UNIVERSITY OF GOTHENBURG

How MAD are we?

Empirical evidence for Model-Driven Agile Development

Bachelor of Science Thesis

Software Engineering and Management

SEBASTIAN HANSSON
YU ZHAO

University of Gothenburg
Chalmers University of Technology
Department of Computer Science and Engineering
Göteborg, Sweden, June 2014

How MAD are we?
Empirical evidence for Model-Driven Development

Sebastian Hansson
Yu Zhao

Examiner: Imed Hammouda

University of Gothenburg
Chalmers University of Technology
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Department of Computer Science and Engineering
Göteborg, Sweden June 2014

How MAD are we? Empirical evidence for Model-Driven Agile Development

Sebastian Hansson, Yu Zhao

University of Gothenburg
Chalmers University of Technology
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Hansson.seb@gmail.com. guszhaoyu@student.gu.se

***Abstract*- Agile development and Model-driven Engineering has both changed the software development industry significantly. Increasing development performance, productivity and software reliability. Agile with its rapid response to change and constant stakeholder involvement and Model-driven development making it easier to communicate within projects and providing high-level designs and architectures. Though they both have some deficiencies, could a combination of them both provide a solution to this? Could Agile inherit the advantages of Model-driven engineering and vice versa? In this paper we will presents a systematic literature review (SLR) collecting practices and experiences from projects that tried to combine these two development styles. We will answer questions like; what is the state of the art, combining Agile and MDD? And what is lacking in empirical literature on Model-driven Agile development?**

***Keywords*- Agile development, Model-Driven Engineering (MDE), Systematic literature review (SLR), Empirical evidence.**

I. INTRODUCTION

How MAD (Model-driven Agile Development) are we?

Both Agile and Model-driven development promises increased productivity, quality [P13, P14], complexity of systems, and stakeholder involvement [P11, P14]. Though they also complement each other, where Agile methodologies strength lies in its rapid response to change [P2, P14] and its emphasis of working software over detailed documentation. Model-driven engineering emphasizes a higher level of abstraction, with comprehensive communication and documentation of complex systems, MDD claims to improve internal communication within the projects [20] with help from models, that usually is easier to understand than code, for someone not directly involved in the development [20]. Why are projects combining Agile development and Model-driven practices seems to lie in their shortcomings. Model-driven practices have difficulties handling changes and including stakeholders in their projects, thus making it less likely to deliver software that meets the customers' expectations. Agile have difficulties with internal and face-to-face communication, and agile promoters seem almost frightened by detailed documentation [23]. Agile development also promotes to always be lightweight, making it difficult to apply agile development in large-scale projects where detailed documentation and high-level design is a must. Can a negotiation between these two styles be a solution? How can you integrate and inherit the best out of both styles avoiding their shortcomings? The most common approach is to include agile practices to MDD practices. To see what empirical evidence there are for MAD we decided to do a Systematic literature review (SLR). "*A systematic literature review is a means of evaluating and interpreting all available research relevant to a particular research question, topic area, or phenomenon of interest. Systematic reviews aim to present a fair evaluation of a research topic by using a trustworthy, rigorous, and auditable methodology"* [2].

We will deliver a comprehensive overview about the subject, providing future software development best practices by

collecting information about all documented cases we can find, and answer our research questions ***What is the state of the art combining Agile and MDD? And what is lacking in empirical literature on Model-driven Agile development?*** Our results tells us that the topic MAD is still too immature for anyone to claim any success or a state of the art over another, and that future work should focus on reporting industrial experience reports to close the current gap in literature.

The rest of this report is structured as follows:

In section 2 will start explaining the background of Agile, MDD, and MAD. In the next section, we will explain our methodology conducting this report, including identifying relevant papers with an inclusion and exclusion criteria. In section 4, Quality Assessment we provide an even more detailed inclusion criteria to ensure our selected papers provide enough quality data. In section 5 we will present our findings based on the eight papers that passed both our inclusion/exclusion criteria and our quality criteria. In section 6, Synthesis, we will answer our research questions. In section 7, threats to validity will be explained. Last, in section 8 we present our conclusion of the entire paper.

II. BACKGROUND

A. Agile development

Since 2001, when the Agile Manifesto [5] was introduced to the software-engineering industry, modern software development was changed forever. The agile manifesto [5] values practices such as, individuals and interactions over processes and tools; Working software over comprehensive documents; customer collaboration over contract negotiation; responding to change over following plans. Agile development includes practices such as, Scrum, XP, and cross-functional teams [5, 17, 18]. Agile development is a group of software development methods based on an iterative development style, using requirements, solutions, and cross-functional teams, to push development. Agile Software Development advocates the incremental development of software based on constant interaction with a customer community (Stakeholders) [P10] and implementation begins much earlier in the life cycle rather than using detailed documentation [1]. Agile has shown to have big advantages over traditional software development, and focuses on rapid delivery of business value, helping teams to constantly evolve and change the technical and functional landscape/business environment. This helps the organizations to minimize the overall risk connected to software development. Agile development use of feedback loops that makes it constantly able to adapt and change the requirements throughout the process with a close contact with the stakeholders [5]. With Agile practices you work with short iterations where all iterations should include the development of features, not tasks because the customers can better understand the purpose of features [16]. Though Agile processes are characterized by considerably less emphasis on analysis and design than almost all other modern life cycle models [P7] and can have difficulties in a large-scale project [P3].

B. Model-driven development

MDD (Model-driven Development) is another software development methodology that uses models to drive development.

MDD uses and manipulates so-called domain models instead of algorithms when developing software. Models are usually combined with code to produce software. Due to the increasing complexity of software systems, model-driven approaches are gaining popularity. In particular, graphical representations like UML diagrams of special system features allow dealing with the complexity and also enabling advanced analysis and validation capabilities [P8]. MDD is an approach to software development where extensive models are created before the source-code is written [P7]. We can categorize MDD and Model-driven Architecture (MDA) as subsections to Model-driven Engineering (MDE). *"MDD is a subset of MDE where the focus is on synthesis transformations, generating more concrete representations from abstractions, while MDE also emphasizes the need for other model-driven activities like reverse engineering, the opposite to synthesis* [3]". *With MDD, a serial approach to development is often taken, which is quite popular with traditionalists* [P7]. The difference between MDD and MDA is that MDA takes use of standards defined by the Object Management Group, OMG inequality for MDA is the detachment of Platform-Independent Models, PIM, from Platform-Specific Models, PSM, where a PIM abstracts away from implementation-specific details that are later perceived through transformations in the PSM. Based on the detachment of PIM and PSM a usually repeated claim is that the PIM can serve two objectives, both as documentation and as an implementation of the software [13], [14], [15]. While model-driven approaches represent a step forward to reduce

development time and work at a higher level of abstraction, most of them practically ignore stakeholder's involvement [P11]. The methodology is not agile to allow frequent requirement changes because they concentrate on modeling activities. Selmi et al. also point out some limitations such as [7]:

- Inability to model business processes.
- Inability to support various abstraction levels.
- No use of standard notation.
- Inability to model interaction aspects.
- Inability to support dynamic generation content.

C. Model-driven Agile development (MAD)

Combining Agile and MDD is a concern raising activity in the current literature [15, 20]. Mellor et al. [15] have discussed combining agile and MDA, the theoretical result demonstrates that, many of the agile practices can be integrated with MDD. Moreover, Kleppe et al. indicates that there are no problems in combining MDA and agile when the MDA tools are more mature [22].

According to similar work of W. El Kaim, P. Studer, and P.A. Muller, the approach to combine Model-Driven engineering and Agile relies on model transformations to promote agility in modeling [21]. Stahl and Völter argue that executable models should work as a better communication media [20]. As R. Gomes et al. state "*This integration can strongly contribute to a common understanding of the system and to improved communications between different stakeholders, as well as to a proficient SE collaborative development environment"* [6]. Rumpe also argues that UML and XP can be combined within MDA, the models will enable static analysis, rapid prototyping, code generation, automated tests, refactoring/transformation as well as detailed documentation [19]. One of the main reasons why organizations want and should combine these two popular development methods seems to lie in their shortcomings as agile 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 an desirable application. "*The models extended from UML focus on the behavior of Web application. The agile process aims to have quick-to-market property and adaptability to requirement changes"* [5].

III. METHOD

We have been reviewing literature regarding combining two development processes (Agile development and Model-driven development) based on a systematic approach called Systematic Literature Review (SLR). SLR is a common way of reviewing literature in the medical field but has gaining popularity also in the Software Engineering field. Kitchenham, 2007 has identified a guideline on how to perform a SLR for software engineering [2].

A. Aims and Objectives

Our aim of this study is to gather all existing empirical evidence about the combination of agile development and model-driven development and to make a synthesis of the data collected [2].

Sub-goals

- Summaries existing evidence [2]
- Identifying gaps in current literature [2]
- Provide framework/background for new research activities [2]

B. Research Questions

This systematic literature review (SLR) aims to analyze two different development methods (Agile and MDD) to find out if you can combine them, however current literature argues that there is little documented empirical evidence about MAD, our research questions aims to find this out. We will present and explain our research questions in Table1 (see page 4).

C. Inclusion Criteria

When performing a Systematic literature review it is critical to set inclusion criteria before you select the final papers, and in some cases also exclusion criteria. This is done so that the researchers will be free from prejudice and would not select the final papers based on their own biases. Our inclusion/exclusion criteria are presented in Table 2 (See page 4).

D. Identification of papers

Our method to identify the most critical and relevant papers to our research starts with a search-string. We spend considerable amount of time to find the best possible search-string that could cover our research area. We used three different digital libraries, IEEE explore, ACM digital library, and the SpringerLink library. These three libraries are covering the vast majority of software engineering publications. To cover all publications discussing agile process and model-driven development, we had to specify our search string carefully, to not miss any important papers. We tried different combinations and at the end we ended up using a search string that included both the words "agile" and "model" based on title search.

Research Questions (RQ)	*Motivation*
- RQ1. What is the state of the art, combining Agile and Model-Driven Engineering?	*- What is the best practice combining agile and MDD? To what extend is it sufficient? Methods/tools/approaches*
- RQ2. What is lacking in empirical literature on Model-Driven Agile Development?	*- What information does future research needs to provide to close the gap in the current empirical literature to make it more mature?*

Table 1. Research Questions

Inclusion	*Exclusion*
- Articles that is explicitly discussing the combination of agile and Model-driven engineering Empirically.	*- Articles discussing the combination of agile development and Model-driven development only theoretically, not providing any clear case or general experience reports.*
- Only articles published after 2001, as the Agile manifesto was then published	*- Articles not written in English*
- If the articles is published in multiple forms we select the most comprehensive one. Our selection priority is: Journal-> Chapter->Conference- >Workshop	*- Articles not published in Journals, Books, Conferences or Workshops*

Table 2. Inclusion/Exclusion criteria

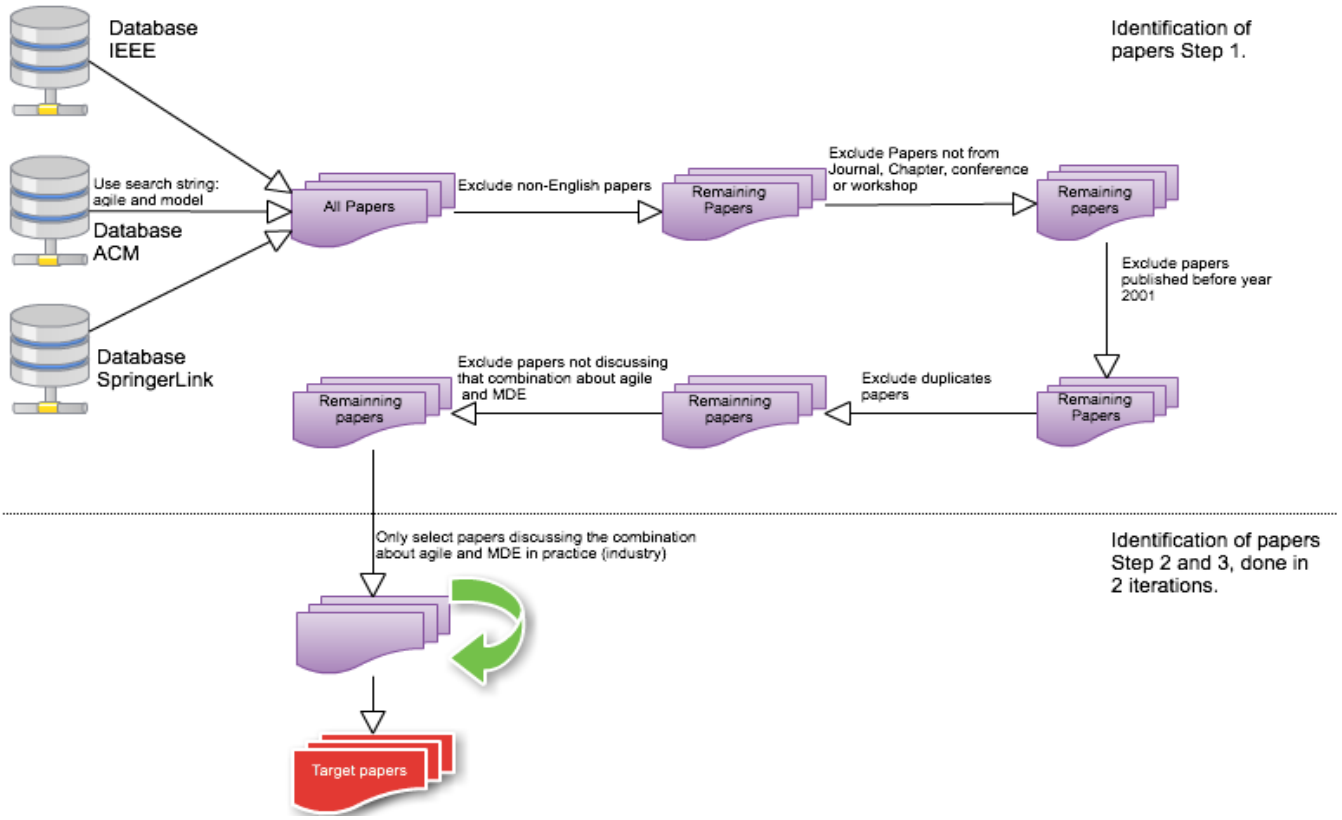

Figure. 1. Identification of papers.

Still, there is a possibility that even a good specified search-string does not cover everything within a specific research area. The papers found searching the databases is called primary studies, collecting these primary studies, reading them and check whether they make references of other papers covering the same topics is a must. These papers were found in the reference lists of our primary studies is called secondary studies. Primary studies are often poorly reported, so it may not be possible to determine how to assess a quality criterion [2]. Secondary studies are usually a better source to extract data from.

We then follow a three-step inclusion/exclusion technique also explained in Figure 1 (See page 4):

1. The first step to include relevant paper is to read the papers title and the papers abstracts and match these to our inclusion/exclusion criteria used.

2. The second step is to read the introduction and conclusion sections of the included papers from the previous step to exclude even more papers not relevant to our topic. In this step we are specifically looking if the papers are empirical or not as the previous step already excluded papers with our basic inclusion/exclusion criteria.

3. The third step is an iteration of step two but in this step we exclude papers by reading the full articles.

In the process of identifying papers, if the articles are published in multiple forms we select the most comprehensive; The Selection priority is Journals, Chapters, Conferences and Workshops. The priority is based on the different venues' credibility. The reason why we only select papers that were published after 2001 is because the Agile Manifesto was introduced in 2001 [5].

E. Identified paper

Primary studied

Results in Databases:

- Springerlink, 86 results, advanced search: where the title contains: ("agile" & "model")
- ACM Digital Library, 84 results, search: (Title:agile and Title:model)
- IEEE Xplore, 121 results, search: (("Document Title":agile) AND "Document Title":model")

Total results: 291 papers

Findings after reading the title and the abstracts:

- IEEE - 24 papers
- ACM - 27 papers
- Springerlink - 27 papers

Total: 78 papers.
Duplicates: 11.
Papers presented in multiple venues: 5.
Total after removing duplicates: 67 papers.
Total after removing publications presented in multiple venues: 62 papers.

Total after reading introduction and conclusion sections: 16 papers.

After reading through these sixteen articles we could exclude even two more articles, they did not mention any empirical case of combining MDD and agile development. We ended up with fourteen papers that mention empirically documented cases of trying the combination out.

Secondary studies

To find secondary studies, we read through all references in the fourteen included papers, surprisingly we could not find any secondary studies that matched our inclusion/exclusion criteria for our identification of papers. Again showing us that empirical evidence about Model-driven Agile development is lacking information in current literature.

Publications Venue

In Appendix section, we have presented three journals, three chapters, seven conferences, and one workshop that has in some way empirically combined Agile development with Model-driven development. According to the venues of publications, there are just two articles published in journals, suggesting that the research area on MAD is not yet mature.

IV. QUALITY ASSESSMENT

The previous section described how we included and excluded papers by specifying an inclusion and exclusion criteria. In this section we will dig deeper and sort out even more papers and identify which papers that are suitable to extract qualitative data from. It is critical to set quality criteria so the data extracted from different papers will be consistent and coherent with another. This makes it possible to make a synthesis, to answer our research questions.

Quality criteria	*Criteria definition*
- Qc1 - Do the paper state any goals/aims why they want to combine agile with MDD?	*- What did the project want to accomplish?*
- Qc2 - Do the paper mention how they combined the two development methods?	*- What was their strategy?*
- Qc3 - Does the paper specify any agile practices used?	*- Ie. scrum, tdd, iterations, xp, stakeholders etc.*
- Qc4 - Does the paper specify any MDD practiced used?	*- Ie. modeling language, tools, code generation, reverse engineering etc.*
- Qc5 - Does the paper specify what kind of team/project that was developing the software?	*- Ie. team-size, responsibilities, configurations, time durations etc.*
- Qc6 - Does the paper specify what kind of software they developed?	*- What domain?*
- Qc7 - Does the paper mention if the project was a success or a failure?	*- What was successful? what was not?*

Table 4. Quality criteria

Paper Nr	*P1*	*P2*	*P3*	*P4*	*P5*	*P6*	*P7*	*P8*	*P9*	*P10*	*P11*	*P12*	*P13*	*P14*
Qc1	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Qc2	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Qc3	✓	✓	✓	✓	✓	✓	✓	X	✓	✓	✓	✓	✓	✓
Qc4	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Qc5	✓	✓	✓	X	✓	X	X	X	✓	X	✓	X	✓	✓
Qc6	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Qc7	✓	✓	✓	✓	✓	✓	X	X	✓	✓	✓	✓	✓	✓

Table 5. Results of applying quality criteria

As Kitchenham, 2007 [2] states, this is done to:

• *To provide still more detailed inclusion/exclusion criteria [2].*

•*To investigate whether quality differences provide an explanation for differences in study results [2].*

•*As a means of weighting the importance of individual studies when results are being synthesized [2].*

• *To guide the interpretation of findings and determine the strength of inferences [2].*

• *To guide recommendations for further research [2]*

"As it is difficult to define what quality is, the CRD Guidelines and the Cochrane Reviewers' Handbook both suggest that quality relates to the extent to which the study minimizes bias and maximizes internal and external validity" [2].

A. Applied Quality assessment

We have provided a checklist based on Kitchenham, 2007 [2] quality check, This list is critical, to know that we can get quality data from the papers selected, that we can extract data that can answer our research question, and to make a valuable synthesis of the data collected. Ensuring sufficient contextual and methodological information is reported [11]. Hence, we want to know how MAD is done.

B. Results of applying Quality criteria

By applying our quality criteria we identified a subset of eight papers that are suitable to extract data from and to make a synthesis from. As Table 5 explains, you can see that paper P4, P6, P7, P8, P10, and P12 fails to report sufficient information about what kind of project, or team that performed in development. They lack information about team-size, responsibilities, and configurations or do not state any time-duration of the project making it difficult to draw parallels and make any synthesis, as they are not coherent with the other papers. Paper seven and eight also fail to deliver any information if the project was a success or failure. This is vital information for answering our research questions. Paper eight also does not mention any agile practices used. These excluded papers will not be a part of our section 5, results nor section 6, synthesis.

V. RESULTS

In this section we present our findings based on the eight papers that passed both our inclusion/exclusion criteria and our quality criteria. Some data will be presented using tables; other data presented is a collection of the most valuable information, which can answer our research questions. In this section the data is presented straightforward and we try to avoid making any conclusions or synthesis. The synthesis of this data will be reported in the next section, section 5.

A. Goals

This section describes why the teams/projects wants to combine MDD and Agile development.

Table 6 (See page 8) is a table of the extracted data from the eight papers which all passed our quality criteria. As Wookjin Lee et al. mention in their University Asset Management System development process, *"We cannot apply agile processes to web application development directly. UML is not sufficient for modeling the navigation of Web applications"* [P5]. Arguable pure agile methodology or pure model-drive software development is not enough any more. Therefore, in the industry, people are starting to combine Agile methods and MDD for different purposes. For instance, to shorten the development cycle, respond to the different domain changes, request more clients' involvement, etc. In Wookjin Lee et al.'s case [P5], Agile and systematic methodology is needed for the Web application development. Moreover, Yuefeng Zhang [P1], Vinay Kulkarni [P3] et al. prefers to shorten the delivery time in their projects. Rainer Burkhardt et al. [P2] and Y. Zhang [P14] tend to use the combination of agile method and MDD to respond to business and technologies changes. Meanwhile, in Y.Zhang 's work [P14], he also indicates that improved customer involvement in the development process could improve productivity and quality, which are reasons why he combines the Agile methods and MDD. Y.Zhang [P14], Julián Grigera [P11] et al. also aims to improve stakeholder' involvement during the development process in their customer satisfaction system. Additionally, Pohjalainen, P [P9] aims to define variability in software architectures with this combination. Y.zhang [P14], James T. Sawyer and David M. Branns [P13] concern is for better productivity and quality in their telecommunication system, meanwhile, they also want to improve verification and validation by combining Agile methods and Model-driven engineering.

- Shorten the delivery time	- Respond to business and technologies changes.	- Stakeholders' involvement in MDD	- Demand for web application	- Define variability in software architectures	- Improve productivity and quality	- Improve verification and validation
P1, P3	P2, P14	P11, P14	P5	P9	P13, P14	P13

Table 6. Data analysis table of goals

B. Strategy

This section describes how the teams/projects combined MDD and Agile development.

The strategy used to achieve agile model-driven development differs from the papers selected. Though many papers also suggest similar approaches. As example, both Rainer Burkhardt et al. [P2] and Vinay Kulkarni et al. [P3] suggest an evolutionary approach that is suitable for change. Pohjalainen, P. [P9] and Julián Grigera et al. [P11] suggest to build mockup models, as a mean of communication and for easy requirement gathering and requirement changes [P5, P11], also James T. Sawyer state to create "just enough" models up front, allowed to be changed along the project [P13]. Thus many papers promote an agile way of building models, with specific tools to easily transform the models [P11]. Yuefeng Zhang suggest using iterations for MDD, and also Wookjin Lee et al. wants sprints and iterations but to still have a high level design [P5]. Zhang, Y strategy is to have both agility and quality build into their development process [P14] and Vinay Kulkarni suggest a modified agile method and presented a tailored approach to address the need of managing evolution using model-based techniques [P3]. Julián Grigera et al. [P11], James T. Sawyer et al. [P13], and Zhang, Y[14] is using a Test-driven development approach combined with modeling.

The most interesting and comprehensive strategy we found is in Y.Zhang and S.Patel work [P1]. They use a methodology called System level agile process (SLAP). SLAP is a Scrum-based agile methodology, constructed by Motorola. SLAP includes extreme programming practices and takes use of SCRUM as a baseline [P1]. SLAP intends to split the software lifecycle in short iterations, where all iterations include three sprints; requirements, architecture, development, and then system integration feature testing (SIFT) [P1].

Y.Zhang and S.Patel then combine SLAP with the MDD process of Motorola, which is based on UML (Unified modeling language), dividing the software developments life cycle in multiple milestone phases where each milestone includes a single or multiple iterations. Combining Motorola's MDD process and SLAP, requires establishing a correspondence between SLAP sprints and MDD development activities. Using SLAP as a backbone process and mapping MDD processes to the corresponding SLAP sprints, building the software from model increments of system functionality in an incremental and iterative approach [P1].

Y.Zhang and S.Patel also state other interesting proposals such as:
"The key in achieving agile MDD is to combine an agile process in a way that can inherit the benefits of both and at the same time avoid their shortcomings", "From MDD perspective the key to success is to maximize automation using the MDD tools chain to enable mistake-free (high-quality) development and significant productivity increase" and "From the Agile perspective, the key is to efficiently achieve end-to-end iterations, from system engineering all the way to system testing. This requires streamlining different process activities such as system engineering, development, and testing" [P1].

Though the most interesting part of Y.Zhang and S.Patel work are their way of presenting a table of different MDD practices where each practices correspond or relate to an agile practice [P1]. They state that a key to implement Model-driven agile development lies in figuring out where and how to apply what agile practices in MDD [P1]. We will present the table as Figure 2 [P1].

C. Challenge

This section discusses what difficulties and challenges the teams/projects had when combining the two different development styles. Also describes challenges connected to Agile and Model-driven engineering separately.

MDD practices and corresponding (c) or related (r) agile practices.

		Agile practices							Agile management practices			
		Paired development	Test-driven development	Iterative and incremental development	Working software over comprehensive documentation	Automated regression testing	Continuous integration	Rapid feedback	Daily standup meeting	Daily development status tracking online	Short sprint cycle (5-week calendar)	Sprint postmortem meeting (retrospectives)
MDD practices	Modeling as coding	r										
	Paired modeling	c										
	Test-driven modeling		c									
	Iterative and incremental modeling			c								
	Modeling as live design documentation				r							
	Automated batch mode simulation					c						
	Continuous modeling						c					
	MDD tools chain							r				
Agile MDD management practices	Daily MDD plan update								r			
	Daily MDD task status update									r		
	MDD plan for the current sprint										r	
	MDD lessons learned in the past sprint (what to keep, change, try)											r

Figure 2. Yuefeng Zhang, Shailesh Patel, 2011.

Simon Urli et al. says that traditional MDD is too heavy for time-boxed iterations, as incremental iterations means change [P9]. Zhang, Y [P14] argues that model changes is hard to handle, thus in the ever-changing world we are living, with technological and business changes, the domain knowledge is a big issue [P1, P2].

Yuefeng Zhang et al. state, "*Agile MDD is still relatively new in real software development. The learning curve is sharp for any new organization to adopt due to process, culture, methodology, and other related changes. Thus, adopting a new agile MDD process is not likely to produce a short-term benefit. But, for the long-term, it's ultimately worth it for large projects with multiple releases"* [P1] also Xufeng (Danny), Liang et al. mention that existing web modeling techniques have a steep learning curve [P12].

To provide lightweight-agility in the development seems to be a solution for Vinay Kulkarni, who argues that traditional, agile development is not suited for larger teams/projects [P3].

Scrum	*XP*	*Iterations/incremental development*	*Test-driven development*	*Feature-driven development*
P1, P3, P5, P9, P11	*P1, P3, P5, P11*	*P1, P2, P3*	*P11, P13, P14*	*P3*

Table 7. Agile Practices

Unified modeling language	*Code generation*	*Model-driven architecture*	*Feature modeling*	*Alt. modeling /web modeling tools*	*Mockup models*
P1, P2, P5, P14	*P1*	*P9*	*P3, P9*	*P5, P11, P13*	*P9, P11*

Table 8. MDD Practices

D. Agile/MDD Practices

In this section the tools and practices used is described.

From the extracted data in our final eight papers. Yuefeng Zhang [P1] et al. suggests a general idea for Agile MDD. *"The key in achieving agile MDD is to combine an agile process with an MDD process in a way that can inherit the benefits out of both and at the same time avoid their shortcomings"*[P1]. It is obvious to see from the data we found, that people have chosen different Agile methods and different MDD practices to achieve the combination for different goals. In general, Scrum, XP, Unified modeling language (UML), and others modeling tools are the most common methods used among the teams. In the paper "Agile Model-Driven Development in Practice" [P1] Yuefeng Zhang and Shailesh Patel have chosen Scrum, XP, iterative development of Agile methods, Unified modeling language and code generation for Model driven methods. In their real-time telecommunications system, they aim to shorten the development cycle, which is also Vinay Kulkarni et al. aim [P3]. Coincidentally, as we can see in the Table 7, Vinay Kulkarni et al. [P3] have chosen the same agile methods as Yuefeng Zhang and Shailesh Patel [P1]. In order to respond quickly changes in business and technologies, Rainer Burkhardt et al [P2] have chosen iterations and unified modeling language as practices for the combination. Meanwhile, Y.Zhang [14] tend to use Test-driven development and Unified modeling language in combination to achieve the same goal as Rainer Burkhardt et al [P2]. Besides of responding the changes in business and technologies, Y.Zhang [14] also uses this combination to improve the stakeholders' involvement during the development process and to improve the productivity. Interestingly, Julián Grigera et al. [P11] have chosen a different agile methods and MDD technology to improve stakeholders' involvement, for instance Scrum, XP, Test-driven development, and alternative modeling tools. Another main purpose to combine Agile and MDD is to improve productivity and quality, which also Y.Zhang [14], James T. Sawyer and David M. Brann does. They decided to use a Test-driven development approach from Agile methods to achieve Agile MDD, but the difference is that James T. Sawyer and David M. Brann are using web modeling tools instead of Unified modeling language of MDDs tool-chain. In addition, James T. Sawyer and David M. Brann use the combination to improve the verification and validation in their projects. In Paper 5, Wookjin Lee et al. [P5] argues that Agile and systematic methodology is needed for web application development. For this integration Wookjin Lee et al. have chosen Scrum, XP, Unified modeling language and other modeling tools. In their Telecommunication account provisioning system [9], Pohjalainen, P [P9] uses Scrum, Model-driven architecture and mock-up models to define variability in software architectures.

E. Team/Project & Domain

As we can see from the Table 9, there are eight systems and applications that have been released successfully. However, some of the development teams actually have been through a tough time during the development process. In the real-time telecommunication system project [P1], *"Brand-new team that formed in batches—not all team members joined at the same time. Many team members had no MDD background or prior agile experience. In addition, most team members needed to pick up new domain knowledge"*. The same situation also happened in Vinay Kulkarni et al [P3]'s business application project, their development team was restructured, and the newly formed team had different domain knowledge and a totally

Paper Nr	*P1*	*P2*	*P3*	*P5*
Domain	Real-time telecommunication system	*Corporate legacy system.*	MDD toolset, MasterCraft for a database-centric business critical application	University Asset Management System. (Web Application)
Team/project	*New team with no agile, mdd, or domain knowledge*	*Web design team. Dynamic, changing team*	*New formed Team*	*Sw analyzer, domain expert.* *UI, component, and db developers.* *Tester.* *Client.*
Paper Nr	*P9*	*P11*	*P13*	*P14*
Domain	Telecommunication account provisioning system.	Customer Satisfaction System	Testing Software	Telecommunication system
Team/project	*Web design team.*	*10 Sw developers experiment*	*Simulation team, High visibility, high-pressure, short delivery project*	60 people divided into groups

Table 9. Team/Project & Domain

different agile and MDD background. Meanwhile, a similar situation happened in Rainer Burkhardt et al. [P2] corporate legacy system, the development teams were not static, and teams' size changed over time, as Rainer Burkhardt et al. [P2] described, *"The size of teams varies over time and from team to team. The weakest team in headcount is the Web Design Team. The strongest was the Development team with a headcount of 12, scaled down later on"* [P2]. However, we can still see roles well-constructed and static teams in some empirical practices. In Paper 5, Wookjin Lee et al. [P5] have defined many roles during the development process, for instance, software analyzer, domain expert, UI developers and testers. When we look into Y.Zhang [P14]'s telecommunication system, he described a detailed team construction process, *"More than 60 people work on this project, divided into groups. Some external groups are in different geographical locations. A big group is subdivided into multiple teams, and each team consists of two to 10 developers. Different teams are responsible for developing different subsystems. The members in a typical team sit close to each other and often work together. Weekly face-to-face group meetings foster communication among teams within a group as do weekly conference call meetings among groups".* Still, in some projects, they just mention vague, and unclear teams' constructions [P8, P11, P13]. All in all, the empirical evidence about Agile Model-Driven Engineering is still too unexplored. People in this industry lack the proper domain knowledge and experience, people are still in the exploration stage.

F. Impact

All the papers P1, P2, P3, P9 and P13 state that they successfully combined Agile and MDD, and many of them deployed a system or application. *"We saw big advantages in applying agile practices to modeling and in applying intensive modeling to our projects"* [P2]. Yuefeng Zhang et al. and Rainer Burkhardt et al. argue that the combination could be useful for future development teams, but that the learning curve is steep, thus it is not likely to produce short-time benefits, but for the longtime it is absolutely beneficial [P1, P2]. Rainer Burkhardt et al. [P2] and Zhang, Y [P14] noticed an increase in commitment, quality and productivity from the developers, also James T. Sawyer et al. [P13] had a better overall team performance and Julián Grigera et al improved in both time and satisfaction [P11]. Wookjin Lee et al. tell us that their approach would not be effective on large projects [P5], where Zhang, Y states that their approach could be beneficial and possible for any-size projects [P14]. Though most papers fail to provide detailed descriptions on what was successful and why it was successful.

VI. SYNTHESIS

This section aims to provide a synthesis of the results and data we collected from our eight final papers in the previous section. This quality data is gathered and collected in such a way that it can answer our research question.

Our findings tell us that the common trend among the reviewed papers is to include a more agile way into existing software development practices. In our case, to include agile practices into model-driven development practices. Especially because of the Agile developments advantages with rapid response to change and close stakeholder involvement. This is a common trend in the software industry, not only from MDD to agile, but also to include agile practices in other traditional development practices. A agile evolution of the Software product line (SPL) [P10].

A. Answering our research questions

- RQ1. What is the state of the art, combining agile and model-driven engineering?

Based on our findings, we can conclude that the area around MAD is somehow immature regarding theoretical approaches, but even more immature when it comes to empirical evidence and industrial experiences. To answer this research question, it is not enough to look at the theoretical part, we need evidence to prove any best practice/state of the art, though the evidence seems to be lacking. The strategies used among the papers found is in many cases also contradictory to each other providing information about different domains, goals and strategies, making it difficult to claim any best practice or success over someone else. Though the most common goals seem to be the agile value of rapid response to change and contact to external stakeholders, and models as an easy way to communicate to internal stakeholders. Most papers wants to include agility into their MDD processes, as this is a huge trend in current software development and has shown great success in big companies. What we need is teams who are willing to experiment using MAD, trying out different combinations and reporting their performance in detailed reports. Only when the area is mature enough it would be possible to claim a state of the art combining Agile and MDD. Though claiming that the area is not yet mature does not mean it is not possible to adapt. As Y.Zhang and S.Patel say *"Agile MDD is still relatively new in real software development, the learning curve is sharp for any new organization to adopt due to process, culture, methodology, and other related changes. Thus adopting a new agile MDD process is not likely to produce a short-term benefit. But, for the long term, it's ultimately worth it for large projects with multiple releases"* [P1]. and other authors such as Wookjun Lee et al. argues that pure agile methodology or pure model-driven software engineering is not longer enough. "*we can not apply agile processes to web application development directly. UML is not sufficient for modeling the navigation of Web applications"* [P5]

As mentioned, it is difficult to claim any state of the art or best practices for MAD yet, but Y.Zhang and S.Patel [P1] still gives us a clue, and the closest answer to our question. For this reason we will explain their findings more comprehensively. As Figure 2 in section 5.2 Strategy explains and Y.Zhang and S.Patel mention; different MDD practices corresponds to a given Agile practice and vice versa.

We will present a short summary of each Agile and MDD practice and it's relationship to each other based on Y.Zhang and S.Patels work [P1].

Modeling as coding: Using UML as a high-level visual programming language and use UML code generator as a UML compiler. Then forward engineering from UML to C or C++. Applying paired UML modeling (paired modeling).

Paired modeling: This corresponds to the paired development agile practice. Two developers or one customer representative work together on UML modeling for one sprint, then rotating pairs, works to detect and resolve modeling issues instantly.

Test-driven modeling: This corresponds to the test-driven development agile practice Test-driven development; they create UML sequence diagrams, and then use them to drive UML design and development of test cases. The focus of UML state machines at the beginning of a sprint is on sending and receiving signals in the right order.

Iterative and incremental modeling: This corresponds to the counterpart of agile practice, Iterative and incremental development. Authors do UML modeling over multiple sprints, repeat the same set of development activities in each sprint, create and evolve an executable model for each sprint with an increment of functionality on top of the model from the previous

sprint, keeping models executable for both simulation and target platform testing.

Modeling as live design documentation: Using UML as a live design document to minimize manual document. Applying *"working software over comprehensive documentation"* [5] to UML modeling.

Automated batch mode simulation: This corresponds to the agile practice of automated regression testing, development, and verifying individual test cases, and add them to the list for automated batch mode execution, then execute all test cases, finally analyze the test results.

Continuous modeling: The corresponds to the agile practice of continuous integration, merging UML design frequently, performing automatic code generation for simulation many times in a sprint, and performing automatic code generation for target platform testing and SIFT, two or more times in a sprint.

MDD tools chain: valuing the MDD tools chain over individual tools (IBM Tau for UML modeling, IBM tester for unit and integration test harness etc.) using it in particular to maximize automation.

Agile MDD management: using the agile project planning and management tool Version One and a daily MDD plan update in daily standup meetings, daily MDD task status updates, frequent reviews of sprint plans, and a sprint postmortem meeting for lessons learned.

Iterative and Incremental development: Aligning with the agile process, MDD has been changed to enable end-to-end iterations, that is, apply iterations to all phases, from system requirements specification all the way down to system testing.

System Engineering: In both Motorola's agile and MDD processes, they create a system engineering UML model to precisely define a system architecture and its dynamic behavior in a layered fashion, and then syntax check the system engineering model and create it with downstream reuse in mind, such as by reusing the interface, signals and etc.

Coding: In the agile process: Forward engineering from UML to C/C++, this is achieved by using and MDD tools chain that supports UML transition-oriented state machines and can automatically generate fully executable code from a UML model.

Unit Testing: For streamline UML unit testing and integration testing, MDD tools chain for unit testing has been used, on the UML side, IBM Tau (a UML model verifier), and on the unit test harness side, IBM tester has been used.

Integration and System Testing: For Streamline unit, Integration and system testing, the UML unit-testing environment is adapted to implement both integration and system testing environments. In this case, authors test the UML model as a whole in unit testing, same unit test cases for integration testing has been reused.

Test-driven development: First, authors create both UML model and test cases (for unit, integration, and system testing), according to the sequence diagrams. Finally, authors execute the test cases until the unit, integration, and system-testing pass.

- RQ2. What is lacking in empirical literature on Model-Driven Agile Development?

As we were performing a Systematic literature review, our initial search gave us around 300 results, by searching on "agile" AND "model" based on title search. After removing articles based on our inclusion and exclusion criteria we had only sixteen papers left who mention any empirical evidence in their papers, this is a rather low value for an SLR, suggesting that the area is still immature and surprisingly, according to the venues of publications, there are just two articles published in journals, suggesting again that the research area on MAD is not mature. By searching for secondary studies, it resulted in not finding a single paper that could match our inclusion/exclusion criteria nor our quality criteria, suggesting that more research in the area is needed. There are authors discussing different theoretical approaches for MAD but few papers describing detailed information how a company or a team implemented such a practice into their development process. Most of the eight papers that passed our quality criteria also fail to deliver detailed information how they performed MAD. They barely mention their team-setup, what practices they combined and what tools they used. Only in paper five the team is properly described. They provide detailed information about their team-setup describing their roles.

We suggest that the literature needs more experience reports and focus on describing in detail variables like:

- Teams/project - team-size, responsibilities, configurations, time durations.
- Practices used - Agile practices, MDD practices.
- Strategy - how they combined Agile and MDD practices, what was successful and what was not.
- Tools to achieve MAD.

VII. THREATS TO VALIDITY

Our research may face validity issues as our research was limited searching only in ACM digital library, Springerlink, IEEE explore. There are other digital libraries, which is widely used in the software engineering field. We plan to extend our study by adding more databases. Another threat to the validity is our search string, as we did a title search on the words agile AND model, it is possible that we have missed articles discussing the combination of agile and model-driven engineering but not mentioned it in their title. Though as searching the references in our primary studies for secondary studies applying our inclusion and exclude criteria and our quality criteria we could not find any article passing the criteria. This gives us some confidence that we could not have missed many articles in our initial search. Our inclusion and exclusion criteria and our quality assessment can also be a threat to validity as we were searching for actual industrial cases where the authors mention certain key information that we as researchers see as empirical evidence, such as teams, project success, MDD and Agile practices among others.

There are many other papers that discusses how and why you should combine the two development methods but they lack these key information, thus failing our identification of relevant papers. The papers not passing our criteria might have important information regarding agile MDD, but could not be included in our data extraction and synthesis. Also our papers included for synthesis might fail to deliver high quality. They might pass our criteria but fail to deliver detailed information in the passed criteria, thus making it difficult for us to be confident if it should be included or not. Same thing goes for documents not passing our criteria; it still might have valuable information regarding our study, but is not included.

Our time-limit and volume of the paper is also a threat to validity, as this project had a time-limit of around two months with limited resources, there are possibilities that we might have missed out important information. Because of this it was also not possible to follow every step of Kitchenham's guidelines on performing an SLR. The biggest issue when it comes to validity threat is the Agile MDD subject itself, finding only 8 documented papers where a team/project performed the combination process, with some documents mention vague information about it, here there is big gap in literature and huge space for further research

VIII. CONCLUSIONS

As Ambler indicates that *"The use of Agile methodology in model-driven development is not prevalent yet, except tailored Agile approaches, such as Agile model driven developmen*t" [4], the practical experiences of Model-Driven Agile Development are still uncommon. Reza Matinnejad also indicates "AMDD is a promising research context and a great practical concept, but it is not mature enough and is still in its infancy" [23].

In this paper we have presented the results of a systematic literature review about empirical evidence on Model-Driven Agile Development. In the eight papers which all passed our quality criteria, the authors wanted to combine the Agile methods and MDD in a way that it could benefit out of both worlds and at the same time avoid their shortcomings. In the result section, we have presented different integrations of Agile practices and MDD for different purposes. Due to this, Reza Matinnejad gives us a proper description of AMDD, *"AMDD can best be described, in our opinion, as an intelligent compromise"* [23]. However, due to the available reference materials (empirical experiences) are rare, challenges were accompanied with their development process. There are multiple authors discussing different theoretical approaches for MAD but just a handful papers describing detailed information how a company or a team implemented such a practice into their development process. Most of the eight papers that passed our quality criteria also fails to deliver comprehensive information how they performed MAD. They barely mention their team-setup, what practices they combined and what tools they used, just in a few papers this is properly described. Discovering that the area around MAD is immature gives plenty of space for future research, especially when it comes to detailed experience

reports, as agile MDD is a relatively new in the software development industry. We aim to extend this study by including more databases, and to provide more comprehensive empirical evidences, digging deeper in the low level of MAD.

IX. ACKNOWLEDGEMENT

We thank our colleagues at the Department of Computer Science and Engineering, especially our supervisor Håkan Burden.

X. REFERENCES

[1] Schach, S.R, "Object-Oriented Classical Software Engineering", 7th edn. McGraw-Hill (2007).
[2] Kitchenham, B. & Charters, S. (2007)," Guidelines for performing systematic literature reviews in
software engineering": Technical Report EBSE-2007-01, School of Computer Science and Mathematics, Keele University, 2007.
[3] T. Mens and P. V. Gorp, "A Taxonomy of Model Transformation," Electronic Notes in Theoretical Computer Science, vol. 152, pp. 125–142, March 2006.
[4] Ambler, S.W, "Agile Model Driven Development": AMDD (2007),
[5] J. Highsmith and M. Fowler, "The agile manifesto,": Software Development Magazine, vol. 9, no. 8, pp. 29–30, 2001.
[6] R. Gomes, G. Hoyos-Rivera, R. Willrich, C. Lima, and J.-P. Courtiat, "A loosely coupled integration environment for collaborative applications": IEEE Trans. Syst., Man, Cybern. A, Syst., Humans, vol. 41, no. 5, pp. 905– 916, Sep. 2011
[7]S. Selmi, N. Kraiem, and H. Ghezala, "Toward a Comprehensive View of Web Engineering": ICWE 2005, LNCS 3579, pp. 19-29, 2005.
[8] P. Abrahamsson, M. A. Babar, and P. Kruchten, "Agility and architecture: Can they coexist?" IEEE Software, 27:16–22, 2010.
[9] Y. Ghanam and F. Maurer, "An iterative model for agile product line engineering": The SPLC Doctoral Sympo- sium, 2008.
[10] C.W.Krueger, "Biglever software gears and the 3-tiered spl methodology": Companion to the 22nd ACM SIGPLAN conference on Object-oriented programming systems and applications companion, OOPSLA '07, pages 844–845, New York, NY, USA, 2007.
[11] Beecham, S, Bowes, D, Gray, D, Counsell, S, "A Systematic Literature Review on Fault Prediction Performance in Software Engineering": Dept. of Inf. Syst. & Comput., Brunel Univ., Uxbridge, UK, 2012.
[12] Jessica Díaz, Jennifer Pérez, Pedro P. Alarcón and Juan Garbajosa, "Agile Product Line Engineering": Wiley Online Library, 2011.
[13] S. J. Mellor and M. Balcer, "Executable UML: A Foundation for Model-Driven Architectures": Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2002.
[14] C. Raistrick, P. Francis, J. Wright, C. Carter, and I. Wilkie, "Model Driven Architecture with Executable UMLTM": New York, NY, USA: Cambridge University Press, 2004.
[15] S. J. Mellor, S. Kendall, A. Uhl, and D. Weise, "MDA Distilled": Redwood City, CA, USA: Addison Wesley Longman Publishing Co., Inc., 2004.
[16] J. Highsmith and A. Cockburn, "Agile Software Development: The Business of Innovation": Computer, vol. 34, no. 9, pp. 120–122, 2001.
[17] H. Takeuchi and I. Nonaka, "The New Product Development Game,": Harvard Business Review, January-February 1986.
[18] K. Schwaber and M. Beedle, "Agile Software Development with Scrum": 1st ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2001.
[19] B. Rumpe, "Agile Modeling with the UML," in Radical Innovations of Software and Systems Engineering in the Future, ser. Lecture Notes in Computer Science, M. Wirsing, A. Knapp, and S. Balsamo, Eds. Springer Berlin Heidelberg, vol. 2941, pp. 297–309, 2004.
[20] T. Stahl and M. Volter, "Model-Driven Software Development: Technology, Engineering, Management": John Wiley & Sons Inc., 2005.
[21] W. El Kaim, P. Studer, and P.-A. Muller, "Model Driven Architecture for Agile Web Information System Engineering": 2003.
[22] Kleppe, J. Warmer, and W. Bast, "MDA Explained: The Model Driven Architecture ™": Practice and Promise. Addison-Wesley Professional, 2005.
[23] Reza Matinnejad, "Agile Model Driven Development: An Intelligent Compromise": Ninth International Conference on Software Engineering Research, Management and Applications, 2011.

Identified papers

[P1] Yuefeng Zhang, Shailesh Patel, "Agile Model-Driven Development in Practice": Software IEEE, 2011.
[P2] Rainer Burkhardt, Volker Gruhn, "Agile Software Engineering: A New System for an Expanding Business Model at SCHUFA": Object-Oriented and Internet-Based Technologies, 2004.
[P3] Vinay Kulkarni, Souvik Barat, Uday Ramteerthkar, "Early Experience with Agile Methodology in a Model-Driven Approach": Model Driven Engineering Languages and Systems, 2011.
[P4] Ramos,A.l, Ferreira, J.V, Barcelo,J, "LITHE: An Agile Methodology for Human-Centric Model-Based Systems Engineering": Systems, Man, and Cybernetics Society, IEEE, 2013.
[P5] W.Lee, S.Park, K.Lee, C.Lee, B.Lee, W.Jung, T.Kim, H.Kim, C.Wu, "Agile development of Web application by supporting process execution and extended UML model": Software Engineering Conference. APSEC '05. 12th Asia-Pacific, 2005.
[P6] Cardoso, N, Rodrigues, P, Ribeiro, O, Cabral,J, Monteiro, J, Tavares, A, "An agile software product line model-driven design environment for video surveillance systems": IEEE 17th Conference on Emerging Technologies & Factory Automation (ETFA), 2012.
[P7] Yen-Chieh Huang, Chih-Ping Chu, "Legacy System User Interface Reengineering Based on the Agile Model Driven Approach": Recent Advances in Computer Science and Information Engineering, 2013.
[P8] Djanatliev, A. Dulz, W. German, R. Schneider, V, "Veritas - a versatile modeling environment for test-driven agile simulation": Simulation Conference (WSC), Proceedings of the 2011 Winter, IEEE, 2011.
[P9] Pohjalainen, P, Bottom-up Modeling for a Software Product Line: "An Experience Report on Agile Modeling of Governmental Mobile Networks": Software Product Line Conference (SPLC), 15th International, 2011.
[P10] Simon Urli, Mireille Blay-Fornarino, Philippe Collet, Sébastien Mosser, "Using composite feature models to support agile software product line evolution": Proceedings of the 6th International Workshop on Models and Evolution, 2012.
[P11] J.Grigera, J.Matías Rivero, E.Robles Luna, F.Giacosa, G.Rossi, "From requirements to web applications in an agile model-driven approach": 12th International Conference, ICWE, Berlin, Germany, July 23-27, 2012.
[P12] Xufeng (Danny), Liang Ioaki (Makis) Marmaridis, Athula Ginige, "Facilitating Agile Model Driven Development and End-User Development for Evolving Web-based Workflow Applications": ICEBE '07 Proceedings of the IEEE International Conference on e-Business Engineering, 2007.
[P13] James T. Sawyer, David M. Brann, "How to test your models more effectively: applying agile and automated techniques to simulation testing": WSC '09 Winter Simulation Conference, 2009.
[P14] Zhang, Y, "Test-Driven Modeling for Model-Driven Development": Software, IEEE, 2004.

XI. APPENDIX

Paper Nr	*Title*	*Author*	*Venue*
Paper 1	*Agile Model-Driven Development in Practice*	*Yuefeng Zhang, Shailesh Patel*	*Journal, Software IEEE, 2011.*
Paper 2	*Agile Software Engineering: A New System for an Expanding Business Model at SCHUFA*	*Rainer Burkhardt, Volker Gruhn*	*Chapter of the book, Object-Oriented and Internet-Based Technologies, 2004.*
Paper 3	*Early Experience with Agile Methodology in a Model-Driven Approach*	*Vinay Kulkarni, Souvik Barat, Uday Ramteerthkar*	*Chapter of the book, Model Driven Engineering Languages and Systems,2011.*
Paper 4	*LITHE: An Agile Methodology for Human-Centric Model-Based Systems Engineering*	*Ramos,A.I, Ferreira, J.V, Barcelo,J.*	*Journal, Systems, Man, and Cybernetics Society, IEEE, 2013.*
Paper 5	*Agile development of Web application by supporting process execution and extended UML model*	*Wookjin Lee, Sanghyun Park, Keeyoull Lee, Chunwoo Lee, Byungjeong Lee, Woosung Jung, Taeksu Kim, Heechern Kim, Chisu Wu*	*Software Engineering Conference, 2005. APSEC '05. 12th Asia-Pacific*
Paper 6	*An agile software product line model-driven design environment for video surveillance systems*	*Cardoso, N, Rodrigues, P, Ribeiro, O, Cabral,J, Monteiro, J, Tavares, A.*	*2012 IEEE 17th Conference on Emerging Technologies & Factory Automation (ETFA)*
Paper 7	*Legacy System User Interface Reengineering Based on the Agile Model Driven Approach*	*Yen-Chieh Huang, Chih-Ping Chu*	*Chapter of the book, Recent Advances in Computer Science and Information Engineering, 2013.*
Paper 8	*Veritas - a versatile modeling environment for test-driven agile simulation*	*Djanatliev, A. Dulz, W. ; German, R. Schneider, V.*	*Simulation Conference (WSC), Proceedings of the 2011 Winter, IEEE*
Paper 9	*Bottom-up Modeling for a Software Product Line: An Experience Report on Agile Modeling of Governmental Mobile Networks*	*Pohjalainen, P.*	*Software Product Line Conference (SPLC), 2011 15th International*
Paper 10	*Using composite feature models to support agile software product line evolution*	*Simon Urli, Mireille Blay-Fornarino, Philippe Collet, Sébastien Mosser*	*Proceedings of the 6th International Workshop on Models and Evolution, 2012.*
Paper 11	*From requirements to web applications in an agile model-driven approach*	*Julián Grigera, José Matías Rivero, Esteban Robles Luna, Franco Giacosa, Gustavo Rossi*	*12th International Conference, ICWE 2012, Berlin, Germany, July 23-27, 2012.*
Paper 12	*Facilitating Agile Model Driven Development and End-User Development for Evolving Web-based Workflow Applications*	*Xufeng (Danny), Liang Ioaki (Makis) Marmaridis, Athula Ginige*	*ICEBE '07 Proceedings of the IEEE International Conference on e-Business Engineering, 2007.*
Paper 13	*How to test your models more effectively: applying agile and automated techniques to simulation testing*	*James T. Sawyer, David M. Brann*	*WSC '09 Winter Simulation Conference, 2009.*
Paper 14	*Test-Driven Modeling for Model-Driven Development*	*Zhang, Y.*	*Journal, Software, 2004.*

Meta-Table for Primary studies