COST-EFFECTIVE MODEL-BASED TEST CASE GENERATION AND PRIORITIZATION FOR SOFTWARE PRODUCT LINE

RABATUL ADUNI BINTI SULAIMAN

A thesis submitted in fulfilment of the
requirements for the award of the degree of
Doctor of Philosophy

School of Computing
Faculty of Engineering
Universiti Teknologi Malaysia

NOVEMBER 2020

DEDICATION

To:

Ma Abah, the two main supporters that always concern on this journey

My husband, the secret of my happiness

The kids, Umar Aisy and Aisyah Aira, you are my sun, my moon and my thousand stars.

Thanks for all your patience and understanding

ACKNOWLEDGEMENT

Bismillahhirahmanirahim, first and foremost, I would like to thank Allah Almighty for giving me opportunity, patience and strength to finish my research.

I wish to express my gratitude and deepest appreciation to my supervisors, Associate Professor Dr. Dayang Norhayati Abang Jawawi and Dr Shahliza Abdul Halim for the guidance to complete my thesis. I would like to acknowledge everyone that has assisted me throughout my doctoral study over the years.

I had been blessed with very understanding and helpful parent (ma abah) who has always supported and prayed for me during my doctoral journey. Not forgotten is my loving husband, Nik Mohd Fakrudin Bin Nik Mat, who was always supportive and helpful throughout this journey with our two kids.

I also would like to extend my appreciation to all my research lab members who always give their valuable comments and suggestion. Not forgotten, Universiti Tun Hussein Onn (UTHM) and Ministry of Higher Education (MOHE), who give me funding throughout this journey. Alhamdulillah.

ABSTRACT

In Software Product Line (SPL), testing is used to manage core assets that comprised variability and commonality in effective ways due to large sizes of products that continue to be developed. SPL testing requires a technique that is capable to manage SPL core assets. Model-based Testing (MBT) is a promising technique that offers automation and reusability in test cases generation. However, there are difficulties to ensure testing in MBT can achieve good test cases generation results based on cost (size of test suite, total execution time) and effectiveness (coverage criteria, fault detection rate) measures. This is due to lack of trade-off between cost and effectiveness in test cases generated in MBT for SPL. This study aims to increase quality of test cases based on cost and effectiveness by using generation and prioritization approaches for MBT in SPL. This study focuses on three parts to enhance quality of test cases. First, test model development based on traceability link. In order to improve test cases quality, this study focused on implementation of hybrid-based and hyper-heuristic based techniques to generate test cases. This is followed by Test Cases Prioritization (TCP) technique that is based on dissimilarity-based technique with string distance. These test cases generation and prioritization approaches are evaluated by using two benchmarks - one test object and one real object. The results are compared with other prominent approaches. The mapping approach showed 10.27% and 32.39% f-measure improvement against existing approach on e-shop object, respectively. For test cases generation using hybrid-based approach, the proposed approach outperformed existing approaches with 11.66% coverage, 17.78% average execution time, and 45.98% average size of test suite on vending machine object. The hyper-heuristic based approach NSGA-II-LHH outperformed other proposed low-level heuristic approaches with 12.00% improvement on coverage, 46.66% average execution time and 42.54% average size of test suite. Furthermore, evaluation of TCP approaches showed fault detection improvement of 21.60%, 10.40% and 12.20% and total execution time improvement of 48.00%, 22.70% and 31.80% in comparison with three existing approaches. The results revealed that proposed model transformations, test cases generation and prioritization approaches significantly improve cost and effectiveness measure in MBT for SPL.

ABSTRAK

Dalam Barisan Keluaran Perisian (SPL), ujian digunakan untuk menguruskan aset teras yang dibahagikan kepada kepelbagaian dan persamaan. Ujian Berasaskan Model (MBT) ialah teknik yang memungkinkan automasi dan guna semula kes ujian. Walau bagaimanapun, terdapat dua masalah yang menyebabkan MBT sukar mencapai penjanaan kes ujian yang bagus dari segi kos (saiz suit ujian, jumlah masa perlaksanaan) dan keberkesanan (kriteria liputan dan kadar kesalahan dikesan) iaitu masalah ketidakseimbangan antara kos dan keberkesanan kes ujian yang dijana oleh MBT. Objektif kajian ini ialah untuk meningkatkan kualiti kes ujian dari segi kos dan keberkesanan dengan menggunakan kaedah penjanaan dan keutamaan dalam MBT untuk SPL. Kajian ini memfokuskan kepada tiga bahagian dalam usaha untuk meningkatkan kualiti kes ujian. Pertama, pembangunan model ujian berasaskan pautan kebolehkesanan. Untuk memperbaiki kualiti kes ujian, kajian ini menumpukan kepada pelaksanaan berasaskan hibrid dan hiper-heuristik untuk menjana kes ujian. Setelah itu, Kes Ujian Keutamaan (TCP) yang berasaskan teknik ketidaksamaan dengan jarak rantaian dijalankan. Ujian penjanaan dan keutamaan ini dinilai menggunakan dua penanda aras – sebuah objek ujian dan sebuah objek sebenar. Keputusan kajian dibandingkan dengan kaedah terdahulu. Teknik pemetaan menunjukkan peningkatan 10.27% dan 32.39% ukuran f untuk objek e-kedai. Untuk ujian penjanaan menggunakan kaedah berasaskan hibrid, kaedah yang dicadangkan menunjukkan 11.66% liputan, 17.78% masa purata pelaksanaan dan 45.98% saiz purata suit ujian untuk objek mesin layan diri. Ujian berasaskan hiper-heuristik NSGA-II mengatasi pendekatan lain dengan 12.00% liputan, 46.66% masa purata pelaksanaan dan 42.54% saiz purata suit ujian. Selain itu, penilaian terhadap kaedah TCP menunjukkan peningkatan kadar pengesanan kesalahan 21.60%, 10.40% dan 12.20% manakala jumlah masa pelaksanaan menurun sebanyak 48.00%, 22.70% dan 31.80% berbanding tiga pendekatan sedia ada. Hasil kajian mendapati kaedah transformasi model, kaedah penjanaan kes ujian dan kaedah keutamaan berjaya menurunkan kos serta meningkatkan keberkesanan dalam MBT untuk SPL.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

APFD	-	Average Percentage Faults Detected
BBA	-	Branch and Bound Algorithm
BFS	-	Best First Search
CIT	-	Combinatorial Interaction Testing
Dice-Jaro-Winkler	-	Dice-Jaro-Winkler
EA	-	Enterprise Architect
EMOA	-	Evolutionary Multi-Objectives Algorithms
FDC	-	Fault Detection Capability
FWA-BBA-BFS	-	Floyd Warshall Algorithm-Branch and Bound Algorithm-Best First Search
FM	-	Feature Model
GA	-	Genetic Algorithm
ISR	-	Improvement Selection Rules
ISR-MCF	-	Improvement Selection Rules-Modified Choice Function
LLH	-	Low-Level Heuristic
MBT	-	Model-Based Testing
NSGA-II	-	Non-Dominated Sorting Genetic Algorithm II
NSGA-II-LLH	-	Non-Dominated Sorting Genetic Algorithm II-Low-Level Heuristic
PEMOA	-	Preference-based + Evolutionary Multi-Objective Algorithm
PSO	-	Particle Swarm Optimization
PSO-LLH	-	Particle Swarm Optimization-Low-Level Heuristic
SPEA 2	-	Strength Pareto Evolutionary Algorithm 2
SPEA 2-LLH	-	Strength Pareto Evolutionary Algorithm 2-Low-Level Heuristic
SPL	-	Software Product Line
TCP	-	Test Case Prioritization
UML	-	Unified Modelling Language
XML	-	Extensible Mark-up Language

LIST OF SYMBOLS

$Access_m$	-	Accessibility Matrix
α	-	Best generation of individuals
C	-	Cost
C_P	-	Crossover Probability
β	-	Generation and previous generation comparison
$\rightarrow$	-	Implies
$\in$	-	Intersection
$JW(T_1, T_2)_j$	-	Jaro-Winkler between two test cases
$Jaro(T_1, T_2)_{djw}$	-	Jaro between two test cases
$Queue$	-	List of queue
$\hat{}$	-	Lebesque measure
G	-	Maximum Generation
M_P	-	Mutation Probability
Max_{PT}	-	Maximum Path
Min_{PT}	-	Minimum Path
!	-	Not Equal
x	-	Number of objectives
ts_n	-	Number of states covered
ts_t	-	Number of states in test model
$score_{+ve}, score_{-ve}$	-	Positive and Negative score
$Q_{next} \in TP_i$	-	Queue list of Test Cases
$r \in R_i$	-	Reference vector
$\{INF\|0\}$	-	Set of irrelevant values
f	-	Set of features
w	-	Set of nodes
c	-	Set of Constraints Considerations
P_S	-	Set of Population Size
p	-	Set of Pareto Front
$[s,i],[s,j]$	-	Two states comparison

$\sum score$	-	Total score
tm	-	Test model
$\sum PM_{score}$	-	Total Permutation score
$\sum_{i=1}^{n} Max\ Score$	-	Total Maximum Permutation
$\sum_{i=1}^{n} Min\ Score$	-	Total Minimum Permutation
n_x	-	Total number of test cases in test suite
$\Delta time_h$	-	Total execution time
$TS(x)$	-	Total Coverage
δ	-	Time elapsed
w	-	Weight coefficient

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 Overview

A Software Product Line (SPL) is one of the paradigms for systematic reuse that guides organization to develop products from core assets rather than develop products from scratch. There are two major activities in SPL that focus of core assets development in domain engineering and product development in application engineering. Development of core assets is based on identification of reusable assets. In order to develop reusable core assets, SPL must have ability to exploit commonality and manage variability. Due to the explosion in the number of products, SPL requires an exhaustive testing technique to manage products. SPL testing is aimed to minimize testing effort while at the same time produce effective testing results. One of the most promising techniques is Model-Based Testing (MBT), which offers systematic automation in test cases generation (Reuys *et al.* 2010). There are two main steps, which are to obtain requirements to be presented in the test model and derive test cases. It offers automated, rigorous and systematic testing early in the software life cycle stage (modelling stage).

Among the challenges of MBT in SPL is to have a test model development that consists of variability and commonality. Cooperation between variability model with test model helps to realize this goal, which requires realization of variability, thus test model can represent SPL core assets. Realization of variability and commonality is important to be well defined in test model because the model will be used for test cases generation. This makes SPL different compared with single system engineering, since SPL features variability property, which is important to be reflected in test model. Researchers addressed mapping linkage to trace between Feature Model (FM) with requirement model. The link between two models is also known as a traceability link. Approaches have been proposed that described the implementation of traceability link in SPL scope (Cichos *et al*. 2011), (Oster 2012), (Wang *et al.* 2013), however, they do not provide any ways to generate the trace link. Many studies proposed traceability link based on model to code transformation (Shen *et al*. 2008), (Zheng *et al*. 2017) and metamodel (Czarnecki and Helsen 2006), (Rose *et al.* 2012), (Machado 2014), (Font *et al.* 2017). Nevertheless, there is lack of standard semantics definition for trace link and it is represented in poor expressibility of the link and causes the misconception of variability interpreted in the test model (Vale *et al.* 2017). One of the ways to overcome the presentation of mapping linkage is to have a clear prerequisite traceability process with proper guidelines. These guidelines can be used to represent model transformation in traceability between variability and commonality with requirement model.

The second difficulty is regarding the quality of test cases generated from test model artefacts. The quality of test cases covers two main aspects, which are the cost of testing and effectiveness of test cases. The multi-objectives technique for MBT in SPL testing is used to handle trade-off between cost and effectiveness measures in generation and optimization techniques (Henard *et al.* 2015), (Wang *et al.* 2015), (Abbas *et al*. 2016). A test case is classified as a good test case if trade-off can be balanced between cost and effectiveness. The single-objectives measure, for example, coverage criteria, can be used to represent effectiveness of test cases; however, lack of cost measure caused the testing cost, for example total execution time and size of test suite, to be ignored (Hemmati *et al*. 2010), (Devroey 2014).

In recent years, the multi-objectives technique has been proposed to cover multiple test cases quality measure in SPL. Effectiveness of testing in MBT for SPL is commonly measured by using coverage criteria. The third challenge is related with lack of studies that implemented a multi-objective criterion that proved effective in fault detection rate in test suites in MBT statechart for SPL. Effectiveness in fault detection can be discovered efficiently by using Test Cases Prioritization (TCP) Technique (Kazmi *et al.* 2017b). However, the lack of TCP implemented by using multi-objective caused the fault to be unable to be revealed earlier. Thus, it highlights the need for test cases generation and prioritization in order to balance a trade-off between cost and effectiveness measure. Studies by (Henard *et al.* 2013), (Arrieta *et* *al.* 2016), (Egyed *et al.* 2016) defined optimization based on generation and prioritization problem by using Search-Based and Heuristic-Based Technique. However, the cost-effectiveness of test suite is still in early phase since there are arguments between the selection of efficient techniques that can give the best cost and effectiveness for SPL testing. It showed the importance of optimization problem of maximizing effectiveness and minimizing the cost for MBT in SPL testing. Summary of challenges in MBT for SPL is illustrated in Figure 1.1.

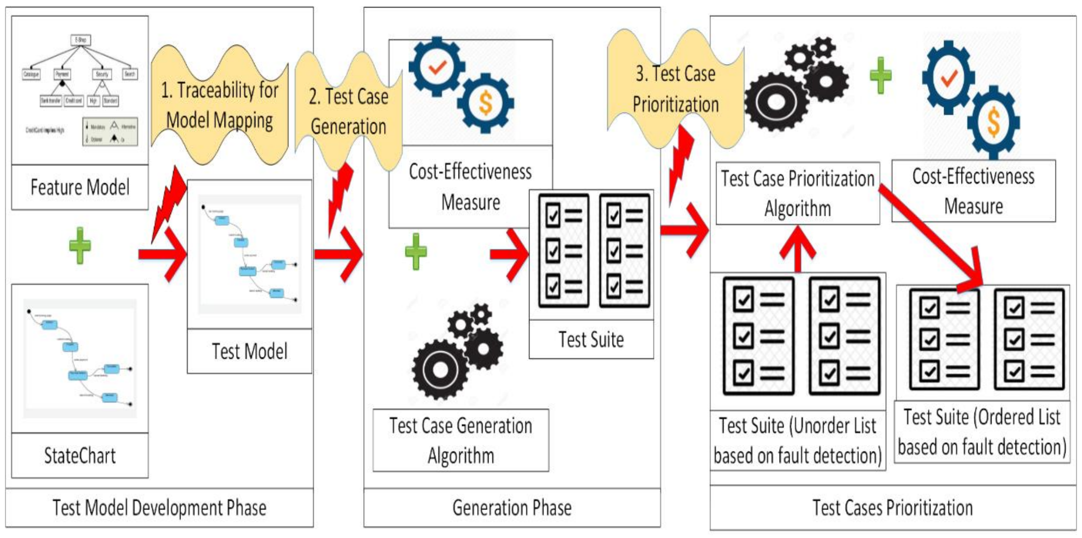
Feature Model
StateChart
1. Traceability for Model Mapping
Test Model
Test Model Development Phase
2. Test Case Generation
Cost-Effectiveness Measure
Test Suite
Test Case Generation Algorithm
Generation Phase
3. Test Case Prioritization
Test Case Prioritization Algorithm
Cost-Effectiveness Measure
Test Suite (Unorder List based on fault detection)
Test Suite (Ordered List based on fault detection)
Test Cases Prioritization

Figure 1.1: Overview of Challenges in MBT for SPL

1.2 Problem Background

In SPL testing context, the explosion in the number of possible products caused exhaustive testing to be infeasible. This issue gives challenge to select relevant subset of product for testing. A basic way to conduct testing for SPL is by using standard technique which are used in single system to be applied for SPL products. However, this takes higher cost and time consumption to evaluate every single product. Thus, there are techniques have been proposed previously to handle issues in SPL testing including MBT. The basic idea of MBT is to systematically minimize effort by exploiting knowledge of core assets. MBT for SPL is used to capture behavior of SPL.

MBT process starts with the test model development which is build based on requirement specifications. Then, test selection criteria will be defined to derive good test cases. A good test case is one that can detect faults earlier with higher effectiveness measure such as structural-based criteria (Utting 2006). In the scope of SPL testing, previous implementation of MBT faced issues of test model development (Ajila and Kaba 2004), (Czarnecki and Helsen 2006), (Heidenreich *et al*. 2008), (Shen *et al*. 2008), (Abbas *et al*. 2016), redundancy in test suites (Perrouin *et al*. 2010), (Wang *et al*. 2015), (Lackner 2017), and quality measure, which are effectiveness of test cases (Devroey 2014), (Machado 2014), (Lackner 2015), (Al-Hajjaji *et al*. 2017) and cost of testing (Hemmati *et al*. 2010), (Ensan *et al*. 2012), (Papadakis *et al*. 2016).

At first, the test model needs to be developed based on core assets reflected in requirement model. This development is related to the process of collection of core assets into the test model. Traceability link is commonly used to handle the different types of models. However, in the scope of SPL testing for derivation of test cases, it required a mapping between models in domain engineering because test cases should be generated in domain engineering before being considered ready to be reused in application engineering. In traceability link for MBT in SPL, approaches have been proposed related to the metamodel (Cavalcanti *et al*. 2011), (Font *et al.* 2017), (Steghöfer *et al.* 2019), model transformation (Czarnecki and Helsen 2006), (Heidenreich *et al*. 2008), (Rose *et al.* 2012), (Jessica *et al.* 2013), (Olsen and Oldevik 2013). Existing studies proposed model transformation approach to trace features in FM with statechart (Czarnecki and Helsen 2006), (Heidenreich *et al,* 2008). These model transformations proposed a process of conversion between FM into test model. However, it faced a problem with scalability due to manual mapping process that required tester to select the features to map with states. Furthermore, these approaches did not fulfill the needs of SPL because the dependency constraints of FM existed were ignored. This caused problem to maintain and evolve the large size of SPL core assets due to tester demand for good quality of test cases based on cost and effectiveness measures. Furthermore, there is a lack of comprehensive steps in development of traceability link. It is very important to have a clear view of traceability link in order to make sure the commonality and variability is undoubtedly defined in test model.

In MBT, redundancy of generated test cases also leads to a large size of test suite and increase in total execution time. This is related with the cost (size of test suite, total execution time) and effectiveness (coverage and fault detection) of test cases that can cause scalability issue. Studies by (Reuys *et al.* 2010), (Cichos *et al.* 2011) and (Lackner 2017) applied coverage-based measures, for example all-states and all-transition to generate test cases. The Ensan *et al.* (2012) discussed the importance of trade-off between different coverage types. However, according to (Ur *et al.* 2018), balance trade-off is not only represented in single measurement types, but needs to be measured between cost and effectiveness of test cases.

For example, once tester has achieved the maximum target to find defects, studies have been unable to ensure that these activities significantly reduce the testing effort and cost. Furthermore, as highlighted by (Inozemtseva and Holmes 2014), a good coverage test suite does not guarantee that the test suite is effective enough. This fact is acceptable in SPL context. The Devroey (2014) proposed a single effectiveness measure aimed to produce a good coverage-based generation. However, it faced scalability issue due to the longer execution time for test case generation. Most existing approaches in MBT for SPL focusing on test case measures were based on effectiveness (coverage). However, due to cost (size of test suite and total execution time), few cases have been evaluated (Cichos *et al.* 2011), (Arrieta *et al*. 2014), (Devroey *et al.* 2014), (Papadakis *et al*. 2016). This brings room for improvement in terms of trade-off between cost and effectiveness measure to improve the test case results. Furthermore, the implementation of Search-Based Technique (SBT) in MBT for SPL was designed to handle trade-off as optimization problem. SBT was based on heuristic technique that consists of cost and effectiveness measures to be transformed into objective functions. The goal of implementation of SBT in MBT for SPL is to generate test cases that balanced trade-off between cost and effectiveness measure.

In order to improve effectiveness (faults detection) in test cases, another technique, which is Test Case Prioritization (TCP), is applied to enhance the fault detection rate. For MBT in SPL, TCP is also used as a technique to improve effectiveness based on earlier fault detection. The lack of TCP implementation made it difficult for faults to be revealed as soon as possible, thus making it possible for wrong products to be executed.

Machado (2014) improved the effectiveness of test cases by prioritizing fault in test model by using mutation testing. Existing studies proposed a similarity-based technique by using string distance method to prioritize test cases from FM and delta-oriented architecture model (Henard *et al.* 2016), (Al-Hajjaji *et al.* 2017), (Sahak 2018). Devroey *et al.* (2017) also proposed generation and prioritization for SPL. However, in terms of TCP, they have reused TCP algorithm proposed by (Henard *et al.* 2016) in test cases generated from mathematical model, for example Markov Chain model.

Moreover, in MBT for SPL, there is a lack of study that implemented TCP based on another test model, especially behavior model statechart. In terms of technique, similarity-based by using string distance to evaluate similarity of test cases is still in early phase. This is due to the lack of extensive view of efficiency of string distance implementation. Only a few string distances, which is Jaccard Distance and Jaro-Winkler hybrid with Hamming Distance, have been discovered. There is no justification on the evaluation of string distance applied in SPL. The string distance can help to reveal faults by measuring the similarity of test cases. The lack of implementation of string distance for similarity measure caused fault techniques to inefficiently detect faults.

In order to handle TCP, a technique was required to reorganize test cases sequence. There are three algorithms that have been previously discussed in similarity-based, which are Local Maximum Distance, Global Maximum Distance and all-yes config. However, there is a lack of evidence to show that the proposed algorithm is the best algorithm to discover faults as early as possible. Furthermore, the cost of TCP (total execution time) for similarity-based technique also required evaluation to make sure the proposed technique is capable to balance trade-off between cost and effectiveness measure. The limited multi-objectives in TCP approach caused difficulties to revealed faults. However, in terms of execution times, it is still important to be evaluated to ensure time can be minimized.

Due to these reasons, three key issues have been identified in this study to be resolved in order to produce a good quality of test cases with trade-off between cost and effectiveness measures. The first involves process of test model development with variability consideration based on FM. The lack of inadequacy of model mapping caused variability and commonality in SPL to be not completely reflected in test model artefacts. Another concern in this study is the demand of SPL testing that aims to obtain test cases with trade-off between cost and effectiveness measure. However, there is a lack of techniques concerned with multi-objective test case measure in SPL test case generation and prioritization. In terms of test case generation, a good technique that can balance trade-off is seen to be important way to enhance test case quality. This is because the existing studies validate test cases based on single measure without implementation or other validation that is also important to be discovered. There is a lack of previous studies that discover fault detection rate from test cases generated in test model artefacts. However, TCP is one of the important techniques to discover faults as early as possible compared with other existing techniques. Thus, test case generation and prioritization were the main factors to achieve trade-off between cost and effectiveness in MBT for SPL.

1.3 Problem Statement

Based on analysis from existing studies in MBT for SPL, there is a demand of test cases to fulfill a good quality measure in terms of cost and effectiveness. In order to enhance cost and effectiveness of test cases in MBT for SPL, there are three important parts that need to be highlighted, which are statechart test model, generation and prioritization of test cases.

First challenge is related to statechart test model development since it is the basic preparation to conduct SPL testing by using MBT. It requires a test model to represent SPL core assets based on requirements of products. The mapping between models is a challenge with respect to accurate generation of test cases. The implementation of single model, for example FM, can represent variabilities and commonalities in simplest ways. FM is based on taxonomic form commonly represented as symbols. Mapping with other models such as requirement model can make FM information to be representable (Czarnecki and Antkiewicz 2005). The traceability link is required to map between two model components. In MBT, it is used to reflect SPL core assets in the test model for test case generation.

Model transformation approach is used to create traceability link between two models. However, the implementation of model transformation approach to conduct traceability mapping caused lack of comprehensive views of creating traceability in clear ways (Anquetil *et al.* 2012), (Vale *et al.* 2017). This is important since it involves variability of products. FM constraints need to be validated in order to prevent an invalid generation (Lochau *et al.* 2012).

The second challenge is related to test cases generation from statechart test model. It is associated with implementation of single measure criteria for test cases generation, for example, coverage criteria. It is used to discover coverage that can be covered by proposed approach based on test model artefacts. Nevertheless, it ignores the remaining validation measure, for example, the cost of testing is also important to be validated in test case generation. The single criteria is not able to represent it as a good test case (Inozemtseva and Holmes 2014), (Kazmi *et al.* 2017b). It required a trade-off between cost (size of test suite, total execution time) and effectiveness (coverage, fault detection capability) to ensure that the generated test cases gain a good quality measurement (Ahmed *et al.* 2020). Most existing approaches in MBT for SPL are concerned with a single objective at a time. This remains valid for certain cases; however, it does not reflect real-life problems in testing. However, trade-off issues can be improved by implementing multi-objective optimization for SPL testing (Henard *et al.* 2013), (Wang *et al.* 2014), (Abbas *et al.* 2016).

In order to enhance effectiveness of testing, earlier fault detection also need to be measured. This is because the test cases generated from MBT are not considered test cases based on fault detection. This has caused test cases in test suite to be listed randomly without any consideration of fault detection rate. TCP is the one of the techniques that can reveal faults earlier by reordering test cases based on fault detection rate. However, there is a lack of studies that measured faults in MBT for

SPL. This has caused faults to be not considered as validation measure to evaluate effectiveness of test cases. Furthermore, in SPL, the proposed TCP approach highlights the concern of test cases from FM and mathematical model (Markov Chain) (Egyed *et al.* 2016), (Henard *et al.* 2016), (Schaefer *et al.* 2016), (Al-Hajjaji *et al.* 2017). There is a lack of TCP approach used to evaluate test cases from UML statechart test model artefacts. This is due to the previous studies that highlighted concern on effectiveness measure of UML statechart test model artefacts by using coverage criteria, for example structural-based coverage. This has led to faults to not be considered an effectiveness measure for test cases generated from MBT in SPL.

The summary of problem to be highlighted in the study is summarized as per Figure 1.2.

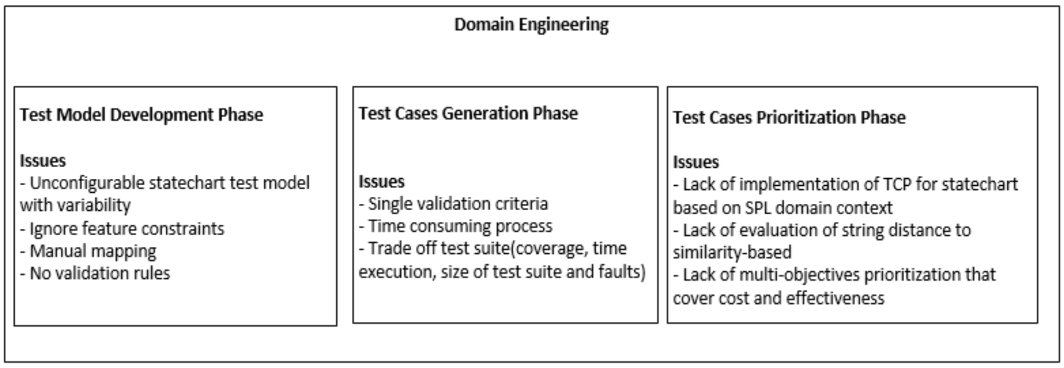
Domain Engineering
Test Model Development Phase
Issues
- Unconfigurable statechart test model with variability
- Ignore feature constraints
- Manual mapping
- No validation rules
Test Cases Generation Phase
Issues
- Single validation criteria
- Time consuming process
- Trade off test suite(coverage, time execution, size of test suite and faults)
Test Cases Prioritization Phase
Issues
- Lack of implementation of TCP for statechart based on SPL domain context
- Lack of evaluation of string distance to similarity-based
- Lack of multi-objectives prioritization that cover cost and effectiveness

Figure 1.2: The recognized problems

To evaluate effectiveness measure based on earlier fault detection rate, similarity and dissimilarity based prioritization approach is shown as a good evaluation method to evaluate faults. It starts with measuring distance between test cases in the test suite. Then, the proposed similarity or dissimilarity-based approach will reorder test cases based on fault detection rate. In order to ensure test cases can be evaluated based on cost and effectiveness measure, it also requires a balance trade-off based on cost (total execution time) and effectiveness (fault detection rate) to make sure the TCP fulfills the software testing demand. Multi-objective optimization is required to ensure the trade-off issues in generation and prioritization approach can be improved. This is because multi-objective optimization offers different evaluation measure at a single time. In addition, the technique can be tuned to ensure cost and effectiveness of test cases can be maximized and minimized.

1.4 Research Question

The aim of this research is to solve issues within the concept of MBT in SPL while targeting the accuracy of managing variability and enhancing the multi-objectives criteria in test cases generation and prioritization.

Derived from the research problem, the following are the formulated research questions:

"*How to minimize cost (size of test suite, total execution time) and maximize effectiveness (coverage and faults) measures for generation and prioritization approach in MBT based on statechart test model for SPL*"

To answer the main research questions, the following sub-question is addressed:

RQ1: How can the traceability mapping approach for MBT in SPL be improved?

RQ 1.1 How can the traceability mapping approach be reflected in explicit mapping between SPL statechart test model and test artefacts?

RQ 1.2: What are the techniques involved in explicit mapping between SPL statechart test model and test artefacts?

RQ2: How can the effectiveness of existing test case generation algorithms be increased?

RQ 2.1: Can the hybrid-heuristic optimization problem handle trade-off between cost and effectiveness in test case generation for MBT in SPL?

RQ 2.2: Does the hyper-heuristic optimization problem handle trade-off between cost and effectiveness in test case generation for MBT in SPL?

RQ3: How can earlier fault detection in test cases by using prioritization technique be discovered?

RQ3.1: How can the existing similarity-based prioritization technique for accelerating fault with minimal execution time be enhanced?

RQ 3.2: How can the existing similarity measurement by using string-based method for similarity-based prioritization be improved?

1.5 Research Goals and Objectives of the Study

The main goal of this research is to obtain increase in the quality of test cases based on cost (size of test suite, total execution time) and effectiveness (coverage, fault detection capability) by using generation and prioritization approach for MBT in SPL. This approach is able to ensure that the derivable test cases can minimize the cost and maximize effectiveness due to demand in SPL testing.

To achieve the mentioned goals, the objective of this study are as follows:

Objective 1: To propose a model transformation approach that can be used to provide explicit traceability mapping in MBT SPL.

Objective 2: To propose hybrid test cases generation technique with trade-off between cost and effectiveness measurement.

Objective 3: To propose a test case prioritization algorithm based on similarity-based with string distance measurement.

1.6 Research Scope

The proposed research is mainly focused on MBT for SPL with three main scopes, which are traceability in model and test case generation and prioritization. Details of the research scope are as follows:

i) Model-based testing for SPL

Model-based in this study covers the scope of testing that starts with test model development. It only covers representations of variability and requirements based on behavior in the scope of test model statechart. The implementation of other test model types are excluded from research.

ii) Traceability on MBT for SPL

The traceability link is based on the model transformation for SPL. Another type of traceability link, which is metamodel transformation, is excluded from the research since the focus is to map between two test models, which required obtaining all the related model components that can be used to derive test cases.

1.7 Significance of the study

SPL core asset management is an important element that needs to be considered before testing is conducted. It is found that constraints validation (Lochau *et al.* 2014) and tracing process (Rose *et al.* 2012) are important elements in order to accurately map test model component. Previous studies do not reflect comprehensive traceability process for SPL in clear way since it implemented a single traceability approach. Thus, the objective is to propose a model transformation approach that can be used to provide explicit traceability mapping in MBT SPL.

The second contribution of this study is to facilitate a good test case evaluation based on cost and effectiveness measures. This is due to the demand in SPL testing to optimize the cost (size of test suite, execution time) (Wang *et al.* 2015), maximize based on coverage (Devroey *et al.* 2014) and fault detection ability (Al-Hajjaji *et al.* 2017). Due to the lack of multi-objectives optimization in MBT for SPL, this study proposed a test case generation approach with aim to enhance quality of test cases in terms of cost and effectiveness. There are two different techniques compared in the proposed approach, which are hybrid-heuristic and hyper-heuristic based for multi-objectives optimization. In terms of test cases generation, the proposed approach will cover the generation of test cases with maximum coverage and optimal cost (size of test suite and execution time). Another concern is due to the lack of trade-off between cost (time execution) and effectiveness (fault detection rate) in test cases from MBT for SPL. Thus, the enhancement of prioritization approach based on prioritization technique and string distance is used to overcome the problem of fault measurements in test cases from MBT. In this light, following points are significant for current research.

I. This research will help to give a complete traceability process based on model transformation and formal method that is used to create traceability link between variability model and requirement model.

II. This research helps to improve the process of mapping between models by using traceability based on automated query.

III. This research provided the test case generation by using hybrid search-based testing with consideration of cost (size of test suite, total execution time) and effectiveness (coverage) measurements.

IV. This research attempts to find the suitable test cases generation technique that can balance trade-off between cost and effectiveness measure.

V. This research attempts to discover faults in test cases by using enhanced prioritization technique and string distance based on dissimilarity measure.

The cost (total execution time) and effectiveness (fault detection) are considered to evaluate the prioritization algorithm proposed.

1.8 Research Organization

The thesis is organized in nine chapters as follows:

Chapter 1: Introduction: It gives an overview on the structure of the research. It described the background of the study that brings the research problem to be discussed and explained. Main sections, which are research question and research objectives, are presented. The importance of this research is also elaborated in this chapter.

Chapter 2: Literature Review: It gives a discussion of the prior studies conducted in the field of MBT and SPL. All related studies are classified, explained and analyzed based on the problem defined in Chapter 1. The advantages and disadvantages of the existing study are also presented.

Chapter 3: Research Methodology: It provides the descriptions of the research framework, research process, case studies, techniques utilized in the current research. Details of each section are described in this chapter.

Chapter 4: The proposed FM_STATE Model Mapping. This chapter described the proposed model mapping approach based on the traceability link of FM and statechart. The design, implementation and results of the study are discussed in this chapter.

Chapter 5: A Hybrid Heuristic Algorithm for MBT Test Cases Generation in SPL. This chapter provides the proposed technique based on the three hybrid algorithms, which are Floyds Warshall Algorithm (FWA), Branch and Bound Algorithm (BBA) and Best First Search (BFS). The design, implementation, results, and existing study comparison are discussed in this chapter.

Chapter 6: Multi-objectives Test Case Generation by using Hyper-Heuristic Algorithm for MBT in SPL will describe the hyper-heuristic approach for MBT in SPL used to generate test cases. This chapter describes the design, implementation and experimental results of the proposed technique.

Chapter 7: Test Case Prioritization by using Hybrid Search Based Algorithm with Similarity Distance describe the TCP technique implemented for MBT statechart. This chapter includes design, implementation and experimental results. This experiment is related to the two proposed generation algorithms.

Chapter 8: The test cases generation from Chapter 5 and 6 with TCP from Chapter 7 are used to evaluate the performance of the proposed approach. The strength and weaknesses of the proposed techniques are also described.

Chapter 9: Conclusion and Future Work. The conclusion of the research is presented while highlighting findings, issues and future works.

REFERENCES

Abbas, A., Siddiqui, I.F. Lee, S. U. J. (2016) "Goal-based Modeling for Requirement Traceability of Software Product Line", *Journal of Theoretical and Applied Information Technology*, 94(2).

Abbas, A., Siddiqui, I. F. and Lee, S. U. (2016) “Multi-Objective Optimization of Feature Model in Software Product Line : Perspectives and Challenges”, *Indian Journal of Science and Technology*, 9(December). doi: 10.17485/ijst/2016/v9i45/106769.

Ahmed, B., Enoiu, E., Wasif. A., Kamal, Z. (2020) “An evaluation of Monte Carlo-based hyper-heuristic for interaction testing of industrial embedded software applications”, *Soft Computing*. Springer Berlin Heidelberg. doi: 10.1007/s00500-020-04769-z.

Ajila, S. A. and Kaba, A. B. (2004) “Using Traceability Mechanisms to Support Software Product Line Evolution”, in *International Conference on Information Reuse and Integration*, pp. 157–162.

Ajmal, F. (2018) *Requirement Engineering and Software Testing : Forward and Backward Traceability in requirement model and system model on the Basis of Failed Test Cases and improved requirement*.

Al-Hajjaji, M. (2014) “Scalable and Efficient Sampling for Product-Line Testing”, *Technical Report FIN-003-2014, University of Magdeburg, Germany, 2014*.

Al-Hajjaji, M., Lily, S., Lachman, R. (2017) “Delta-Oriented Product Prioritization for Similarity-Based Product-Line Testing”, *Proceedings - 2017 IEEE/ACM 2nd International Workshop on Variability and Complexity in Software Design, VACE 2017*, pp. 34–40. doi: 10.1109/VACE.2017.8.

Al-shamri, M. Y. H. (2014) “Power coefficient as a similarity measure for memory-based collaborative recommender systems”, *EXPERT SYSTEMS WITH APPLICATIONS*. Elsevier Ltd, 41(13), pp. 5680–5688. doi: 10.1016/j.eswa.2014.03.025.

Anjorin, A. Oster, S., Zorcic, I. (2012) “Optimizing Model-Based Software Product Line Testing with Graph Transformations”, *Electronic Communications of the EASST 11th International Workshop on Graph Transformation and Visual*

Modeling Techniques (GTVMT 2012), 47.

Anquetil, N. Kulesza, U., Mitschke, R. (2010) “A model-driven traceability framework for software product lines”, *Software & Systems Modeling*, pp. 427–451. doi: 10.1007/s10270-009-0120-9.

Anquetil, N. Mitschke, R., Moreire, A. (2012) “A Model-Driven Traceability Framework for Software Product Lines”, *Software & Systems Modeling*, pp. 427–451.

Anquetil, N. Grammel, B., Galvao, I. (2015) “Traceability for Model Driven , Software Product Line Engineering”, in *ECMDA Traceability Workshop Proceedings*, pp. 77–86.

Arif, T. (2015) “Exploring The Use Of Hybrid Similarity Measure”, *INTERNATIONAL JOURNAL OF SCIENTIFIC & TECHNOLOGY*, 4(12), pp. 171–175.

Arrieta, A. Wang, S., Sagardui, G. (2016) “Test Case Prioritization of Configurable Cyber-Physical Systems with Weight- Test Case Prioritization of Configurable Cyber-Physical Systems with Weight-Based Search Algorithms”, in *Proceedings of the Genetic and Evolutionary Computation Conference 2016*. doi: 10.1145/2908812.2908871.

Arrieta, A., Sagardui, G. and Etxeberria, L. (2014) “A model-based testing methodology for the systematic validation of highly configurable cyber-physical systems”, in *6th International Conference on Advances in System Testing and Validation Lifecycle, VALID 2014*, pp. 66–72.

Baller, H. Lily, S., Lochau, M. (2014) “Multi-objective test suite optimization for incremental product family testing”, *Proceedings - IEEE 7th International Conference on Software Testing, Verification and Validation, ICST 2014*, pp. 303–312. doi: 10.1109/ICST.2014.43.

Baudry, B. LeGuen, H., Samih, H. (2014) “An Approach to Derive Usage Models Variants for Model-Based Testing”, *FIP International Conference on Testing Software and Systems. Springer, Berlin, Heidelberg*, pp. 80–96. doi: 10.1007/978-3-662-44857-1_6.

Belli, F. Budnik, C., Hollman, A. (2016) “Model-based mutation testing - Approach and case studies”, *Science of Computer Programming*. Elsevier B.V., 120, pp. 25–48. doi: 10.1016/j.scico.2016.01.003.

Belli, F. and Hollmann, A. (2014) “Test Generation and Minimization with “ Basic ”

Statecharts”, (January 2008). doi: 10.1145/1363686.1363856.

Bertolino, A. (2007) “Software Testing Research: Achievements, Challenges, Dreams”, *Future of Software Engineering*, (September), pp. 85–103. doi: 10.1109/FOSE.2007.25.

Boghdady, P. Badr, N., Hashem, M., (2011) “Test Case Generation and Test Data Extraction Techniques”, *International Journal of Electrical & Computer Sciences*, 11(3), pp. 82–89.

Cartaxo, E. G., Neto, F. G. O. and Machado, P. D. L. (2007) “Test case generation by means of UML sequence diagrams and labeled transition systems”, *Conference Proceedings - IEEE International Conference on Systems, Man and Cybernetics*, pp. 1292–1297. doi: 10.1109/ICSMC.2007.4414060.

Cavalcanti, Y.C., do Carmo Machado, i., da Mota, P.A., Neto, S., Lobato, L.L., de A. (2011) “Towards Metamodel Support for Variability and Traceability in Software Product Lines”, in *Proceedings of the 5th Workshop on Variability Modeling of Software-Intensive Systems*, pp. 49–57. doi: 10.1145/1944892.1944898.

Cichos, H., Oster, S., Lochau, M. (2011) “Model-Based Coverage-Driven Test Suite Generation for Software Product Lines”, *Model Driven Engineering Languages and Systems*, 6981, pp. 425–439.

Cichos, H., Oster, S. and Lochau, M. (2011) “Extended Version of Test Suite Generation for Software Product Lines”, *International Conference on Model Driven Engineering Languages and Systems*, pp. 425–439.

Coutinho, B., Gadelha, E. and Emı, A. (2016) “Analysis of distance functions for similarity-based test suite reduction in the context of model-based testing”, *Software Quality Journal*, 24, pp. 407–445. doi: 10.1007/s11219-014-9265-z.

Czarnecki, K. and Antkiewicz, M. (2005) “Mapping Features to Models : A Template Approach Based on Superimposed Variants Background : Feature Modeling”, *Proceedings of the 4th International Conference Generative Programming and Component Engineering, GPCE 2005*, pp. 422–437. Available at: http://link.springer.com/chapter/10.1007/11561347_28.

Czarnecki, K. and Helsen, S. (2006) “Feature-based survey of model transformation approaches”, *IBM Systems Journal*, 45(3), pp. 621–645. doi: 10.1147/sj.453.0621.

Damiani, F., Lienhardt, M. and Paolini, L. (2019) “A formal model for Multi Software Product Lines”, *Science of Computer Programming*. Elsevier B.V., 172(644298), pp. 203–231. doi: 10.1016/j.scico.2018.11.005.

Deb, K. Agrawal, S., Pratap, A. (2000) “A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization : NSGA-II”, *International conference on parallel problem solving from nature*, pp. 849–858.

Devroey (2017) *Behavioural model-based testing of software product lines*. University of Namur.

Devroey, X., Perrouin, G., Cordy, M. (2014) “A Variability Perspective of Mutation Analysis”, *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pp. 841–844.

Devroey, X. (2014) “Behavioural Model Based Testing of Software Product Lines”, *Software Product Line Conference,* (August), pp. 1–8. Available at: https://www.researchgate.net/profile/Xavier_Devroey/publication/265211912_Behavioural_Model_Based_Testing_of_Software_Product_Lines_Research_Abstract/links/540581380cf23d9765a6f0dc.pdf.

Devroey, X., Perrouin, G., Legay, A., (2014) “Coverage Criteria for Behavioural Testing of Software Product Lines”, *Proceedings of the 6th International Symposium On Leveraging Applications of Formal Methods, Verification and Validation (to appear)*, pp. 336–350. doi: 10.1007/978-3-662-45234-9_24.

Devroey, X., Perrouin, G., Legay, A. (2017) “Dissimilar Test Case Selection for Behavioural Software Product Line Testing”, in. doi: 10.1145/1235.

Devroey, X., Perrouin, G., Cordy, M. (2017) “Statistical prioritization for software product line testing : an experience report”, *Software & Systems Modeling*. Springer Berlin Heidelberg, 16(1), pp. 153–171. doi: 10.1007/s10270-015-0479-8.

Devroey, X., Perrouin, G. and Schobbens, P.-Y. (2014) “Abstract test case generation for behavioural testing of software product lines”, *18th International Software Product Line Conference, SPLC 2014*, pp. 86–93. doi: 10.1145/2647908.2655971.

Dice, L. R. (1943) “The biotic provinces of North America”, in *The biotic provinces of North America,* p.440.

Dragan Gaševic, Ralf Lämmel, E. van W. (2010) “Engineering a DSL for Software Traceability”, in *Software Language Engineering*, p. 153.

Egyed, A. Segura, S., Lopez-Herrejon, R. (2016) “Multi-objective test case prioritization in highly configurable systems: A case study”, *Journal of Systems and Software*, 122, pp. 287–310. doi: 10.1016/j.jss.2016.09.045.

Emília, A. and Barbosa, V. (2015) *Similarity-based test suite reduction in the context of Model-Based Testing*. Universidade Federal de Campina Grande.

Engström, E., Runeson, P. (2015) “Software Product Line Testing - A Systematic Mapping Study”, *Information and Software Technology*, (October), pp. 2–13. doi: 10.1007/978-3-642-29578-2.

Ensan, F., Bagheri, E. and Gasevic, D. (2012) “Evolutionary Search-based Test Generation for Software Product Line Feature Models”, in *International Conference on Advanced Information Systems Engineering*, pp. 613–628.

Etien, A. (2008) "Fine Grained Traceability for an MDE Approach of Embedded System Conception", in *Oldevik, J. Aagedal, JØ (eds.) ECMDA Traceability Workshop (ECMDATW),* pp. 27–38.

Fang, C. and Chen, Z. (2014) “Similarity-Based Test Case Prioritization Using Ordered Sequences of Program Entities”, *Software Quality Journal*, 22.2, pp. 335–361.

Farrag, M. (2013) *Colored Model Based Testing for Software Product Lines (CMBT-SWPL).* Technical University of Ilmenau.

Ferreira, T. Kuk, J., Pozo, A. (2016) “Product selection based on upper confidence bound MOEA/D-DRA for testing software product lines”, *2016 IEEE Congress on Evolutionary Computation, CEC 2016*. IEEE, pp. 4135–4142. doi: 10.1109/CEC.2016.7744315.

Floyd, R. W. (1962) “Algorithms 97”, *Communications of the ACM*, pp. 344–348.

Font, J. Arcega, L. (2017) “Leveraging variability modeling to address metamodel revisions in Model-based Software Product Lines”, *Computer Languages, Systems & Structures*. Elsevier, 48, pp. 20–38. doi: 10.1016/j.cl.2016.08.003.

Gao, L., Mishra, S. K. and Shi, J. (2012) “An extension of branch-and-bound algorithm for solving sum-of-nonlinear-ratios problem”, pp. 221–230. doi: 10.1007/s11590-010-0232-8.

García, B. and Navas, A. (2010) “An automated Model-based Testing Approach in Software Product Lines Using a Variability Language”, in *Fraunhofer Institute for Open Communication Systems*.

Gargantini, A. and Vavassori, P. (2012) “CITLAB: A laboratory for combinatorial interaction testing”, *Proceedings - IEEE 5th International Conference on Software Testing, Verification and Validation, ICST 2012*, pp. 559–568. doi: 10.1109/ICST.2012.141.

Gebizli, C. S. and Sozer, H. (2016) “Model-Based Software Product Line Testing by Coupling Feature Models with Hierarchical Markov Chain Usage Models”, *Proceedings - 2016 IEEE International Conference on Software Quality, Reliability and Security-Companion, QRS-C 2016*, pp. 278–283. doi: 10.1109/QRS-C.2016.42.

Groher, I. and Voelter, M. (2011) "Aspect-Oriented Model-Driven Software Product Line Engineering", in *Transactions on aspect-oriented software development VI*, pp. 111–152.

Heidenreich, F., Kopcsek, J. and Wende, C. (2008) "FeatureMapper: mapping features to models", *International Conference on Software Engineering*, 30, pp. 943–944. doi: 10.1145/1370175.1370199.

Heider, W. *et al.* (2010) "Simulating evolution in model-based product line engineering", *Information and Software Technology*. Elsevier B.V., 52(7), pp. 758–769. doi: 10.1016/j.infsof.2010.03.007.

Hemmati, H., Arcuri, A. and Briand, L. (2010) "Reducing the Cost of Model-Based Testing through Test Case Diversity", *IFIP International Conference on Testing Software and Systems. Springer, Berlin, Heidelberg*, (2), pp. 63–78.

Henard, C. Papadakis, M., Perrouin, G. (2013) "Multi-objective Test Generation for Software Product Lines", in *Proceedings of the 17th International Software Product Line Conference*, pp. 62–71.

Henard, C. Papadakis, M., and Harman, M. (2015) "Combining Multi-Objective Search and Constraint Solving for Configuring Large Software Product Lines", in *Proceedings of the 37th International Conference on Software Engineering*, pp. 517–528. doi: 10.1109/ICSE.2015.69.

Henard, C. (2015) *Enabling Testing of Large Scale Highly Configurable Systems with Search-based Software Engineering : The Case of Model-based Software Product Lines Dissertation Defense Committee*. Universite Du Luxemborg.

Henard, C. Papadakis, M., Perrouin, G. (2016) "Bypassing the Combinatorial Explosion : Using Similarity to Generate and Prioritize T-wise Test Configurations for Software Product Lines", *Proceedings of the National Academy of Science*, no. 37, pp. 10442–10447.

Her, J. S. Oh, S. (2017) "A framework for evaluating reusability of core asset in product line engineering", *Information and Software Technology*, 49, pp. 740–760. doi: 10.1016/j.infsof.2006.08.008.

Hervieu, A. and Baudry, B. (2011) "P ACOGEN : Automatic Generation of Pairwise Test Configurations from Feature Models", *International Symposium on Software*

Reliability Engineering, pp. 120–129.

I.Machado (2014) *Fault Model-Based Variability Testing (Ph.D. Thesis)*. Universidade Salvador.

Inozemtseva, L. and Holmes, R. (2014) "Coverage Is Not Strongly Correlated with Test Suite Effectiveness", in *Proceedings of the 36th International Conference on Software Engineering*, pp. 435–445.

Jaccard, P. (1929) "Considerations sur le coefficient générique et sa signification floristique et phytosociologique", *Bulletin de la Société Botanique de France*, 76.1, pp. 47–66.

Jacelyn Simmonds, M. C. B. (2015) "Modeling Variability in Software Product Family", *Journal of Software*, 16(1), p. 37. doi: 10.1360/jos160037.

Jakubovski Filho, H. L., Ferreira, T. N. and Vergilio, S. R. (2019) "Preference based multi-objective algorithms applied to the variability testing of software product lines", *Journal of Systems and Software*. Elsevier Inc., 151, pp. 194–209. doi: 10.1016/j.jss.2019.02.028.

Jawawi, Dayang N. A, Zaki, Rosbi Mamat, Fakhitah Ridzuan, Muhammad Khatibsyarbini, M. Z. M. (2015) "Introducing computer programming to secondary school students using mobile robots", in *10th Asian Control Conference (ASCC)*, pp. 1–6.

Jessica, D. Jennifer, P., Fern, C., (2013) "Model-to-Code transformation from Product-Line Architecture Models to AspectJ", in *Euromicro Conference Series on Software Engineering and Advanced Applications*, pp. 98–105. doi: 10.1109/SEAA.2013.11.

Jirapanthong, Waraporn, Zisman, A. (2009) "XTraQue: traceability for product line systems Journal", *Software and Systems Modeling*, pp. 117–144. doi: 10.1007/s10270-007-0066-8.

Jirapanthong, W. and Zisman, A. (2009) "XTraQue : traceability for product line systems", *Software & Systems Modeling*, 1, pp. 117–144. doi: 10.1007/s10270-007-0066-8.

Kanstrén, T. Puolitaival, O. (2012) "Experiences in setting up domain-specific model-based testing", *2012 IEEE International Conference on Industrial Technology, ICIT 2012, Proceedings*, pp. 319–324. doi: 10.1109/ICIT.2012.6209957.

Kazmi, R. Jawawi, D., Mohamad, R. (2017a) "Effective Regression Test Case Selection", *ACM Computing Surveys*, 50(2), pp. 1–32. doi: 10.1145/3057269.

Kazmi, R., Jawawi, D., Mohamad, R. (2017b) *EFFECTIVE REGRESSION TEST*

CASE SELECTION TECHNIQUE USING WEIGHTED AVERAGE SCORING. Universiti Tekologi Malaysia.

Kennedy, J. and Eberhart, R. (1995) “Particle Swarm Optimization”, in *Proceedings of ICNN’95-International Conference on Neural Networks*, pp. 1942–1948.

Kesserwan, N. Dssouli, R., Bentahar, J. (2019) “From use case maps to executable test procedures : a scenario-based approach”, *Software & Systems Modeling*. Springer Berlin Heidelberg, 18(2), pp. 1543–1570. doi: 10.1007/s10270-017-0620-y.

Kim, J. (2014) “A Comparison of Software Product Line Traceability Approaches from End-to-End Traceability Perspectives”, *International Journal of Software Engineering and Knowledge Engineering*, 24(4), pp. 677–714. doi: 10.1142/S0218194014500260.

Kolb, R. (2014) “A Risk-Driven Approach for Efficiently Testing Software Product Lines”, *5th GPCE Young, Researches Workshop, Erfurt, Germany*, (August 2003).

Lackner, H. (2015) “Model-based product line testing: Sampling configurations for optimal fault detection”, *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9369, pp. 238–251. doi: 10.1007/978-3-319-24912-4_17.

Lackner, H. (2017) *Domain-Centered Product Line Testing*. Humboldt-Universität zu Berlin.

Lapeña, R., Pérez, F. and Cetina, C. (2017) “On the Influence of Models-to-Natural-Language Transformation in Traceability Link Recovery among Requirements and Conceptual Models”, in *ER Forum/Demos*, pp. 271–284.

Lee, J., Kang, S. and Jung, P. (2020) “Test coverage criteria for software product line testing: Systematic literature review”, *Information and Software Technology*. Elsevier B.V., p. 106272. doi: 10.1016/j.infsof.2020.106272.

Lee, J., Kang, S. and Lee, D. (2012) “A survey on software product line testing”, *Proceedings of the 16th International Software Product Line Conference*, Volume 1, pp. 31–40. doi: 10.1145/2362536.2362545.

Lee, K., Kang, K. C. and Lee, J. (2007) “Concepts and Guidelines of Feature Modeling for Product Line Software Engineering”, *International Conference on Software Reuse*, pp. 62–77. doi: 10.1007/3-540-46020-9_5.

Lity, S. Seidl, C., Nahrendorf, S. (2018) “175 % Modeling for Product-Line Evolution of Domain Artifacts”, *ACM Computing Machinery*, (1), pp. 27–34.

Lity, S. B. (2019) *Model-Based Product-Line Regression Testing of Variants and*

Versions of Variants. Technische Universitat Braunschweig.

Lizhang, X. L. (2014) “An Evolutionary Methodology for Optimized Feature Selection in Software Product Lines”, in *Proceedings of the International Conference on Software Engineering and Knowledge Engineering, SEKE*, pp. 2–5.

Lochau, M. Oster, S., Goltz, U. (2012) “Model-based pairwise testing for feature interaction coverage in software product line engineering”, *Software Quality Journal*, 20(3–4), pp. 567–604. doi: 10.1007/s11219-011-9165-4.

Lochau, M. Lily, S., Lachmann, R. (2014) “Delta-oriented model-based integration testing of large-scale systems”, *Journal of Systems and Software*. Elsevier Inc., 91(1), pp. 63–84. doi: 10.1016/j.jss.2013.11.1096.

Lochau, M. Burdek, J., Holzle, S. (2017) “Specification and automated validation of staged reconfiguration processes for dynamic software product lines”, *Software and Systems Modeling*. Springer Berlin Heidelberg, 16(1), pp. 125–152. doi: 10.1007/s10270-015-0470-4.

Luiz, H. Filho, J., Ferreira, T. (2018) “Incorporating User Preferences in a Software Product Line Testing Hyper-Heuristic Approach”, *2018 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, pp. 1–8.

Machado, I., McGregor, J., Cavalcanti, Y. (2014) “On strategies for testing software product lines: A systematic literature review”, *Information and Software Technology*. Elsevier B.V., 56(10), pp. 1183–1199. doi: 10.1016/j.infsof.2014.04.002.

Mamun, A. Al, Djatmiko, F. and Das, M. K. (2016) “Binary Multi-objective PSO and GA for Adding New Features into an Existing Product Line”, *19th International Conference on Computer and Information Technology (ICCIT)*. IEEE, pp. 581–585. doi: 10.1109/ICCITECHN.2016.7860263.

Manning, C. D. (2015) *Introduction to Information Retrieval*. Available at: http://www.math.unipd.it/aiolli/corsi/0910/IR/irbookprint.pdf%5Cnfiles/601/irbookprint.pdf.

Marcus, A. and Maletic, J. I. (2005) “RECOVERY OF TRACEABILITY LINKS BETWEEN SOFTWARE”, *International Journal of Software Engineering and Knowledge Engineering*, 15.05, pp. 811–836.

Marino, J., Alexandre, V. and Júnior, D. S. (2019) “A systematic mapping addressing Hyper-Heuristics within Search-based Software Testing ☆”, *Information and Software Technology*. Elsevier B.V., 114(September 2018), pp. 176–189. doi:

10.1016/j.infsof.2019.06.012.

Morgan, S. P. (1998) “Richard Wesley Hamming”, *Notices of the AMS*, 45.8.

Nebut, C. Fleurey, F., Traon, Le. (2010) “A Requirement-Based Approach to Test Product Families”, *International Workshop on Software Product-Family Engineering*, pp. 198–210. doi: 10.1007/978-3-540-24667-1_15.

Olimpiew, E.M. Gomaa, H. (2008) “Model-based Test Design for Software Product Lines”, *Software Product Line Conference*, (January), pp. 173–178.

Olimpiew, E. M. and Gomaa, H. (2009) “Reusable Model-Based Testing”, *International Conference on Software Reuse*, pp. 76–85.

Olsen, G. K. and Oldevik, J. (2013) “Scenarios of Traceability in Model to Text Transformations”, *European Conference on Model Driven Architecture-Foundations and Applications*, pp. 144–156.

Oster, S. (2012) *Feature Model-based Software Product Line Testing, Phd Thesis*. Technische Universität. Available at: http://tuprints.ulb.tu-darmstadt.de/2881/.

Oster, S. (2012) *Feature Model-based Software Product Line Testing*. Technische Universität.

P. Asirelli, M.H. ter Beek, S. G. (2009) “Deontic Logics for Modeling Behavioural Variability”, in *Proceedings of the 3rd International Workshop on Variability Modelling of Software-intensive Systems*, pp. 71–76.

Papadakis, M., Klein, J. and Traon, Y. Le (2016) “Assessing Software Product Line Testing via Model-based Mutation: An Application to Similarity Testing”, *International Conference on Software Testing, Verification and Validation Workshops*, (November), pp. 188–197. doi: 10.1109/ICSTW.2013.30.

Perrouin, G. Sen, S. (2010) “Automated and scalable T-wise test case generation strategies for Software Product Lines”, *ICST 2010 - 3rd International Conference on Software Testing, Verification and Validation*, pp. 459–468. doi: 10.1109/ICST.2010.43.

Perrouin, G., Oster, S., Sen, S. (2011) “Pairwise testing for software product lines: comparison of two approaches”, *Software Quality Journal*, 20(3–4), pp. 605–643. doi: 10.1007/s11219-011-9160-9.

Reis, S., Metzger, A. and Pohl, K. (2006) *A reuse technique for performance testing of software product lines, Proceedings of the International. Workshop on Software Product Line Testing*.

Reuling, D. Burdek, J., Rotarmel, S. (2015) “Fault-based product-line testing”, *SPLC*

- *International Conference on Software product lines*, pp. 131–140. doi: 10.1145/2791060.2791074.

Reuys, A. Kamsties, E., Pohl, K. (2010) “Model-Based System Testing of Software Product Families”, *International Conference on Advanced Information Systems Engineering,* pp. 519–534. doi: 10.1007/11431855_36.

Rose, L. and Matragkas, N. (2012) “A Feature Model for Model-to-Text Transformation Languages”, *2012 4th International Workshop on Modeling in Software Engineering (MISE).* IEEE, pp. 57–63. doi: 10.1109/MISE.2012.6226015.

Ross, P. (2003) “HYPER-HEURISTICS”, in *Algorithm for Sofware Engineering. Boston, MA: Springer,* pp. 529–556

S.Kang, J.Kim, J. lee (2014) “A comparison of software product line traceability approaches from end-to-end traceability perspectives”, *International Journal of Sofware Engineering Knowledge Engineering,* 24(04), pp. 677–714.

SA Halim; DN Jawawi, M. S. (2019) “SIMILARITY DISTANCE MEASURE AND PRIORITIZATION ALGORITHM FOR TEST CASE PRIORITIZATION IN SOFTWARE PRODUCT LINE TESTING”, *Journal of Information & Communication Technology*, 18.1.

Saeed, A., Ab Hamid, S. H. and Mustafa, M. B. (2016) “The experimental applications of search-based techniques for model-based testing: Taxonomy and systematic literature review”, *Applied Soft Computing Journal*. Elsevier B.V., 49, pp. 1094–1117. doi: 10.1016/j.asoc.2016.08.030.

Sahak, M. (2018) *Effective similarity based test case prioritization technique for software product lines*. Universiti Teknologi Malaysia.

Saikia, A. Baruah, R., Sarma, U. (2018) “Jaro Winkler Fuzzy match algorithm to calculate a similarity index between two strings using open source platform”, *INTERNATIONAL JOURNAL FOR INNOVATIVE RESEARCH IN MULTIDISCIPLINARY FIELD,* 4(8), pp. 138–143.

Samih, H and Guen, H. (2014) “Deriving usage model variants for model-based testing: An industrial case study”, *Proceedings of the IEEE International Conference on Engineering of Complex Computer Systems, ICECCS*, pp. 77–80. doi: 10.1109/ICECCS.2014.19.

Samimi-dehkordi, L., Zamani, B. and Kolahdouz-rahimi, S. (2019) “Leveraging product line engineering for the development of domain-specific metamodeling languages ☆”, *Journal of Computer Languages*. Elsevier, 51(February), pp. 193–213. doi: 10.1016/j.cola.2019.02.006.

Sánchez, Ana B., Sergio Segura, A. R.-C. (2014) “A Comparison of Test Case Prioritization Criteria for Software Product Lines”, *IEEE Seventh International Conference on Software Testing, Verification and Validation*, pp. 41–50. doi: 10.1109/ICST.2014.15.

Sayyad, A. S. (2013) “On the Value of User Preferences in Search-Based Software Engineering : A Case Study in Software Product Lines”, in *Proceedings of the 2013 International Conference on Software Engineering*, pp. 492–501.

Schaefer, I. *et al.* (2016) “Fine-grained test case prioritization for integration testing of delta-oriented software product lines”, 1(212), pp. 1–10. doi: 10.1145/3001867.3001868.

Schaefer, I. and Seidl, C. (2018) “175% Modeling for Product-Line Evolution of Domain Artifacts”, *Proceedings of the 12th International Workshop on Variability Modelling of Software-Intensive Systems*, (1), pp. 27–34. doi: 10.1145/3168365.3168369.

Shen, L., Peng, X. and Zhao, W. (2008) “A Comprehensive Feature-Oriented Traceability Model for Software Product Line Development”, in *ECMDA Traceability Workshop Proceedings*, pp. 77–86.

Steghöfer, J. Brink, C. (2019) “A Generic Traceability Metamodel for Enabling Unified End-to-End Traceability in Software Product Lines”, in *34th ACM/SIGAPP Symposium on Applied Computing*, pp. 2344–2353.

Strickler, A. Guen, H. (2016) “Deriving products for variability test of Feature Models with a hyper-heuristic approach”, *Applied Soft Computing Journal*. Elsevier B.V., 49, pp. 1232–1242. doi: 10.1016/j.asoc.2016.07.059.

Strickler, A. Kuk, J., Vergillio, S., (2017) “Hyper-Heuristic Based Product Selection for Software Product Line Testing”, (may), pp. 34–45.

Sulaiman, R. A., Jawawi, D. and Halim, S. A. (2018) “Coverage-based approach for model-based testing in Software Product Line”, *International Journal of Engineering and Technology(UAE)*, 7(4). doi: 10.14419/ijet.v7i4.15.21373.

Szasz, N. and Vilanova, P. (2008) “Statecharts and Variabilities”, *VaMoS*, pp. 131–140.

Tsafarakis, S., Marinakis, Y. and Matsatsinis, N. (2011) “Particle swarm optimization for optimal product line design”, *International Journal of Research in Marketing*.

Elsevier B.V., 28(1), pp. 13–22. doi: 10.1016/j.ijresmar.2010.05.002.

Tumeng, R. A. (2017) *Test case prioritization with requirements change using string metrics*. Universiti Teknologi Malaysia.

Ur, S. Khan, R., Lee, S. (2018) "A Systematic Review on Test Suite Reduction : Approaches , Experiment ' s Quality Evaluation , and Guidelines", *IEEE Access*. IEEE, 6(ii), pp. 11816–11841. doi: 10.1109/ACCESS.2018.2809600.

Utting, M., Pretschner, A.Legeard, B. (2010) "A taxonomy of model-based testing approaches", *Software Testing Verification and Reliability*, 24(8), pp. 591–592. doi: 10.1002/stvr.

Utting, Mark, B. L. (2010) *Practical model-based testing: a tools approach*. Elsevier.

Utting, M. and Legeard, B. (2006) *A Taxonomy of Model-Based Testing*.

Vale, T. Santana, E., Almeida, D. (2017) "Software product lines traceability : A systematic mapping study", *Information and Software Technology*. Elsevier B.V., 84, pp. 1–18. doi: 10.1016/j.infsof.2016.12.004.

Varshosaz, M. and Mousavi, M. R. (2019) "Comparative Expressiveness of Product Line Calculus of Communicating Systems and 1-Selecting Modal Transition Systems", (Sofsem), pp. 490–503. doi: 10.1007/978-3-030-10801-4_38.

Varshosaz, M. and Schneider, G. (2017) *Test Models and Algorithms for Model-Based Testing of Software Product Lines*.

Varshosaz, M., Schneider, G. and Mostowski, W. (2019) *Modeling and Model-Based Testing of Software Product Lines*. Halmstad University Press.

Voelter, M. and Groher, I. (2013) "Product Line Implementation using Aspect-Oriented and Model-Driven Software Development", in *11th International Software Product Line Conference Product*, pp. 233–242. doi: 10.1109/SPLINE.2007.23.

Wang, S. Gotlieb, A., Ali, S. (2012) "Automated Selection of Test Cases using Feature Model for Product Lines : An Industrial Case Study", (October).

Wang, S. Gotlieb, A., Ali, S. (2013) "Using feature model to support model-based testing of product lines: An industrial case study", *Proceedings of the International Symposium on the Physical and Failure Analysis of Integrated Circuits, IPFA*, pp. 75–84. doi: 10.1109/QSIC.2013.51.

Wang, S. Gotlieb, A., Ali, S. (2014) "Multi-Objective Test Prioritization in Software Product Line Testing : An Industrial Case Study", *Proceedings of the 18th International Software Product Line Conference*, 1, pp. 32–41.

Wang, S. (2014) “Systematic Product Line Testing : Methodologies , Automation , and Industrial Application”, (November).

Wang, S. Gotlieb, A., Ali, S. (2016) “A systematic test case selection methodology for product lines: results and insights from an industrial case study”, *Empirical Software Engineering*. Empirical Software Engineering, 21(4), pp. 1586–1622. doi: 10.1007/s10664-014-9345-5.

Wang, S. and Ali, S. (2013) “Minimizing Test Suites in Software Product Lines Using Weight-based Genetic Algorithms”, *Proceedings of the 15th annual conference on Genetic and evolutionary computation. ACM, 2013*, pp. 1493–1500.

Wang, S., Ali, S. and Gotlieb, A. (2015) “Cost-effective test suite minimization in product lines using search techniques”, *Journal of Systems and Software*. Elsevier Ltd., 103, pp. 370–391. doi: 10.1016/j.jss.2014.08.024.

Wang, Y., Xing, Y., Gong, Y. (2014) “Optimized Branch and Bound for Path-wise Test Data Generation”, *International Journal of Computers Communications & Control 9.4*, 9(4), pp. 497–509.

Weißleder, S. (2010) *Test Models and Coverage Criteria for Automatic Model-Based Test Generation with UML State Machines DISSERTATION*. Diss. Humboldt University of Berlin.

Weißleder, S. and Lackner, H. (2013) “Top-Down and Bottom-Up Approach for Model-Based Testing of Product Lines”, *Electronic Proceedings in Theoretical Computer Science*, 111(Mbt), pp. 82–94. doi: 10.4204/eptcs.111.7.

Weißleder, S., Sokenou, D. and Schlingloff, B.-H. (2008) “Reusing State Machines for Automatic Test Generation in Product Lines”, *1st Workshop on Model-based Testing in Practice (MoTiP “08)*, 6, p. 10. Available at: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.169.5699&rep=rep1&type=pdf#page=21.

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B. and Wesslén, A. (2012) "*Experimentation in software engineering"*. Springer Science & Business Media.

Xhevahire Ternava (2017) "*Handling Variability at the Code Level : Modeling , Tracing and Checking Consistency"*. Universiite Cote d Azur.

Xing, Y. (2014) “An Intelligent Method Based on State Space Search for Automatic Test Case Generation”, *Journal of Software*, 9(2), pp. 358–364. doi: 10.4304/jsw.9.2.358-364.

Xing, Y. Gong, Y., Wang, Y. (2015) “A Hybrid Intelligent Search Algorithm for

Automatic Test Data Generation”, 2015.

Yoo, S. and Harman, M. (2012) “Regression testing minimization , selection and prioritization : a survey”, *Software testing, verification and reliability*, 22.2(March 2010), pp. 67–120. doi: 10.1002/stvr.

Zamli, K. Z., Alkazemi, B. Y. and Kendall, G. (2016) “A Tabu Search hyper-heuristic strategy for t-way test suite generation”, *Applied Soft Computing Journal*. Elsevier B.V., 44, pp. 57–74. doi: 10.1016/j.asoc.2016.03.021.

Zheng, Y., Cu, C., & Asuncion, H. U. (2017) “Mapping Features to Source Code through Product Line Architecture: Traceability and Conformance”, in *IEEE International Conference on Software Architecture (ICSA)*, pp. 225–234.

Zisman, A. (2012) “Using Rules for Traceability Creation”, *Software and Systems Traceability*, (2010), pp. 147–170. doi: 10.1007/978-1-4471-2239-5.

Zitzler, E., Laumanns, M. and Thiele, L. (2001) *SPEA2 : Improving the Strength Pareto Evolutionary Algorithm*.