

Developing Model Transformations: A Systematic Literature Review

Ana Patrícia Magalhães^{1,3}^a, Rita Suzana P. Maciel²^b and Aline Andrade²^c

¹State University of Bahia, Department of Exact Sciences and Earth, 2555 Silveira Martins St. Cabula, Bahia, Brazil

²Federal University of Bahia, Computer Science Department, Bahia, Brazil

³Salvador University, Post Graduate Program in Computing and Systems, Bahia, Brazil

Keywords: Model Transformation, Development Strategies, Development Process, Modeling Language.

Abstract: Model Driven Development is an approach that makes use of models instead of code in software development. At its core, there is a transformation chain responsible for the (semi) automation of the development process converting models into models until code. The development of transformations has been a challenge as there is an inherent complexity of the transformation domain in addition to the complexity of the software being constructed using these transformations. In order to assist this development as well as improve transformation quality, it is important to adopt software engineering facilities such as processes, languages and other techniques. This paper presents a systematic literature review of strategies currently proposed to develop model transformations. We aim to investigate development processes or any other strategies used to guide transformation development, the phases of software development life cycle considered, modeling languages adopted for specification and also the level of automation provided. The study selected and analyzed 23 papers to identify which aspects are addressed by research and any gaps in this area. We identified four different strategies in guiding transformation development and perceived the lack of a modeling language standard.


1 INTRODUCTION


Model Driven Development (MDD) (Mellor, 2004) is a software development paradigm that makes intensive use of models to represent systems at different levels of abstraction. At the core of MDD is a transformation chain which converts models into other models (model transformations - MT) and model into text (program transformation) in order to generate application code (Stahl, 2010). In this study we are initially interested in MT development.


The specification of a MT is defined between metamodels of source and target languages, which define application domains (Brambilla, 2012). Any models that are instances of the metamodels can be processed by the transformation. In general, developers are used to manipulating models for software documentation, but they are not used to working with metamodels, which require

specification at higher abstraction levels than in models. Moreover, the environments, for both metamodel implementation and model specification according to the metamodel, should be adopted to support MT development. Additionally, model transformations are usually written in specific languages, e.g. ATL (Atlas Transformation Language), which are appropriate for model manipulation, instead of coded in common program languages such as C and Java. Therefore, the development of MT involves expertise in new languages, the adoption of development environments customized for these languages and engines to execute the transformations.

In summary, model transformation development involves elements that are not common in traditional software development and this increases its complexity, such as domain specific modeling languages, and specific programming languages as well as comprising metamodel definition and manipulation. In addition, software, in general, has become increasingly complex due to the need to work

^a <https://orcid.org/0000-0002-8608-4553>

^b <https://orcid.org/0000-0003-3159-6065>

^c <https://orcid.org/0000-0002-9926-9303>

with wide domains, integration with other software, among others, which may also reflect in the size and complexity of the MT used in its construction. Therefore, the development of MT requires software engineering facilities, for example, development processes, modeling languages, and automation to achieve quality and improve maintenance and evolution (Guerra, 2010), (Bollati, 2013).

Initially, MT were usually specified in natural language and implemented directly in code (Guerra, 2010), (Bollati, 2013). As a consequence, this hindered the adoption of best practices in software engineering, such as reuse, which might compromise its evolution and maintenance. This scenario is changing and some proposals consider other phases of the software development life cycle, instead of only codification. For example, there are works that focus on design and implementation phases (Del Fabro, 2009), (Bollati, 2013), (Tavac, 2013), while others concern formal specification (Sani, 2011).

As there is still no consensus about which methods and techniques are more appropriate, it is necessary to understand what strategies are being used to develop MT in order to obtain a better understanding and comprehension of current trends in this area. In this direction, this paper provides a Systematic Literature Review (SLR) of the strategies used to guide MT development, i.e. analysis, design and implementation of MT. SLRs aim to aggregate knowledge about a software engineering topic or research technique using a rigorously methodology review of resource results. They are important to collect evidence on a particular topic of interest and support practitioners in developing appropriate solutions for specific context (Kitchenham, 2004). This definition and the protocol of Kitchenham are adopted in our work. Therefore, our review aims to know how the community is developing MT, i.e. identify which methods have been used and collect evidence of its use. The SLR covers the period from 2003 to 2019, including research databases of most publications related to MDD. We selected and analyzed 23 works and significant results were obtained, which are important for both developers and researchers in need of support in developing transformation chains e.g. this SLR can help to make easier the identification of the most used strategies when a software architect is proposing a new model transformation approach.

The text is organized as follows: Section 2 introduces some background about MDD; Section 3, 4 and 5 respectively present the research method used to guide this SLR, its execution and the results obtained; Section 6 analyzes related works about MT

development. Section 7, discusses the results of the review identifying some current gaps in MT development; Section 8 discuss some threats to our review; and, Section 9 presents our conclusions.

2 BACKGROUND

In MDD, models at high abstraction levels are converted, through a transformation chain, into models at a low abstraction level until the application code is generated. MDD comprises two main elements: models, which are the artifacts that represent application at different abstraction levels and transformations, the software responsible for the automation of the development (Brambilla, 2012).

Models are abstract representations of a system, which comprise the structure and behavior (Mellor, 2004). MDD models are not mere documentation of software, they are the artifacts used as input of the transformations to generate other models or code. Thus, they must be defined in modeling languages with a well-defined syntax and semantic. MDD usually uses Domain Specific Modeling Languages (DSML) for model specification. In MDD, the abstract syntax and the static semantics of a modeling language are expressed in metamodels. Thus, models are designed conforming to metamodels. UML, a general purpose language, can be customized to specific domains through the definition of UML profiles to be used as a modeling language in MDD.

In MDD, transformations can be classified according to the artifact produced as output from model transformation, when generate models as output; or program transformation when generates code as output. A transformation can be seen from two different viewpoints, as a function that maps models between domains or as a terminating algorithm that applies structural and/or semantic changes to models (Ma, 2016). This work focuses on the first viewpoint called relational transformation. Therefore, a MT comprises a set of rules that describes how models, instances of source metamodels, are converted into models, instances of target metamodels (Mens, 2006).

3 RESEARCH METHOD

The SLR reported in this paper follows the guidelines presented in (Kitchenham, 2004). The Start tool (Start, 2013) was used to support the process. The review was performed by three researchers, a Ph.d.

student, and her two supervisors. Extra information about this SLR can be accessed at <https://drive.google.com/drive/folders/1Vxi2j9-SfHYbt6YmlaGpTHjHgVP6ObrQ?usp=sharing>

3.1 Planning the Review

The goal of our study was specified according to the GQM template (Solingen, 2002), shown in Figure 1.

Analyze proposals of model transformations development
 From the perspective of identifying the strategies used in model transformation development
 With respect to phases of software development life cycle which were considered, modeling languages adopted and automatic resources provided
 From the perspective of the researchers
 In the context of current works

Figure 1: Goal of the SLR according to GQM template.

For this review, we formulated the following hypothesis and research questions (RQ).

Hypothesis 1: The development of model transformations is a trend and different strategies have been proposed in literature to support it. However, there is still no well-defined strategy that can be widely used by the community. We considered as a well-defined strategy the proposal that: (i) comprises the necessary elements, e.g. tasks and artifacts, well organized to be followed as a guide, such as development processes or tools; and (ii) is stable enough, i.e. has been extensively tested, to be widely used. This hypothesis motivated us to construct the following research questions: **RQ01: What are the current strategies used to guide the development of model transformations?** In this question, we want to identify the kinds of methods that have been used in model transformation development. **RQ02: Which phases of software development life cycle have been considered in model transformations development?** When adopting a development strategy it is important to consider the level of coverage of it concerning software development life cycle, e.g. analysis, design, implementation. **RQ03: How automated is the proposed strategy?** Automation is an important issue in any kind of software development because it can promote better productivity as well as makes the use of the approach easier. We are also interested in strategies that generate the code of the MT. **RQ04: Which validated methods have been used to evaluate the current proposals?** We aim to evaluate the feasibility of the approach, investigating how the proposals were validated. **RQ05: How many examples of model transformations are tested in each validation?** We want to know if the strategy was sufficiently tested to be considered stable for use.

Hypothesis 2: In MT specification, there is no consensus on a notation to be adopted as a standard. During a software development life cycle there are some phases where the adoption of specific notations is necessary, such as in software analysis and design. Therefore, we wanted to map which modeling languages have been used in each phase of transformation development. This hypothesis led us to construct the following research question. **RQ06: Which languages/notations have been used for model transformations specification?** The goal of this research question is to identify if there is a modeling language that could be considered a standard for the model transformation domain.

To perform the review, we considered four important research databases of Software Engineering, ACM Digital Library, IEEE Library, Science Direct and Springer where we selected works using the following search string:

(MDE or MDD or MDA or Model Driven Development) and (model transformation) and (specification or development or approach or strategy or framework or systematic or process or methodology or method or life cycle)

This search string was applied to title, abstract and key words. Besides the search string, we analyzed the references of the selected papers and used the snowballing method to find other relevant works.

Our research included works written in English published between 2003 to 2019. The studies selected from the automatic search were refined manually according to the following inclusion/exclusion criteria: **Inclusion Criteria 1:** the study identifies and organizes the activities to develop MT. *Rationale:* the focus of our research is on the strategies to guide development; **Exclusion Criteria 1:** the study presents a strategy to develop program transformations. *Rationale:* program transformation involves different activities and programming languages from MT transformation; **Exclusion Criteria 2:** the study focuses on non-relational transformations. *Rationale:* these studies involve other approaches of development (e.g. graph transformations). We focus on relational-transformation; **Exclusion Criteria 3:** the study is about MT for a specific domain (e.g. transformations for embedded systems). *Rationale:* These studies do not focus on MT development, but support the development of software in specific domains. **Exclusion Criteria 4:** the study is about bidirectional MT. *Rationale:* we focus on unidirectional MT. They have different purposes. In order to evaluate the quality of the selected articles we defined five quality assessment criteria: Is there a proper introduction to

Table 1: Features defined to evaluate selected works.

Id	Questions	Answers	RQ
F1	What is the strategy used to guide development?	1 - development process 2 - tool 3 - systematic approach 4 - other (specify)	01
F2	Does the study consider requirements specification phase?	1 - no 2 - yes, informally described 3 - yes, described in detail 4 - not mentioned	02
F3	Does the study consider design phase independent of platform?	1 - no 2 - yes, informally described 3 - yes, described in detail 4 - not mentioned	02
F4	Does the study provide instructions to map requirements specification into design?	1 - no 2 - yes, but not automated 3 - yes, automated	04
F5	Does the study consider design phase for specific platform?	1 - no 2 - yes, informally described 3 - yes, described in detail 4 - not mentioned	02
F6	Does the study provide instructions to map models (design independent) into models (in specific platform)?	1 - no 2 - yes, but not automated 3 - yes, and automates it	04
F7	Does the study consider implementation phase?	1 - no 2 - yes 3 - not mentioned	02
F8	Does the study provide instructions for code generation in specific languages?	1 - no 2 - yes 3 - not mentioned	04
F9	Which are the methods used to validate/verify model transformations?	1 - test case 2 - formal method 3 - other 4 - not mentioned	02
F10	Are validation/verification activities automated?	1 - no 2 - yes 3 - not mentioned	02
F11	Does the study use MDD approach?	1 - no 2 - yes	01
F12	Which are the modeling language adopted?	1 - natural language 2 - UML/light extension of UML 3 - heavy extension of UML 4 - proprietary notation 5 - formal language 6 - not mentioned	02
F13	Does the study provide a tool?	1 - no 2 - yes, proprietary 3 - yes, open source 4 - not mentioned	04
F14	What are the methods used to validate the proposal feasibility?	1 - none 2 - example/proof of concept 3 - case study 4 - experiment 5 - other (specify)	05

the article? Does the introduction have a clear idea of the research objectives? Is there a clear statement of the results? Does it clearly describe the contributions? Is the strategy validated? All the selected works fulfil these criteria.

In order to help in the analysis of the articles we defined certain features, based on the development life cycle of model transformations (Table 1). The first and second columns present, respectively, the Id and a brief description of each feature. The third column shows what was observed in the study in order to analyze each feature and the last column identifies the desired research question that we wanted to answer with this feature. It is important to highlight that some research questions may be related to more than one feature.

4 CONDUCTING THE REVIEW

Table 2 shows the records returned according to each research database.

The first column shows the research database, and the others present the following results: number of records identified after the search string application in each database (column *Initial*); papers selected after scanning titles (column *Title*); papers selected after reading the abstract (column *Abstract*); papers

Table 2: Summary of the works.

Base	Initial	Title	Abstract	Introd	Final
ACM	270	46	18	8	5
IEEE	737	100	34	15	6
Science Direct	343	41	16	9	6
Springer	995	9	6	4	4
Others	-	-	-	-	2
Total	2345	196	74	36	23

selected after reading the introduction (column *Introd*); and papers whose full text was analyzed (column *Final*). The row *Others* refers to the papers added from other sources, after analyzing the references of the previously selected papers.

5 REPORTING THE REVIEW

This section presents the results of the SLR. We used *R Software* (RSoftware, 2019) to perform the statistical analysis and generate graphics.

Figure 2 shows a timescale with the papers selected after applying the inclusion / exclusion criteria of our protocol. As can be seen, the number of articles increased from 2003 until 2019. From 2005 (when the first paper was published) on, the number of articles fluctuated until 2016 when an increasing number of articles were published. This increase may

indicate the community effort to establish effective methods to aid the development of model transformation. After that, we could perceive a decrease in the number of works, which may indicate that the proposals are becoming more stable.

Table 3 lists the Selected Papers (SP) in chronological order with an id for each, their publication year, title and the database where we found them. These works were analyzed based on the features defined in Table 1 in order to confirm/refused the hypothesis formulated in Section 3, according to the research questions.

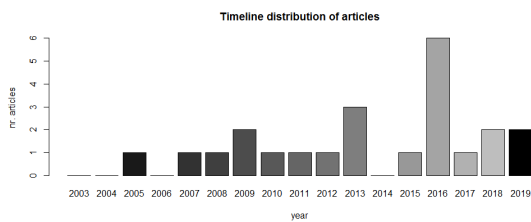


Figure 2: Distribution of articles along the years.

Concerning **RQ01: What are the current strategies used to guide the development of model transformations?** from the 23 studies considered relevant we identified four different strategies to

guide MT development: (i) systematic approach, which comprises a description of steps, written in natural language, to be followed by developers; (ii) algorithm, with steps described in natural language and organized using control structures such as conditionals and loops; (iii) software tools with an embedded methodology to drive development; and (iv) development process in a Process Modeling Language (PML) with the relevant elements of a process and their relationships (Figure 3).

As can be seen in Figure 3, systematic approaches are the most commonly used strategy for MT development (15 articles, 65%). In general, they were the first initiatives in systematizing the tasks related to MT development. Two works (9%) proposed algorithms to organize the activities of development in programming structures. Using these structures, algorithms reduce the level of ambiguity that exists in systematic approaches, although in these works they were not implemented and had to be followed manually. More recently, tools have been proposed (5 works, 22%) to improve MT development. In this case, activities to support development are encapsulated in proprietary environments. One work proposes a development process, specified in a PML, suitable for a MT domain.

Table 3: Selected Studies.

Id	Year	Title	Search Base
SP1	2005	A systematic approach to design model transformation (Kuster, 2005)	IBM
SP2	2007	Towards Model Transformation Generation By-Example (Wimmer, 2007)	IEEE
SP3	2008	Transformation have to be developed ReST assured (Siikarla, 2008)	Springer
SP4	2009	Model Transformation by Demonstration (Sun, 2009)	Springer
SP5	2009	Towards the efficient development of model transformations using model weaving and matching transformation (Del Fabro, 2009)	Springer
SP6	2010	Method of constructing model transformation rule based on reusable pattern (Li, 2010)	IEEE
SP7	2011	Model transformation specification for automated formal verification (Sani, 2011)	IEEE
SP8	2012	A model based development approach for model transformation (Kolahdouz-Rahimi, 2012)	ACM
SP9	2013	Engineering model transformation with transML (Guerra, 2013)	Springer
SP10	2013	Applying MDE to the (semi-) automatic development of model transformation (Bollati, 2013)	Science Direct
SP11	2013	The general algorithm for the MDA transformation models (Tavac, 2013)	IEEE
SP12	2015	Specifying model transformation by direct manipulation using concrete visual notation and interactive recommendation (Avazpour, 2015)	Science Direct
SP13	2016	Requirements engineering in model transformation development: a technique suitability framework for Model Transformation Applications (Tehrani, 2016)	ACM
SP14	2016	Multi-Step learning and adaptive search for learning complex model transformations from examples (Baki, 2016)	ACM
SP15	2016	Design pattern oriented development of model transformation (Ergin, 2016)	Science Direct
SP16	2016	Model-based M2M transformations based on drag-and-drop actions: Approach and implementation (Skersys, 2016)	Science Direct
SP17	2016	A Model-Driven approach for model transformation (Ma, 2016)	IEEE
SP18	2016	Designing and describing QVTo model transformation (Tikhonova, 2016)	Scopus
SP19	2017	Formal concept analysis for specification of model transformation (Berranla, 2017)	IEEE
SP20	2018	EVL-Strace: a novel bidirectional model transformation approach (Semimi-Dehkordi, 2018)	Science Direct
SP21	2018	A generic approach to model generation operation (Kleiner, 2018)	Science Direct
SP22	2019	Model driven transformation development (MDTD): An approach for developing model to model transformation (Magalhaes, 2019)	Science Direct
SP23	2019	Applying a Data-centric framework for Developing Model Transformations (Camargo, 2019)	ACM

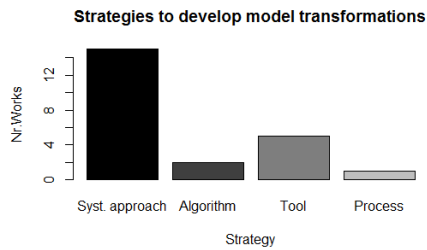


Figure 3: Strategies to develop model transformations.

As well as the approach used in development, we also identified the adoption of a wide number of techniques to support development, such as transformations by examples, transformation by demonstration, weaving models, patterns, formal methods, MDD, requirements engineering and mathematical methods as illustrated in Figure 4.

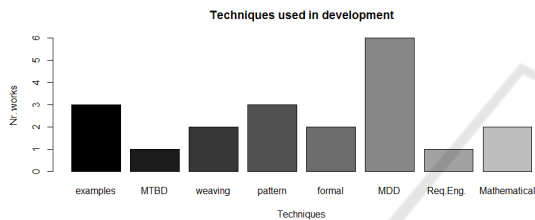


Figure 4: Techniques used in transformation development.

Related to **RQ02: Which phases of software development life cycle have been considered for model transformations development?** Following (Magalhaes, 2016) we considered that a MT development life cycle comprises at least four phases: Requirements, Design (and Design for a specific platform), Implementation and Validation/Verification. Figure 5 shows which of these phases were considered in the papers analyzed. Features F2, F3, F5, F7 and F9 were used to analyze data.

As shown in Figure 5, Requirements phase is considered in 43% of the proposals (10 works), but only 22% (not shown in the graph) of these works detail how to perform the tasks. The others only describe what might be specified. Most of the proposals focus on Design, regardless of and/or specific to a platform (78%) and Implementation phases (18 works, 78%). The Validation/Verification, a crucial phase for any software, is considered in 39% (9 works). The method used in Validation/Verification (not showed in this article) for those who considered this phase, 3% use test cases (3 works), 56% (5 works) apply formal methods, and 11% (1 work) use other techniques. Moreover, 40% of them (4 works) use automatic techniques to perform validation.

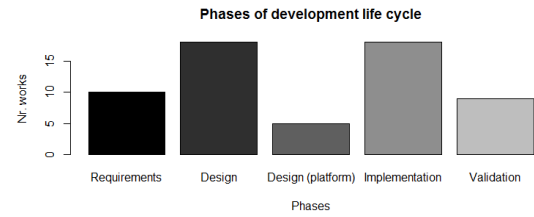


Figure 5: Phases of model transformation development life cycle.

To answer **RQ03: How automated is the proposed strategy?** we analyzed the level of automation between the phases of software development life cycle (features F4, F6 and F8), for example, supporting the map between requirements specification and design phases and automation in code generation. Four works provide automation through modeling phases, while 13 works support code generation. This shows that despite the change in focus promoted by MDD from code to model, transformation development still does not follow this same direction. However, according to the percentage of support for code generation, we can assume that this scenario is beginning to change. The level of abstraction is gradually increasing because some years ago transformations used to be developed manually directly in code.

Related to **RQ04: What methods have been used to validate the strategies?** some methods have been used to validate the selected proposals (feature F14), according to Figure 6, 35% of the works (8 works) were validated using examples.

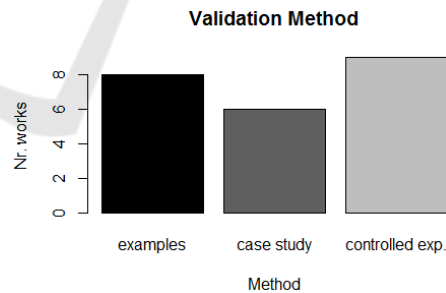


Figure 6: Validation methods used.

This method is important for obtaining initial results. However, empirical methods are required nowadays in order to provide more reliable results. We therefore found an increase in the number of works using case studies (26%, 6 works) and controlled experiments (39%, 9 works), particularly in more recent works.

Related to the notations used in specification stated in **RQ06: What languages/notations have been used in model transformations specification?** As shown in Figure 7 there are some options of

notation available in literature. Two works (9%) adopt UML or a lightweight extension of this (called UML profiles). There are also two works (9%) that use a heavyweight extension of UML. In this case, UML semantics is modified making the use of existing tools difficult. Thus, specific tools should be used. Formal languages are still used in six works (26%). Most works (10, which represent 43%) propose new languages specific for transformation domain (shown as a Proprietary language in the graph). There is also one work that uses natural languages. The others did not mention which language was adopted.

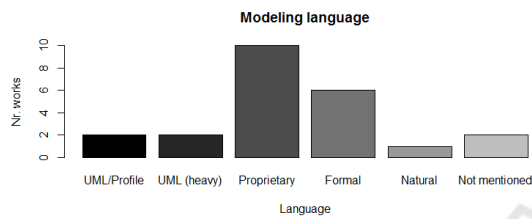


Figure 7: Modeling languages.

Figure 8 synthesizes the results of our review according to each RQ and the features used to answer them in order to analyze the hypothesis.

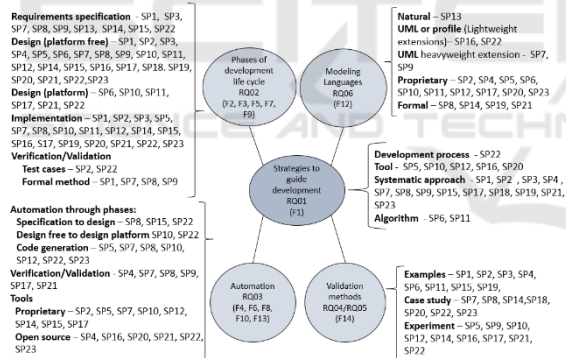


Figure 8: Summarizing the selected studies.

For each research question, the figure shows the related features and maps the articles (using its Id) that provide the feature. For example, concerning the strategy used to guide MT development (RQ01), the first initiatives, called systematic approaches, were proposed by SP1, SP2, SP3, SP4, SP7, SP8, SP9, SP15, SP17, SP18, SP19, SP21 and SP23. Other works detail the necessary activities in steps structured as algorithms (SP6 and SP11) or encapsulate them in a tool (SP5, SP10, SP12, SP16, SP20). The formalization of a process using PMLs has recently been proposed, in SP22.

Summarizing these data, regarding **hypothesis 1** - The development of model transformations is a trend and different strategies have been proposed in literature to support it, but there is still no well-defined strategy that can be widely used by the community. We found four different strategies presented in RQ1. However, we perceive that although the number of works has been increasing, they focus on specific aspects of development and do not cover the entire life cycle (RQ2) and that automation support also focuses on specific tasks, usually in code generation (RQ3). There are processes for the domain of model transformation development, e.g. (Magalhaes, 2016), and tools, e.g. (Bollati, 2013) and (Avazpour, 2015), however, we perceive that they are not validated enough to be widely used. As shown in RQ4, we observed the use of different validation methods, but none of the proposals have been exhaustively tested (RQ5). As a result, we come to the conclusion that there is a lack of a well-defined strategy, confirming hypothesis 1. Finally, we come to the same conclusion for **hypothesis 2** - In model transformation specification, there is no consensus on a notation to be adopted as a standard, where in general, a proprietary language is adopted in most works.

6 RELATED WORKS

Different strategies of software development have been used to produce model transformations.

The survey presented in (Silva, 2015) discusses the concepts that surround MDD, such as model, metamodel and platform. Despite the importance of this work for the community, its focus on the definition of concepts, it does not discuss how to develop transformations as we do in this paper.

Similar to our work, (Berranla, 2017) first discusses the problem statement concerning MT generation, i.e. proficiency in programming languages and knowledge in metamodels. Then it reports proposals to automate the development of model transformations considering some dimensions, such as the transformation inputs, outputs and algorithm as the main elements to define a development approach. The main difference between our review and this is the viewpoint adopted to classify the works as we use as reference the phases of a software development life cycle. Moreover, we also provide a quantitative result using graphics.

In (Bollati, 2013) the author carries out a SLR, the main goal of which is to find proposals that use MDD to develop model transformations. The review was

performed in five digital libraries and selected six approaches. Compared to our literature review, she includes works that use the MDA approach, from PIM to code, while we cover any strategy related to MT development life cycle. In addition, modeling languages as well as methods used in proposal validation are not mapped. Apart from these points, the review dates from 2011 and there have been new proposals since.

In the same direction as our work (Silva, 2014) presents a literature review of MT development approaches divided into three main groups: (a) MT foundations, that identify which concepts of model transformation are considered in each proposal (b) features implemented in the approaches, e.g. which phases of life cycle are considered; and (c) the applicability of approaches. The review was performed in the IEEE digital library and eight approaches were selected. This review analyzes MT foundations, which we do not. With regards to life cycle, the review briefly defines some phases of software development in order to evaluate the proposals, e.g. modeling source and target metamodels. In this, the concepts of phase and activity are placed at the same level of abstraction, which makes comparison with other works difficult. In our review, in order to guide the analysis of the proposals, we consider the main aspects of the life cycle of software development processes presented in the literature (Magalhaes, 2016), e.g. phases, activities, and artifacts. Unlike (Silva, 2014), we also explore the kind of modeling language adopted, e.g. UML, proprietary and natural language, as well as resources of automation. Furthermore, (Silva, 2014) only use works in the IEEE database while we consider four data sources.

In summary, we considered the review of (Bollati, 2013) included in our work, as we use a wider scope in terms of goals and items analyzed in each proposal. Moreover, we can say that the reviews of (Berranla, 2017) and (Silva, 2014) are complementary to our work. The former uses a different viewpoint to analyze the approaches and it interferes in the set of selected works. In the second review, certain aspects of MT foundations are analyzed that we did not take into consideration. On the other hand, we analyzed aspects such as automation and languages, which were not covered in the work.

7 DISCUSSIONS

In this section, we discuss some of the issues addressed in the selected papers and identify the strengths and gaps in this research area.

It is well known that the quality of a product is influenced by the process used to produce it (Sommerville, 2011). Through our review we observed an increasing number of proposals concerning systematized methods in transformation development. This might indicate that the community is worried about improving transformation quality. These proposals differ from each other in three aspects mainly: (i) the level of specification granularity; (ii) the level of formalism used in method specification; and (iii) the coverage of phases concerning to development life cycle.

According to (Sommerville, 2011), a software development process must define what should be done, how to perform it, when and by whom. Most proposals analyzed focus on the definition of what should be performed, i.e. identify and organize the necessary tasks, but they do not detail how to perform them. We perceived that this scenario is directly related to the level of formalism used to specify the proposed method. The first initiatives, e.g. (Kuster, 2005), (Wimmer, 2007), (Siikarla, 2008), (Sun, 2009), provide a description of the method in natural language, what we call in our review a systematic approach. Over the years proposals have become more formal e.g. algorithms have been used to structure the method, as in (Li, 2010) and in (Tavac, 2013), giving better support. Recently, some tools have been proposed as in (Bollati, 2013) and in (Avazpour, 2015) as well as a development process specified in PMLs such as SPEM, e.g. (Magalhaes, 2019), providing resources for automation and enactment that facilitate the adoption of the proposal by others. Improving the strategies towards a well-defined process is important to enable its replication.

Concerning the development life cycle, we perceive that works usually focus on specific phases but do not cover an entire life cycle. For example, (Del Fabro, 2009) focus on design and implementation, but do not support the requirements specification phase, and (Sani, 2011) focus on formal specification in order to enable transformation verification. This feature may lead to the need to adopt more than one method to cover all the development. Consequently, problems, such as selection of tools and document interchange, may occur. Leaving the responsibility to solve this to the Software Engineer may lead them to adopt ad-hoc methods. Therefore, strategies should be specified to

cover the entire development cycle of model transformations.

The specification of transformations, as in other software, requires the adoption of modeling languages to produce the necessary documentation. This is also essential for model refinement and code generation. We found no consensus on the adoption of a modeling language suitable for model transformation development. As a result, different proprietary modeling languages (and also UML profiles and heavyweight extensions of UML) have been used which may hamper communication and interchange of documents between tools. QVT (Query/View/Transformation), proposed by OMG as a standard is in fact not widely adopted. As a result, the definition of a modeling language which can be used for both industry and academia, as there is in programming languages (e.g. Java), is still required.

Finally, the development of model transformation involves different elements, e.g. development processes, modeling languages, modeling environments, formal languages and automation, and the current strategies usually do not cover all of these aspects. Thus, developers have a hard job choosing and integrating them. Therefore, integrated approaches are also required in order to reduce the complexity of the development.

8 THREATS TO VALIDITY

This section discusses some threats to the validity of our review according to the guidelines of Wohlin(2012). Considering construction validity, to decrease bias of the reviewers we established a protocol for how the reviews should be conducted. Related to internal validity, to reduce the chance of relevant papers being excluded we defined features to be observed in each study analyzed in our review. Concerning external validity, we considered studies in four major research databases however, we may have missed some relevant studies. There is, therefore, a threat to validity related to the generalization of the conclusions of the study which is minimized if we consider that the selected databases contain the main publications in MDD. Finally, in order to enable future replications of our review, we used the Start tool. Thus all the definitions, i.e. goal, research questions and the protocol information, as well as the metadata of the analyzed works in each stage, e.g. identification, selection and extraction, are stored and can be used by other researchers.

9 CONCLUSIONS

Over the last decade, many proposals have emerged to reduce transformation development complexity as well as improve transformation quality. Motivated by the lack of consensus about which techniques and methods are more appropriate, this paper presented a SLR in an attempt to investigate the strategies currently used in model transformation development. This systematic review not only identified the strategies most frequently used in transformation development but also analyzed relevant issues to support this development such as development phases, modeling languages and the level of automation provided.

Four strategies have been used to conduct transformation development, systematic approaches, algorithms, development processes, and tools. All of them identify the main activities and organize them in a structure to support development. In this direction, different approaches to software development are adopted, such as incremental and iterative process model, MDD, transformation patterns or a combination of these. Depending on the main goal, specific phases of development life cycle are considered. With regard to modeling languages, we also found a wide range of proprietary languages adopted in specification phases. Moreover, automation techniques to improve productivity have also been considered in more recent proposals.

In summary, we observed that much effort has gone into reducing the level of complexity that surrounds model transformation development. Developers have been very busy experimenting, however, there remains a lack of a stable strategy, one which is well tested, i.e. in real scenarios, to be replicated on a larger scale.

To conclude, it is well known that the definition and validation of new development approaches, which involve methods, languages and tools, are very challenging and should be done gradually. We can say that it has been almost fifteen years since MDD was introduced in software construction, however, transformation development practices are still in their infancy. Defining an integrated approach, i.e. which comprises method, languages and automation, that covers the entire life cycle, and that is well-tested in real scenarios remains a challenge. Understanding the specificities of transformation development can contribute to the widespread adoption of MDD.

REFERENCES

- Avazpour I, Grundy J, Grunske L, 2015. Specifying model transformation by direct manipulation using concrete visual notation and interactive recommendations. 195 – 211. *Journal of Visualization Language and Computing*.
- Baki I, Sahraoui H, 2016. Multi-step learning and adaptive search for learning complex model transformations from examples. *ACM Trans Softw Eng Methodol*. Available from: <http://doi.acm.org/10.1145/2904904>.
- Berranla K, Deba E, Benhamamouch D, et al, 2017. Formal concept analysis for specification of model transformations. *First International Conference on Embedded Distributed Systems (EDiS)*
- Bollati V, Vara J, Jimnez A, et al, 2013. Applying mde to the (semi-)automatic development of model transformations. 699–718, *IST*.
- Brambilla M, Cabot J, Wimmer M, 2012. *Model-driven software engineering in practice*. 1st ed. Morgan Claypool Publishers.
- Camargo LC, Del Fabro MD. 2019. Applying a data-centric framework for developing model transformations. *ACM SAC Program Language track*.
- Del Fabro M, Valduriez P, 2009. Towards the efficient development of model transformations using model weaving and matching transformations. *SSM*.
- Ergin H, Syriani E, Gray J, 2016. Design pattern oriented development of model transformations. *Comput Lang Syst Struct*. 106–139.
- Guerra E, Lara J, Kolovos D, et al., 2010. Transml: A family of languages to model model transformations.
- Guerra E, Lara J et al. 2013. Engineering model transformations with TransML. *SSM*. Pp 555-577.
- Kitchenham B, 2004. Procedures for performing systematic reviews. *Keele University Technical Report TR/SE-0401 / NICTA Technical Report 0400011T.1*.
- Kleiner M, Del Fabro MD. 2018. A generic approach to model generation operations. *JSS*, pp. 136-155.
- Kolahdouz-Rahimi S, Lano K, 2012. A model-based development approach for model transformations. *4th IPM International Conference on Fundamentals of Software Engineering*.
- Kuster J, Rynduna K, Hauser R, 2005. A systematic approach to designing model transformations. *Computer Science. Research Rep. Computer Science*.
- Li J, Yin G, 2010. Method of constructing model transformation rule based on reusable pattern. *V8-519–524. International Conference on Computer Application and System Modeling*.
- Ma Z, He X, 2016. A model-driven approach for model transformations. *1199–1205. SAI Computing Conf*.
- Magalhaes AP, Andrade A, Maciel RS, 2016. A model driven transformation development process for model to model transformation. *Proc. 30th Brazilian Symposium on Software Engineering. ACM*.
- Magalhaes APF, Andrade AMS, Maciel RSP, 2019. Model driven transformation development (mdtd): An approach for developing model to model transformation. *IST 55–76*.
- Mellor S, 2004. *Mda distilled*. Addison-Wesley.
- Mens T, Czarnecki K, Gorp P, 2006. A taxonomy of model transformation. 125–142. *Proc International Workshop on Graph and Model Transformation*.
- RSoftware, 2019. Available at <https://www.r-project.org/>
- Sani A, Polack F, Paige R, 2011. Model transformation specification for automated formal verification. 76–81. *5th Malaysian Conference in Software Engineering*.
- Semimi-Dehkordi L, Zamani B, Kolahdouz-Rahimi S, 2018. Evt+strace: a novel bidirectional model transformation approach. *47–72. IST*.
- Silva G, Rose LM, Calinescu R. A qualitative study of model transformation development approaches: Supporting novice developers. 2014; *Proceedings of Model-Driven Development Processes and Practices*.
- Silva A, 2015. Model-driven engineering: A survey supported by the unified conceptual model. 139 155.
- Skersys T, Danenas P, Rimantas B, 2016. Model-based m2m transformations based on drag-and-drop actions: Approach and implementation. 327–341. *JSS*.
- Solingen R, Basili V, Caldiera H Gand Rombach. 2002. *Goal question metric (GQM) approach*. John Wiley Sons. Inc.
- Sommerville I, 2011. *Software engineering*. Pearson Prentice Hall.
- Stahl T, Volter M, 2010. *Model-driven software development. technology, engineering, management*. Wiley.
- Siikarla M, Laitkorpi M, Selonen P, et al, 2008. Theory and practice of model transformations. *ICMT*.
- Start, 2013. State of art in systematic review - start Available <http://lapes.dc.ufscar.br/tools/starttool>
- Sun Y, White J, Gray J, 2009. Model transformation by demonstration. *Proc. Model Driven Engineering Languages and Systems*.
- Tavac M, Tavac V, 2013. The general algorithm for the design of the mda transformation models. *5th Int. Conf. on Computational Intelligence, Communication Systems and Networks*.
- Tehrani SY, Zschaler S, Lano K, 2016. Requirements engineering in model-transformation development: An interview-based study. In: *Proceedings of the 9th International Conference on Theory and Practice of Model Transformations*.
- Tikhonova U, Willemsse T, 2016. Documenting and designing qvto model transformations through mathematics. 349–364 *International Conference on Software Technologies*.
- Wimmer M, Strommer M, Kramler G, 2007. Towards model transformation generation by example. 285- 294. *Proceedings of the 40th Hawaii International Conference on System Sciences - HICSS*.
- Wohlin, C. et al. *Experimentation in Software Engineering*. [S.l.]: Springer, 2012. ISBN 978-3-642-29043-5.