Model-Driven Development and Formal Methods: A Literature Review

Valdemar Vicente Graciano Neto[1,2]

[1]Instituto de Informática – Universidade Federal de Goiás (UFG)
Alameda Palmeiras, Quadra D, Câmpus Samambaia Caixa Postal 131 -
CEP 74001-970 - Goiânia - GO - Brazil

[2]Laboratório de Engenharia de Software (LabES)
Instituto de Ciências Matemáticas e Computação (ICMC)
Universidade de São Paulo - USP
Avenida Trabalhador São-carlense, 400 - Centro.
CEP: 13566-590 - São Carlos - SP - Brazil

valdemarneto@inf.ufg.br

Abstract. Background*: Model-Driven Development (MDD) has increased. However, MDD still has some lacks that becomes hard the entire adoption of the paradigm. On the other hand, Formal Methods (FM) has been applied in a lot of areas inside Software Engineering once they are mathematically based techniques for the specification, analysis and development of software systems, increasing software quality.*
Aim/Research Question*: This paper aims to answer this research question: How and how much MDD and FM have been associated in research? Answering that delivered a diagnostic report about this research field status.*
Method*: A literature review was conducted following Mapping Studies principles.*
Conclusions*: This research field has increased along the years, what shows promising results and cross-fertilizing benefits when these research fields are associated.*
Contribution*: An indication of research direction for Software Engineering area in the present and next years.*

1. Introduction

Model-Driven Development (MDD) has increased. It consists in a new prescriptive model of software development process [Pressman 2010, Graciano Neto and de Oliveira 2013] where models are the first class citizens [Sendall and Kozaczynski 2003, Amrani et al. 2012].

However, MDD still has some lacks that becomes hard the entire adoption of the paradigm to generate software in industry using exclusively this fashion, for instance:

1. the difficulties to automatically generate the entire software (a part is still manually generated),
2. the problems to validate model transformations once they are code as other programs and suffer from the same classic problems of Verification and Validation activities,

3. techniques and tools to check the models and assure their conformance against their respective metamodels, and so on.

On the other hand, Formal Methods has been applied in a lot of areas inside Software Engineering, specially in the critical systems area as avionics, health and militar, where errors can not be found.

Formal Methods are welcome once they are mathematically based techniques for the specification, analysis and development of software systems. Correctness of software systems can be improved by formalising different products and processes in the life-cycle, enabling rigorous analysis of the system properties [Oquendo 2006].

Thus this paper presents a literature review conducted following Mapping Study principles [Petersen et al. 2008, Kitchenham et al. 2010] to evaluate how Formal Methods could be associated to MDD. This could be a way to benefit and make it possible to construct and assure MDD artifacts quality and to become the use of Formal Methods more abstract, high-level, and human-readable.

The remainder of this paper presents some background in section 2, the research methodology in section 3, results in section 4, and some conclusions and future work in section 5.

2. Background

Several models are used to express the concepts of a knowledge domain for which a software is built. There are specific models for each phase of the software development process. The requirements model is used as input for discussion and production of architecture/design models. These models are considered for structuring the source code and the source code is the input for test cases specification. In fact, software development process is a succession of models transformations[Graciano Neto et al. 2010].

Model-Driven Development (MDD) is a software production process based on MDA (Model-Driven Architecture), a model-centered approach, where models, metamodels, and plugabble transformations are stored in *cartridges* that should be changed against a model transformer to produce many different kinds of software products in many technology flavors from a same source model.

MDE appeared after as an evolution of MDA and MDD to denote an complete process composed by methods, tools, technologies and principles that should be followed in the entire software development process to produce software as a sequence of model transformations.

MDE followed the natural evolution of any technology. It emerged as a coding activity in the software development life cycle. However, the principles started to be migrated by the software development team for the earliest moments of the software development life cycle as requirements engineering, software architecture definition and so on [Junior and Winck 2006], originating a totally new software development life cycle, as happened with aspects, agents, components, and tests, creating paradigms as Aspect Oriented Software Development/Engineering, Agent-Oriented Software Engineering and so on.

For simplicity, this paper uses MDD as the acronym to denote every sort of model-driven technologies and acronyms.

MDD lives a deadlock. Tools and promising technologies as Eclipse Modeling Framework[1] and Kermeta [Manset et al. 2006] have been developed to support the MDD process. However, some studies have reported that MDD is not still mature [Westfechtel 2010] once some difficulties has been faced as:

1. The total automation is still a legend in most cases. Manual code addition is still necessary in a lot of model transformations;
2. Models are not enoughly expressive to cover every aspect of a software representation;
3. Model Transformations code is big and its production is still error-prone [Gonzalez and Cabot 2012]. Validation relies on the traditional software development techniques and the lack of validation techniques of models and model transformations are critical barriers to a wide industrial adoption [Baudry et al. 2010];
4. There is no guarantees that the software produced has quality and it is the expected result comparing it to one that could be manually produced.

The fact is that the representing software as models is an excelent idea, but there is no guarantees about its correctness, conformance, quality or rigor.

Formal Methods (FM) and *Model-Driven* do not often appear in the same sentence. Really, in a first moment we can see these words as representing disjoint areas.

However, if we reflect over this subject, we can envision that FM could strongly benefit MDD in some problems highlighted above as:

1. models could be **formally specified**, that is, models, metamodels and transformations could be specified using some formal method. This makes it possible to enforce and restrict the model expressiveness (through some formal lexical, syntatical and semantical value), becoming this totally measurable, allowing increasing or reducing the models according to the projects necessities;
2. models could be **formally verified**: again the models, metamodels and transformations specified in a formal way could be, either, formally verified. Additionally, this verification could be automatic once there are already tools and languages that do this kind of verification.

Additionally, the inverse situation can happen as well.

Formal methods allow a software engineer to create a specification that is more complete, consistent, and unambiguous than those produced using conventional or object-oriented methods. Set theory and logic notation are used to create a clear statement of facts (requirements). This mathematical specification can then be analyzed to prove correctness and consistency. Because the specification is created using mathematical notation, it is inherently less ambiguous that informal modes of representation [Pressman 2014].

FM have not been widely adopted in Software Engineering Industry (except Critical Systems, and Critical Embedded Systems) because [Pressman 2010]:

[1]http://www.eclipse.org/modeling/emf/

1. few software developers have the necessary background to apply formal methods, extensive training is required;
2. it is difficult to use the models as a communication mechanism for technically unsophisticated customers;
3. the development of formal models is currently quite time consuming and expensive.

But, considering MDD encompasses as least the high level and abstract software representation as models and software producing automation, MDD could help FM to increase level of abstraction in Formal Methods adoption, decreasing training time; to provide communication alternatives for customer and other stakeholders, and to automate the production of formal models synthesis from abstract formal models, minimizing time and cost.

Next section presents the research methodology we followed to investigate how these methods (FM and MDD) have been associated and reported in the software engineering literature.

3. Research Methodology

This paper aims to answer this research question: ***How and how much MDD and FM have been associated in research?*** Answering that delivered a diagnostic report about this research field status.

To investigate this research topic, a literature review was conducted following Mapping Studies principles [Petersen et al. 2008, Kitchenham et al. 2010]. This is an scientific investigation approach classified as an Exploratory Study once it has not the responsibility to compare any tools or methods, but just investigate and present a broad vision about some subject [Petersen et al. 2008].

A search was conducted using the string *Model-Driven Development and Formal Methods* in the ACM Digital Library[2] and Google Scholar[3]. Other scientific bases were not considered.

Two selection criteria were chosen: 1) Title Reading, and 2) Abstract Reading. Just papers and articles were considered in this research. After this step, the appropriate ones were included to a subsequent paper entire reading, and the inappropriate were discarded.

Once the search string was so broad, the amount of recovered results had been huge. Research validity discussions are done in a following section. Next section presents the results.

4. Results

Following the research methodology cited above, twenty-seven (27) papers were analysed. From those papers, eighteen (18) were considered appropriate and included while nine (9) were discarded because they were not treating the expected theme.

[2]http://dl.acm.org/

[3]http://scholar.google.com

Next subsections bring a quantitative analysis covering just numeric data, a qualitative analysis that discusses how MDD and MF have been associated in software engineering literature, and a discussion about the research validity.

4.1. Quantitative Analysis

Two aspects were observed during the data collecting: the kind of formal techniques focused on the papers, and the popularity of themes along the years.

Figure 1 presents the subject focused by the selected papers regarding to Formal Methods. Six focused just on Formal Verification. These papers did not focused their discussion on Formal Specification. Eight of them focused on Formal Specification and did not consider the formal validation of these formal models. Four of them reported both activities.

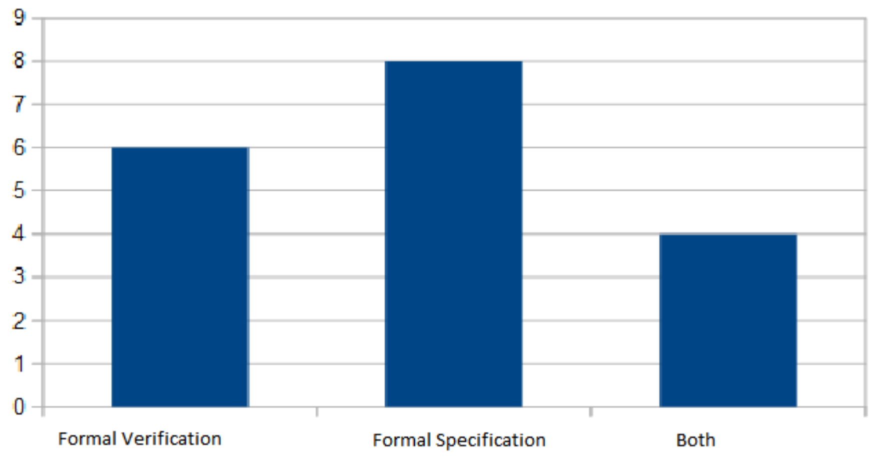

Figure 1. Research interest focused in papers.

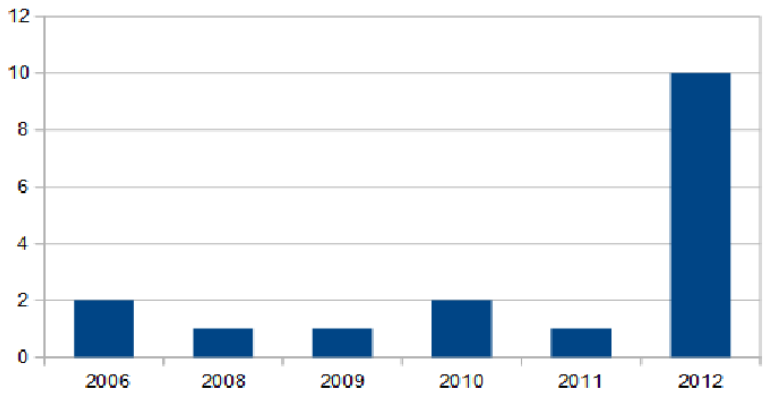

Figure 2. Popularity of the research topic in years.

Figure 2 analyses the popularity of this topic. It is evident that considering the amount of publications per year and the period covered by the selected results, the year 2012 has experienced a boom in this research topic, increasing in ten times the interest considerind with the preceding year (2011).

The only one paper written in 2013 recovered by the search was discarded. The research has been performed during 2013 November.

4.2. Qualitative Analysis

Between the works analysed, we found most of papers dealing with the association between FM and MDD as expected: some of them presented solutions to *formally specify* MDD artifacts, while others presented solutions to *formally verify* MDD artifacts.

Between the formalisms used we can highlight B, Z, Z3, formal diagrams as Statecharts, Petri Nets and UML extensions; EMF Modeling Operations (EMO), OCL, NuSMV, and so on.

About Model Transformations and FM, Three approaches have been used in the verification of Model Transformations [Lano et al. 2012] :

1. transformation code verifying (syntathic analysis);
2. mapping from transformation to a formalism in which the semantic analysis can be performed;
3. specifying transformations in a formalism for proof and implementation synthesis as a proof by construction.

Not so much papers from 2013 were recovered. One possible interpretation is that once the research has been performed in 2013, some conferences have been not indexed yet by the Scientific Search Engines while the research was being performed. However, the tendence of the results shows that 2013 could have even more papers dedicated to the highlighted research topic if the statistical increase keeps ascending.

No one result was found about how MDD could be used to increase formal methods level of abstraction to face those features that becomes the Formal Methods broad adoption in software engineering industry a hard task, except for those domains where it is imperative, as aeronautics, critical embedded systems, critical medical systems, military, and others.

This is a valuable contribution of this research, once it opens a new direction of research inside the conjunction of Formal Methods and MDD, introducing a new thread inside a research topic that can be considered, after these results, a hot topic and buzz words in Software Engineering research.

4.3. Research Credibility Discussion

This research was conducted under a strict available time, into the context of a PhD course in Formal Software Specification. This is the reason because not all the papers recovered were considered for the research.

Considering the nature of the study (exploratory) and the high-level and abstractedness of the search string, it is possible to suppose that the credibility of the study was not affected, once there was not an intention to be rigorous about the Scientific Search Bases amount, the papers number coverage or a strict comparing between mappings as it happens in Systematic Reviews.

The intention of this research is to give a panorama of how these research fields have been associated along the years, demonstrating a research tendence and an increasing interest.

The slice of papers cutted from the recovered results was specific. Considering the papers more aligned with the search string are recovered in the first positions of result in Information Retrieval Systems[Graciano Neto and Ambrosio 2009, Graciano Neto et al. 2009], the first ones were taken as more representative for the research question. So, with some percentual and statistic variation, we believe the first results can be considered the more relevant results for this research question.

5. Conclusions and Future Work

This paper presented results of an exploratory study conducted to answer the following research question: *How and how much Model-Driven Development (MDD) and Formal Methods (FM) have been associated in research*?

Answering that delivered a diagnostic report about this research field status, giving a panorama of how these research fields have been associated along the years, demonstrating a research tendence.

The main contribution of the paper is the opening of a new direction of research inside the conjunction of FM and MDD: the use of MDD to increase formal methods level of abstraction to face FM low adoption in Software Engineering industry. This can be glimpsed as a bew research direction for Software Engineering area in the present and next years.

It is possible to conclude that this research field has increased along the years, what shows promising results and cross-fertilizing benefits when these investigation areas are associated.

As future work, a more strict exploration could be performed over this theme, establishing a more restrict protocol and considering a larger slice of recovered papers, amplifying the research coverage. Furthermore, the new research direction identified can be explored, proposing new methods, models and techniques to deliver some building blocks to edify the basis for a more tangent, human, high-level and touchable formal methods process.

References

Amrani, M., Lucio, L., Selim, G., Combemale, B., Dingel, J., Vangheluwe, H., Traon, Y. L., and Cordy, J. R. (2012). A tridimensional approach for studying the formal verification of model transformations. *Software Testing, Verification, and Validation, 2008 International Conference on*, 0:921–928.

Baudry, B., Bazex, P., Dalbin, J.-C., Dhaussy, P., Dubois, H., Percebois, C., Poupart, E., and Sabatier, L. (2010). Trust in mde components: The domino experiment. In *Proceedings of the International Workshop on Security and Dependability for Resource Constrained Embedded Systems*, S&#38;D4RCES '10, pages 2:1–2:7, New York, NY, USA. ACM.

Gonzalez, C. and Cabot, J. (2012). Atltest: A white-box test generation approach for atl transformations. In France, R., Kazmeier, J., Breu, R., and Atkinson, C., editors, *Model Driven Engineering Languages and Systems*, volume 7590 of *Lecture Notes in Computer Science*, pages 449–464. Springer Berlin Heidelberg.

Graciano Neto, V. V. and Ambrosio, A. P. L. (2009). A WordNet-based Search Engine for the Moodle Learning Environment. In *Proceedings of the International Conference EDUTEC*, EDUTEC '09, pages 1–10.

Graciano Neto, V. V., Ambrosio, A. P. L., and e Silva, L. O. (2009). Automatic Retrieval of Complementary Learning Material for Slide Presentations. In *Proceedings of the International Conference on Interactive Computer Aided Blended Learning*, ICBL '09, pages 1–8.

Graciano Neto, V. V., da Costa, S. L., and de Oliveira, J. L. (2010). Lessons Learned about Model-Driven Development after a Tool Refactoring (In portuguese). In *Proceedings of the VIII Annual Meeting of Computing*, ENACOMP '10, pages 68–75.

Graciano Neto, V. V. and de Oliveira, J. L. (2013). Evolution of an Application Framework Architecture for Information Systems with Model Driven Development(In Portuguese). In *Proceedings of the IX Brazilian Symposium on Information Systems*, SBSI'13, pages 1–12.

Junior, V. G. and Winck, D. V. (2006). *AspectJ - Aspect Oriented Programming with Java (In portuguese)*. Novatec, Sao Paulo, Brazil, 1 edition.

Kitchenham, B., Brereton, P., and Budgen, D. (2010). The educational value of mapping studies of software engineering literature. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 1, ICSE 2010, Cape Town, South Africa,*, pages 589–598.

Lano, K., Kolahdouz-Rahimi, S., and Clark, T. (2012). Comparing verification techniques for model transformations. In *Proceedings of the Workshop on Model-Driven Engineering, Verification and Validation*, MoDeVVa '12, pages 23–28, New York, NY, USA. ACM.

Manset, D., Verjus, H., Mcclatchey, R., and Oquendo, F. (2006). A formal architecture-centric, model-driven approach for the automatic generation of grid applications. In *Proc. of the 8th Int. Conf. on Enterprise Inform. Systems.*

Oquendo, F. (2006). Pi-methodd: A model-driven formal method for architecture-centric software engineering. *SIGSOFT Softw. Eng. Notes*, 31(3):1–13.

Petersen, K., Feldt, R., Mujtaba, S., and Mattsson, M. (2008). Systematic mapping studies in software engineering. In *Proc. of ICEASE*, EASE'08, pages 68–77, Swinton, UK, UK. British Computer Society.

Pressman, R. (2010). *Software Engineering: A Practitioner's Approach*. McGraw-Hill, Inc., New York, NY, USA, 7 edition.

Pressman, R. S. (2014). Available in: http://www.rspa.com/spi/formal-methods.html. Access: February 2014.

Sendall, S. and Kozaczynski, W. (2003). Model transformation: The heart and soul of model-driven software development. *IEEE Software*, 20(5):42–45.

Westfechtel, B. (2010). A formal approach to three-way merging of emf models. In *Proceedings of the 1st International Workshop on Model Comparison in Practice*, IWMCP '10, pages 31–41, New York, NY, USA. ACM.