Testing M2T Transformations
A Systematic Literature Review

André Abade[1,2], Fabiano Ferrari[1] and Daniel Lucrédio[1]
[1]*Computing Department, Federal University of São Carlos, Rodovia Washington Luis, km 235, 13565-905, São Carlos, SP, Brazil*
[2]*Federal Institute of Education, Science and Technology of Mato Grosso, Estrada de Acesso a BR-158, s/n, 78600-000, Barra do Garças, MT, Brazil*

Keywords: Model-Driven Development, MDD, Model-to-Text, M2T, Testing Transformation Model, Complex Data.

Abstract: **Context:** Model-Driven Development (MDD) is about to become a reality in the development of enterprise information systems due to its benefits, such as reduction of development and maintenance costs, and support for controlled evolution. Consequently, testing model transformations, considering their high complexity particularly regarding Model-to-Text (M2T) transformations, plays a key role to increase the confidence in the produced artefacts. **Objective:** this paper aims to characterize testing approaches and test selection criteria that focus on M2T transformations, in particular white-box approaches. **Method:** the objective is accomplished through a systematic literature review. We defined research questions regarding the testing of M2T transformations and extracted and analyzed data from a set of primary studies. **Results:** we identified a variety of incipient white-box testing approaches for this context. They mostly rely on mapping strategies and traceability of artefacts. Most of them focus on well-formedness and correctness of models and source code, although we could notice a change of focus in most recent research. **Conclusions:** current solutions for testing M2T transformations have begun to change the initial focus on well-formedness and correctness of models. Some approaches involve techniques that establish coverage criteria for testing, whereas others try to solve the testability across many transformations languages.

1 INTRODUCTION

A crucial success factor in the development of information systems is the alignment of system and business goals, business semantics and business processes. Recently, Model-Driven Development (MDD) concepts, methodologies and technology have been employed in the development and evolution of web information systems (Soltani and Benslimane, 2012; Panfilenko et al., 2013; Basso et al., 2014; Mayo et al., 2014; Garzotto, 2011). Overall, research initiatives aim to integrate MDD with other paradigms and technologies such as BPM (Business Process Model), SOA (Service-Oriented Architectures) and Agile Methods. Note that MDD does not only provide a structured and systematic approach to system development, but also offers developers the possibility of using model transformation technologies to derive models of a lower abstraction level that can be further refined, and even generating software code automatically (Pastor and España, 2012).

The need for software solutions in the business practice faces the increasing complexity of these solutions and the permanent rising of requirements related to performance, reliability and shorter technology cycles. There is also the pressure for cost reduction and frequent requirement changes. Together, these factors do not allow the industry to undervalue the integration of new concepts and techniques for software engineering (Panfilenko et al., 2013). Thus, in the MDD field, ensuring that models and transformations are effective and efficient becomes a major challenge. Note that the transformations of models involve different abstraction levels, and the complexity of the MDD paradigm is directly related to three elements involved in these transformations: the input (or source) models, the transformation rules and the output artefacts (Sendall and Kozaczynski, 2003; Guerra et al., 2013).

Software testing can be an effective approach for validating model transformations. According to Fleurey et al. (2004), the testing of model transformations is different from testing of traditional implementations because the former requires complex models as test input data, in particular when compared with simple data types. Testing in MDD represents a new challenge compared to testing in traditional systems (Beizer, 1990) or object-oriented systems (Binder, 1999). This is due to the fact that the involved data are models, which have significantly more complex structures than simple data types or traditional objects (Fraternali and Tisi, 2010). In fact, the testing-related problems in MDD more closely resembles those of testing interpreters or virtual machines (Sirer and Bershad, 1999), and are also related to the testing of grammars (Lämmel, Ralf, 2001) or compilers (Kalinov et al., 2003).

Studies conducted by Harman (2008) claim that the testing of model transformations leaves a number of open issues, which are related with the combination of software testing and the types of transformation that happen in MDD. Much effort has been put into the establishment of Model-to-Model (M2M) transformation testing techniques in the last years (Amrani et al., 2012; Vallecillo et al., 2012).

Most of the approaches for testing Model-to-Text (M2T) transformations are derived from M2M-related techniques. Consequently, specific knowledge regarding M2T testing is still limited, and hence provides only moderate support to developers, e.g. to those using the *Template Method* behavioural pattern. Thus, Wimmer and Burgueo (2013) highlight the need for testing approaches that are able to support the correct generation of software artefacts by means of M2T transformations.

Motivated by research opportunities determined by numerous issues involving the testing of model transformations, Selim et al. (2012) provided a historical perspective in order to recognise the main research investigations. The authors identify core and emerging topics and proposed approaches. Despite the fact that Selim et al. (2012) work dates from 2012 and that their discussions did not exclusively address the testing in context of M2T transformations, three phases are highlighted and need further research and investigations, namely: (i) generation of test data; (ii) test adequacy criteria; and (iii) the definition of test data sets.

This paper presents the results of a systematic literature review that summarises the state-of-the-art regarding the testing of M2T transformations. More specifically, this study aims to identify which structural (white-box) testing techniques and test selection criteria have been investigated to date, how complex data has been characterised within the identified approaches, and which is the predominant behavioural pattern used in transformation engines. A systematic literature review is a type of scientific research which aims to present an unbiased review about a research topic, following a methodology that is reliable, accurate and that allows auditing (Kitchenham, 2004; Fabbri et al., 2013).

The remainder of this paper is organised as follows: The next section provides basic background. The study planning is described in Section 3; the section also lists the selected primary studies. In the sequence, Section 4 presents an analysis of data extracted from the selected studies. Section 5 compares the identified testing approaches and Section 6 concludes this paper.

2 BACKGROUND

2.1 Model-Driven Development – MDD

In Software Engineering, MDD is a software development paradigm that handles models as primary artefacts from which code or other artefacts are generated. The Model Driven Architecture (MDA), proposed by the Object Management Group (OMG), is one of the embodiments of the MDD concept. In MDA-based processes, models play a key role in the software development (Object Management Group, 2003).

Model transformation has been characterised as a primary step for the MDD paradigm. It can be defined by a set of rules that together describe how a model in the source language can be transformed into one or more models in the target language (Kleppe et al., 2003).

A transformation involving two or more models can be classified as: transformation from model to model (M2M or Model-to-Model); and transformation from model to text (M2T or Model-to-Text). M2T transformations are used to generate the source code and documents, implement serialised models, and create visualisation and exploration models, thus automating several software engineering tasks (Rose et al., 2012).

One issue addressed in M2T transformations refers to the way of implementing the code generator. According to Czarnecki and Helsen (2006) and Rose et al. (2012), the most common approaches to accomplish M2T transformations are based on the *Visitor* and *Template Method* behavioural patterns. A Template Method is an abstract definition of an algorithm; it defines the algorithm step by step. Each step invokes an abstract operation or a primitive operation. The *Visitor* pattern, on the other hand, represents an operation to be performed on the elements of an object structure. It allows the creation of a new operation without changing the class of the elements on which it operates.

To define these transformations, a tool that allows software engineers to build mapping rules from M2M or M2T is necessary. It should allow mapping rules to be specified and created as naturally as possible, since the construction of a transformer is already a complex task by itself. Finally, a mechanism to apply the specified transformations, possibly maintaining some form of traceability between the mapped elements, is required (Lucredio et al., 2012).

2.2 Testing and MDD

Testing consists of a dynamic analysis of the product which is relevant to the identification and elimination of defects that remain. It represents the latest revision of the specification, design and coding (Harrold, 2000). In general, the testing techniques can be divided into code-based (white-box or structural) and specification-based (black-box or functional). It is important to notice that neither is complete, since they aim to identify different types of defects. When applied together, they can raise the level of software reliability.

The high abstraction level of MDD is a key concern that rules the complexity of the three main elements in this paradigm, namely: the input models, the transformation rules and the output artefacts. Consequently, applying existing testing techniques to model transformations becomes a cumbersome task.

The testability of model transformations aims at facilitating the generation of test data and ensuring better coverage through adjustments of the test criteria. Studies of Harman (2008) and Selim et al. (2012) claim that the models of transformer tests open a number of issues which relate to the combination of software testing and the types of transformations that occur in MDD. Fleury et al. (2004) and Selim et al. (2012) highlight that functional testing, on the one hand, guarantees the correctness of transformations. White-box testing, on the other hand, aims to guarantee high coverage of the transformations, both in terms of the input/output artefacts and the transformation rules themselves.

The relevance of the issues surrounding the transformation of models and their combination with software testing has motivated and underlie the systematic review reported in the paper. Since the MDD paradigm aims to produce effective, defect-free source code at a late stage in the software lifecycle, it important to investigate the evolution and application of testing techniques and test selection criteria in the context of the complexity that comes along with, specifically, M2T transformations.

3 THE REVIEW PROCESS AND RESULTS

This section presents the study planning and summarises the main procedures to perform the systematic review herein described. A systematic review aims to obtain the pieces of work that provide research evidence on a specific subject, which are called *primary studies*[1].

3.1 Study Planning

To identify the white-box testing approaches (techniques and test selection criteria) that have been proposed and applied to M2T transformations, we planned a systematic literature review (or simply *systematic review*) following the process defined by Fabbri et al. (2013). To provide transparency and allow replicability and audit of the whole process (which are requirements of systematic reviews (Kitchenham, 2004), we used the StArt Tool (State of the Art through Systematic Review) (Hernandes et al., 2012), which supports all process phases and stores all intermediate results together with the study selection decisions.

Research Questions

n Software Engineering, the research questions of a systematic review usually address solutions for a specific problem with a given research topic. When it comes to M2T transformations, there are several interesting related issues. For our review, we defined three research questions, each one associated with inclusion and exclusion criteria, as follows:

- Primary Question (PQ) - Which white-box testing approaches have been investigated for M2T transformations in the context of MDD so far?
- Secondary Question 1 (SQ1) - Amongst the testing approaches that have been investigated for M2T transformations in the context of MDD, which ones are specific regarding the use of the *Template Method* behavioural pattern?
- Secondary Question 2 (SQ2) - Amongst the testing approaches that have been investigated for M2T transformations in the context of MDD, which ones characterise the complexity of the data used in the transformation mechanisms?

[1]The term "primary study" has so far been used in the Evidence based Software Engineering domain (Kitchenham et al., 2004)to describe a variety of research results, from well-founded experimental procedures to incipient research approaches. Systematic reviews, on the other hand, are considered "secondary studies".

Search String

According to Petersen et al. (2008) and Fabbri et al. (2013), the process of collecting the set of primary studies begins by defining the search string. Therefore, the search string should be formulated on the experience of researchers and reviewers involved and must have a comprehensive scope, for the purpose of recovering all primary studies of interest. Reading the primary studies retrieved with the search string enable the researcher to verify their relevance. For initial selection, key information should be spotted in the study title and/or summary. Subsequently, final selection relies on the full reading of the study.

Even though different indexed research repositories define their own syntax for building search strings, we initially defined the following generic string that was later customised for each search engine:

```
(``model driven development'' OR ``model driven
   software development'' OR MDD OR MDSD OR
   ``model-driven engineering'' OR MDE)
                    AND
(test* OR ``coverage criteria'' OR ``software
         testing'' OR ``white box'')
                    AND
(``model transformation'' OR ``model to text
      transformation'' OR ``model to code
          transformation'' OR M2T OR M2C)
```

Primary Study Sources

The search strategy and selection of primary studies were defined according to the sources of studies, keywords, selection criteria, types of studies and languages of the studies. We selected the following sources to search for primary studies: indexed electronic databases (IEEE Xplore Digital Library [2], ACM Digital Library [3] and Elsevier Scopus [4]). In addition, we hand searched 18 journals and conference proceedings suggested by specialists[5], all related to software testing and MDD, published from 2008 to 2013. The list of journals and conference proceedings is presented in Table 1.

The searches in the indexed repositories returned 202 entries[6]. The full list of primary studies identified by the searches was evaluated against a set of inclusion and exclusion criteria which are described in the sequence. We also describe the selection process (preliminary and final selection).

[2]http://www.ieeexplore.ieee.org – access: october/2014

[3]http://dl.acm.org – access: october/2014

[4]http://www.scopus.com – access: october/2014

[5]Approaching specialists in a given field is one of the recommendations of (Kitchenham, 2004).

[6]Note that not all entries are in fact primary studies. For example, some are proceedings' front matters, while others are abstracts of conference invited talks and tutorials.

Table 1: List of Conferences and Journals indicated.

ID	Description
1	ACiS - Software Engineering Research and Applications
2	ASE - Automated *Software* Engineering
3	CAiSE - Conference on Advanced Information Systems Engineering
4	ENASE - Evaluation of Novel Approaches to *Software* Engineering
5	GPCE - Generative Programming and Component Engineering
6	GTTSE - Generative and Transformational Techniques in *Software* Engineering
7	ICMT - International Conference on Model Transformation
8	ICSR - International Conference on *Software* Reuse
9	ICST - International Conference on *Software* Testing, Verification, and Validation
10	ICTSS - International Conference on Testing *Software* and Systems
11	ISSTA - International Symposium on *Software* Testing and Analysis
12	MoDELS - Model Driven Engineering Languages and Systems
13	MODELSWARD - Model-Driven Engineering and *Software* Development
14	SBCars - Brazilian Symposium on *Software* Components, Architectures and Reuse
15	SBES - Brazilian Symposium on *Software* Engineering
16	SLE - *Software* Language Engineering
17	SoCO - Soft Computing Models in Industrial and Environmental Applications
18	SoSyM Journal - *Software* and System Modeling

Inclusion and Exclusion Criteria

Inclusion criteria were used to determine which studies were relevant to the research questions of the systematic review. The following inclusion criteria were applied:

+ Publications, technical reports or "grey" literature that describe empirical studies, of any particular study design, in which the testing of M2T transformations was applied to any technology.
+ Studies that describe the problems / barriers for the use of white-box testing approaches in M2T transformations.
+ Studies describing techniques / methods to testing M2T transformations.
+ Studies which allowed the identification of behavioural pattern used in M2T transformations.

Exclusion criteria were defined to protect the interests of the main theme and the research questions. Several studies were excluded for not addressing the testing of M2T transformations, or not even mentioning the behavioural pattern used in the M2T transformations.

Studies that met the following criteria were excluded from the review:

- Studies that are not relevant to the research questions.
- Studies that did not allow access to full text.
- Studies that did not describing techniques / methods to testing M2T transformations.

Preliminary Selection

After applying the inclusion and exclusion criteria to each primary study based on their titles and abstracts, the preliminary set of relevant primary studies included 49 entries. At this stage, 21 primary studies were included as a result of the analysis of specific conference proceedings and journals, thus summing up 70 entries in the preliminary list. Table2 presents the number of studies accepted, discarded and duplicated (i.e. retrieved by more than one search engine) after the preliminary selection phase.

Table 2: Number of studies accepted, discarded and duplicated during the preliminary selection.

Source	Preliminary Selection		
	Accepted	Discarded	Duplicated
ACM	14 (20,00%)	43 (36,13%)	12 (35,29%)
IEEE	11 (15,71%)	23 (19,33%)	01 (02,94%)
Scopus	24 (34,29%)	53 (44,54%)	21 (61,76%)
Specialists	21 (30,00%)	00 (00,00%)	00 (00,00%)
Total	70 (100,0%)	119 (100,0%)	34 (100,0%)

Data Extraction Strategy and Final Selection

We extracted data from the selected studies to address each of the research questions. We classified studies according to the following categories: behavioural pattern; research type (evaluation research, solution proposal, validation research, philosophical paper, opinion paper and personal experience); characterisation (or not) of the complexity of the data used in the transformation mechanisms; testing approach. Discrepancies observed during the analysis were discussed amongst the involved researchers (co-authors of this paper).

In a second, final iteration, from the 70 preselected studies, we selected only those ones which fulfilled a set of relevant research issues and specified testing approaches used in the model transformations. In order to perform this final selection, studies accepted in the preliminary selection phase went through a complete reading and were again classified based on the inclusion and exclusion criteria. This resulted in a final list of 9 primary studies. Details are provided in the sequence.

3.2 Final Selection Results

Table 3 presents the list of 9 selected primary studies. Studies are sorted by year of publication and grouped by source. Other information items available in the table are study identifier (ID), authors, and information related to the research questions which are detailed later in the paper. Note that the ID of selected studies will be used hereafter to facilitate the identification of each piece of work.

According to Kitchenham (2004), the final set of selected studies should avoid overlapping of primary studies. For example, if we identify two or more pieces of work that describe the same testing approach (e.g. the latter as the evolution of the former), the final selection should include only the more recent one. Taking this into consideration, Table 4 lists all overlapping results we identified.

Studies listed in Table 3 were classified according to a set of relevant research issues and specified approaches used in the model transformations. Each characteristic was named *facet*. Figure 1 depicts the distribution of studies according to each facet. It provides a quantification of studies associated with the research questions defined for the review.

4 ANALYSIS AND DISCUSSION

This section presents the analysis and discussion of the results obtained in the systematic review. The description of results is organised according to the facets depicted in Figure 1. The facets concern properties which are relevant to the research questions that will be discussed in the sequence.

It should be clarified that the behavioural pattern defined for the transformations, as well as the characterisation of complex data, impacts on the formulation of techniques and approaches to test the M2T transformations. For instance, the *Template Method* often makes use of a reference implementation (Lucredio et al., 2012), allowing that coverage criteria can be defined with respect to the input models, the templates and the generated code.

Facet 1 - Behavioural Pattern – This facet groups the studies according to the behavioural pattern of the transformer design. The classification is based on the categories defined by Czarnecki and Helsen (2006) and Rose et al. (2012). After analysing the characteristics of the transformations languages used in each study, we obtained the following groups: (a) approaches that apply the *Visitor* pattern (S6, S7); and (b) approaches that apply the *Template Method* pattern (S1, S2, S3, S4, S5, S8, S9).

Table 3: Final selection of primary studies.

Year	Source	ID	Title	Authors	Characterise Complex Data	Structural Test	
						Visitor Pattern	Template Method
2008	Specialist	S1	Unit Testing Model Management Operations	(Polack et al., 2008)	No	-	X
	Scopus	S2	Using model transformation to support model-based test coverage measurement	(Naslavsky et al., 2008)	No	-	X
2010	ACM	S3	Multi-level Testes for Model Driven Web Applications	(Fraternali and Tisi, 2010)	No	-	X
2011	Scopus	S4	Model-driven testing: Transformations from test models to test code	(Lamancha et al., 2011)	No	-	X
2013	Specialist	S5	A Method for Testing Model to Text Transformations	(Tiso et al., 2013)	Parcial	-	X
	Specialist	S6	An Approach to Testing Java Implementation against Its UML Class-Model	(Chavez et al., 2013)	No	X	-
	ACM	S7	Transformation Rules for Platform Independent Testing: An Empirical Study	(Eriksson et al., 2013)	Parcial	X	-
	Scopus	S8	MTC Flow: A Tool to Design, Develop and Deploy Model Transformation Chains	(Alvarez and Casallas, 2013)	No	-	X
	Specialist	S9	Testing M2T/T2M transformations	(Wimmer and Burgueño, 2013)	No	-	X

Table 4: Overlapping results of primary studies.

Year	**Final Selection**	**Year**	**Subsumed studies**
2013	(Tiso et al., 2013)	2012	(Tiso et al., 2012)
2013	(Eriksson et al., 2013)	2012	(Eriksson et al., 2012)

The results shown in Figure 1 are consistent with the standardisation efforts of the OMG MOFM2T (Object Management Group, 2012). They show that 78% of the selected studies support transformations by applying the *Template Method* pattern, while only 22% emphasises the application of the *Visitor* pattern. Following this trend, the work of Rose et al. (2012) defines a model of features that supports the decision making process regarding the use of M2T transformation languages, based on MOF standards.

Facet 2 - Characterisation of Complex Data – Basically, the complexity of transformations is defined by the high level of abstraction used in input models, by the complexity of the transformation rules and by the diversity of software artefacts that may be generated (Baudry et al., 2010). In detriment of the analytical model and of the formal model, the computational model encapsulates all semantics of the developed solution, *i.e.*beyond the abstract data types, the computational models also establish relationships, cardinality, and the directional flow of associations between objects belonging to a given context.

According to Baudry et al. (2010), the complexity of the input/output data manipulated by model transformations bring new challenges, thus requiring adaptations of testing techniques. In spite of it, we found out that 78% of studies did not address issues related to complex data involved in transformations (S1, S2, S3, S4, S6, S8, S9). Only 22% of the studies *partially* characterise complex data in transformations using some representation strategy (S5, S7).

Facet 3 - Testing Approach – Software testing consists in a dynamic analysis of the product. It is relevant to the detection and subsequent elimination of software defects. Testing represents the latest revision of the specification, design and coding. In general, testing techniques can be classified between those based on source code (structural) and those based on the speci-

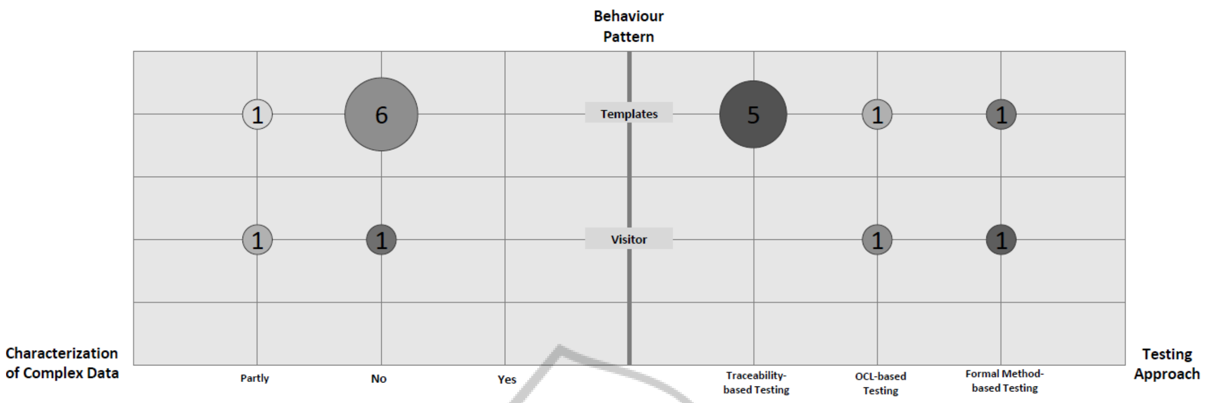

Figure 1: Bubble chart - Results distribution by facet.

fication (functional).

Overall, the studies investigate structural testing for M2T transformations. Such technique examines the internal structure of the software to derive test requirements. This facet groups primary studies that investigate structural-based testing approaches that focus on the following property or software specification methods: (a) Traceability, with 56% of studies (S1, S2, S3, S4, S8); (b) OCL, with 22% of studies (S6, S9); and (c) Formal Methods, with 22% of studies (S5, S7). The next section provides more details of this facet.

5 COMPARISON BETWEEN APPROACHES

To keep our analysis aligned with the facets defined in the previous section, we next compare the testing approaches by means of their common characteristics. For this, three groups are defined as follows: (A) M2T Testing with *Visitor Pattern*; (B) M2T Testing with *Template Method* Pattern; and (C) Complex Data Characterisation for M2T Testing. The analyses are grouped according to the categories of the *Testing Approach* facet. Following this organisation, for groups (A) and (B) we compared the selected studies based on the categories included in the third facet depicted in Figure 1, namely: *Traceability-based Testing*, *OCL-based Testing* and *Formal Method-based Testing*.

A - M2T Testing with Visitor Pattern

Two testing approaches show these characteristics. They are the studies S6 and S7 listed in Table 3. Note that this group does not include any primary study that addresses Traceability-based Testing.

OCL-based Testing

Study S6, by Chavez et al. (2013), proposes an innovative approach named CCUJ. The authors describe how to automate such an approach, which allows the conformance checking (well-formedness) between a UML design and Java implementation using traceability techniques and mappings. The CCUJ approach checks whether the Java implementation is consistent with the OCL specification in terms of pre- and post-conditions associated with the class diagrams; consequently, conformance checking is needed. In this work, the authors do not explicitly define the coverage criteria for structural testing, since much of the methodology was concerned with validation of well-formedness. The study describes the methodology, however does not address the issue of complex data characterisation.

Formal Method-based Testing

Studies S7, by Eriksson et al. (2013) propose that the structural testing for M2T transformations and their coverage criteria are defined by means of arrays composed of predicates and their clauses. One of the processes of these approaches consists in quantifying how many clauses form a predicate; the goal is to estimate the number of test requirements based on logical expressions. These approaches discuss how to resolve the discrepancy in the number of test requirements between the model and generated source code, considered in their empirical evaluations on average 60% higher. These approaches partially characterise the input/output complex data involved in M2T transformations. Basically, both studies cover all the issues that define the problems for Testing M2T Transformations. However, the authors highlight that only one compiler was used for transformations, thus invalidating the results to other transformers.

B - M2T Testing with Template Method Pattern

Eight testing approaches address the testing of M2T transformations that focus on the *Template Method* pattern. These studies are S1, S2, S3, S4, S5, S8 and S9 listed in Table 3. We next provide an analysis and comparison between them.

Traceability-based Testing

Study S1, by Polack et al. (2008), makes use of unit testing to ensure the correctness of the operations of the model with respect to the generated source code. For that, the authors propose a prototype which specifies the execution of this transformation upon the Epsilon platform. This approach uses the know-how of testing of M2M transformations to tackle the testing of M2T transformations. Even though unit testing is not a new concept for the development of software, it is considered an important strategy for developers to improve code quality, during the understanding of the functional requirements of a class or method (Osherove, 2009). A variety of approaches for structural testing of M2M transformations apply this strategy to establish good training models and correctness criteria.

Studies S2 Naslavsky et al. (2008) and S4 Lamancha et al. (2011) deal with testing of M2T transformations by applying Model-Based Testing (MBT). MBT is a technique for automatically generating test cases based on models extracted from software artefacts. Thus, its main goal is the creation of artefacts that describe the requirements and behaviour of the system itself, aimed at automating the testing process. Once the model is developed, it can be used in various ways and for various stages of the software development process, such as requirements specification, code generation, analysis of system reliability and testing (Broy et al., 2005). The approach proposed by Naslavsky et al. (2008) (S2) relies on a strategy of mapping and traceability, via annotations in the code, to measure the coverage achieved by test cases exercised by the models derived with the MBT approach. Lamancha et al. (2011) (S4) propose a framework to automate the testing in the context of MBT. This framework consists in creating test models for both types of transformations: M2M and M2T. To generate test cases with this approach, MOFScript is used according to OMG standards (Oldevik, 2011). The contribution of this study is guided primarily by automated generation of test cases through mapping and traceability technique.

The approach of Fraternali and Tisi (2010) (S3) focuses on testing environment issues. Testing environment refers to the infrastructure upon which the test runs, including hardware configurations, software, automation tools, testing team, organisational aspects, supplies, and network documentation. Its purpose is to facilitate testing under well-known and controlled conditions. In this perspective, (Fraternali and Tisi, 2010) address the definition, management and execution of tests for web applications under the MDD paradigm, in its various abstraction levels. The approach is aligned with the flow of transformations MDA and source code. The proposed methodology makes use of WebML and WebRatio platform to implement a framework that manages the test cases, keeping traceability across the levels.

Study S8 Alvarez and Casallas (2013) describes a tool called MTC Flow that allows developers to create applications according to the MDD paradigm. The tool provides a graphical DSL to specify the MTC workflow model independently of the transformation technology. Another important feature is the management of alternative execution paths using a Tag System to mark the elements in the MTCs design. These marks later will guide the MTC execution engine. In this approach, the integration of the different layers of the MDA provides interoperability. Testing techniques available in the tool's environment underlies the validation of correctness and well-formedness of the models.

OCL-based Testing

Wimmer and Burgueño (2013) (S9) make know-how inherited from testing in M2M context to compose a proposal for M2T / T2M test regardless of the languages used both for source models and output artefacts. The authors specify the contracts and transformation rules through an extension of OCL, supporting the design and execution of unit tests in order to validate the established rules and contracts.

Formal Method-based Testing

Studies S5 by Tiso et al. (2013), present a method to test M2T transformations, which uses a set of integrated approaches. The development of model transformations is part of this method. This proposal is a set of rules that characterise the well-formedness and correctness of the model, in addition to the characterisation by means of stereotypes of UML diagrams for selecting source models. Thus, this method has the objective of organising test input data to establish test coverage criteria of the model.

C - Complex Data Characterisation for M2T Testing

Most real-world applications use complex input data structures, which are represented as objects that result from the aggregation or composition of other objects. In a context of structural testing, which requires knowledge about the internal implementation structure to devise relevant test input data, understanding such input data structure is fundamental. This comprehension shall enable the definition of effective test data that will determine high coverage, and hence high confidence on the software correctness.

Once it becomes possible to adequately represent the elements of the source models in a MDD approach, testers becomes able to compose complex input data sets that exercise specific parts of the transformer. Such specific parts are related, for example, to a particular type of artefact that will be generated. As an example, while a set of input elements can exercise the transformer to generate Java artefacts (*i.e.*Java classes), another particular set can exercise the transformer to generate SQL artefacts (*i.e.*database queries).

In this systematic review, two studies (S5 and S7) partially characterise complex data for M2T transformations using some data representation strategy. Study S7 (Eriksson et al., 2013) use predicate logic to compose arrangements of complex data in order to resolve the discrepancy in the number of test requirements between the source model and the generated code. The limitations reported by the authors concerns the use of only one transformer with the *Visitor* behavioural pattern, and the lack of reports regarding the effective characterisation of complex data for different types of artefacts that can be generated according to the MDD paradigm.

As previously described in this section, study S5 (Tiso et al., 2013) have the objective of organising test input data to establish test coverage criteria of the model. However, the data characterisation described in these studies has a generic nature. The authors do not particularly address complex data representation for textual models; the generic approach might be customised for the output of M2T transformations.

6 CONCLUSION

A systematic review is not a simple summary of the published research on a particular topic. It is a methodology for building knowledge by means of evidence reported in primary studies, which allows researchers to go a step beyond an ordinary review. In this context, this paper reported on the results of a systematic review that characterised the state-of-the-art with respect to the testing of M2T transformations in the context of MDD. The results provide a categorisation of the common features found in testing approaches, as well as their recurring problems and limitations.

In a high-level analysis, we conclude that the existing solutions for the testing of M2T transformations have begun to change the initial focus on well-formedness and correctness of models. Even though they are still in a small number, some approaches are prominent proposals regarding the use of techniques that establish coverage criteria for testing, whereas others already try to solve the testability across many transformation languages. Nevertheless, the solutions found in the literature cannot yet be considered established, so this research topic needs to be further explored.

Regarding our research questions, the results reveal that the focus is on structural testing of M2T transformations. Approaches in this context are evolving, especially with the use of traceability mappings and transformations. However, most of the evaluated approaches deal exclusively with issues of well-formedness and correctness of models. We could not identify approaches that characterised the complexity of the data manipulated by model transformations that somehow optimise the coverage criteria by establishing a discipline for test design.

Still regarding the research questions, we conclude that structural testing approaches for M2T transformations focus on the *Template Method* behavioural pattern, when compared with the *Visitor* pattern. This corroborates the OMG effort for defining standards for M2T transformations, raising the relevance of templates as transformation artefacts, and hence their quality assurance through testing.

We highlight that a main threat to the validity of our study concerns the small number of selected papers after the application of the search string and the criteria to include papers for evaluation. As a future work and as a way to mitigate such threat, we plan to consider additional databases, increasing the scope of results obtained through the search machines. Another possible measure to reduce the threat consists in the application of the backward snowballing technique in the selected primary studies (Jalali and Wohlin, 2012).

Even though we were able to identify approaches for testing M2T transformations, the main problems still require much research and investigation. The complexity of the source model, the transformer itself and the output artefacts pose an important challenge for future research.

ACKNOWLEDGEMENTS

The authors would like to thank the financial support received from the Federal Institute of Education, Science and Technology of Mato Grosso and the Graduate Degree Program in Computer Science of the Federal University of São Carlos .

REFERENCES

Alvarez, C. and Casallas, R. (2013). Mtc flow: A tool to design, develop and deploy model transformation chains. In *Workshop on ACadeMics Tooling with Eclipse (ACME)*, pages 1–9. ACM.

Amrani, M., Lucio, L., Selim, G., Combemale, B., Dingel, J., Vangheluwe, H., Le Traon, Y., and Cordy, J. R. (2012). A tridimensional approach for studying the formal verification of model transformations. In *5th International Conference on Software Testing, Verification and Validation (ICST)*, pages 921–928. IEEE Computer Society.

Basso, F. P., Pillat, R. M., Frantz, R. Z., and Roos-Frantz, F. (2014). Assisted tasks to generate pre-prototypes for web information systems. In *16th International Conference on Enterprise Information Systems (ICEIS)*, pages 14–25.

Baudry, B., Ghosh, S., Fleurey, F., France, R., Le Traon, Y., and Mottu, J.-M. (2010). Barriers to systematic model transformation testing. *Communications of the ACM*, 53(6):139–143.

Beizer, B. (1990). *Software Testing Techniques*. Van Nostrand Reinhold Co., New York, NY, USA, 2nd edition.

Binder, R. V. (1999). *Testing Object-Oriented Systems: Models, Patterns and Tools*. Addison Wesley, Reading/MA - USA, 1st edition.

Broy, M., Jonsson, B., Katoen, J.-P., Leucker, M., and Pretschner, A. (2005). *Model-Based Testing of Reactive Systems: Advanced Lectures*, volume 3472. Springer-Verlag.

Chavez, H. M., Shen, W., France, R. B., and Mechling, B. A. (2013). An approach to testing java implementation against its uml class model. In *16th International Conference on Model Driven Engineering Languages and Systems (MODELS)*, pages 220–236 (LNCS 8107). Springer.

Czarnecki, K. and Helsen, S. (2006). Feature-based survey of model transformation approaches. *IBM Systems Journal*, 45(3):621–645.

Eriksson, A., Lindström, B., Andler, S. F., and Offutt, J. (2012). Model transformation impact on test artifacts: An empirical study. In *Workshop on Model-Driven Engineering, Verification and Validation (MoDeVVa)*, pages 5–10. ACM.

Eriksson, A., Lindström, B., and Offutt, J. (2013). Transformation rules for platform independent testing: An empirical study. In *6th International Conference on Software Testing, Verification and Validation (ICST)*, pages 202–211. IEEE Computer Society.

Fabbri, S. C. P. F., Felizardo, K. R., Ferrari, F. C., Hernandes, E. C. M., Octaviano, F. R., Nakagawa, E. Y., and Maldonado, J. C. (2013). Externalising tacit knowledge of the systematic review process. *IET Software*, 7(6):298–307.

Fleurey, F., Steel, J., and Baudry, B. (2004). Validation in model-driven engineering: testing model transformations. In *First International Workshop on Model, Design and Validation*, pages 29–40. IEEE Computer Society.

Fraternali, P. and Tisi, M. (2010). Multi-level tests for model driven web applications. In *10th International Conference on Web Engineering (ICWE)*, pages 158–172 (LNCS 6189. Springer-Verlag.

Garzotto, F. (2011). Enterprise frameworks for data intensive web applications: An end-user development, model based approach. *Journal of Web Engineering*, 10(2):87–108.

Guerra, E., Lara, J., Kolovos, D., Paige, R., and Santos, O. (2013). Engineering model transformations with transml. *Software and Systems Modeling*, 12(3):555–577.

Harman, M. (2008). Open problems in testability transformation. In *Workshops of the International Conference on Software Testing Verification and Validation (ICSTW)*, pages 196–209. IEEE Computer Society.

Harrold, M. J. (2000). Testing: A roadmap. In *Conference on the Future of Software Engineering - in conjunction with ICSE*, pages 61–72. ACM.

Hernandes, E. M., Zamboni, A., Fabbri, S., and Di Thommazo, A. (2012). Using GQM and TAM to evaluate StArt - a tool that supports systematic review. *CLEI Electronic Journal*, 15(1).

Jalali, S. and Wohlin, C. (2012). Systematic literature studies: Database searches vs. backward snowballing. In *6th International Symposium on Empirical Software Engineering and Measurement (ESEM)*, pages 29–38. ACM.

Kalinov, A., Kossatchev, A., Petrenko, A., Posypkin, M., and Shishkov, V. (2003). Coverage-driven automated compiler test suite generation. *Electronic Notes in Theoretical Computer Science*, 82(3):500–514.

Kitchenham, B. (2004). Procedures for performing systematic reviews. Technical report, Departament of Computer Science, Keele University.

Kitchenham, B. A., Dyba, T., and Jorgensen, M. (2004). Evidence-based software engineering. In *26th International Conference on Software Engineering (ICSE)*, pages 273–281, Washington, DC, USA. IEEE Computer Society.

Kleppe, A. G., Warmer, J., and Bast, W. (2003). *MDA Explained: The Model Driven Architecture: Practice and Promise*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Lamancha, B. P., Mateo, P. R., Polo, M., and Caivano, D. (2011). Model-driven testing - transformations from test models to test code. In *6th International Conference on Evaluation of Novel Approaches to Software Engineering - (ENASE)*, pages 121–130. SciTePress.

Lämmel, Ralf (2001). Grammar testing. In Hussmann, H., editor, *Fundamental Approaches to Software En-*

gineering, volume 2029 of *Lecture Notes in Computer Science*, pages 201–216. Springer Berlin Heidelberg.

Lucredio, D., de Almeida, E., and Fortes, R. (2012). An investigation on the impact of MDE on software reuse. In *Sixth Brazilian Symposium on Software Components Architectures and Reuse (SBCARS)*, pages 101–110. IEEE Computer Society.

Mayo, F. J. D., Escalona, M. J., Mejías, M., Ross, M., and Staples, G. (2014). Towards a homogeneous characterization of the model-driven web development methodologies. *Journal of Web Engineering*, 13(1&2):129–159.

Naslavsky, L., Ziv, H., and Richardson, D. J. (2008). Using model transformation to support model-based test coverage measurement. In *3rd International Workshop on Automation of Software Test (AST)*, pages 1–6. ACM.

Object Management Group (2003). MDA guide version 1.0.1. online - http://www.omg.org/cgi-bin/doc?omg/03-06-01.pdf – accessed on 26/02/2015.

Object Management Group (2012). MOF model to text transformation language - MOFM2T 1.0. online. http://www.omg.org/spec/MOFM2T/1.0/ – accessed on 26/02/2015.

Oldevik, J. (2011). MOFScript User Guide. online. http://eclipse.org/gmt/mofscript/doc/MOFScript-User-Guide-0.9.pdf.

Osherove, R. (2009). *The Art of Unit Testing: With Examples in .Net*. Manning Publications Co., Greenwich, CT, USA, 1st edition.

Panfilenko, D. V., Hrom, K., Elvesæter, B., and Landre, E. (2013). Model transformation recommendations for service-oriented architectures. In *15th International Conference on Enterprise Information Systems (ICEIS)*, pages 248–256.

Pastor, O. and España, S. (2012). Full model-driven practice: From requirements to code generation. In *Advanced Information Systems Engineering*, volume 7328 of *Lecture Notes in Computer Science*, pages 701–702. Springer Berlin Heidelberg.

Petersen, K., Feldt, R., Mujtaba, S., and Mattsson, M. (2008). Systematic mapping studies in software engineering. In *12th International Conference on Evaluation and Assessment in Software Engineering (EASE)*, pages 68–77, Swinton, UK, UK. British Computer Society.

Polack, D., Paige, R., Rose, L., and Polack, F. (2008). Unit testing model management operations. In *Workshops of the International Conference on Software Testing Verification and Validation (ICSTW)*, pages 97–104. IEEE Computer Society.

Rose, L., Matragkas, N., Kolovos, D., and Paige, R. (2012). A feature model for model-to-text transformation languages. In *ICSE Workshop on Modeling in Software Engineering (MISE)*, pages 57–63. IEEE Computer Society.

Selim, G. M. K., Cordy, J. R., and Dingel, J. (2012). Model transformation testing: The state of the art. In *First Workshop on the Analysis of Model Transformations (AMT)*, pages 21–26, New York, NY, USA. ACM.

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

Sirer, E. and Bershad, B. (1999). Testing Java virtual machines - An Experience Report on Automatically Testing Java Virtual Machines. In *International Conference on Software Testing And Review*.

Soltani, M. and Benslimane, S. M. (2012). From a high level business process model to service model artifacts - A model-driven approach. In *14th International Conference on Enterprise Information Systems (ICEIS)*, pages 265–268. SciTePress.

Tiso, A., Reggio, G., and Leotta, M. (2012). Early experiences on model transformation testing. In *First Workshop on the Analysis of Model Transformations (AMT)*, pages 15–20, New York, NY, USA. ACM.

Tiso, A., Reggio, G., and Leotta, M. (2013). A method for testing model to text transformations. In *Second Workshop on the Analysis of Model Transformations (AMT)*, volume 1077. CEUR-WS.org.

Vallecillo, A., Gogolla, M., Burgueño, L., Wimmer, M., and Hamann, L. (2012). Formal specification and testing of model transformations. In *Formal Methods for Model-Driven Engineering*, volume 7320, pages 399–437. Springer.

Wimmer, M. and Burgueño, L. (2013). Testing M2T/T2M transformations. In *Model-Driven Engineering Languages and Systems*, volume 8107, pages 203–219. Springer Berlin Heidelberg.