Validating Model Transformations using Search-based Software Testing

Author

Momina Ramzan

00000205124

Supervisor

Dr. Farooque Azam

DEPARTMENT OF COMPUTER & SOFTWARE ENGINEERING

COLLEGE OF ELECTRICAL & MECHANICAL ENGINEERING

NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY

ISLAMABAD

AUG, 2021

Validating Model Transformations using Search-based Software Testing

Author

Momina Ramzan

00000205124

A thesis submitted in partial fulfillment of the requirements for the degree of

MS Computer Software Engineering

Thesis Supervisor

Dr. Farooque Azam

Thesis Supervisor's Signature: _____

DEPARTMENT OF COMPUTER & SOFTWARE ENGINEERING

COLLEGE OF ELECTRICAL & MECHANICAL ENGINEERING

NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY, ISLAMABAD

AUG, 2021

Declaration

I certify that this research work titled "*Validating Model Transformations using Search-based Software Testing"* is my own work under the supervision of Dr. Farooque Azam. This work has not been presented elsewhere for assessment. The material that has been used from other sources has been properly acknowledged / referred.

_____

Signature of Student

Momina Ramzan

00000205124

Language Correctness Certificate

This thesis has been read by an English expert and is free of typing, syntax, semantic, grammatical and spelling mistakes. Thesis is also according to the format given by the university.

_____

Signature of Student

Momina Ramzan

00000205124

_____

Signature of Supervisor

Copyright Statement

- Copyright in text of this thesis rests with the student author. Copies (by any process) either in full, or of extracts, may be made only in accordance with instructions given by the author and lodged in the Library of NUST College of E&ME. Details may be obtained by the Librarian. This page must form part of any such copies made. Further copies (by any process) may not be made without the permission (in writing) of the author.
- The ownership of any intellectual property rights which may be described in this thesis is vested in NUST College of E&ME, subject to any prior agreement to the contrary, and may not be made available for use by third parties without the written permission of the College of E&ME, which will prescribe the terms and conditions of any such agreement.
- Further information on the conditions under which disclosures and exploitation may take place is available from the Library of NUST College of E&ME, Rawalpindi.

Acknowledgements

In the name of Allah, the Most Beneficent and the Most Merciful

First and foremost, I am highly grateful to Allah Almighty for blessing me with this life, with the abilities, skills and environment to carry out this work successfully.

Next, I am greatly indebted to my supervisor Dr. Farooque Azam for his kind supervision, invaluable feedback, unwavering support, constant encouragement, timely availability and patience throughout the course of this research work, without which a work of this magnitude would not have been possible. I am also thankful to Mr. Muhammad Waseem Anwar for his guidance and feedback at various stages of this work.

I am grateful to all respected committee members who include Dr. Wasi Haider Butt and Dr. Urooj Fatima for their kind and encouraging words. I would like to acknowledge all faculty members at our department who taught us in-depth and hands-on courses and always went out of their way to answer any queries that students had. Dr. Saad Rehman who taught us the course of Research Methodology deserves a special mention here, who often shared his wisdom and insights during the course which helped me unstuck any convoluting problem that I faced during this research work.

Moreover, several of my dear batch fellows that shared their experiences with me and offered their help, I am indebted to you all as well. I am grateful to all the administrative staff of our college for their timely coordination and cooperation.

Last but definitely not the least, I am very much grateful to my family – my rock, who inspire me with their strength, resilience and perseverance, who is always by my side, who believes in me, who celebrates my every achievement and who reminds me that no matter how difficult it may get, there is always a way out – a lesson which is apt in the prevailing times of uncertainty where the global pandemic of COVID-19 looms above our head.

Dedicated to My Parents

Abstract

Model Transformations (MTs) are the cornerstone of Model-Driven Engineering (MDE). MTs systematically transform input models to output models. Validating MTs establishes credibility of MDE which finds its applicability in avionics and automotive industries, however; it is a non-trivial task as models are complex structures consisting of attributes and associations of various cardinalities. These models conform to their corresponding Metamodel (MM) which defines the model structure. A MM further imposes additional constraints on models that they must satisfy. These constraints manifest as Boolean expressions adding to the complexity of models, making it all the more challenging to validate MTs for which test models need to be generated. Previous studies showed that formal techniques for Test Model Generation (TMG) involved overhead of intermediate formalism, were time consuming and suffered from combinatorial explosion. In contrast to formal techniques, Search-based Software Testing (SBST) demonstrated effective and efficient TMG. SBST relies on search algorithms guided by heuristics and a Fitness Function (FF) defined using different coverage criteria targeting model constraints. Previously, FF based on weaker criteria such as Decision Coverage (DC) has been widely studied. Few studies cater stronger condition-based coverage criteria only by reusing DC's FF. This results in inadequate coverage for stronger criteria that are often mandated as standards in MDE industries. To better cater condition-based criteria, we propose a five-step approach employing SBST. A novel condition-based FF is also proposed. Modified Condition/Decision Coverage (MC/DC) is selected as the coverage criterion. Alternating Variable Method (AVM) is selected as the search algorithm. An existing tool named EsOCL is extended to realize our approach. Two case studies of varying complexity are used to evaluate our approach in terms of coverage and success rate. Our condition-based FF is compared with the widely studied DC's FF. Results are verified by means of an extensive analysis. Our results demonstrate a significant improvement of ~36.2% in terms of coverage and ~0.3% in terms of success rate. Our proposed approach advocates for the efficacy of our condition-based FF which delivers promising results, ranging from weaker to stronger coverage criteria, in comparison to existing DC's FF.

Key Words: *Model Transformations, Validation, DC, MC/DC, SBST, Fitness Function*

Table of Contents

List of Figures

List of Tables

CHAPTER 1: INTRODUCTION

This chapter introduces some of the major concepts that build the foundation for our research work. These concepts are briefly described along with the challenges that exist as open problems in the field and require further attention. The scope of the research work is also set within this chapter. Problem statement is identified and idea of the proposed solution is presented. The contributions made in the light of this research are highlighted, concluding with the organization of the upcoming chapters.

1.1 Model-Driven Engineering (MDE)

MDE is a software engineering paradigm in which models are treated as the primary artifacts. Models abstract away complexities of a system and can represent a system at various abstraction levels ranging from Platform Independent Models (PIMs) to Platform Specific Models (PSMs). Moreover, different dimensions of a system can be represented using structural, behavioral and deployment models. These models enable automated generation of several other artifacts of the Software Development Lifecycle (SDLC) such as code, test cases, user documentation among others. In contrast to traditional SDLC, MDE offers abstraction, reusability and scalability as well. Owing to its benefits, it is extensively used in critical systems such as avionics and automotive industries.

1.1.1 Model Transformations (MTs)

MDE involves the systematic engineering of models by means of Model Transformations (MTs). MTs are the "heart and soul" of MDE [1]. They transform input models to output models following a set of transformation rules and satisfying certain constraints. This transformation process is the key to automation.

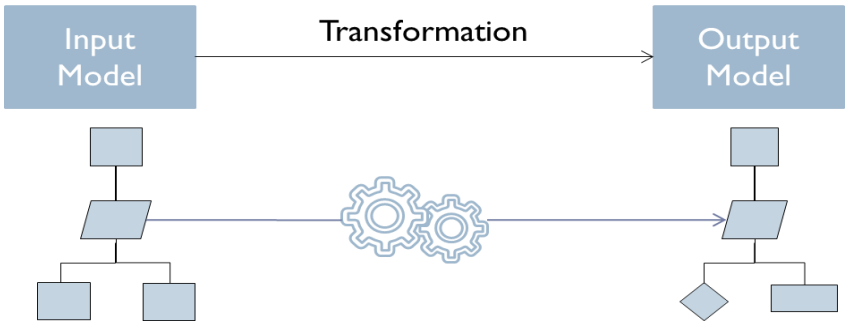

Figure 1.1: A Simple Visualization of a Model Transformation (MT)

Generally, MTs are classified into two main types: Model-to-Model (M2M) transformations and Model-to-Text (M2T) transformations [2]. In M2M transformations, the input as well the output is a model. Whereas, in M2T transformations, the input is a model but the output is text[1] which can represent code, test cases or documentation.

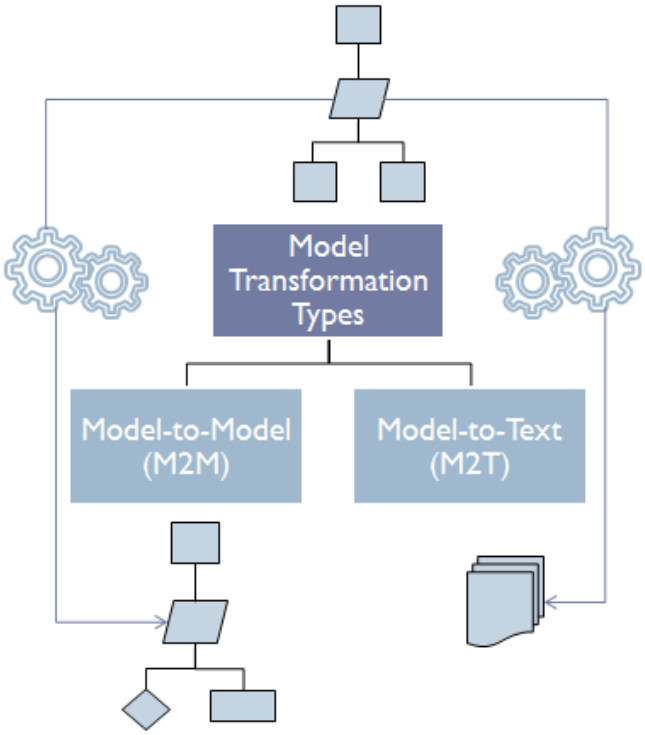

Figure 1.2: Types of Model Transformations (MTs)

1.1.2 Validating MTs

MDE involves the systematic engineering of models via MTs till the desired outcome is achieved. The output model of one transformation can serve as the input model to another transformation whose output model can then again serve as input model to another transformation, hence resulting in a chain of MTs. It is, therefore, imperative to establish validity of MTs in order to ensure credibility of the MDE process chain.

However, validating MTs is not an easy task. As the input and output under consideration are models. Models are complex data structures comprising of attributes, operations and associations of various cardinalities. Moreover, models also

[1] The output of M2T transformations can also considered as a model provided that it conforms to a certain structure and semantics.

need to conform to the structure and semantics of a Meta-Model (MM), which is defined at a higher abstract level, for a model to be declared as a valid instance of that MM.

Apart from the inherent structural complexity of models, they are also annotated with constraints that enrich the structural or behavioral elements of the model with additional details. These constraints can manifest in the form of pre-conditions, post-conditions or invariants that need to be satisfied throughout various stages of the transformation process. A model satisfying these constraints would be considered as a well-formed model.

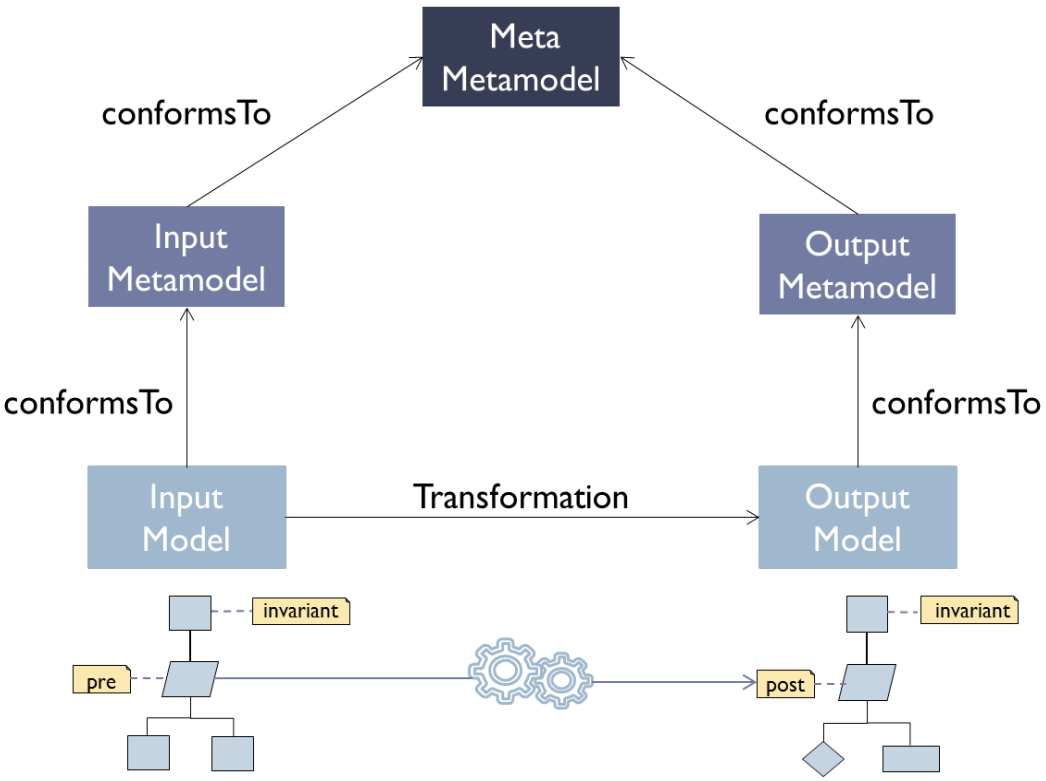

Figure 1.3: An Extended Visualization of a Model Transformation (MT)

In MDE, to represent these models, a well-known industrial standard [3] such as the Unified Modeling Language (UML) is used [4]. To complement UML, the constraints are most commonly expressed using another standard known as the Object Constraint Language (OCL) [5], which is a formal declarative language [6] to expresses constraints in the form of decisions and conditions.

1.1.2.1 Process of Validating MTs

The validation process for MTs consists of three main steps [7] that are enlisted below and briefly described in the figure that follows:

- Test Model Generation (TMG)
- Quality Assessment
- Oracle Checking

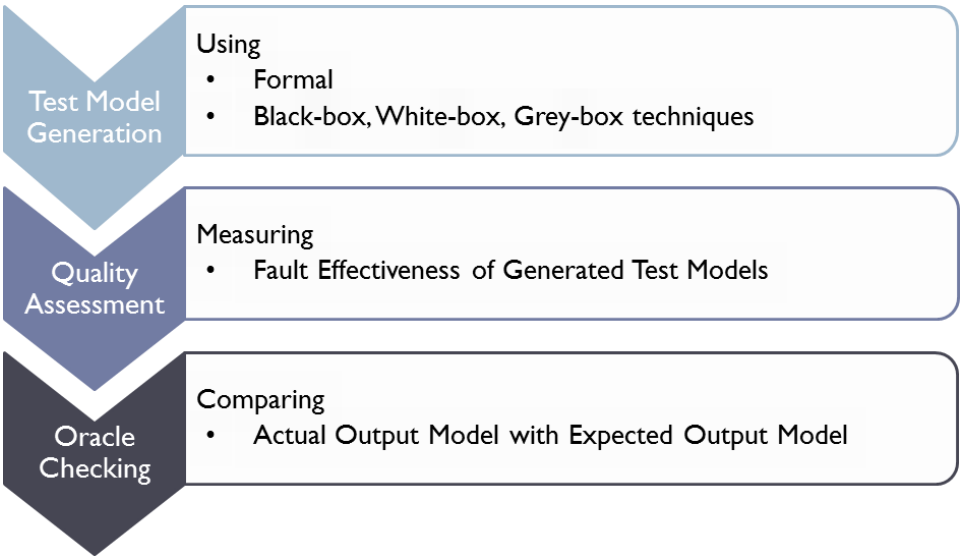

Figure 1.4: Validation Process for MTs

In order to initiate the validation process, we would limit our scope to focus on the very first step of the validation process i.e. TMG, which is a known problem in MDE.

1.1.2.1.1 Techniques for TMG

Common techniques for TMG make use of formal approaches such as Boolean SATisfiability (SAT) [16], Satisfiability Module Theories (SMTs) [17], Theorem Proving or Automated Reasoning [18], Model Checking [19] [20], Symbolic Execution [21] to name a few. According to [15], in all these approaches, what is common is the conversion of constraints from one formalism to another in order to determine test models satisfying the given constraints. Apart from that, it is time consuming to determine a test model satisfying a constraint within reasonable time. Combinatorial explosion is also a common consequence when using these approaches as the constraints grow in complexity.

Other than the aforementioned formal techniques, there are three validation[2] views for MTs [7] that are inspired from traditional software testing [8] but adapted in the context of MDE. These validation views are mentioned and described below:

- Black-box (BB) Validation
- White-box (WB) Validation
- Grey-box (GB) Validation

For BB validation, the implementation of MT is not available. However, the input model to the MT is available. This input model conforms to an input meta-model (MM) to be considered as a valid model (instance of that MM) [2]. Moreover, it may also be annotated with constraints that determine the well-formedness of that model [2]. So, the input model along with the well-formedness constraints is the artifact under consideration for BB validation.

For WB validation, implementation of the MT is available. Different types of model transformation languages (MTLs) implement the transformation specifications as a set of rules that map an input model to an output model. These MTLs can either be declarative, imperative or hybrid [2] in nature based on the constructs offered by the MTL to implement the model transformation. Hence, apart from the input model and constraints, the internal structure of the MT or its implementation is under consideration for WB validation.

Grey-box (GB) validation implies a combination of BB and WB validation. For GB, input model with annotated constraints and expected output model is available, similar to BB validation. However, the MT implementation may only be partially available [7], which makes it similar to WB validation. Apart from that, traces of MT execution [9] may also be available if supported by the Model Transformation Language (MTL) used to implement that MT. Traces map the MT rules or specifications to input model elements [9]. Apart from the input model and constraints, the available implementation of MT along with the traces, if any, is the artifact under consideration for GB validation.

[2] In the context of our research work, we consider Validation and Testing synonymous.

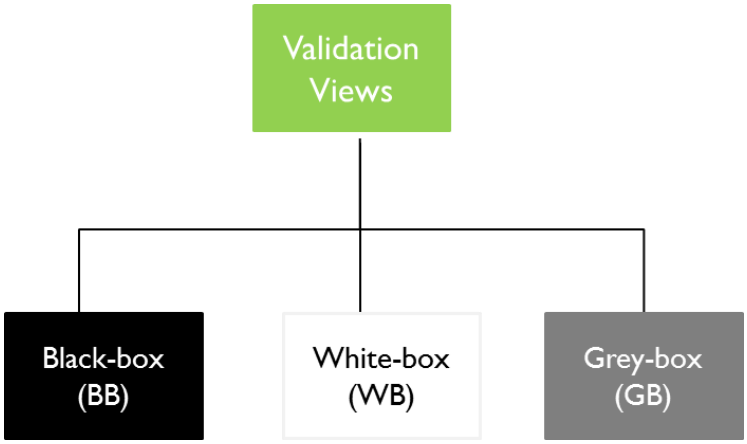

Figure 1.5: Validation Views for MTs

Considering the aforementioned validation views, coverage criteria are often utilized for both BB and WB validation in the context of model constraints which consists of Boolean expressions. These criteria are inspired from the Control-Flow Graphs (CFGs) and Data-Flow Graphs (DFGs) [22] commonly used for traditional software testing. Statement Coverage, Decision (or Branch) Coverage (DC), Condition Coverage (CC) are commonly known CFG-based coverage criteria, often used in the context of MDE as well. The figure that follows enlists those criteria that are either often employed or can be further investigated for TMG problem.

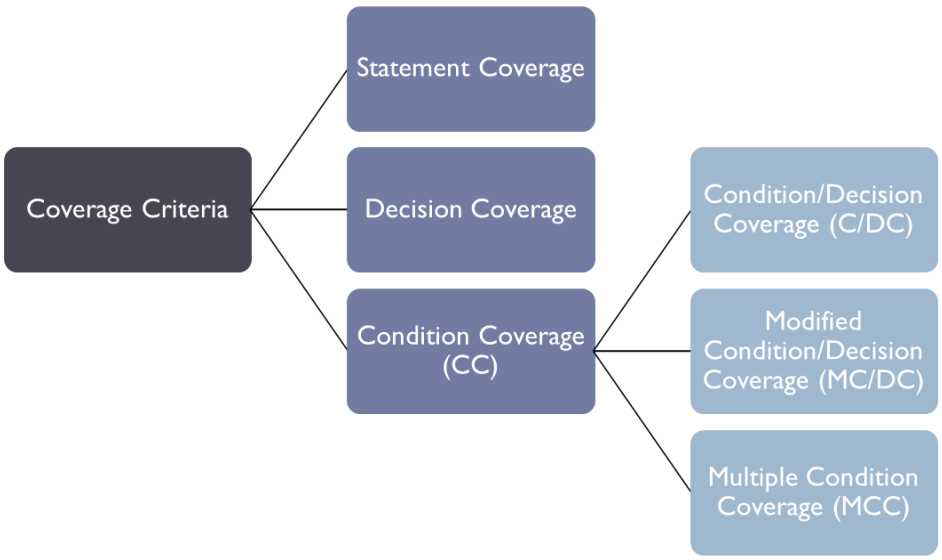

Figure 1.6: Coverage Criteria for Validation

A comparison of coverage criteria is depicted in the following table as we move from weaker to stronger criteria from left to right.

Table 1-1: Coverage Criteria Comparison in terms of Criterion Requirements from Weakest to Strongest Criteria (adapted from [41])

S. No.	Coverage Criteria	Statement Coverage	Decision/ Branch Coverage (BC)	Condition Coverage (CC)	Condition/ Decision Coverage (C/DC)	Modified Condition/ Decision Coverage (MC/DC)	Multiple Condition Coverage (MCC)
1	Every entry & exit point invoked atleast 1x		✓	✓	✓	✓	✓
2	Every statement invoked atleast 1x	✓					
3	Every decision exercised all possible outcomes atleast 1x		✓		✓	✓	✓
4	Every condition in a decision exercised all possible outcomes atleast 1x			✓	✓	✓	✓
5	Every condition in a decision independently affected that decision's outcomes					✓	✓
6	Every combination of condition outcomes in a decision, invoked atleast 1x						✓
		Weakest					Strongest

1.2 Problem Statement

Our research work addresses the problem of TMG for validation of MTs. The input model to a MT is inherently a complex structure owing to the attributes and associations by which the model elements are specified and linked to one another respectively. Moreover, this structural complexity of models increases manifold when they are annotated with constraints to express additional dependencies on the models. Hence, the complexity of models and imposed constraints remains a bottleneck for validation of MTs.

The validation approaches that address this problem for automated generation of test models rely on techniques that involve conversion of model constraints to intermediate formalisms. It can also be time consuming to find a test model in a given search space satisfying the given constraints. Moreover, another limitation is that

often such techniques suffer from combinatorial explosion as the number of operands in a constraint increases. Our research work aims to overcome the identified problems by making use of a technique that can overcome the aforementioned limitations.

1.3 Proposed Solution

For validation of constraints imposed on a model, one of the techniques that is recently gaining momentum is that of Search-based Software Testing (SBST) [23]. SBST does not suffer from the limitation of conversion to intermediate formalisms as the search algorithms are applicable to the artifact under consideration as-is. Moreover, restricting the search space of possible test models can overcome the problem of combinatorial explosion [15]. The search algorithms are able to find test models satisfying the respective coverage criteria within a reasonable amount of time [15] in comparison to formal approaches.

For coverage criteria that is often utilized for validation of MTs, SBST has already been applied by utilizing Decision Coverage (DC) [15], [24], [25], however, there are many other coverage criteria that can be utilized in the context of MT validation such as various condition-based criteria [26].

Owing to the benefits mentioned above in using SBST for TMG, this technique is further explored in detail and forms a part of our proposed solution. The search algorithms under consideration for SBST are of the following three types based on exploration of the search space. Some commonly known algorithms following under each search category [27] are also mentioned:

- Local Search
    - Hill Climbing (HC), Alternating Variable Method (AVM), Simulated Annealing (SA)
- Global Search
    - Genetic Algorithm (GA), Particle Swarm Optimization (PSO)
- Hybrid Search
    - A combination of both Local Search and Global Search such as:
        - Genetic Algorithm (GA) along with Particle Swarm Optimization (PSO)

For a given search space that consists of all possible test models, the search algorithm is guided by means of heuristics [28]. These heuristics can be expressed in the form of distance functions that help determine the distance of a searched test model in the search space with respect to the given constraint i.e. they calculate how far is a test model from satisfying the given constraint. Additionally, a fitness function (FF) [28], which makes use of these heuristics helps in quantifying the quality of the searched test model based on the distance calculation and assists the search algorithm in moving towards the direction of the fittest test model that can satisfy the given constraint. FFs are problem dependent and can vary based on the coverage criteria which is under consideration for the constraints. Success rate [29] determines how many times the search algorithm has been successful in finding a test model that satisfies the test case [15]. It is calculated using the equation below:

$$Success\ Rate = {}^{No\ of\ times\ Search\ Algorithm\ found\ a\ Test\ Model}/_{100}$$

Success rate is averaged over a 100 runs as a standard practice [14], [24] in order to accommodate for the random nature of the search algorithm.

In terms of coverage criteria, there are many other coverage criteria identified above that can be utilized in the context of MT validation such as various condition-based criteria. Coverage percentage [29] for an OCL constraint is determined by how many test cases in the coverage criterion's test suite have been satisfied by finding the fittest test model. It is calculated using the equation below (adapted from DC):

$$Coverage = ({}^{No\ of\ Test\ Cases\ Satisfied}/_{Total\ No\ of\ Test\ Cases}) * 100$$

Effectiveness of SBST techniques for MT against these other various coverage criteria is yet to be investigated.

1.4 Contribution

Mentioned below is a list of contributions made in the context of TMG for validation of MTs using SBST:

- A five-step approach is devised for this purpose which holistically solves the constraint in the context of the input model, generating test model instance(s) satisfying that constraint as per the selected coverage criteria.
- A novel condition-specific FF has been proposed for stronger coverage criteria such as CC, C/DC, MC/DC and MCC, while also remaining compatible with weaker criteria such as DC.
- This condition-specific FF is also compatible with weaker criteria such as DC according to the subsumption relationship as stronger criteria subsume weaker criteria.
- An existing tool [25] which implements SBST in the context of MTs, is extended to cater to stronger coverage criteria by implementing our novel condition-based FF.
- Comparison between existing approach and our approach is performed where we have demonstrated improvement in terms of:
    - Coverage %
    - Success Rate
    - Along with reduction in False Positives

1.5 Thesis Organization

The rest of the thesis is organized as follows:

- Chapter 2 presents the literature review that covers various techniques for different validation views of MTs, identifies the research gaps and enlists the research questions that would derive our research while focusing on the TMG problem.
- Chapter 3 presents the proposed methodology by devising our five-step approach for TMG using SBST for the MC/DC criterion. Our novel FF is also introduced. Our approach is demonstrated on an example whereby which we also compare our novel FF with the existing FF.
- Chapter 4 presents the validation of our five-step approach using two case studies. Brief details regarding the extended tool are also shared. Results obtained against each case study in terms of coverage and success rate are also enlisted along with the answers to our research questions.
- Chapter 5 discusses the implication of our results in the light of the analysis performed. Moreover, overall limitations of our approach along with possible mitigations are also mentioned.
- Chapter 6 offers suggestions for future work that may serve as potential research opportunities and concludes the thesis by reiterating our problem, proposed approach, novelty, comparison and major findings.

CHAPTER 2: LITERATURE REVIEW

Validation of model transformations (MTs) is an actively researched area making use of Black-box (BB), White-box (WB) and Grey-box (GB) validation [7]. These validation views are greatly inspired from traditional software validation [8]. However, to be applicable in the domain of MDE, these techniques are adapted to suit the needs of various challenges presented by models and model transformations, as explained in Chapter 1. Although our research work focuses on BB validation of MTs for test model generation, for the sake of completeness, we also touch upon other validation views focusing on test model generation techniques.

2.1 Validation for MTs

2.1.1 Black-box (BB) Validation

For BB validation of MTs, different coverage criteria based on the input model have been defined. The model considered can either be a structural model such a Class Diagram or a behavioral model such as a State Machine Diagram. For instance, in case of a Class Diagram, covering all-classes, all-attributes, all-associations [22] constitute some of the coverage criteria based on the structure of the model. Similarly, for a State Machine Diagram, covering all-states, all-transitions, all-paths [22] constitute some of the coverage criteria based on the model structure.

These structural coverage criteria are not only extensively utilized in the context of Model-based Testing (MBT) [22] but are also extensible in the context of BB validation of model transformations [7] where the input model to a MT is the artifact under consideration for test model generation. Some of the commonly known techniques in MDE literature for coverage of model structures are adapted from traditional software testing techniques. For instance, techniques such as Input Domain Reduction [15], Partitioning [13] and Boundary Value Analysis [14] have also been utilized for structural coverage of complex and large-scale models.

For input models annotated with constraints, apart from the coverage criteria relying merely on model structure, the structure of constraints can also be utilized for BB validation by covering the decisions and conditions involved in the constraints. This leads to the coverage criteria common in traditional software testing such as decision coverage and condition coverage among others [22] inspired from control-

flow testing of software programs. Such coverage criteria have often been explored for generating input models of MTs [7], [22]. Apart from the formal techniques already identified in Chapter 1, few authors [15], [26] have also utilized SBST techniques for generating test models for MTs with promising results that outperform the formal approaches.

2.1.2 White-box (WB) Validation

For WB validation of MTs, different coverage criteria based on the internal structure of the MT, have been defined. Such coverage criteria are again inspired from traditional WB software testing techniques. Some of the commonly used criteria rely on abstract representations such as CFG and DFG that are adapted to be applicable for MT implementation. Decision, condition and path coverage of MTs that has been reported mostly utilizes techniques such as constraint analysis [13].

For the input models, different strategies have been devised to derive test models when the MT is available such as Effective Meta-Model (EMM) [10] followed by the Extended Effective Meta-Model (EEMM) [11].

For input models annotated with constraints, apart from the coverage criteria relying on MT structure, the structure of constraints can also be utilized to generate test models that perform effective WB validation by covering the decisions and conditions involved in the constraints. Moreover, MTLs that are imperative and hybrid in nature often express constraints within the MT implementation using helpers and rule filters. Due to this commonality, BB validation for MTs can also be extended to carry out WB validation. This leads to coverage criteria such as decision and condition coverage for constraints in the MT as well [24]. Apart from the formal techniques already identified in Chapter 1, few authors [24], [30]* have also utilized SBST techniques for generating test models for WB validation of MTs that offers promising results in comparison to formal approaches.

2.1.3 Grey-box (GB) Validation

For GB validation, both BB and WB techniques can be leveraged either separately or in combination according to the artifacts available. It focuses more on fault identification, detection and localization by exploiting the traces that are available [9]. However, no techniques unique to GB test model generation have been

identified. More commonly, existing BB or WB techniques are utilized for this purpose [7].

2.2 TMG using SBST

From among the techniques that are mentioned above, SBST techniques for test model generation have stood out in contrast to other techniques that suffer from limitations as described in Chapter 1. Being a relatively newer research area in the domain of MDE, strength is that they have also been applied in industrial contexts where models can be large and scalable with complicated constraints and where search space for test models satisfying those constraints, can be adjusted accordingly to gain promising results.

Even though, SBSE has been applied to other problems such as searching for optimal MT in a given search space [31], we however, explicitly focus on the TMG works identified above [11], [14], [24], [25], [30] where SBST has been applied for validation of MTs by utilizing constraints annotated on the input model or occurring within the MT. Strengths and limitations of identified works as described in the table below:

Table 2-1: A Comparison of Existing Work based on SBST for TMG

Year, Author	Approach Followed	Strengths	Limitations
2013, S Ali	• SBST for BB Validation of MTs • BC	• Extensible to Model-based Testing (MBT) & MTs (in WB context)	• Heuristics & fitness function specific to BC; not scalable to stronger criteria
2014, AAA Jilani	• SBST for WB Validation of MTs • BC	• Existing BC fitness equation is improved by utilizing WB-specific objective function	• Limited generalizability due to Language-specific constructs of MTs • Limited Scalability due to (test) model instantiation by Effective Meta-Model (EEFM)
2016, S Ali	• SBST for BB Validation of MTs; • BC; • complemented by dependency analysis at the condition level	• Simplifies the process of determining BC; in terms of fewer test cases • Suited to constraint languages that support short-circuiting	• Limited generalizability as Object Constraint Language (OCL) does not support short-circuiting
2019, H Sartaj	• SBST for BB Validation of MTs; • MC/DC; • complemented by Case-based Reasoning	• Extensible to Model-based Testing (MBT) & MTs (in WB context) • Stronger coverage criterion • Reusing existing solution of test cases for newer similar test cases	• simpler fitness function is reused for stronger criteria; no condition-level equation

Followed below is a look into the intricacies of SBST and comparison of existing work based on the FF used.

Table 2-2: A Comparison of Existing SBST Work based on FF for TMG

Year, Author	Validation View	Technique Used for TMG	Coverage Criteria	Fitness Function (FF)
2013, S Ali	Black-Box (BB)	SBST	Branch Coverage (BC)	Branch-based
2014, AAA Jilani	White-Box (WB)	SBST	Branch Coverage (BC)	Branch-based (Improved for WB)
2016 S Ali	Black-Box (BB)	SBST	Branch Coverage (BC)	Branch-based (Improved)
2019, H Sartaj	Black-Box (BB)	SBST	Modified Condition/Decision Coverage (MC/DC)	Branch-based

From the works that we have shortlisted above, it is observed that a BB approach can also be applicable in the WB context. Infact, for WB approach, the FF is indeed derived from BB approaches but improved as per the needs of WB validation of MTs.

2.3 Research Gaps

Considering the above table, it has been identified that the existing work [11], [14], [24], [25] whether BB or WB in nature, mostly focuses on simple coverage criterion such as Branch Coverage (BC) (also known as Decision Coverage (DC)) is considered. To the best of our knowledge, only one paper [26] goes a step further and considers stronger coverage criterion such as Modified Condition/Decision Coverage (MC/DC).

For DC, the fitness functions to guide the search algorithms are designed adequately as per the decision-level. However, even for MC/DC, the fitness function used is still that of DC. As fitness functions are problem dependent, reusing a fitness function that reflects DC may not perform adequately when dealing with a different criterion such as MC/DC that leverages both DC as well as condition coverage (CC). As identified, no specific fitness function has been proposed that deals with constraints at the condition-level; as needed for MC/DC. This highlights the need to

further utilize SBST techniques for coverage of stronger criteria in the context of BB validation of MTs.

2.3.1 Research Questions (RQs)

For our research work, we would focus on BB validation of MTs as BB approach is scalable to WB approach as mentioned earlier. Following research questions have been devised for BB TMG of MTs based on the research gaps identified above:

RQ1: Can a condition-based FF achieve better coverage for MC/DC criterion in comparison to reusing a branch-based FF?

RQ2: Can a condition-based FF achieve better success rate for MC/DC criterion in comparison to reusing a branch-based FF?

We will aim to answer these RQs by devising our 5-Step proposed approach with our novel condition-based FF which would be demonstrated and validated against a two case studies in the upcoming chapters.

CHAPTER 3: PROPOSED METHODOLOGY

Generation of test models is the initial step that leads towards BB validation of MTs. Our proposed methodology generates test models by satisfying MC/DC criterion for OCL constraints. We utilize SBST guided by our novel condition-based FF to achieve such test models. The following sections enlist and describe the various steps to realize our approach.

3.1 5-Step Approach for MC/DC TMG

We generate test models as per MC/DC of OCL constraints by utilizing our five-step approach which is briefly described below:

1st Step: Taking Input

- UML Model
- OCL Constraint

2nd Step: Parsing

- Decisions
- Conditions
- Operands (Variables and/or Constants)
- Logical and Relational Operators

3rd Step: Determining MC/DC Test Suite

- MC/DC (Unique-Cause) Criterion

4th Step: Applying Search Algorithm

- AVM

5th Step: Returning Output

- Test Model
- Coverage
- Success Rate

Each of the previously mentioned steps is elaborated as follows:

3.1.1 Taking Input

The first step of the five-step approach starts with taking a UML model and OCL constraint defined in the context of the same model. The input model is specified by its file path ending with the .uml extension whereas the constraint to the input model is defined as a string. At present, UML class diagram model is supported along with one OCL constraint at a time.

3.1.2 Parsing

After taking the input, the second step involves the process of parsing it. The OCL constraint to be resolved in the context of the input model is parsed into decisions, conditions, operands (variables and/or constants), logical and/or relational operators.

3.1.3 Determining MC/DC Test Suite

The MC/DC test suite is determined for the parsed OCL constraint. This requires generating a truth table for the decision under consideration and identifying the independence pairs for each condition of that decision. An independence pair for a condition helps in locating those test cases from the truth table whereby the condition is solely responsible for affecting the decision's outcome, hence the said condition being the unique-cause for a change in the corresponding decision's truth value. These independence pairs once determined for all conditions in a decision are combined to form the MC/DC test suite.

3.1.4 Applying Search Algorithm

The fourth step of the approach is the application of search algorithm to find test models that satisfy the MC/DC test suite obtained against an OCL constraint. The algorithm is applied against one test case at a time till all the test cases have been covered in the test suite. AVM, a local search algorithm [32], [15], [25], [11], [26] is chosen to find a test model in the search space which satisfies the test case under consideration. A set of distance functions (heuristics) used at the condition-level of an OCL decision is given by the table below:

Table 3-1: Condition-level Distance Functions to Guide the Search Algorithm [15]

Condition c	**Distance Function d(c)**
x = y	if (x-y = 0) then d(c) = 0 else d(c) = abs(x-y)
x <> y	if (x-y <> 0) then d(c) = 0 else d(c) = abs(x-y) + k
x < y	if (x-y < 0) then d(c) = 0 else d(c) = abs(x-y) + k
x > y	if (x-y > 0) then d(c) = 0 else d(c) = abs(x-y) + k
x <= y	if (x-y <= 0) then d(c) = 0 else d(c) = abs(x-y) + k
x >= y	if (x-y >= 0) then d(c) = 0 else d(c) = abs(x-y) + k

Where k is the smallest positive integer i.e. k = 1.

Our novel condition-based FF is proposed and described below:

$$FF = \sum_{c \in C_D} d(c,\ t)$$

Where

- FF: Fitness Function (to be minimized)
- D: decision
- C_D: set of conditions in decision D
- c: a single condition in C
- t: a test case for c
- d(c, t): distance function for condition c, provided a test case t

This is the step which distinguishes our condition-based FF from the existing branch-based FF [26]. A comparison of both would be demonstrated through an example in the upcoming sections.

The above process (1st Step to 4th Step) is repeated as such for each test case of the MC/DC test suite.

3.1.5 Returning Output

The last step of the five-step approach is returning the fittest test model that is obtained against a test case of the MC/DC test suite for an OCL constraint. The fittest test model is the one for whom the fitness function is minimized. For such a test model, the distance with respect to the test case under consideration is also reduced to a minimum. Hence, this test model satisfies the OCL constraint, one MC/DC test case at a time. Apart from the test model(s), the success rate of the search algorithm and the coverage of the MC/DC test suite is also returned.

The five-step approach elaborated above takes a UML class diagram model along with its OCL constraint and searches for a test model that is able to satisfy the given OCL constraint as per the MC/DC criterion.

Figure 3.1: Our 5-Step Proposed Approach for MC/DC TMG using SBST

3.2 Example

We demonstrate our 5-step proposed approach on an example below:

1st Step: Taking Input

Consider the following UML experimental model consisting of 2 classes A and B for which attributes of different class and data types have been declared. A and B are connected to one another by means of a bi-directional association link. Moreover, an OCL constraint has been defined in the context of B, involving integer and double data type attributes. The experimental model annotated with the constraint would serve as the input.

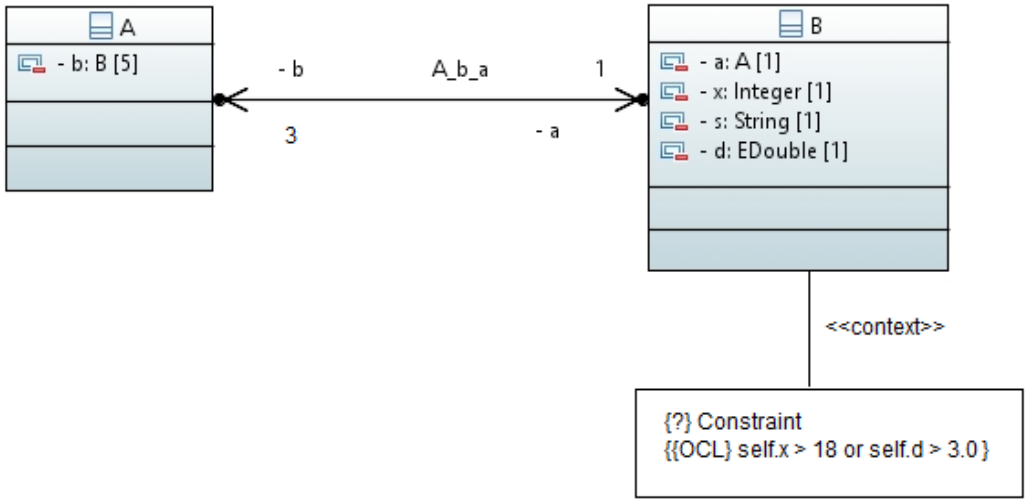

Figure 3.2: 1st Step - UML Experimental Model Annotated with an OCL Constraint, Serving as Input

2nd Step: Parsing

For the constraint in the given example (1st Step), which would need to be resolved in the context of B, consists of a Boolean expression serving as one single decision. This decision consists of two conditions. The first condition involves a comparison of integer variable (attribute) involving the '>' relational operator, while the second condition involves a comparison of double variable (attribute), also involving the '>' relational operator. These conditions are conjuncted by means of an 'or' logical operator (at the decision level). Details such as the left hand side (LHS) and right hand side (RHS) of each condition and decision, serving as operands, respective operators involved and the operands (variables and/or constants) extracted for test data generation as a result of parsing are mentioned in the following table:

Table 3-2: 2nd Step - Parsing of OCL Constraint with the Extracted Details

OCL Constraint	Decision(s)	1	Condition(s)	2
Context B inv self.x > 18 or self.d > 3.0	Logical Operator(s)	or	Relational Operator(s)	>
	LHS (operand1)	self.x > 18	LHS (operand1)	self.x
			RHS (operand 2)	18
	RHS (operand 2)	self.d > 3.0	LHS (operand1)	self.d
			RHS (operand 2)	3.0
Variable(s) (for Test Data Generation)			self.x self.d	

3rd Step: Determining MC/DC Test Suite

For the parsed constraint (2nd Step), the truth table is given by the following table. Each row of the truth table serves as a test case. For condition 1, row 2 and 4 are the identified independence pairs while for condition 2, row 3 and 4 serve as the independence pairs. These independence pairs or MC/DC test cases after combining together consist of rows 2, 3 and 4 resulting in the MC/DC test suite.

Table 3-3: 3rd Step - MC/DC Test Suite Determined for the Parsed OCL Constraint

Test Case #	Condition 1 self.x > 18	Condition 2 self.d > 3.0	Decision self.x > 18 or self.d > 3.0	Independence Pairs
1	T	T	T	
2	T	F	T	(2, 4)
3	F	T	T	(3, 4)
4	F	F	F	
MC/DC Test Suite		{(2, 4) U (3, 4)} = {2, 3, 4} i.e. {TF, FT, FF}		

4th Step: Applying Search Algorithm

For the MC/DC test suite (3rd Step), each of the three test cases when substituted against the given constraint result in three corresponding constraints as depicted in the following table. For each corresponding constraint (or a test case), the operators, heuristics and FF to guide AVM towards a test model satisfying this constraint are also mentioned. How our condition-based FF approach (green) differs from the existing branch-based FF approach (red) [26] for the 'or' logical operator involved in the corresponding substituted constraints, is also depicted in the following table.

Table 3-4: 4th Step (1/2) – Heuristics and FF (our vs. existing [26]) to guide AVM against each Substituted Constraint corresponding to each Test Case of the MC/DC Test Suite for TMG

[1st Step] OCL Constraint		[2nd Step] Variable (s)	[3rd Step] MC/DC Test Suite		Heuristics (Distance Calculation)		
			Test Case(s)	Substituted OCL Condition(s)	Operator (s)	T	F
Decision	self.x >18 or self.d > 3.0	self.x self.d	TF	self.x >18 or self.d <= 3.0	>	if (LHS > RHS) then d(condition) = 0	else d(condition) = abs(LHS-RHS)+1
Condition 1	self.x >18		FT	self.x <=18 or self.d > 3.0	<=	if (LHS <= RHS) then d(condition) = 0	else d(condition) = abs(LHS-RHS)+1
Condition 2	self.d >3.0		FF	self.x <=18 or self.d <= 3.0	or	Existing Branch-based Approach	min [d(condition 1), d(condition 2)]
						Our Cond-based Approach	d(condition 1) + d(condition 2)

For the sake of completeness, we also compare our condition-based FF approach with the existing branch-based FF approach [26] for one of the substituted constraints with respect to the given constraint, details of which are as follows:

- OCL Constraint: self.x >18 or self.d > 3.0
- Test Case: self.x >18 or self.d <= 3.0 (TF)
- Search Algorithm: AVM
- Assume (initial test data): x = 20, d = 3.5

The calculated distance and the result of both the FFs are illustrated in the following figure.

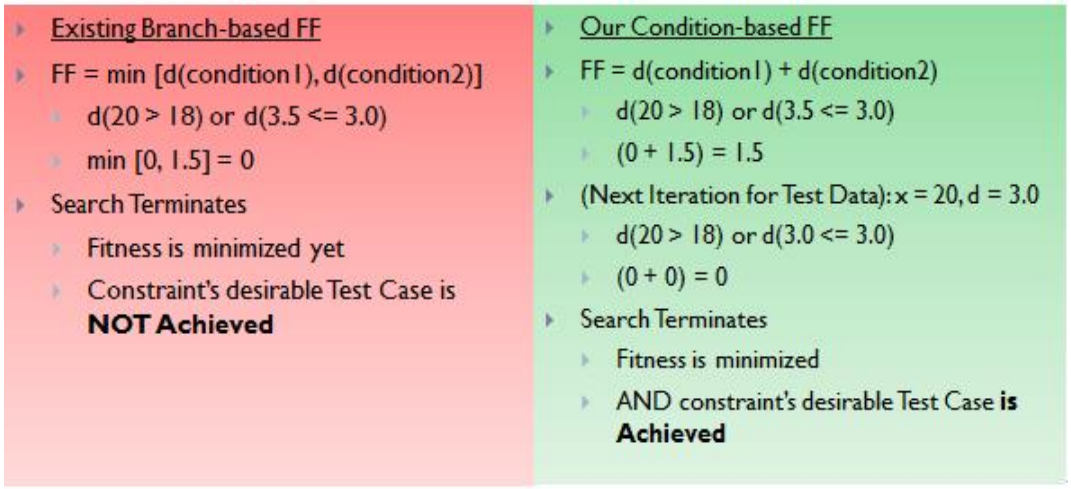

Figure 3.3: 4th Step (2/2) – A Comparison of Our Condition-based FF (green) with Existing Branch-based FF (red) [26] for one of the Substituted Constraints/Test Cases

The main distinguishing feature of our approach is that it helps eliminate the false positives that are garnered as a result of using the existing branch-based FF, due to which the test model returned by AVM in the latter case does not satisfy the desirable substituted constraint/test case.

5th Step: Returning Output

The test model satisfying one of the test cases is returned as an output whereby which test data is generated for integer and double variables.

```
Constraint:  inv initial: self.x > 18 or self.d > 3.0
Search Algorithm: AVM
Result: true
Iterations:2

****************** ModelInstance: 1 ******************

Class Name: root::Test2::B
    Attribute Name: root::Test2::B::d  ------  Attribute Type:root::double ------  Attribute Value: 7.18689
    Attribute Name: root::Test2::B::s  ------  Attribute Type:root::String ------  Attribute Value: null
    Attribute Name: root::Test2::B::x  ------  Attribute Type:root::Integer ------  Attribute Value: 58.0
    Attribute Name: root::Test2::B::a  ------  Attribute Type:root::Test2::A

    Class Name: root::Test2::A
        Attribute Name: root::Test2::A::b  ------  Attribute Type:root::Test2::B ------  Attribute Value: null
```

Figure 3.4: 5th Step – Test Model Returned (using our FF) against an MC/DC Test Case/Substituted Constraint

Output results that help determine success rate and coverage are elaborated in the upcoming chapters.

CHAPTER 4: VALIDATION

In order to validate our approach, empirical evaluation is performed on two case studies by extending an existing tool. Brief details regarding the tool are mentioned and each of the case studies is described below which is followed by the results that are obtained from the returned output (i.e. the 5th Step of our proposed approach).

4.1 Tools Used

We extend an existing tool named Evolutionary Solver for OCL (EsOCL) [15] to realize our 5-Step proposed approach. To the best of our knowledge, it is the only open-source[3] tool available which implements SBST-based algorithms in the context of MDE and does so adequately. It has been developed in Java and its output is console-based. More details on this tool can be found in [15].

We only utilize a subset of this tool which pertains to our 5-Step proposed approach. It, however, already implements the existing branch-based FF [26]. We extend EsOCL to implement our novel condition-based FF, and then compare our approach and results with the already implemented FF. For execution and extension of EsOCL, we utilize the Eclipse Integrated Development Environment (IDE) and use Papyrus for visualization of our case study models and constraints.

```
private double impliesOp(IModelInstanceObject env, OclExpression leftExpression, OclExpression rightExpression) {


    double notA_bdc = notOp(env, leftExpression);       // where A decision, recursively solved the not operator
    double B_bdc = handleBooleanOp(env, rightExpression);

    // existing [not A or B]
    // return Math.min(notA_bdc, B_bdc);

    // ours
    return A_bdc + B_bdc;

}
```

Figure 4.1: Coding Interface of EsOCL where ours and existing FFs are implemented for the ‘implies’ operator

[3] https://github.com/Simula-COMPLEX/EsOCL

```
public void test01() {

    String inputModelPath = "src/main/resources/model/RoyalAndLoyal.uml";
    /**
     * Original
     * context Service inv invariant_Service1 :
     * (self.pointsEarned > 0) implies not (self.pointsBurned = 0)
     */

    String constraint = "context Service inv:  self.pointsEarned >  0  implies  not ( self.pointsBurned =  0  )

    OCLSolver oclSolver = new OCLSolver();
    try {
        Result result = oclSolver.solveConstraint(inputModelPath, constraint, new String[]{"AVM"}, 500);
        oclSolver.printResults(result);
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

Figure 4.2: A Glimpse of a Test Case in EsOCL where RoyalandLoyal can be seen as the Input Model with a Constraint for which AVM Search Algorithm is applied for 500 Independent Runs

4.2 Case Studies

Two case studies ranging from small to medium scale are selected to perform validation of our approach. One of them is a small-scale experimental model whereas the other one is a medium-scale benchmark model that is similar to an industrial-level setting. Both of them are described below along with the results (in terms of success rate and coverage) that we obtain against each of them.

4.2.1 Case Study 1: Experimental Model

First case study is an experimental model adapted from [15], [25]. It is a UML class diagram consisting of two classes namely A and B. Their structural properties are specified by means of attributes of varying class and data types. Classes A and B are associated to one another by means of a bi-directional association namely A_b_a. Each end of the association specifies its cardinality in terms of a lower and upper limit. For the sake of simplicity, the cardinality of A_b_a is fixed at one i.e. both upper and lower limit of the association is equal to one. An instance of A is associated with one and only one instance of B and vice versa. Moreover, an instance of B is owned by A and vice versa. The following figure visualizes this experimental model:

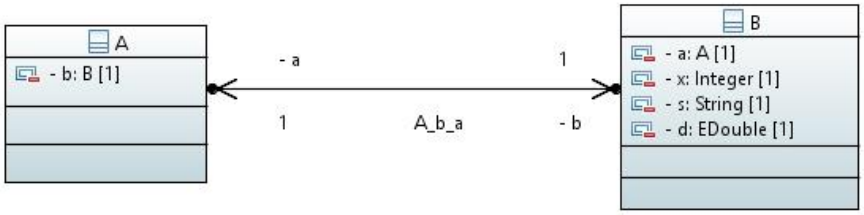

Figure 4.3: UML Class Diagram for Experimental Model

Seven OCL constraints have been defined in the context of both classes A and B. These constraints impose additional restrictions in terms of attribute values and associations between instances of A and B. This results in a total of twenty MC/DC test cases for which test models are generated. The following figure visualizes the experimental model annotated with its constraints:

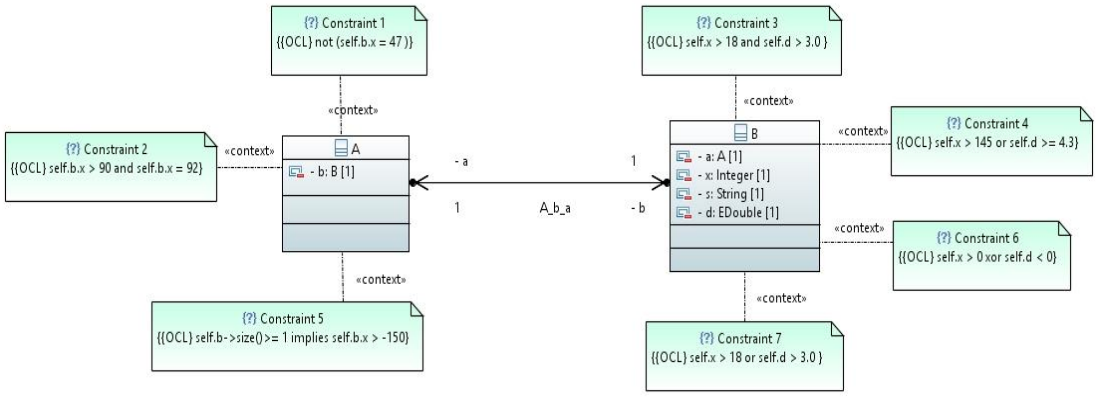

Figure 4.4: Case Study 1 - UML Class Diagram for Experimental Model Annotated with OCL Constraints

4.2.2 Case Study 2: RoyalandLoyal Model

The second case-study is a benchmark model often used for OCL constraint evaluation, named as RoyalandLoyal.uml [6], [26]. It represents a loyalty management program that facilitates program partners to offer different services to their loyal customers. It is a Class Diagram consisting of 14 classes that models different entities of the loyalty management program such as ProgramPartner, Membership, Customer, LoyaltyAccount, Service, Transaction among others. These classes are linked to one another either by simple association or by an inheritance which then establishes the superclass and subclass relationship among the linked classes. Ten OCL constraints have been specified in the context of various classes in the model covering almost each class and each association link atleast once. These constraints have been expressed by using both relational and logical operators. This results in a total of twenty nine MC/DC test cases for which test models are generated. The figure that follows shows this case-study model:

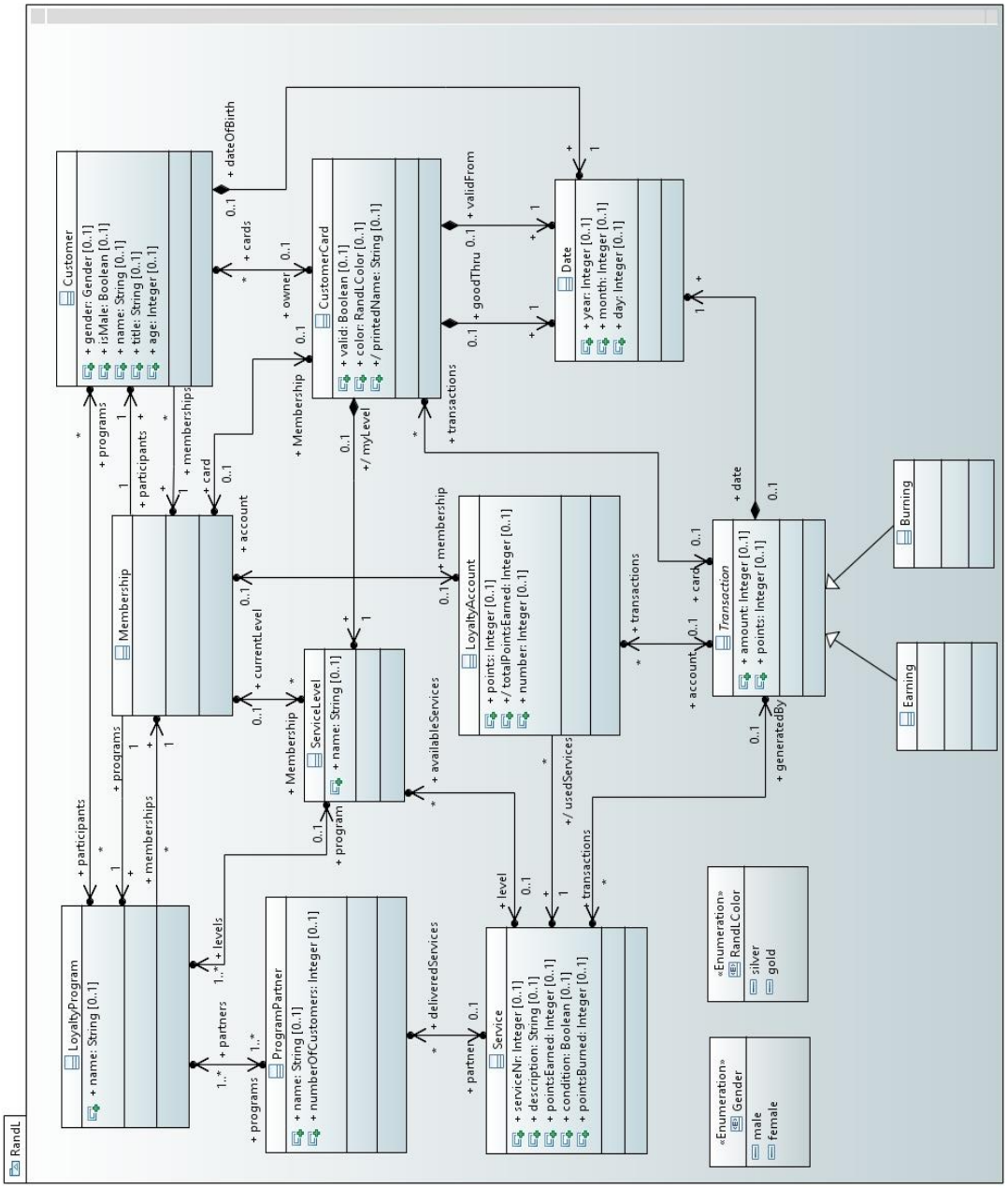

Figure 4.5: Case Study 2 - UML Class Diagram for RoyalandLoyal Model

Details of the two case-studies are summarized in the following table.

Table 4-1: Summarized Details of Two Case Studies used for Validation of our Approach

S. No.	Case-study	No. of Classes	No. of OCL Constraints	No. of MC/DC Test Cases	Relational Operators Used	Logical Operators Used
1	Experimental Model	2	7	20	>, >=,<,<=, =, <>	not, and, or, implies, xor
2	Royaland Loyal model	14	10	29		

4.3 Results

The following tables enlist the results obtained for the case studies in terms of coverage and success rate. The constraints and their corresponding MC/DC test cases are also enlisted and results obtained for our condition-based approach are compared with the existing branch-based approach [26] cumulatively and try to answer our research questions.

For the tables that follow, the rows in red highlight those constraints (or test cases) where the existing approach lagged behind our approach.

Table 4-2: Results Obtained for Case Study 1 - Comparing Our Approach with Exiting Approach [26] in terms of Coverage and Success Rate

S. No.	Constraints	MC/DC Test Cases	Our Condition-based Approach		Existing Branch-based Approach [26]	
			Coverage %	Success Rate	Coverage %	Success Rate
1	context A inv initial: not (self.b.x = 47)	T, F	100	1	100	1
2	context A inv initial: self.b.x > 90 and self.b.x = 92	TT, TF, FT	100	1	100	1
3	context B inv initial: self.x > 18 and self.d > 3.0	TT, TF, FT	100	1	100	1
4	context B inv initial: self.x > 145 or self.d >= 4.3	TF, FT, FF	100	1	~33.3	0.3
5	context A inv initial: self.b->size()>= 1 implies self.b.x > -150	TT, TF, FF	100	1	50	0.5
6	context B inv initial: self.x > 0 xor self.d < 0	TF, FT, FF	~66.6	0.6	0	0
7	context B inv initial: self.x > 18 or self.d > 3.0	TF, FT, FF	100	1	~33.3	0.3

Answer to RQ1: In terms of coverage, for case study 1 (Experimental Model), the first four rows do not record any difference between ours and existing approach where the ‘not’ and ‘and’ operators are commonly involved in the constraints. However, as we move further from row 4 to 7, where operators such as ‘or’, ‘implies’ and ‘xor’ are involved, we observe a significant difference whereby which our approach clearly outperforms the existing approach. For row 4 and 7, our approach performs ~66.6% better than the existing approach. For row 5, ours performs 50% better than the existing while for row 6, again our approach comes close to performing ~66.6% in comparison to existing approach. For all the seven constraints and twenty MC/DC test cases, our approach demonstrates an average coverage of ~95.2% in contrast to the existing approach which only demonstrates it upto ~59.5%, hence suggesting an improvement of ~35.7%.

Answer to RQ2: In terms of success rate, similar behavior is observed as both coverage and success rate are interrelated. For case study 1, the first four rows do not record any difference between ours and existing approach where the ‘not’ and ‘and’ operators are commonly involved in the constraints. However, as we move further from row 4 to 7, where operators such as ‘or’, ‘implies’ and ‘xor’ are involved, we observe a significant difference whereby which our approach clearly outperforms the existing approach. For row 4 and 7, our approach performs ~0.6% better than the existing approach. For row 5, ours performs 0.5% better than the existing while for row 6, again our approach comes close to performing ~0.6% in comparison to existing approach. For all the seven constraints and twenty MC/DC test cases, our approach demonstrates an average success rate of ~0.9% in contrast to the existing approach which only demonstrates it upto ~0.5%, hence suggesting an improvement of ~0.3%.

Table 4-3: Results Obtained for Case Study 2 - Comparing Our Approach with Exiting Approach [26] in terms of Coverage and Success Rate (1/2)

S. No.	Constraints	MC/DC Test Cases	Our Condition-based Approach		Existing Branch-based Approach	
			Coverage %	Success Rate	Coverage %	Success Rate
1	context Membership inv: self.account.points = 0 or self.account.points > 0	TF, FT, FF	100	1	~66.6	0.6
2	context LoyaltyAccount inv : self.transactions.points >= 0 and self.transactions.points >= 10000	TT, TF, FT	100	1	~66.6	0.6
3	context ProgramPartner inv: self.numberOfCustomers >= 0	T, F	100	1	100	1
4	context ProgramPartner inv: self.deliveredServices.transactions.points >= 0 and self.deliveredServices.transactions.points < 10000	TT, TF, FT	100	1	100	1
5	context Membership inv: self.account.points >= 0 implies self.account->notEmpty()	TT, TF, FF	100	1	~33.3	0.3

Table 4-4: Results Obtained for Case Study 2 - Comparing Our Approach with Exiting Approach [26] in terms of Coverage and Success Rate (2/2)

S. No.	Constraints	MC/DC Test Cases	Our Condition-based Approach		Existing Branch-based Approach	
			Coverage %	Success Rate	Coverage %	Success Rate
6	context Membership inv: self.card.color = RandLColor:: silver or self.card.color = RandLColor::gold	TF, FT, FF	100	1	100	1
7	context Service inv: self.pointsEarned > 0 implies not (self.pointsBurned = 0)	TT, TF, FF	100	1	~33.3	0.3
8	context Membership inv: self.programs.partners.deliveredServices.pointsEarned = 0 implies self.account->isEmpty()	TT, TF, FF	100	1	~33.3	0.3
9	context Membership inv: self.account.points = 0 or self.account.points > 0	TF, FT, FF	100	1	~66.6	0.6
10	context LoyaltyAccount inv: self.points > 0 implies self.transactions.points > 0	TT, TF, FF	100	1	~33.3	0.3

Answer to RQ1: In terms of coverage, for case study 2 (RoyalandLoyal Model), the rows 3, 4 and 6 do not record any difference between ours and existing approach where a single condition/decision is used or operators such as ‘and’ and ‘or’ are involved in the constraints. However, we observe different trend in row 1, 2 and 9 where ‘or’ and ‘and’ operators are again involved in the constraints. Stark difference is observed in rows 5, 7, 8 and 10 where the ‘implies’ operator is commonly used in the constraints. For row 1, 2 and 9 our approach performs ~33.3% better than the existing approach. For row rows 5, 7, 8 and 10, our approach performs significantly better than the existing approach and demonstrates an improvement of ~66.6%. For all the ten constraints and twenty nine MC/DC test cases, our approach demonstrates an average coverage of 100% in contrast to the existing approach which only demonstrates it upto ~63.3%, hence suggesting an improvement of ~36.7%.

Answer to RQ2: In terms of success rate, for case study 2, the rows 3, 4 and 6 do not record any difference between ours and existing approach where a single condition/decision is used or operators such as ‘and’ and ‘or’ are involved in the constraints. However, we observe different trend in row 1, 2 and 9 where ‘or’ and ‘and’ operators are again involved in the constraints. Stark difference is observed in rows 5, 7, 8 and 10 where the ‘implies’ operator is commonly used in the constraints. For row 1, 2 and 9 our approach performs ~0.3% better than the existing approach. For row rows 5, 7, 8 and 10, our approach performs significantly better than the existing approach and demonstrates an improvement of ~0.6%. For all the ten constraints and twenty nine MC/DC test cases, our approach demonstrates an average success rate of 1 in contrast to the existing approach which only demonstrates it upto ~0.6%, hence suggesting an improvement of ~0.4%.

From the results, it can be seen that our condition-based approach outperforms the existing branch-based approach by ~36.2% in terms of coverage or by ~0.3% in terms of success rate. We also observe how the results obtained for the RoyalandLoyal model which is an industrial-level case study are consistent with the results obtained for the Experimental model, which also justifies how the latter, albeit being small-scale is representative of medium to large-scale industrial case-studies [15], [25].

CHAPTER 5: DISCUSSION AND LIMITATIONS

This chapter presents an analysis of the results obtained in terms of coverage (or success rate). More importantly, we try to determine the reasons behind why our condition-based approach performs significantly better in comparison to the existing branch-based approach. Moreover, a few limitations regarding our research work are also enlisted along with the suggested mitigations that can help overcome them.

5.1 Discussion

From the returned output against a case study which consists of test model(s), coverage percentage and success rate, it is imperative to verify these obtained results in order to ascertain the validity of our approach. Some of the major findings resulting from a keen and extensive manual analysis of both existing and our results are discussed below with possible reasons:

- It was observed that the existing branch-based FF and our novel condition-based FF mostly demonstrated equivalent results for the ‘and’ logical operator.
    - The reason for this finding is that both existing and our FF sums up distance functions of conditions that are connected by an ‘and’ operator, hence resulting in the same results (i.e. equivalent coverage and success rate).
    - However, these results cannot be generalized as we also had constraints with the ‘and’ operator where the results were not equivalent. The existing branch-based FF performed poorly for such constraints with a drop in its performance, which can be attributed to the random nature of SBST algorithms.
- For other logical operators such as ‘or’, ‘implies’, ‘xor’, several false positives were identified in the existing results [26] whereas these false positives were overcome by our approach. A false positive occurs when the search process is terminated assuming that the fittest test model is found against a test case, however, in actual, the test model returned does not satisfy that test case.
    - The probable reason for this is that the branch-based FF used for existing results calculates the distance function according to the logical operator involved in the decision i.e. it is dependent on that logical

operator. Due to which, it operates at the decision-level. As soon as the decision is satisfied the search process is terminated irrespective of whether the test model returned satisfies the given test case or not.

- In contrast to that, our fitness function is independent of the logical operator(s) involved and hence operates at the condition-level. Due to which the MC/DC test cases, that are a combination of truth values of conditions, can be satisfied without any false positives i.e. the search process terminates when the fittest test model returned is infact the one which satisfies the given test case. This also suggests that our approach can also better handle complex operators such as ‘xor’ and ‘implies’ apart from the commonly used ‘or’ operator or ‘and’ operators.

The elimination of false positives from existing results is what leads to a significant improvement in our results.

5.2 Limitations

Enlisted below are the limitations, which are inherent to the technique or the coverage criterion considered for the problem of TMG by BB validation of MTs:

- Unlike formal approaches, SBST algorithms are random in nature. They lack the rigor that comes with formal techniques and can offer limited guarantee of the obtained results.
    - However, to mitigate the randomness of the SBST algorithms, the obtained results are averaged over several runs using small to medium-scale case-studies. Representative results can be obtained by following best practices [28].
- MC/DC is a strong coverage criterion which results in stringent test cases. The type of MC/DC used in our work is Unique-Cause MC/DC, which can often lead to infeasible test cases [25], [31]. Although, infeasible test cases are eliminated before coverage or success rate is determined, but it can involve tedious manual analysis that can be time consuming.
    - Another type of MC/DC is known as Masking MC/DC [34], [35], [33]. Unlike Unique-Cause, this type is flexible and is able to overcome the limitations of the former type.
    - However, the fitness function that we proposed is still applicable to Masking MC/DC. In order to limit the scope of this work, we only focus on Unique-Cause and demonstrate results for the same.

CHAPTER 6: FUTURE WORK AND CONCLUSION

This chapter suggests improvements for future that can be potentially explored as research opportunities. It concludes with an overview of our approach, the novelty that we introduced and shares the major findings of our research work especially in comparison to existing approach.

6.1 Future Work

Below are a few suggestions that can be implemented as part of future work:

- For the sake of completeness, augment TMG with other steps of the validation process for MTs i.e. perform
    - Quality assessment of test models
        - Using techniques such as Mutation Analysis [7], [9], [36]
    - Oracle checking
        - Using techniques such as Metamorphic Testing [37]
- Extend this approach to apply stronger criterion for carrying out validation of various MTLs, targeting language-specific constructs other than OCL constraints
- Incorporate another metric [38], [39] along with the coverage objective in the fitness equation i.e.
    - Propose a multi-objective fitness function that utilizes
        - Complexity metrics in the context of MDE
        - Or incorporate performance-based factors for SBST such as minimizing no of iterations to find a test model, limit memory or time consumption
- Apply SBST algorithms to other open problems for validating MTs such as
    - Regression Testing
    - Performance Testing
- Experiment with variations in configuration parameters of SBST algorithms to gain optimal results [40].

6.2 Conclusion

TMG for validation of MTs is a known problem in the field of MDE. Complexity of models and annotated constraints remain the bottleneck in overcoming the challenge of TMG. Formal techniques for TMG requires conversion of given constraints into intermediate formalism, are time consuming and suffer from combinatorial explosion. SBST techniques overcome these challenges and offer promising results to the TMG problem by eliminating the need for any intermediate formalism and being able to find a test model in a given restricted search space within a reasonable amount of time. Existing work on SBST for MTs only covers weaker coverage criteria for constraints such DC (or BC) or it reuses the DC solution for stronger coverage criteria such as MC/DC. No condition-specific solution has been proposed to cater to the needs of condition-based criteria.

While considering the need of MC/DC as a standard mandated in the avionics and automotive industries that extensively utilize MDE in their process chain, we propose a solution that addresses the above problem by devising a novel FF which is condition-specific. AVM – a local search algorithm has been applied for TMG using MC/DC criterion for model constraints. The approach is validated against 2 case studies ranging from small to medium scale. In comparison to existing results that reuse DC solution for MC/DC, our approach outperforms with an improvement of ~36.2% in terms of coverage or 0.3% in terms of success rate. Moreover, our FF is generalizable to other condition-based criteria such as C/DC, MCC and is even applicable to DC due to the subsumption relationship as we move from weaker to stronger criteria.

REFERENCES

[1] S. Sendall and W. Kozaczynski, “Model transformation: The heart and soul of model-driven software development,” *IEEE Software*, vol. 20, no. 5. pp. 42–45, 2003.

[2] M. Brambilla, J. Cabot, and M. Wimmer, *Model-Driven Software Engineering in Practice: Second Edition*. Morgan & Claypool, 2017.

[3] ISO, “Unified Modeling Language Specification Version 1.4.2,” *ISO/IEC 19501:2005(E)*, vol. 4, no. 1, pp. 1–432, 2005.

[4] OMG, “UML Unified Modeling Language Specification Version 2.5.1.” [Online]. Available: https://www.omg.org/spec/UML. [Accessed: 14-Jun-2021].

[5] OMG, “OCL Object Constraint Language Specification Version 2.4.” [Online]. Available: https://www.omg.org/spec/OCL/. [Accessed: 12-Jun-2021].

[6] J. Warmer and A. Kleppe, *The Object Constraint Language: Getting Your Models Ready for MDA, 2nd Edition*. Addison Wesley, 2003.

[7] G. M. K. Selim, J. R. Cordy, and J. Dingel, “Model transformation testing: The state of the art,” in *Proceedings of the 1st Workshop on the Analysis of Model Transformations, AMT 2012*, 2012, pp. 21–26.

[8] J. Tian, *Software Quality Engineering: Testing, Quality Assurance, and Quantifiable Improvement*, 1st ed. Wiley-IEEE Computer Society Press, 2005.

[9] L. Burgueño, J. Troya, M. Wimmer, and A. Vallecillo, “Static fault localization in model transformations,” *IEEE Trans. Softw. Eng.*, vol. 41, no. 5, pp. 490–506, 2015.

[10] F. Fleurey, J. Steel, and B. Baudry, “Validation in Model-Driven Engineering: Testing Model Transformations,” in *First International Workshop on Model, Design and Validation (MoDeVa’04)*, 2004.

[11] A. A. A. Jilani, “Automated Test Data Generation for Model Transformation Testing,” National University of Computer & Emerging Sciences, 2018.

[12] E. Brottier, F. Fleurey, J. Steel, B. Baudry, and Y. Le Traon, “Metamodel-based Test Generation for Model Transformations: an Algorithm and a Tool,” in *17th International Symposium on Software Reliability Engineering (ISSRE’06)*, 2006.

[13] C. A. González and J. Cabot, “Test Data Generation for Model Transformations Combining Partition and Constraint Analysis,” in *International Conference on Theory and Practice of Model Transformations (ICMT). Lecture Notes in Computer Science*, 2014, vol. 8568 LNCS, pp. 25–41.

[14] S. Ali, T. Yue, X. Qiu, and H. Lu, “Generating boundary values from OCL constraints using constraints rewriting and search algorithms,” in *IEEE Congress on Evolutionary Computation (CEC)*, 2016, pp. 379–386.

[15] S. Ali, M. Zohaib Iqbal, A. Arcuri, and L. C. Briand, “Generating Test Data from OCL Constraints with Search Techniques,” *IEEE Trans. Softw. Eng.*, vol. 39, no. 10, pp. 1376–1402, 2013.

[16] E. Guerra and M. Soeken, “Specification-driven model transformation testing,” *Softw. Syst. Model.*, vol. 14, no. 2, pp. 623–644, 2015.

[17] F. Büttner, M. Egea, and J. Cabot, “On verifying ATL transformations using ‘off-the-shelf’ SMT solvers,” in *15th International Conference on Model Driven Engineering Languages & Systems (MODELS). Lecture Notes in Computer Science*, 2012.

[18] K. Berramla, E. A. Deba, and M. Senouci, “Formal validation of model transformation with Coq proof assistant,” in *First International Conference on New Technologies of Information and Communication (NTIC)*, 2015, pp. 1–6.

[19] D. Varró and A. Pataricza, “Automated Formal Verification of Model Tranformations,” in *Workshop on Critical Systems Development in UML (CSDUML)*, 2003, no. Otka 038027, pp. 63–78.

[20] L. Lúcio, B. Barroca, and V. Amaral, “A Technique for Automatic Validation of Model Transformations,” in *International Conference on Model Driven Engineering Languages and Systems (MODELS). Lecture Notes in Computer Science*, 2010, vol. 6394 LNCS, no. PART 1, pp. 136–150.

[21] B. J. Oakes, J. Troya, L. Lúcio, and M. Wimmer, “Full contract verification for ATL using Symbolic Execution,” *Softw. Syst. Model.*, vol. 17, pp. 815–849, 2018.

[22] M. Utting and B. Legeard, *Practical Model-Based Testing: A Tools Approach*, 1st ed. Morgan Kaufmann, 2006.

[23] M. Harman, Y. Jia, and Y. Zhang, “Achievements, Open Problems and Challenges for Search Based Software Testing,” in *8th International Conference on Software Testing, Verification and Validation (ICST)*, 2015, pp. 1–12.

[24] A. A. Jilani, M. Z. Iqbal, and M. U. Khan, “A search based test data generation approach for model transformations,” in *International Conference on Theory and Practice of Model Transformations (ICMT). Lecture Notes in Computer Science*, 2014, vol. 8568 LNCS, pp. 17–24.

[25] S. Ali, M. Z. Iqbal, M. Khalid, and A. Arcuri, “Improving the performance of OCL constraint solving with novel heuristics for logical operations: a search-based approach,” *Empir. Softw. Eng.*, vol. 21, no. 6, pp. 2459–2502, 2016.

[26] H. Sartaj, M. Z. Iqbal, A. A. A. Jilani, and M. U. Khan, “A Search-Based Approach to Generate MC/DC Test Data for OCL Constraints,” in *Search-Based Software Engineering (SSBSE). Lecture Notes in Computer Science*, 2019, vol. 11664 LNCS, no. August, pp. 105–120.

[27] M. Harman and P. McMinn, “A Theoretical and Empirical Study of Search Based Testing: Local, Global and Hybrid Search,” *IEEE Trans. Softw. Eng.*, vol. 36, no. 2, pp. 226–247, 2010.

[28] M. Harman, P. McMinn, J. T. de Souza, and S. Yoo, “Search Based Software Engineering: Techniques, Taxonomy, Tutorial,” in *Empirical Software Engineering and Verification (LASER 2008-2010). Lecture Notes in Computer Science*, 2012, vol. 7007 LNCS, pp. 1–59.

[29] R. Ferguson and B. Korel, “The Chaining Approach for Software Test Data Generation,” *ACM Trans. Softw. Eng. Methodol.*, vol. 5, no. 1, pp. 63–86, 1996.

[30] B. Alkhazi, C. Abid, M. Kessentini, D. Leroy, and M. Wimmer, “Multi-criteria test cases selection for model transformations,” *Autom. Softw. Eng.*, vol. 27, no. 1–2, pp. 91–118, 2020.

[31] M. Fleck, J. Troya, and M. Wimmer, “Search-based Model Transformations,” *J. Softw. Evol. Process*, vol. 28, no. 12, pp. 1081–1117, 2016.

[32] B. Korel, “Automated Software Test Data Generation,” *IEEE Trans. Softw. Eng.*, vol. 16, no. 8, pp. 870–879, 1990.

[33] G. Gay, A. Rajan, M. Staats, M. Whalen, and M. P. E. Heimdahl, “The Effect of Program and Model Structure on the Effectiveness of MC/DC Test Adequacy Coverage,” *ACM Trans. Softw. Eng. Methodol.*, vol. 25, no. 3, 2016.

[34] K. J. Hayhurst, D. S. Veerhusen, J. J. Chilenski, and L. K. Rierson, “A Practical Tutorial on Modified Condition/Decision Coverage,” 2001.

[35] J. J. Chilenski, “An Investigation of Three Forms of the Modified Condition Decision Coverage (MCDC) Criterion,” 2001.

[36] J. Troya, A. Bergmayr, L. Burgueño, and M. Wimmer, “Towards Systematic Mutations for and with ATL Model Transformations,” in *8th International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, 2015, no. Mutation.

[37] J. Troya, S. Segura, and A. Ruiz-Cortés, “Automated inference of likely metamorphic relations for model transformations,” *J. Syst. Softw.*, vol. 136, pp. 1339–1351, 2018.

[38] M. Harman and J. Clark, “Metrics Are Fitness Functions Too,” in *10th International Symposium on Software Metrics (METRICS)*, 2004.

[39] M. F. Van Amstel and M. G. J. Van Den Brand, “Using Metrics for Assessing the Quality of ATL Model Transformations,” in *CEUR Workshop*, 2011, vol. 742, pp. 20–34.

[40] J. Kempka, P. McMinn, and D. Sudholt, “Design and analysis of different alternating variable searches for search-based software testing,” *Theor. Comput. Sci.*, vol. 605, no. 618091, pp. 1–20, 2015.

[41] M. D. Natale “An Introduction to MC/DC Coverage”, Scuola Superiore S. Anna-Pisa, Italy [Accessed: 14-Jun-2020].