD in practice, and the observed problems and challenges of the current Agile MDD approaches.

1 INTRODUCTION

Agile methods are lightweight software development processes that emerged as a reaction to plan-driven limitations by compromising between no process and excessive process (Fowler, 2005). They impose an iterative approach to develop systems incrementally. Agile methods include Extreme Programming (XP) (Beck and Fowler, 2001), Scrum (Schwaber and Beedle, 2002), Crystal (Cockburn, 2002), etc. Each of these share common principles and values defined by the Agile Manifesto (Beck, 2001).

Model-Driven Development (MDD) methods (Selic, 2003) have emerged as a new software development paradigm where models play a fundamental role. They are used to specify the required system and then to automatically generate the source code. MDD is established with the aim of raising the level of abstraction and increasing automation in code generation (Selic, 2003; Frankel, 2003). In this way, MDD claims to improve productivity, portability, interoperability and maintainability of the systems (Kleppe et al., 2003).

The integration of Agile and MDD (Agile MDD) is of growing interest for many reasons. Firstly, the benefits that Agile and MDD provide can lead to improved software development (Vijayasarathy and Turk, 2008; **?**). Secondly, MDD can be considered as a heavyweight process which can be a barrier to industry adoption, and we believe that adding agility to it will ease its adoption. Moreover, there exist fundamental differences and conflicts between Agile and MDD that might make the integration quite challenging.

This paper provides a detailed SLR on the integration of Agile methods and MDD. This SLR identifies the main characteristic of current Agile MDD approaches and explores the benefits as well as the problems encountered by these approaches. The remainder of this paper is structured as follows: in Section 2, we describe our research methodology that we followed to conduct the SLR. In Section 3, we present the main results of our SLR while in Section 4 we answer the research questions and present the data analyses from our findings. Section 5 presents the related work followed by a conclusion of this paper in Section 6.

2 REVIEW METHOD

This review has been undertaken based on the guidelines provided by Kitchenham (Kitchenham, 2004). The steps in SLR are elaborated in the following subsections.

Alfraihi H. and Lano K.
The Integration of Agile Development and Model Driven Development - A Systematic Literature Review.
DOI: 10.5220/0006207004510458
In *Proceedings of the 5th International Conference on Model-Driven Engineering and Software Development (MODELSWARD 2017)*, pages 451-458
ISBN: 978-989-758-210-3

2.1 Research Questions

In order to investigate the empirical evidence of current Agile MDD approaches, we identified the following questions:
RQ1: What are the main characteristics of current Agile MDD approaches?
RQ2: What are the benefits of adopting Agile MDD for the software development process?
RQ3: What are the problems and challenges of adopting Agile MDD?

2.2 Source and Primary Studies Selection

In this study, we carried out an automatic search within the following digital libraries and databases: **ACM Digital Library**, **IEEE Xplore**, **SpringerLink (MetaPress)**, **ScienceDirect (Elsevier)**, and **Google Scholar**. To make sure that we retrieve the most relevant papers, search strings and keywords should be well-chosen. Therefore, based on our research questions we used the following search strings (with adaptations for some digital libraries): (" *agile*" OR "*lightweight*") AND ("*model-driven*" OR "*model-based*). To complement the automatic search, a manual search was conducted on relevant journals which are: **International Journal on Software and Systems Modeling (SoSyM)** and **Journal of Software and Systems (JSS)**.

2.3 Inclusion and Exclusion Criteria

The studies that were taken into consideration were papers that present approaches for combining Agile methods and MDD for software development. Moreover, only peer-reviewed papers published after 2001 and written in English are included. When a publication has multiple versions, only the recent one was included. The following type of publications were excluded: short papers with less than 4 pages, papers proposing Agile methods without a focus on MDD, papers proposing approaches for MDD without considering Agile methods, papers concerned with MDD perspectives where models are only used for specifying and designing the system without a link to automatic code generation, papers proposing theoretical studies without practical implementation, papers proposing partial Agile MDD approach, i.e. Agile MDD should be applied for the full development life cycle (starting from requirements all the way down to testing), and papers combining Agile and MDD with other approaches (e.g. Agile, MDD, and user-centred design).

2.4 Data Extraction

The data extraction form was designed to ensure that sufficient data is extracted and collected to address the research questions. It has been checked by the first and second authors to ensure its validity. Here, we describe the key aspects of Agile MDD that we are concerned with:

- **The Characteristics of Agile MDD Approaches:** in this context, we are concerned with the main features of the approaches such as: (i) *the aim of Agile MDD approach*: in this aspect, we want to understand the different aims and reasons behind combining Agile and MDD. (ii) *the application domain*: the approaches are classified according to the target application domain for Agile MDD. (iii) *methodology type*: there are different ways to combine Agile methods and MDD techniques. Matinnejad (Matinnejad, 2011) has classified these as: Agile-based approach where MDD process is introduced into a current Agile software project, MDD-based approach where Agile methods are applied to an existing MDD process, and Assembly-based approach which has some elements from Agile methods and others from MDD process. (iv) *modelling approach*: the models can be defined using different modelling languages ranging from general purpose languages such as UML to Specific Domain Language (SDL) such as Matlab. (v) *Agile practices*: in this aspect, we are interested in finding which Agile practices have been employed in Agile MDD approach. (vi) *MDD practices*: here, we are interested in finding out which MDD practices have been used in Agile MDD approach. (vii) *Agile MDD process*: in this dimension, we are interested with understanding how Agile and MDD have been integrated. (viii) *verification and validation*: verification is meant to check if a system meets a set of pre-determinant specifications. This can involve running a simulation for a part of the system or performing special testing, while validation is used to check if the system being developed meets the customer's requirements. (ix) *evaluation methods*: here, we want to now how the approach has been evaluated? This includes: a case study, experiments, running an example, ..etc.
- **The Benefits of Agile MDD Approaches:** the benefits of combining Agile and MDD reported

Table 1: The general context of Agile MDD approaches.

Study ID	Reference	Aim	Application domain	Methodology type	Based-upon
S1	(Zhang and Patel, 2011)	Shorten delivery cycle time, improve quality	Telecommunication	MDD-based	Scrum, XP, MDD
S2	(Guta et al., 2009)	Apply MDD approach to small-sized projects	Web applications	Assembly-based	Parallel Agile, MDD
S3	(Grigera et al., 2012)	Involve customers, create high level designs	Web applications	MDD-based	Scrum, mockup models
S4	(Eliasson et al., 2014)	Shorten development cycle, get early feedback	Mechatronic systems	MDD-based	General Agile process, MDE
S5	(Zhang, 2004)	Enhance MDD for agility and quality	Telecommunication	MDD-based	XP, MDD
S6	(Kirby Jr, 2006)	Get early feedback	Reactive systems	MDD-based	General Agile process, MDD
S7	(Nakićenović, 2012)	Shorten development cycle	Financial systems	MDD-based	Scrum, MDA
S8	(Kulkarni et al., 2011)	Shorten development time	Business systems	MDD-based	General Agile process, MDD
S9	(Basso et al., 2015)	Involve customers, quick designs	Web applications	Agile-based	Scrum, MDE
S10	(Lano et al., 2015)	Improve agility in MDD	Transformation	MDD-based	Scrum, MDD
S11	(Luna et al., 2009)	Improve user involvement	Web applications	Agile-based	TDD, MDD
S12	(Rivero et al., 2014)	Improve user involvement	Web applications	Agile-based	Scrum, MDD
S13	(Rivero et al., 2013)	Improve requirements gathering and customer involvement	Web applications	Agile-based	Scrum, MDD
S14	(Cáceres et al., 2004)	Combine the advantages of Agile methods and MDD	Web applications	MDD-based	XP, MDA
S15	(Krasteva et al., 2013)	Improve the modernisation development process	Web applications	MDD-based	Scrum, MDE

in the studies are highlighted and discussed.

- **The Problems and Challenges of Agile MDD Approaches:** the problems and the limitations observed in the current Agile MDD approaches are identified.

2.5 Conducting the Review

To select the eligible studies, the search process has been conducted in three phases. Firstly, 299 papers were found after applying the search strings for the digital sources as follows: 81 papers from IEEE, 95 papers from ACM, 44 papers from Science Direct, and 79 papers from SpringerLink. Then, 58 papers were selected based on the titles and abstract scanning. Finally, the full text of the papers were reviewed for further refinement. After applying our inclusion and exclusion criteria, fifteen studies were included as shown in Table 1. To make sure that we have not missed any relevant study, the list of references of the included studies were searched manually. Moreover, we performed a manual search on some journals to complement the automated search without finding any further relevant study.

3 RESULTS

In this section, we report the results of the SLR research questions in regards to Agile MDD.

3.1 The Characteristics of the Approaches

The Aim of Combining Agile and MDD: keeping the customer involved and actively collaborated in the development process was the most frequently stated aim for adopting Agile MDD (5 of 15 cases). Moreover, accelerating development process was another common aim for combining Agile and MDD. Other aims for incorporating Agile and MDD included: obtaining early feedback, improving the quality of the final product, and creating quick design. Some characteristics of Agile MDD are presented in Table 1.

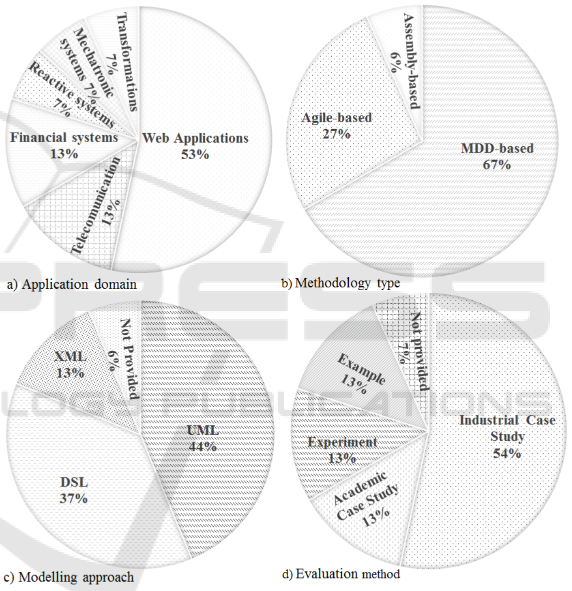

Figure 1: Statistics of some Agile MDD aspects.

The Application Domain: the most common domain for Agile MDD approaches was web applications (8 of 15 cases). Some other domains include: financial services, reactive systems, mechatronic systems and telecommunication. The majority of approaches were targeting a specific single domain, however, few studies explicitly mentioned that they can be applicable more generally for small-medium size projects such as **S2** and **S10** (Fig. 1a).

Methodology Type: we found that the majority of the reviewed Agile MDD approaches were MDD-based. That may be due to the fact that MDD can be considered as a heavyweight process and developers tend to relax its recommendations and to make it more lightweight by introducing agility to the process. Consequently, we think that the motivation for adopting Agile MDD is more relevant for MDD as opposed to Agile-based. A few approaches follow the Agile-based methodology whilst we found only one approach that is Assembly-based (Fig. 1b).

Modelling Approaches: Most of the approaches (6 of 15) used UML to define models while two more studies (**S2** and **S7**) used XML format. On the other hand, **S3**, **S9**, **S11**, **S12** and **S13** defined their own DSL models (Fig. 1c).

Agile Practices: in terms of Agile methods, the most common method was Scrum (8 out of 15 cases). This is not surprising, since the latest Agile survey (One, 2016) identified Scrum as the most widespread Agile process (58%). XP was also adopted in some approaches like in **S1** , **S5** and **S14**. On another hand, **S7** used Scrum throughout the whole project but when approaching the end of the project, tasks became difficult and could not be planned in sprints, and developers switched to Kanban. However, some Agile MDD approaches did not specify which Agile methods they used. In terms of Agile practices, most, if not all, Agile MDD approaches have followed an iterative and incremental development. Moreover, Test-Driven Development, refactoring, prioritised backlogs and direct customer involvement were the most frequently used. Frequencies of these practices and occurrences in reviewed studies are summarised in Table 2.

MDD Practices: most the approaches support the development of systems by automatically generating code (model-to-text transformation). However, only few studies explicitly mentioned they support model-to-model transformations without specifying how the transformations have been implemented such as **S3**, **S4**, **S9** and **S14**. In terms of reverse engineering, the majority of approaches provide only one-way forward engineering. Only **S7** supports round-trip engineering to synchronise the divergence between code and models. Executable models - which are models that can be run- have been used by some approaches such as **S1**,**S4**,**S5**, and **S6**. Frequencies of these practices and occurrences in reviewed studies are summarised in Table 3.

The Process of Agile MDD: most of the approaches failed to define a systematic process for combining Agile and MDD except mentioning that some Agile and MDD practices were used. The most comprehensive approach is presented in **S1**. **S1** used the System Level Agile process (SLAP) which is a Scrum-based process adopted by Motorola. Its development life cycle is divided into three successive iterations, each of which consists of application requirements and architecture, development, and system integration feature testing. To achieve their Agile MDD approaches, MDD activities (requirements, requirements analysis and high-level design, detailed design, code generation, and system testing) are mapped into these three iterations to build a new increment of the system.

From another perspective, Kulkarni et al. in **S8** argue that Agile methodology is not suitable as is with MDD as they found some activities, which require in-depth analysis and detailed documentation, cannot be conducted within short time-boxed iterations. For this reason, they proposed to introduce meta-sprints to be executed in parallel with the normal sprints where sprints are used for understood functionalities while meta-sprints are dedicated to features that require detailed investigation such as evaluation.

Another interesting approach is proposed in **S7**. In order to reduce the effort of model management, platform-independent models (PIM) and platform-specific models (PSM) are combined into a single model which contains all information for all programming languages.

Verification and Validation Techniques: through surveying the studies, verification procedures involved performing tests or running a simulation. For instance, **S2**, **S7**, and **S8** used integration testing while **S3**, **S9**, **S12**, **S13**, and **S15** used acceptance testing. Both **S3** and **S11** used interaction testing to check interactions between dynamic pages whilst **S10** and **S14** used unit testing. Moreover, simulation is used in **S1**, **S4**, and **S5** to verify the system earlier. However, explicit verification was omitted in **S6**.

Evaluation Methods: in terms of methods used to evaluate the Agile MDD approaches, most of the studies described an industrial case study (8 out of 15) or academic case study such as **S10** and **S14** to evaluate their approaches. Few studies presented a running example to illustrate their approaches such as **S11** and **S13** while another two studies (**S3** and **S12**) conducted an experiment to provide the proof of concept of their approaches (Fig. 1d).

3.2 The Benefits

The approaches of Agile MDD reported the benefits of combining the two approaches in different aspects. With respect to productivity, **S1** noticed threefold increase and 93% of code was generated automatically, whereas **S5** reported five times productivity improvement in terms of number of lines of code per staff. Similarity, **S8** observed an improvement in productivity due to continuous focus on the deliverables and shorter turnaround time (four weeks as compared to six months in traditional approach). In regard of quality, **S1** observed significant improved quality of code compared to hand-craft one. Likewise, **S5** reported a 20% improvement in quality by simulation in terms

Table 2: Agile practices used in reviewed Agile MDD approaches.

Agile practice	freq.	Studies that reported the practice
Test-driven development	5	S1, S3, S5, S7, S11
Pair development	5	S1, S3, S7, S14, S15
Continuous integration	5	S1, S2, S7, S14, S15
Refactoring	5	S3, S10, S11, S13, S15
Prioritised backlog	5	S4, S8, S9, S13, S15
Direct customer involvement	5	S10, S11, S13, S12, S14
Stand-up meeting	4	S1, S4, S5, S8
Collective ownership	2	S7, S15
Self-selected team	1	S7
Burn-down chart	1	S8
Release planning	1	S14

Table 3: MDD practices used in reviewed Agile MDD approaches.

MDD practice	freq.	Studies that reported the practice
Automated code generation (Model-to-text transformation)	13	S1, S2, S3, S4, S5,S6,S7,S8,S9, S10, S11, S12, S13
UML modelling	7	S1, S5, S6, S9, S10, S14, S15
DSL modelling	6	S3, S4, S9, S11,S12,S13
Executable models	4	S1,S4,S5,S6
Model-to-model transformation	4	S3,S4,S9, S14
XML modelling	2	S2, S7
Model-based testing	1	S3
round-trip engineering	1	S7

of number of defects that escaped from testing. In another respect, **S3** found an improvement in terms of time and customers satisfaction while **S9** found an improvement in code modularisation and simpler mockup design.

From Agile development perspectives, **S7** stated that Agile practices make the starting curve shorter. They also help to relax some recommendations of MDD and ease its adoption by allowing developers to spend a little time on small increments of functionality. Furthermore, it has been noticed that the development of Agile MDD is accelerated by using Test-Driven Development technique. Early feedback through frequent increments resulted in reduction in rework effort and better team commitment in **S8**. Likewise, **S4** stated that it helps to gain an early knowledge and reduce the assumptions developers have to make. Moreover, in **S13**, requirements gathering was improved by facilitating user involvement in the development process. **S14** found that pair development practice was helpful in terms of increasing team responsibility and commitment.

Compared to a traditional model-driven approach, **S12** noticed reduction in the errors and effort during modelling stage while an increase in reusing predefined architecture. **S14** found that the principle of aspects separation from MDD was helpful in facilitating requirements gathering and analysis and planning the user stories during iterations.

On the other hand, some studies such as **S2**, **S6**, **S10**, **S11** and **S15** found the experiences of adopting Agile MDD successful without showing what and how it was successful.

3.3 The Problems and Challenges

From all studies, we identified some problems that occur when combining Agile and MDD approaches. The first problem is a steep learning curve as reported in **S1**, **S5**, and **S9**. This arise due to the lack of experience, process, or culture. Zhang and Patel in **S1** discuss that Agile MDD is relatively new and due to the sharp learning curve, it is less likely to produce benefits in the short-term. However, there are long-term benefits for large projects with multiple releases.

Since models are created and transformed at different levels of abstractions with multiple views, insufficient management of models is another problem, which is expected since it is a main issue in MDD processes (France and Rumpe, 2007). Different studies report difficulties in keeping track of the relationship between requirements and models which impedes responding to changes incrementally such in **S3**, models merging such in **S5**, and models migration to new meta-models upon change request such as in **S8**. Kulkarni et al. in **S8** reported that the lack of configuration management of different parallel teams is a problem that needs to be addressed. Also, the lack of automation in testing made sprint durations longer. They argue that purely Agile methods are not compatible with MDD as they suggest using Agile methods in limited scope; i.e. mature development teams and projects less critical requirements. In the same context they conclude that *"we argue that true agility in model-driven development is possible only when code generators can also be adapted as quickly as application models"*.

4 SYNTHESIS

In this section, collected data is synthesised and the research questions are answered.

RQ1: What are the main characteristics of current Agile MDD approaches?

Due to the low number of the primary studies, it was difficult to provide a comparative analysis. Instead we provide a general insights into the main characteristics of Agile MDD approaches. We found that publications have different strategies of adopting Agile MDD, motivations, and application domains, although introducing Agile methods into MDD was common to achieving Agile MDD. Although generalisation is difficult here, we found that approaches used in the same domain have a similar context in terms of their aims and practice, eg., as in **S3**, **S12** and **S13**. In order to make an efficient and effective assessment of the area, more case studies, industrial reports, and experiments are needed.

RQ2: What are the benefits of adopting Agile MDD for the software development process?

When it comes to the benefits, many Agile MDD approaches reported different positive impacts of incorporating Agile and MDD such as improvement in productivity and quality, faster development rate and better customer satisfaction. Furthermore, Agile MDD has not been used only to develop a system from scratch but also it has been used successfully for the evolution of legacy systems as discussed in **S7** and **S15**. Nakicenovic in **S7** states that "*Through our industrial report we are able to provide strong support in favour of the claim that MDD and Agile practices can be used together, preserving the benefits of each*" and "*an Agile MDD could be a key success factor for organizations, which are not ready for the introduction of the full-scale MDA*".

In spite of the fact that most of the approaches had a successful experience in adopting Agile MDD, they failed to show what has succeeded and how. This makes understanding and comparing the results quite difficult. Moreover, it is unclear what kind of improvements MDD bring to Agile and vice versa.

RQ3: What are the problems and challenges of adopting Agile MDD?

From all studies, only few discuss the limitations and problems they face in Agile and MDD integration. Unfortunately, it has been observed in a lot of publications that success projects are reported more than failure. This review revealed that the most often reported problems are: lack of model management, lack of verification, and steep learning curve and start-up overheads. To address the model management problem, **S7** proposed to minimise the number of models and hence the management effort by combining combining PIM and PSM into a single model. Also, it is important to define a systematic guidelines on the integration of Agile development and MDD that can help cutting steep learning curve. To evaluate the approaches precisely, we urge researchers and practitioners to report both failure and success projects.

5 RELATED WORK

To the best of our knowledge, there is only one related survey (Matinnejad, 2011) and one SLR (Burden et al., 2014) in this area. In (Matinnejad, 2011), the author proposes a criteria-based evaluation framework to review and compare four Agile MDD approaches. Based on the results, he presented an empirical analysis. Although this is the first work that represents a significant attempt to examine Agile MDD approaches, it is limited to a narrow scope and it is not a systematic review. In (Burden et al., 2014), the authors conduct a SLR for the experiences of Agile MDD approaches from an empirical point of view. They propose two research questions to understand the state-of-art of Agile MDD and to investigate what is lacking in the literature. Seven publications are reviewed in this study. They conclude that the area in Agile MDD is still immature and there is a need for more reports on industrial experience of Agile MDD. The drawback of this work is that the quality assessment criteria for selecting the publications are too specific (e.g. details of development team must be presented). As a consequence, many well-known publications are missing such as (Nakićenović, 2012) and (Luna et al., 2009). Also, they are concerned with investigating the state-of-art of Agile MDD in general. In this study, we reviewed more studies and investigated different aspects of Agile MDD such as the main characteristics of current Agile MDD approaches, the benefits, and challenges of Agile and MDD integration.

6 CONCLUSION AND FUTURE WORK

In this paper, we have presented details of SLR on Agile and MDD integration. This study included a total of 15 papers that were published from the year 2001 till 2016. The main characteristics of Agile MDD were explored and the observed benefits and problems were identified. The results show that there is still a lot of confusion about what Agile MDD is and how the two approaches can be effectively integrated and what the real benefits and challenges are. This proves that Agile MDD is still in its early stages. This SLR should contribute in advancing the state of research of Agile MDD and can be used by researchers to bridge the gap in this area, while industrial practitioners can utilise the description of Agile MDD approaches and corresponding practices to identify both the success factors and potential challenges of the integration of Agile and MDD. One of the initial conclusions we came up is that many studies failed to explain how Agile and MDD have been combined. Moreover, methodological aspects have been only discussed by few studies. Most approaches present illustrative examples but lack comparative in-depth evaluation of the effectiveness of the approaches. As suggested by (Burden et al., 2014), more experience reports and evaluations are required to advance the area of Agile MDD. As future work, we will conduct an interview-based study to examine current practices for Agile methods and MDD in order to verify and complement the findings of the SLR.

REFERENCES

Basso, F. P., Pillat, R. M., Roos-Frantz, F., and Frantz, R. Z. (2015). Combining mde and scrum on the rapid prototyping of web information systems. *International Journal of Web Engineering and Technology*, 10(3):214–244.

Beck, K. (2001). Manifesto for agile software development.

Beck, K. and Fowler, M. (2001). *Planning Extreme Programming*. The XP series. Addison-Wesley.

Burden, H., Hansson, S., and Zhao, Y. (2014). 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. CEUR.

Cáceres, P., Díaz, F., and Marcos, E. (2004). Integrating an agile process in a model driven architecture. Citeseer.

Cockburn, A. (2002). *Agile Software Development*. Agile Software Development. Addison-Wesley.

Eliasson, U., Heldal, R., Lantz, J., and Berger, C. (2014). Agile model-driven engineering in mechatronic systems-an industrial case study. In *Model-Driven Engineering Languages and Systems*, pages 433–449. Springer.

Fowler, M. (2005). The new methodology.

France, R. and Rumpe, B. (2007). Model-driven development of complex software: A research roadmap. In *2007 Future of Software Engineering*, pages 37–54. IEEE Computer Society.

Frankel, D. S. (2003). *Model Driven Architecture: Applying MDA to Enterprise Computing*, volume 25. John Wiley & Sons.

Grigera, J., Rivero, J. M., Luna, E. R., Giacosa, F., and Rossi, G. (2012). From requirements to web applications in an agile model-driven approach. In *Web Engineering*, pages 200–214. Springer.

Guta, G., Schreiner, W., and Draheim, D. (2009). 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.

Kirby Jr, J. (2006). Model-driven agile development of reactive multi-agent systems. Technical report, DTIC Document.

Kitchenham, B. (2004). Procedures for performing systematic reviews. *Keele, UK, Keele University*, 33(2004):1–26.

Kleppe, A., Warmer, J., and Bast, W. (2003). *MDA Explained: The Model Driven Architecture : Practice and Promise*. The Addison-Wesley object technology series. Addison-Wesley.

Krasteva, I., Stavros, S., and Ilieva, S. (2013). Agile model-driven modernization to the service cloud. In *The Eighth International Conference on Internet and Web Applications and Services (ICIW 2013). Rome, Italy*.

Kulkarni, V., Barat, S., and Ramteerthkar, U. (2011). Early experience with agile methodology in a model-driven approach. In *Model Driven Engineering Languages and Systems*, pages 578–590. Springer.

Lano, K., Alfraihi, H., Yassipour Tehrani, S., and Haughton, H. (2015). 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).

Luna, E. R., Grigera, J., and Rossi, G. (2009). Bridging test and model-driven approaches in web engineering. In *International Conference on Web Engineering*, pages 136–150. Springer.

Matinnejad, R. (2011). Agile model driven development: An intelligent compromise. In *Software Engineering Research, Management and Applications (SERA), 2011 9th International Conference on*, pages 197–202. IEEE.

Mohagheghi, P. and Dehlen, V. (2008). Where is the proof?-a review of experiences from applying mde in industry. In *European Conference on Model Driven Architecture-Foundations and Applications*, pages 432–443. Springer.

Nakićenović, M. B. (2012). An agile driven architecture modernization to a model-driven development solution. *International Journal on Advances in Software Volume 5, Number 3 & 4, 2012*.

One, V. (2016). 10th annual state of agile development report.

Rivero, J. M., Grigera, J., Rossi, G., Luna, E. R., Montero, F., and Gaedke, M. (2014). Mockup-driven development: providing agile support for model-driven web engineering. *Information and Software Technology*, 56(6):670–687.

Rivero, J. M., Luna, E. R., Grigera, J., and Rossi, G. (2013). Improving user involvement through a model-driven requirements approach. In *Model-Driven Requirements Engineering (MoDRE), 2013 International Workshop on*, pages 20–29. IEEE.

Schwaber, K. and Beedle, M. (2002). *Agile Software Development with Scrum*. Agile Software Development. Prentice Hall.

Selic, B. (2003). The pragmatics of model-driven development. *IEEE software*, 20(5):19–25.

Vijayasarathy, L. and Turk, D. (2008). Agile software development: A survey of early adopters. *Journal of Information Technology Management*, 19(2):1–8.

Zhang, Y. (2004). Test-driven modeling for model-driven development. *Software, IEEE*, 21(5):80–86.

Zhang, Y. and Patel, S. (2011). Agile model-driven development in practice. *IEEE software*, 28(2):84.