

CAPITAL UNIVERSITY OF SCIENCE AND
TECHNOLOGY, ISLAMABAD



Test Case Prioritization for Software Product Line Testing

by

Sundus Ali Qureshi

A thesis submitted in partial fulfillment for the
degree of Master of Science

in the

Faculty of Computing
Department of Computer Science

2018

Copyright © 2018 by Sundus Ali Qureshi

All rights reserved. No part of this thesis may be reproduced, distributed, or transmitted in any form or by any means, including photocopying, recording, or other electronic or mechanical methods, by any information storage and retrieval system without the prior written permission of the author.

Dedication

This thesis work is dedicated to my parents, who have been a continuous source of support, love, and encouragement during the challenges of life. I also dedicate this work to my teachers, who always thought me to work hard with their constant supervision and encouragement.



CAPITAL UNIVERSITY OF SCIENCE & TECHNOLOGY
ISLAMABAD

CERTIFICATE OF APPROVAL

Test Case Prioritization for Software Product Line Testing

by

Sundus Ali Qureshi

MCS163027

THESIS EXAMINING COMMITTEE

S. No.	Examiner	Name	Organization
(a)	External Examiner	Dr. Rizwan Bin Faiz	RIU, Islamabad
(b)	Internal Examiner	Dr. Muhammad Tanvir Afzal	CUST, Islamabad
(c)	Supervisor	Dr. Aamer Nadeem	CUST, Islamabad

Dr. Aamer Nadeem

Thesis Supervisor

October, 2018

Dr. Nayyer Masood

Head

Dept. of Computer Science

October, 2018

Dr. Muhammad Abdul Qadir

Dean

Faculty of Computing

October, 2018

Author's Declaration

I, **Sundus Ali Qureshi** hereby state that my MS thesis titled “**Test Case Prioritization for Software Product Line Testing**” is my own work and has not been submitted previously by me for taking any degree from Capital University of Science and Technology, Islamabad or anywhere else in the country/abroad.

At any time if my statement is found to be incorrect even after my graduation, the University has the right to withdraw my MS Degree.

Sundus Ali Qureshi

Registration No: MCS163027

Plagiarism Undertaking

I solemnly declare that research work presented in this thesis titled “**Test Case Prioritization for Software Product Line Testing**” is solely my research work with no significant contribution from any other person. Small contribution/help wherever taken has been dully acknowledged and that complete thesis has been written by me.

I understand the zero tolerance policy of the HEC and Capital University of Science and Technology towards plagiarism. Therefore, I as an author of the above titled thesis declare that no portion of my thesis has been plagiarized and any material used as reference is properly referred/cited.

I undertake that if I am found guilty of any formal plagiarism in the above titled thesis even after award of MS Degree, the University reserves the right to withdraw/revoke my MS degree and that HEC and the University have the right to publish my name on the HEC/University website on which names of students are placed who submitted plagiarized work.

Sundus Ali Qureshi

Registration No: MCS163027

Acknowledgements

First I would like to thank ALLAH (S.W.T) for providing me with the strength, knowledge, and blessing to complete this thesis, without His help and blessing it was not possible.

I would like to thank my supervisor Dr. Aamer Nadeem for his patient guidance, which encouraged me for self-learning and advices he has provided throughout my times as his student. I have been very lucky to have a supervisor who cared so much about my work, and who responded to my queries. My sincere gratitude to him for the tremendous ongoing encouragements that have throughout enabled me to put in the efforts and all-time dedication for the attainment of my research work. I am thankful to all my teachers who helped me and all the members of CSD (Center for Software Dependability) group especially Mr. Qamar uz Zaman for their comment and feedback on my thesis.

I cannot thank my parents enough for the education privilege that they have given me; without this, it would have never been possible to accomplish this work. I am thankful to my brothers and sister-in-law from whom I have got constant and unconditional support with affection. I am grateful to my friend Asra Ishtiaq who remained a source of moral support for me, which was the constant energy boost for me.

In the last, I am grateful to everyone from my family to friends, from teachers to colleagues who have always been a part of this research work.

Abstract

Software Product Line (SPL) is a collection of similar products using a common set of features to develop the product to fulfill a particular mission and to satisfy the needs of the market segment. Instead of developing each product from scratch, a common set of features combined to build the new product. The goal of the product line is to provide reusability and variability of the software and reduce the cost and time in producing the new software products with the improved quality. Feature model (FM) provides the information about the features and relations among them, it represents SPL too. FM enables to generate the products from the combination of features. Combination of features may lead to exponential growth in the number of products. Due to the combinatorial explosion of the products, testing of SPL becomes a challenging task. A fault present in any feature can appear in many products generated from the FM and can cause failure in many products. To avoid such circumstances, the proper testing strategy is required to be implemented.

To test all the derivable products is not feasible in terms of time and cost, as the number of products increases exponentially with the numbers of features in the FM. Test case prioritization for SPL is used to order the products based on some criteria so that they are effective in time and cost with the ability to detect maximum faults in it. Various approaches have been proposed for the prioritization of the products based on some criteria. Existing criteria for prioritization are feature priority criteria, feature coverage criteria, feature frequency criteria and feature coupling criteria. Most of the techniques considered the feature coverage and feature coupling complexity criteria for prioritization of SPL product.

In this research, we have proposed a prioritization criterion that is the combination of two criteria. First, individual feature complexity criteria and second, feature coupling complexity criteria. Individual feature complexity is calculated from the use case descriptions of the feature and feature coupling complexity criterion is used from the existing approach in which we calculated how tightly features are coupled. With the combination of these two criteria, a product's complexity is

calculated and the highest priority is assigned to the product that achieved the highest complexity.

Proposed approach significantly improves the rate of fault detection. We have performed detailed experimentation to measure the effectiveness of criterion based on the rate of fault detection and evaluated our work with the existing technique that performs the best in terms of fault detection rate. Three case studies are used to evaluate the performance of our work. The results show that different ordering of the same test suite leads to the considerable differences in the average percentage of fault detection rate.

Contents

Author’s Declaration	iv
Plagiarism Undertaking	v
Acknowledgements	vi
Abstract	vii
List of Figures	xii
List of Tables	xiv
List of Abbreviations	xvi
1 Introduction	1
1.1 Software Product Lines	1
1.1.1 Motivation for Software Product Line Development	3
1.1.2 Feature Model	4
1.2 Software Product Line Testing	8
1.3 Test Case Prioritization for SPL Testing	9
1.3.1 Prioritization Criteria	10
1.3.2 Evaluation Metric	10
1.4 Problem Statement	11
1.5 Research Question	12
1.6 Research Methodology	12
1.7 Thesis Organization	13
2 Literature Review	14
2.1 Prioritization Criteria based on Feature Coverage	14
Devroey et al., 2014	15
Henard et al., 2014	15
Al-Hajjaji et al., 2014	16
Wang et al., 2014	16
Sánchez et al., 2014	17
Al-Hajjaji et al., 2016	17

	Al-Hajjaji et al., 2017	18
	Al-Hajjaji et al., 2017	18
2.2	Prioritization Criterion based on Feature	19
	Ensan et al., 2011	19
	Sánchez et al., 2014	20
2.3	Prioritization Criteria based on Feature Coupling	20
	Sánchez et al., 2014	20
2.4	Analysis and Comparison	21
	2.4.1 Gap Analysis	25
3	Proposed Approach	26
3.1	Proposed Prioritization Algorithm	27
3.2	Proposed Criterion for Product Prioritization	34
	3.2.1 Feature Complexity	35
	Use Case Model	35
	Use Case Metrics	37
	3.2.2 Feature Coupling Complexity	40
	3.2.3 Combining Feature Complexity and Feature Coupling Com- plexity	41
3.3	Example of Proposed Approach	42
4	Implementation	52
4.1	Implementation Details	52
	4.1.1 Test Suite	54
	4.1.2 Feature Complexity	55
	4.1.3 Feature Coupling Complexity	56
	4.1.4 Test Suite Prioritization	57
5	Results and Discussion	59
5.1	Case Studies	60
	5.1.1 E-Commerce SPL	60
	5.1.2 Social Network SPL	61
	5.1.3 Transport Network SPL	61
5.2	Fault Injection	62
5.3	Evaluation Metric	62
	5.3.1 Example	63
5.4	Experiment 1. Evaluation of Proposed Criterion Based on Different values of α	64
	5.4.1 Prioritized Test Suites Based on Proposed Criterion	64
	Prioritized list for E-Commerce	64
	Prioritized list for Social Network	66
	Prioritized list for Transport Network	67
	5.4.2 Comparison	69
5.5	Experiment 2. Evaluation of Proposed and Existing Criteria	70

5.5.1	Prioritized Test Suites Based on Existing and Proposed Criteria	70
5.5.2	Comparison	72
6	Conclusion and Future Work	77
6.1	Future work	78
	Bibliography	80
	Appendices	87
A	Prioritized Test Suite for Subject Product Line	87
B	Subject Product Line	95
C	Test Suite for Subject Product Line	98

List of Figures

1.1	Feature Model Example.	5
1.2	Mandatory Feature.	5
1.3	Optional Feature.	6
1.4	Alternative Relation.	6
1.5	Or Relation.	7
1.6	Cross-tree Constraints.	7
3.1	An Illustration of Proposed Solution.	29
3.2	Feature-Model-Example.	43
3.3	Use Case Description.	44
4.1	FM in FeatureIDE.	53
4.2	Constraint Editor of FeatureIDE.	53
4.3	Configuration of FeatureIDE.	54
4.4	Class Diagram of Proposed Algorithm.	55
4.5	Output of Prioritized Test Suites on Console.	58
5.1	Graphical Representation of APFD for E-Commerce Product Line Based on Different Values of α	66
5.2	Graphical Representation of APFD for Social Network Product Line.	67
5.3	Graphical Representation of APFD for Transport Network Product Line Based on Different Values of α	69
5.4	APFD Comparison of Existing and Proposed Criterion for E-Commerce Subject Product Line.	72
5.5	Graphical Representation of Fault Detection of Test Cases for E- Commerce Subject Product Line.	73
5.6	APFD Comparison of Existing and Proposed Criterion for Social Network Subject Product Line.	73
5.7	Graphical Representation of Fault Detection of Test Cases for Social Network Subject Product Line.	74
5.8	APFD Comparison of Existing and Proposed Criterion for Trans- port Network Subject Product Line.	74
5.9	Graphical Representation of Fault Detection of Test Cases for Trans- port Network Subject Product Line.	75
5.10	Graphical Representation of APFD Comparison for Subject Prod- uct Lines	76

B.1	E-Commerce Feature Model.	95
B.2	Social Network Feature Model.	96
B.3	Transport Network Feature Model.	97

List of Tables

2.1	Comparison of Techniques Based on Prioritization Criteria	23
3.1	Weights(α) for Test Suite	42
3.2	Test Suite	44
3.3	Use Case Metric Values for the Use Case of Figure 3.3	45
3.4	Complexity of Features	46
3.5	FC of Test Suite	47
3.6	Normalized FC of Test Suite	47
3.7	VC and CC of Test Suite	48
3.8	FCC of Test Suite	49
3.9	FCC of Feature Model	49
3.10	Normalized FCC of Test Suite	50
3.11	Test Suite Complexities	51
3.12	Prioritized Test Suite	51
5.1	Summary of Subject Product Lines.	61
5.2	APFD for E-Commerce Product Line Based on Different Values of α . 65	
5.3	APFD for Social Network Product Line Based on Different Values of α	66
5.4	APFD for Transport Network Product Line Based on Different Values of α	68
5.5	Priority List for E-Commerce Product Line	70
5.6	Priority List for Social Network Product Line	71
5.7	Priority List for Transport Network Product Line	71
5.8	Comparison of APFD for Subject Product Lines.	76
A.1	Prioritized Test Suite for E-Commerce Product Line Based on Proposed Criterion	87
A.2	Prioritized Test Suite for Social Network Product Line Based on Proposed Criterion	90
A.3	Prioritized Test Suite for Transport Network Product Line Based on Proposed Criterion	92
C.1	Alphanumeric Character for Features of E-Commerce Product Line	98
C.2	E-Commerce Product Line Test Suite	99
C.3	Alphanumeric Character for Features of Social Network Product Line	102

C.4	Social Network Product Line Test Suite	103
C.5	Alphanumeric Character for Features of Transport Network Product Line	107
C.6	Transport Network Product Line Test Suite	108

List of Abbreviations

SPL	Software Product Line
SPLE	Software Product Line Engineering
FM	Feature Model
VC	Variability Coverage
CC	Cyclomatic Complexity
FC	Feature Complexity
FCC	Feature Coupling Complexity
TCC	Test Case Complexity
APFD	Average Percentage of Faults Detected
SPLOT	Software Product Lines Online Tool

Chapter 1

Introduction

In this chapter, we give an introduction of the Software Product Line (SPL), Feature Model (FM), and the test case prioritization of the SPL. The initial section discusses the importance of SPL and FM, afterward, the challenges for testing the SPL and prioritization criteria are discussed. In the last section of this chapter problem statement and questions of research work is elaborated and the briefly express the steps of the methodology of the proposed solution.

1.1 Software Product Lines

Software Product Line (SPL) is a concept of family, which is a collection of related products that share a common parts of the products known as core assets to fulfill the needs of the particular market segment. Core assets are common in all the products of the family and variable assets are used in particular products [1]. These assets are commonly known as 'feature' and features in SPL consider as a commonality and variability of product. Behavior and capabilities (commonality and variability) of the features are used to differentiate among the products. Instead of developing each product from scratch, a common set of features (core assets) are combined with the variable features to build the new product. Product line approach is known to be a successful approach for reusability because of the

common features in a product line that can be used in many other products of the family [2].

Product line allows producing a family of products, where a feature is incremented in the functionality of the product to achieve the goal of the market with a significant reduction in cost and time. The aim is to reuse the existing assets of a product that is a part of SPL. Several assets are frequent in the products of the product line and few of them are particular for individual products. Consider an example of a product line of social messaging application in which core assets can be the 'messages' and 'call' features whereas 'group video call' and 'status updates' features can be the variable assets of the product line. Which means all the products of the social messaging application contains a 'messages' feature and 'call' feature, as they are the basic functionality of the social messaging application. However, in some social messaging application 'group video call' and 'status updates' may or may not be present as they are optional features. WhatsApp is an example that introduced 'status updates' feature and now recently they introduced 'group video call' feature. These optional facilities are the variability of the product line. Thus, Software Product Line Engineering (SPLE) is generating the group of related software systems instead of generating the single software system. For that reason, there is a need for a basic artifact that specifies and states the variability and commonality between the products of related systems.

SPL represents prominent innovation to support the derivation of an extensive range of applications. SPL gained significant momentum in recent years. Product line approach is extensively used in hardware manufacturing fields but recently it revealed a greater influence on the software development process [3]. For example, all the car models introduced by car manufacturing company that introduces the new cars with the additional functionality such as variation with gears and doors. Many industrial case studies represent the effectiveness of SPL in reducing time and development cost such as Nokia, Boeing, and Video Conference Systems (VCSs) developed by Cisco Systems (Norway)[4, 5]. There are many other examples of successful companies that are using the SPL mentioned in the website the Hall of Fame [6].

1.1.1 Motivation for Software Product Line Development

The motivation to develop the product line instead of single system development are as follows:

- The fundamental use of SPL is the reduction in cost. In a single system, if there is a need to use the part of the software in several kinds of software systems. It involves an extra cost and time in order to reuse them in the different software platform. On the other hand, if parts of the software are managed for the re-usability and the platform is created before then it can reduce the cost per product as schedules and resources can be used again from the preceding project [7]. The world's largest company of diesel engines, Cummins Inc. claimed that they reduced the effort of 360 engineers that desired to manufacture the new software for the engine to 100 engineers [8]. Before implementing the product line approach, teams of software engineers developed the softwares that perform the operations of an engine such as control the ignition and the fuel delivery, for each new product with different standards, which conflict the end quality of resulting application. They decided to build the core assets that are commonly used in all the applications with the same quality that also reduced the number of software engineers required to build the application.
- Quality of the software is enhanced by reviewing and testing of many products. The extensive testing in various kinds of a product entails a greater opportunity of detecting faults and to accurate them, in that way products quality can be increased [7]. It also beneficial for testing as test plans, test cases and all the related test data are already available.
- Comparing with the single software development time, initially, product line needs greater time to develop the common part of software but after that, it requires considerably less time because the common part is reused in each new products [7]. Most of the customer requirements are same and can be

reused from the earlier project so that time and cost of requirement analysis is saved and feasibility is assured.

SPL is also called families of products because in SPL one product is design and multiple (families of products) products can be produced by reusing the features. As SPLE leads to less development effort and increased productivity in terms of cost, quality and time so, everyone in software industry wants to gain benefit by adopting SPLE. SPL are represented through the model named as, feature models (FM), they are most commonly used model for SPL in terms of variability [9]. There are some other models also that deals with the variability such as decision model [10] and orthogonal variability model [7].

1.1.2 Feature Model

Feature models are renowned to be the most significant contributions of SPLE [11, 12]. It was first presented in 1990 [13]. The major task of SPLE is to represent the variability and commonality of the products. FM is used to fulfill the stated purpose [14]. FM represents all achievable products of SPL. It is graphical and tree-like hierarchical representations of product lines with assemble set of features created by:

- Parent and child relations
- Constraints relations

The first proposed FM was the basic. Benavides explains two types of relations depicted from the basic FM; Parent and child features relationships [15]. Root feature is the feature that includes all the products. Child feature appears in product only if parent of that child feature appears in the product. Relationships between sub features i.e. cross-tree constraints consist of inclusion and exclusion. figure 1.1 shows the simple example of FM.

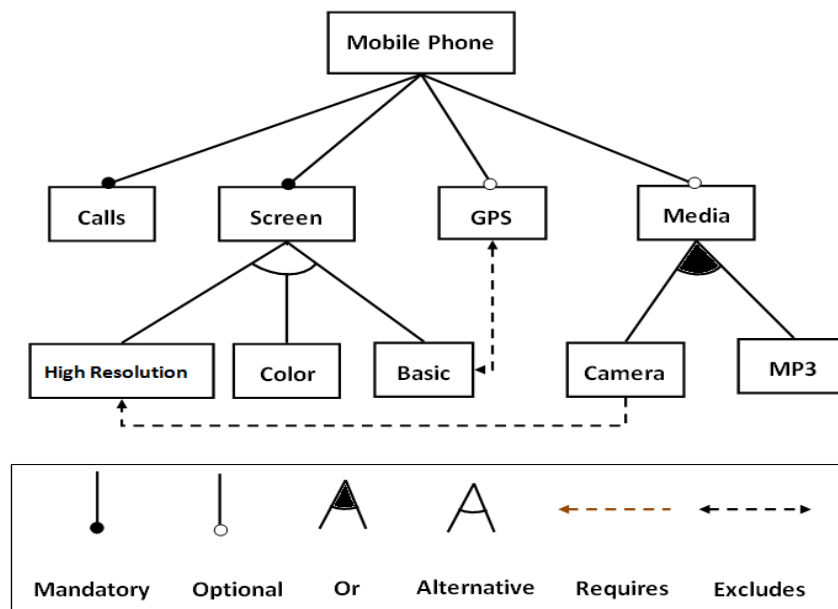


FIGURE 1.1: Feature Model Example.

Relationships between parent-child features categorized as:

1. **Mandatory** relation exists among parent and child feature which means that child feature must be present in all the products in which parent feature is included. Mandatory features are a compulsory feature in the product that included in all the product of the SPL whenever the parent feature is included. 'Calls' is the mandatory child feature in the figure 1.2 with parent feature 'Mobile Phone'.

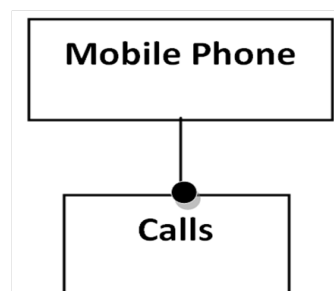


FIGURE 1.2: Mandatory Feature.

2. **Optional** relation exists among parent and child feature which means that child feature is not compulsory to include in all product in which parent

feature is included. GPS is the optional feature of parent 'Mobile Phone' in figure 1.3.

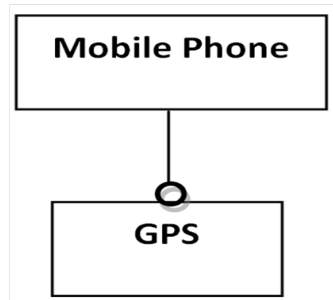


FIGURE 1.3: Optional Feature.

3. **Alternative** relation exists among parent and the set of child features from which only one feature can be selected from the set of child features if parent feature is selected in any product. 'High Resolution', 'Color' and 'Basic' are the set of child features with alternative relation with 'Screen' parent feature in figure 1.4.

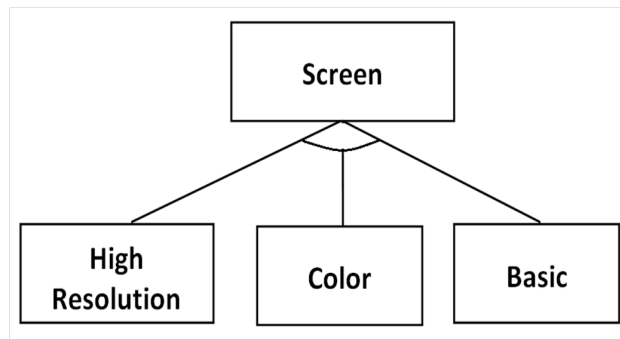


FIGURE 1.4: Alternative Relation.

4. **Or** relation exists among parent and the group of child features from which one or more than one feature can be selected from the set of child features if the parent feature is selected in any product. 'Camera' and 'MP3' are the set of child features with or relation with 'Media; parent feature in figure 1.5.

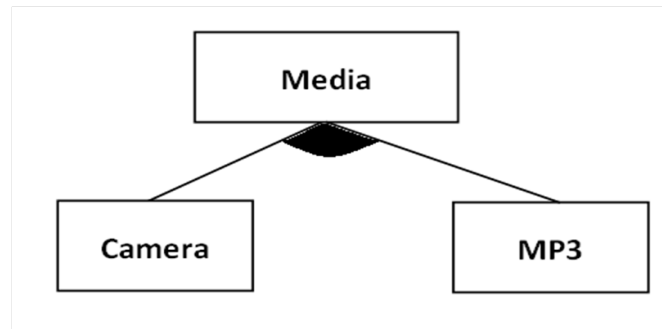


FIGURE 1.5: Or Relation.

Relationships between cross-tree constraint features that restrict the feature combination categorized as:

1. **Requires** cross-tree constraints shows in figure 1.6 with single arrow. Feature 'A' requires 'B' feature that means if feature 'A' appears in the product, feature 'B' must also appear in the same product with feature 'A'. 'Camera' feature requires 'High Resolution' feature in figure 1.1 whenever it appears in any product.
2. **Excludes** cross-tree constraints shows with double arrow in the figure 1.6. Feature 'A' excludes 'B' feature that means features 'A' and 'B' should not be included together in the product. As shown in figure 1.1 whenever 'GPS' feature is selected, 'Basic' feature must be deselected. Feature 'A' along with feature 'B' cannot be appear in the same product.



FIGURE 1.6: Cross-tree Constraints.

1.2 Software Product Line Testing

Software testing is a process to evaluate the effectiveness of software by detecting errors and improving it. It is a significant activity of the Software Development Life Cycle (SDLC), which consumes more than 50% of software resources. Software development is a complex and error-prone task as faults might occur in any stage of SDLC, they must be identified and removed as early as possible because if they propagate to other stages, a lot of resources and efforts required to detect them [16]. The objective of testing is to make ensure that the application is working correctly by executing its code. This objective can be accomplished in two ways. First, assist developers to find defects in program or application during the development so that they can fix it. Second, to validate whether an application is performing as mentioned in requirements [17].

Testing of a solo software system is a complicated task; SPL testing is more difficult and tricky because of the exponential increase in the number of products. Including only one feature from the FM in the product line may cause to increase a number of products generation. Testing each product from SPL would be ideal but it is time-consuming and too costly, nearly impossible. The quantity of products generating from the SPL is increasing by the combination of features that would lead to increases exponentially, resulting in millions of different products.

In SPL, **test case** is specifies as product or configuration (set of features) of the product line [18–20]. A product (also known as configuration) is generated from the FM by the selection of features; not all the feature selections are valid which means in-valid feature selection can also be selected that produce meaningless products. Constraint on the feature selection is known as feature dependency. Configuration of a product line is generated by the selection of valid features and feature selection is valid only if it fulfills the feature dependency.

Quantity of features increases in the FM might lead to generate millions of products from FM that means thousands of SPL test cases are generated from the FM. Testing a huge number of test cases is a challenging task in SPL. Several

approaches introduced to trim down the number of products required to be tested but reduced numbers of products are still high and it is difficult to test a large number of products within limited budget in terms of time and cost. Hence, tester wishes to find more faults in less time within limited resources.

1.3 Test Case Prioritization for SPL Testing

Prioritization testing techniques for SPL can increase the effectiveness of test suite by ordering the test cases to meet some performance goal. There are few approaches to prioritize products in order to make a list that suggest the order in which these products are to be tested so that they are cost effective and less time consuming with the ability to detect maximum faults in it. Prioritization technique does not select the sets of test cases from the test suite relatively it allows each test case to be executed. However, Elbaum et al. stated that elimination of test cases can be used in some cases by using test case prioritization with test case minimization technique or with test case selection technique [21]. In SPL, prioritization technique can be used with the test case selection or test case minimization techniques. Since the large number of test suite require to reduce the test suite and then prioritize it.

Tester uses the priority list so that it can decide when to quit testing in case of insufficient assets to execute all test cases. Probability is increased by the prioritization list that, in any case, if testing is ended before; because of some reason, the most imperative tests with higher priority being executed, hence increasing the rate of fault detection and facilitate troubleshooting in beginning periods of testing. One benefit of prioritization testing is increases in the rate of fault detection. If more faults detected by the minimum number of test cases and those test cases are on the top of the prioritized list then the fault detection rate is increased. The tester may be applied different prioritization criteria to assign priority to the test cases in order to meet some objective.

1.3.1 Prioritization Criteria

In SPL testing different criteria are used to prioritize the SPL test suite. Criteria are used to reorder the test cases to meet some goal. Feature coverage criteria is used which consider the coverage of features. The test case that covers the maximum number of features obtained the highest priority in the ordering list. Individual feature criteria is used in which the individual feature's importance or the individual feature's frequency can be considered. Importance of the feature may be based on the stakeholder's priority or some other criteria. The test case that contains a maximum number of important features attains the highest priority in the ordering list whereas, the frequency of the feature is based on the highly reused feature; a test case that contains a maximum number of those features that have the highest commonality achieves the highest priority in the ordering list. Another criterion, feature coupling complexity that measures how tightly features are coupled. The test case that obtains the highest coupling complexity obtained the highest priority in the ordering list.

1.3.2 Evaluation Metric

Test case prioritization order test cases in a way that give maximum benefit to the tester. Each test case in test suite assigns priority so that test cases can sort according their priorities and run earlier in the testing process [22]. Goal of prioritization is to attain high fault detection. Fault detection rate is affected by the ordering of test cases if the test case that cover maximum faults, run prior to the test case that cover only one fault which is already covered by the previous test case can increase the rate of fault detection within limited budget and time. After prioritization of test suite, effectiveness of test suite in terms of fault detection can be calculated by Average percentage of faults detected (APFD) metric developed by Elbaum et al. [21]. It is the measure to check how early faults are detected by the test suite [23]. Weighted average of faults detection for the system under test

is used to measure the APFD metric. Range of values from 0 to 1. High APFD value indicates faster fault detection rate [24]. APFD metric is given below:

$$1 - \frac{TF_1 + TF_2 + \dots + TF_m}{nm} + \frac{1}{2n}$$

T is the test suite consists of n test cases where F is the set of m faults detected by T . For ordering T' , TF_i is the order of first case that detects the i^{th} fault.

1.4 Problem Statement

Test case prioritization allows the tester to order the test cases based on some criteria, so that test cases with the highest priority execute earlier in the testing process to achieve some goal. Mostly used prioritization criteria for prioritizing the products of SPL are:

- Feature coverage criteria
- Feature coupling complexity criteria

The first criteria considered the maximum coverage of the features in the product and the highest priority is assigned to the product that covered the maximum number of features. The other criteria considered the coupling of the features and the highest priority is assigned to the product that contained the tightest coupled features. Existing approaches considered the feature coverage and feature coupling complexity. Not a single technique considered the feature complexity criterion for the prioritization. Feature complexity can also be considered in order to get the most complex test case. Also, most of the techniques used the single criterion for prioritization and combination of the criteria is not considered. Prioritization based on a single criterion ignored the other criteria, as feature coupling complexity criteria ignored the feature complexity criteria. Combination of different criteria and the complexity of the feature helps to improve the prioritization order of the SPL.

1.5 Research Question

In this research, we use feature model and use case descriptions to calculate the complexities of features to order the test cases. However, the following factor must also be taken into account:

RQ1. Does existing approaches consider the complexity of feature for prioritization?

In order to answer the research question, literature survey is carried out to investigate the existing prioritization criteria and gaps in these criteria are identified through depth analysis.

RQ2. Can test case prioritization through feature complexity improves fault detection rate?

To answer this question, different experiments are performed to get the APFD values of test suites using feature complexity criterion with feature coupling criterion and compared with the strongest existing criterion.

1.6 Research Methodology

1. In the first phase, we have done the literature review to find the most common and relevant prioritization technique use for SPL testing. After studying various techniques, we conclude that most of the proposed techniques use feature coverage criterion in which they consider only the differences among the features of the product and does not achieve high APFD value. While the approach that considers the complexity of the product is not appropriate, as it does not consider complexity the feature individually.
2. To conquer the gap in existing techniques, we have proposed a new criterion for ordering the SPL products in order to achieve a high rate of fault detection.

3. Implementation of our approach discussed in the following steps:
 - (a) In the initial phase, we will get the dataset of product lines from SPLOT repository [25] after that test suite will generate from feature models by using Feature IDE tool [26].
 - (b) Use case descriptions of the features from the feature models will generate. After that, we will measure the individual feature complexity from the use case description.
 - (c) Next, we will measure the feature-coupling complexity using the existing criterion.
 - (d) After that, product complexity will be calculated by combining the individual feature complexity and feature coupling complexity with the help of an α factor in order to propose the new prioritization criterion.
 - (e) Afterward, prioritization will perform to order the test cases and get the prioritized SPL test suite.
4. After generating the prioritized list, evaluation is performed. For evaluation, we will compare the proposed criterion with the existing criterion that achieved the highest APFD [20]. The test suite will prioritize based on our proposed prioritization criterion and with the existing prioritization criterion. Average Percentage of Faults Detection metric will use for the comparison of both prioritized test suites.

1.7 Thesis Organization

Rest of the thesis is organized as; Chapter 2 surveys the existing approaches on SPL prioritization and gap analysis. Chapter 3 defines a proposed technique for prioritization and in chapter 4 the implementation of the proposed technique is given. Chapter 5 discusses the results and chapter 6 conclude the research and give direction for future work.

Chapter 2

Literature Review

In this chapter, we discuss related work that has been done in the field of SPL test case prioritization. We conducted a detailed survey analysis to find the gap and comparison is performed among the existing approaches. The challenges of software product lines testing are extensively discussed in many research work. One of the most complex challenges is the interaction between the features that is a large number of possible configurations to be tested. Since large number of test cases, prioritization testing is an effective testing in early fault detection.

2.1 Prioritization Criteria based on Feature Coverage

In the following section, criteria based on feature coverage are discussed. T-wise coverage is used, some approaches used one value of T and few used the 2-wise coverage in which they consider the coverage of pair-wise features, and in one approaches T-wise coverage is used in which value of $T > 3$ is considered. All the criteria in this section are feature coverage with different values of T.

Devroey et al., 2014 In this approach, Devroey et al. used feature coverage criterion for prioritization [27]. They proposed a statistical approach in which behavior of products were considered for prioritization by using usage model (Markov Chain). They used three models, one was feature diagram that represents the relations and constraints between features, second was feature transition system (FTS) that was used to model the system behavior, that is, mapping feature to system behavior and third was usage model to extract the product selection and prioritize them. First, the usage model selects the traces of the products that were executed than second model FTS filter the transitions of products and keep those products that were valid products and executed at least once, in the third step each valid product that was executed combined with other valid products to generate the set of products. Probabilities of execution of products used to prioritize the products. Products were prioritizing according to the probability after that; test cases generated by applying the algorithm to the FTS. The evaluation was not performed although the proposed technique is based on operational profile of the software and highly depends on the nature of case study.

Henard et al., 2014 Feature coverage criterion is used in this approach for prioritization [28]. Henard et al. proposed similarity based prioritization to conquer the combinatorial issues comes from large feature models. The author stated that most combinatorial interaction testing (CIT) approaches fail to solve T-wise coverage in which $t > 3$. Hence, use search-based approach to combine sampling technique and prioritization based on similarity among configurations. They samples the product based on selected items. SAT solver used to produce the valid configuration of feature models in the boolean formula with $t = 3$ to 6 and use dissimilarity among the configuration to cover the maximum coverage. Prioritization is applied on test cases with the highest summation of distance between them. While their approach mainly targets the scalability of large feature models but constraint solver takes a large amount of time. However, their goal is to achieve maximum feature coverage instead of fault detection.

Al-Hajjaji et al., 2014 Feature coverage criterion and proposed similarity based prioritization is used by Al-Hajjaji et al.[29]. It was applied to the products before they were generated and implemented their criteria in FeatureIDE tool. They did not considered the behavior of SPL as used by Devroey et al.[27]. The goal was to increase the interaction coverage of SPL under test to grow faster after some time. A methodology based on by selecting the product with a maximum number of features as the first product to be tested. After that, the product that has the minimum similarity with the previous product selected using the hamming distance similarity measure. The distance was calculated by considering deselected features as well with selected features. Then the least minimum similar product with last two products was selected and repeats this process until all the products were selected for prioritization order. FeatureIDE tool was used and evaluation was performed by comparing with three sampling algorithms named as CASA, ICPL, Chavatal and random orders. More errors were not detected by the proposed approach when compared with sampling approach but it took limited time for testing. When compared with random order, similarity-based prioritization order was better.

Wang et al., 2014 This approach is based on feature pair wise coverage criterion for prioritization with search based technique by Wang et al.[30]. Proposed approach is different with all other approach as they focus on the cost aware prioritization. The objective was to minimize the execution time and maximize the rate of fault detection within a limited budget of test resources. For multiple criteria, four measures were defined one was cost measure (Overall Execution Cost) and three were effectiveness measures (Prioritized Extent of test cases, Fault Detection Capability and Feature Pair-wise Coverage). Considering all four measures fitness function was introduced. Evaluation performed using three search-based algorithms GA, AVM and (1+1) EA with defined fitness function on 500 artificial industrial problems. Two models are used to extract the test cases from the repository, Feature model and component family model . To test a product, relevant

features extracted from FM and the corresponding test cases for a product automatically selected from the repository, then optimal order for selected test cases find by proposed search based technique. (1+1)EA performed best in finding the optimal solution for test case prioritization.

Sánchez et al., 2014 Five prioritization criteria have proposed in this approach by Sánchez et al. based on the feature model metrics [20]. One is based on the feature coverage criterion. Similar to Henard et al.[28] dissimilarity of the products are considered and assign the highest priority to those products that are most dissimilar to cover the higher feature coverage criterion and rate of fault detection. For measuring the difference among products, Jaccard distance is used. Prioritized list is obtained by calculating the highest distance among the products; most dissimilar products are added to the prioritized list. The process continues to add the product to the list that has the maximum distance with the list until all the products are added. They have performed two experiments, in the first experimentation, prioritization is applied to the test suite that generated randomly. In the second experimentation, prioritization is applied on the test suite based on combinatorial selection. Evaluation based on the APFD values and comparison is performed. APFD of coverage criterion is high when compared with random and 2-wise order, that is 87.4% and 86.9% respectively that was better than the random and 2-wise selection.

Al-Hajjaji et al., 2016 Similarity based prioritization evaluation is performed by Al-Hajjaji et al.[31] proposed previously in their paper [29]. They performed two experimentations one was to compare the effectiveness and interaction coverage of similarity based approach with the random and interaction based approach with real faults. The second was performed with artificially injected faults in the product line to compare the effectiveness of default order of sampling algorithms and prioritized order of proposed approach. When compared with real product lines; they conclude that their approach significantly improved the effectiveness results with 92%, 95% and 97.5% APFD values but approximately identical in term

if increasing interaction coverage. When compared with default order of sampling algorithms achieved 71%,81.1% and 81.8% APFD values.

Al-Hajjaji et al., 2017 Another feature coverage criterion for prioritization is used in this approach that is based on the sampling process [32]. They used delta to find the difference among the products whereas Sánchez et al., Wang et al., Henard et al. and Devroey et al. uses different method for feature coverage criterion. They used similarity measure for finding the difference between the products of SPL. Al-Hajjaji et al. proposed two approaches, first; prioritizing product using delta modeling to increase the rate of fault detection second; delta-oriented prioritization combined with the configuration in similarity-based product prioritization using weight factor. In the first approach, the author focused on solution space artifacts rather on the problem space i.e. selection of features. Architecture model was used as an approach based on solution space. Products prioritized by using similarity approach whereas similarity in proposed techniques calculated by the deltas. Deltas were calculated based on dissimilarity using hamming distance. On evaluation, proposed approach outperformed the random order, default order of sampling algorithm, and achieved 90.7% APFD value. APFD of combined approach was 90.5%.

Al-Hajjaji et al., 2017 In this approach, clustering technique is used. Al-Hajjaji et al. used feature coverage criterion for prioritization with clustering [33]. Contrary to all the previous approaches, they proposed feature coverage criterion but with clusters. The products that have similar features group into clusters to reduce the testing effort and improves the effectiveness. Products were a cluster (k-means algorithm) into subsets such that each set shares common set of properties i.e. similar products that share a common set of features were cluster in one set. After that, products in clusters are prioritized using similarity-based approach. Proposed approach work slightly better than random order but when compared with heuristic similarity based prioritization Al-Hajjaji et al. which

gives slightly worse results [33]. AFPD value of cluster-based prioritization depends on the number of clusters. Increasing in a number of clusters decreases the AFPD value. Average results showed that proposed approach did not perform better than current approaches. However, it worked better than random orders.

2.2 Prioritization Criterion based on Feature

In the following section two types of feature is considered. In one criterion, importance or the priority of individual feature is taken into account based on some goal. In other criterion, frequency of the feature in all the products is considered to prioritize the test cases. In both criteria, feature is considered for the prioritization.

Ensan et al., 2011 Feature priority criterion used by Ensan et al. to prioritize the test suite [34]. They introduced an approach in which most desirable features were extracted to reduce the size of feature model. The goal of their study was, to accomplish the higher error coverage by testing less number of test cases. Criterion is based on the assumption that the feature, which is important with the stakeholder's perspective, has a higher chance to appear in more products. To achieve a goal, size of the feature model was reduced by selecting important feature from the model that was based on the goals, described by the stakeholders. Features that cover important goals were selected and remaining features were removed from the model. For prioritizing the test cases, goals of stakeholders ranked according to the weight of goal (stakeholder's point of view). The highest rank goal was the most important goal to meet. Features that is used to meet the highest ranked goal also ranked high. After that, test cases were ranked as; a test case that covered the collection of important features was ranked high. However, the proposed criterion only cover the feature complexity.

Sánchez et al., 2014 Another feature criterion based on the frequency of the feature, proposed by the Sánchez et al. depends on the reusability of products names as Commonality (Comm) [20]. Frequency of features are calculated from the test suite and assigns highest priority to those test cases that contains the highest priority features (as they use repeatedly in multiple products) because they cover the more portion of products. It represents the percentage of the feature that is present in the test suite and give a chance to the test case that contain highly reused feature to test first because it cover the more part of the product. In the first experiment 74.9% of APFD is achieved and in the second experiment 55% is achieved.

2.3 Prioritization Criteria based on Feature Coupling

As discussed before, Sánchez et al. proposed five prioritization criteria based on the feature model metrics [20]. In this section, three of them are discussed, based on feature coupling criteria. Their approach is different with all the previous approaches, none of them consider the coupling complexity of the feature. The goal of their work was to increase the early fault detection rate. Hence, three criteria measure the complexity of the products named as Cross Tree Constraint Ratio (CTCR), Coefficient of Connectivity Density (CoC) and Variability Coverage and Cyclomatic Complexity (VC&CC). As it was assumed that products that are more complex are likely to be more error-prone, therefore, they were given higher priority to test first. They have performed two experiments, first is with the random order of test suite and second, with the 2-wise interaction coverage.

Sánchez et al., 2014 CTCR criterion calculates the ratio of features with constraints. CTCR of each test case is calculated by dividing the number of features that involves with constraints to the total number of features involve in the test case. Test case with highest CTCR is the most complex and assigns the highest

order in the list. Upon two experiments, first experiment of CTCR obtains 84.5% and second experiment achieves 83.2% of APFD.

CoC criterion represents the connectivity of features in the test case, it calculates the edges and constraints of the test case. Total number of edges in the test case is considered that is the relation between parent and child, and divided with the total number of features of the test case. Test case with highest CoC is the most complex test case. Evaluation results achieve 90.6% and 88% APFD on first and second experiments respectively.

VC&CC is the combination of two feature model metrics, VC and CC. VC provides the information about the variation points of the test case that represent different variants to create the new product. VC is the optional features and all non-leaf features whereas CC is the number of cycles that create in the test case. As features are represented by the tree-like structure that is feature model so cycles are calculated as the number of constraints in the test cases. The highest value of VC&CC represents the most complex test case and effective in revealing faults therefore assigned highest priority in the list. SPLAR tool was used to analysis of feature model. 96.5% and 90.7% APFD is achieved with first and second experiments respectively. The best APFD average value was achieved by the VC&CC metric followed by CoC.

2.4 Analysis and Comparison

All of the techniques defined above prioritized the test cases based on the particular criteria by using different measuring instruments. List of test cases are prioritized using the proposed criterion and for evaluating their criteria, they have used different measures. Some of them used fault analysis and few of them used coverage analysis, as mentioned in table 2.1.

From the table 2.1, we observe that highest APFD, that is 96.5% is achieved by the feature coupling criterion named as VC&CC proposed by Sánchez et al. [20], in

which they used metrics of the feature model as a measuring instrument. Second highest is achieved by the feature coverage criterion proposed by Al-Hajjaji et al. [33] that is 90.7%. Al-Hajjaji et al. criterion was based on the coverage of the features in which measuring instrument was the similarity metrics among the features of test cases.

TABLE 2.1: Comparison of Techniques Based on Prioritization Criteria

Author	Prioritization Criteria	Measuring Instrument	Feature Complexity	Feature Coupling Complexity	Feature Coverage	APFD
Ensan et al.[34]	Feature priority	Features Weight	No	No	No	-
Devroey et al.[27]	Feature coverage	Probability Measure	No	No	Yes	-
Henard et al.[28]	Feature t-wise coverage	Similarity Measure	No	No	Yes	-
Al-Hajjaji et al.[29]	Feature coverage	Similarity Measure	No	No	Yes	81%
Wang et al.[5]	Feature pairwise coverage	GA, RS,(1+1)EA and AVM	No	No	Yes	-
Sanchez et al.[20]	Feature coupling	FM Metrics	No	Yes	No	90.6%
	Feature coupling	FM Metrics	No	Yes	No	84.5%
	Feature coupling	FM Metrics	No	Yes	No	96.5%
	Feature frequency	FM Metrics	No	No	No	74.9%
	Feature coverage	Similarity Measure	No	No	Yes	87.4%
Al-Hajjaji et al.[32]	Feature coverage	Similarity Measure	No	No	Yes	90.7%
Al-Hajjaji et al.[35]	Feature coverage	Similarity Measure	No	No	Yes	68%

Whereas, other techniques that used feature coverage as a criterion does not achieve high APFD than Al-Hajjaji et al.[32]. Although, all the techniques used different case studies and they cannot be compared with each other. Subsequently, we can compare those techniques that used the same case studies. Sánchez et al.[20] proposed five criterion from which one is based on the feature coverage criterion, one is based on feature complexity and others are based on feature coupling criteria.

We can compare the criteria proposed by Sánchez et al.[20] as they are based on the same case studies. 15 test suites were used with 100 to 500 products. Rate of fault detection of test suites based on the five criteria are calculated. From their criteria, Feature coverage criterion achieved the lowest value of APFD that is 74.9% (average). It portrays that the criterion is not good enough as the random order of the test suite obtained the better rate of fault detection than this criterion that is 77.4% on the same case studies. So we can say that the criterion based on only the individual feature is not good enough in terms of fault detection. Feature coupling complexity (CTCR) criterion achieved 84.5% of APFD, feature coverage criterion achieved 87.4%, other feature coupling complexity criteria (CoC) and (VC&CC) achieved 90.6% and 96.5% respectively. CoC and VC&CC based on the complexity of the features and achieved the highest fault detection rate, whereas third complexity based criterion did not get the better results than the coverage criterion. Coverage criterion detected all the faults by using 45% of the test suite but coverage criterion does not consider the complexity of the test case and it did not achieve the APFD better than the feature coupling criterion. So the best value is achieved by the VC&CC and detected all the faults by using only 15% of the test suite that is, only 75 out of 500 test cases are used to detect all the faults. Therefore, we can conclude that feature coupling complexity is the best criterion among them all.

2.4.1 Gap Analysis

All the proposed techniques discussed in literature review have developed to increase the rate of fault detection by prioritizing the test cases. Many prioritization techniques based on the coverage criterion in which products are prioritized based on the similarity or the dissimilarity of their features. If two products are less similar with respect to their features, they consider for prioritization order. However, these techniques do not consider the complexity of product itself and from the comparison of APFD it is concluded that coverage based criterion did not achieve the better APFD. Few techniques consider optimization and search based algorithm for prioritization which is randomized and does not guarantee that next time generated prioritization list will follow the same order and same time. Remaining techniques as shown in table 2.1 did not perform evaluation as their APFD values are not given. so, we cannot say how these techniques perform in terms of fault detection.

From the given table we are able to perceive that APFD of feature coupling criterion proposed by Sánchez et al.[20] is better than other techniques in terms of fault detection rate, among the criterion which consider the complexity of features names as VC&CC is better than the other feature coupling criteria. while, they partially consider the product's complexity. They consider the product's complexity measure in terms of features and achieved 96.5% fault detection rate but they did not focus on product's complexity completely.

Chapter 3

Proposed Approach

In the previous chapter, literature survey is conducted from which we came to know that many techniques used feature coverage criterion for prioritization but when compared with other prioritization criteria, the highest APFD value is achieved by the metric that use the feature coupling complexity. Therefore, we conclude that feature coupling complexity is an important factor for measuring the complexity of the products in order to find a prioritized set of products. Nevertheless, only feature coupling complexity is considered separately and feature's individual complexity is being ignored. Two techniques considered the individual feature as a factor but according to our knowledge, only one technique considered the feature priority for prioritizing the test cases. They disregard the feature coupling complexity. Feature complexity is also an important factor that can be considered to determine the product's complexity. To overcome the gaps in the existing techniques, we have proposed new criterion that considers both types of complexities, discussed in this chapter.

This chapter presents our proposed approach of prioritization criterion for the software product lines. The proposed approach is demonstrated with an example for better understanding. In the first part of this chapter, the algorithm of proposed approach and the context diagram is presented. The second part holds the proposed criterion and the last part contains the example of proposed criterion.

3.1 Proposed Prioritization Algorithm

In proposed prioritization criterion, test cases will be prioritized based on the complexities that will be obtained from the two factors.

1. Feature complexity
2. Feature coupling complexity

For feature complexity, use case descriptions will be used. From the use case descriptions, complexity of all the features that are part of FM will be computed with the help of the use case metrics. Once the complexity of features are computed, we will find the feature complexity of a test case. As, a test case may contain multiple features, so for each feature that is the part of test case, feature's complexity will be combined. Similarly, feature complexity of test suite will be calculated.

For the second factor, feature coupling complexity, we will use the metric from the existing technique proposed in [20]. They measure the complexity of test case that is based on the feature model metrics. For each test case complexity, variation points and cyclomatic complexity will be calculated from the formula 3.5. Variation point is the feature that provide different variation in the FM to create product. All the non leaf features and the optional features in the product are the variation points. Whereas, cyclomatic complexity is the number of cycles in the product, which can be computed by counting the number of constraints in the product.

After measuring the feature complexity and feature coupling complexity, both complexities will be combined in order to get the Test Case Complexity (TCC). For combining these two factors, we have introduced α as a weighting factor. This factor adjust the weights of both complexities, FC and FCC. We will perform some experiments in order to know which value of α factor give more importance than the other. For this, we use the value of α from 0-1 and get different number of the test suite. Each test suite than prioritized according to the complexities. After that, APFD will be calculated for each test suite. The test suite with the highest

APFD value provides information about the importance of α values which helps us to find how much weight to assign the FC and FCC.

The context diagram of proposed approach is given in the figure 3.1 and the algorithm of proposed approach is given in Algorithm 1, 2 and 3. Algorithm 1 is for calculating the feature complexity that is the first part of proposed approach, algorithm 2 is for calculating the feature coupling complexity and algorithm 3 is for combining both factors to measure the test case complexity.

The running time for calculating the feature complexity of proposed criterion is constant as the time complexity is based on size of the input. Size of input in our algorithm is the number of statements that is, the number of actor actions and the number of system actions. These statements are constant and independent to the size of input. Therefore, the time complexity is constant, that is, $\mathcal{O}(1)$ as it requires same amount of time to execute the statement regardless with the input size.

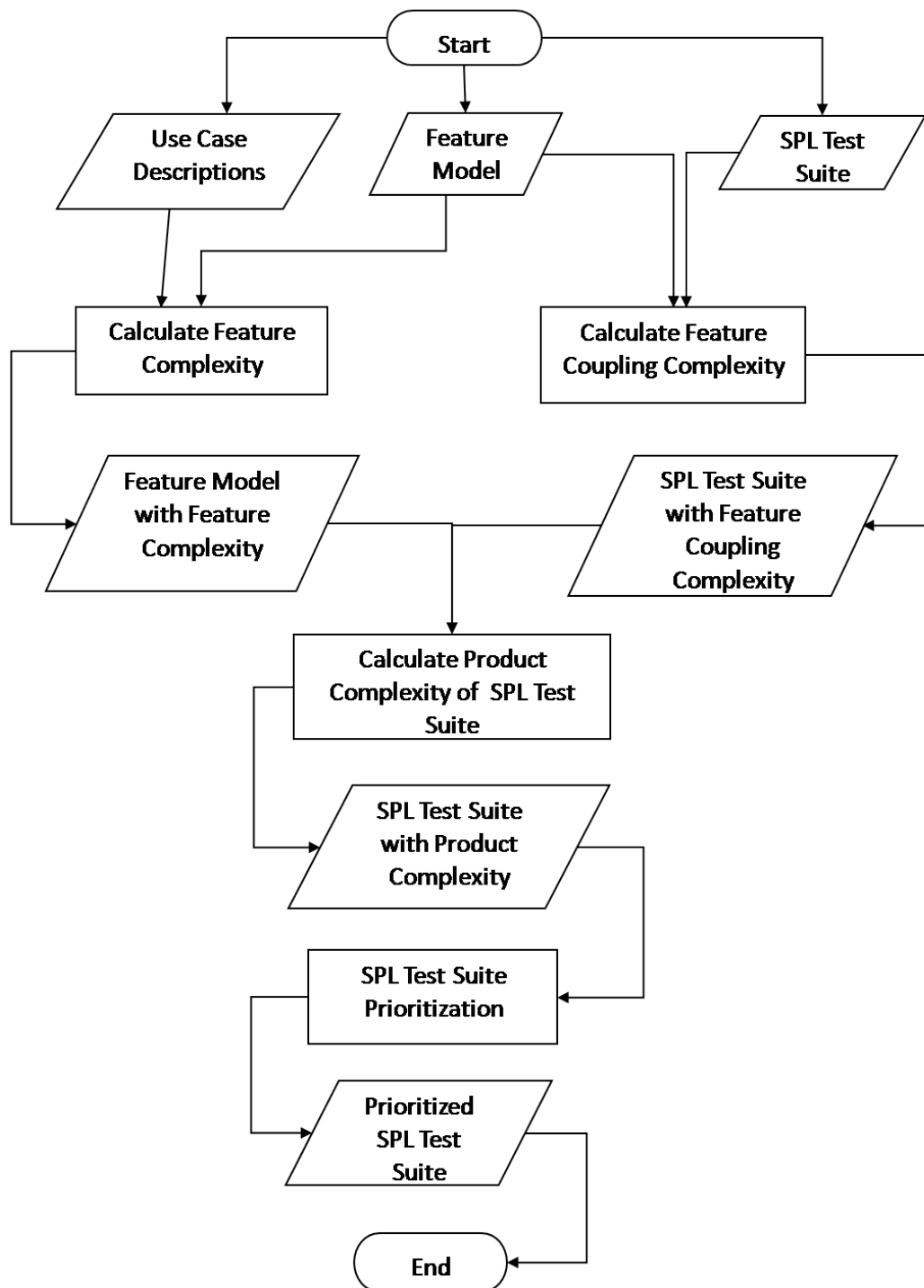


FIGURE 3.1: An Illustration of Proposed Solution.

Algorithm 1 Calculating Feature Complexity

Require:

FM: { $f_i, i = 1, \dots, n$ }	▷ Feature Model
T: { $t_j, j = 1, \dots, m$ }	▷ Test Suite
t_j : { $f_k, k = 1, \dots, o$ }	▷ Test case
UC: { $UC_l, l = 1, \dots, p$ }	▷ Use Case

Ensure:

FC_T	▷ Feature complexity of test suite
--------	------------------------------------

Declaration

$FC_f = 0$	▷ FC of features in FM
$FC_{t_j} = 0$	▷ Feature complexity of t
UC_l : { $UC_m, m = 1, \dots, q$ }	▷ UC lines
$NOAS = 0$	▷ Number of actor actions steps in UC
$NOUS = 0$	▷ Number of use case action steps in UC
$NOSS = 0$	▷ Number of system action steps in UC
$NOCS = 0$	▷ Number of conditional steps in UC
$NOE = 0$	▷ Number of exceptions in UC
$NOS = 0$	▷ Number of steps in UC
$CC = 0$	▷ Cyclomatic complexity of UC
Max = 0	▷ Maximum value of FC

1: **procedure** FC(T, FM, UC)

2: **for all** f_i in FM **do** ▷ FC of FM features

3: **for all** UC_l in UC **do**

4: **if** $UC_{m1} == f_i$ **then**

5: break

6: **for all** UC_l in UC_m **do**

7: **if** $UC_l =$ actor action **then**

8: $NOAS_{f_i} = +1$

9: **else if** $UC_l =$ system action **then**

10: $NOSS_{f_i} = +1$

```

11:         else if  $UC_l = \text{use case action}$  then
12:              $NOUS_{f_i} = +1$ 
13:         else if  $UC_l = \text{if condition}$  then
14:              $NOCS_{f_i} = +1$ 
15:         else if  $UC_l = \text{exception}$  then
16:             for all  $UC_l$  in exception do
17:                  $NOE_{f_i} = +1$ 
18:             end for
19:         end if
20:     end for
21:      $NOS_{f_i} = NOAS_{f_i} + NOUS_{f_i} + NOSS_{f_i}$ 
22:      $CC_{f_i} = NOE_{f_i} + NOC_{f_i} + 1$ 
23:      $FC_{f_i} = NOS_{f_i} + CC_{f_i}$ 
24: else
25:      $FC_{f_i} = 0$ 
26: end if
27: end for
28: end for
29: Max = maximum of  $FC_{f_i}$ 
30: for all  $t_j$  in T do ▷ FC of test suite
31:     for all  $f_k$  in  $t_j$  do ▷ FC of test case
32:          $FC_{t_j} = +FC_{f_k}$ 
33:     end for
34:      $FC_{t_j} = \frac{FC_{t_j}}{Max}$ 
35:      $FC_T[t_j] = FC_{t_j}$ 
36: end for
37: return  $FC_T$ 
38: end procedure

```

Algorithm 2 Calculating Feature Coupling Complexity

Require:

- FM: $\{ f_i, i = 1, \dots, n, C_h, h = 1, \dots, l \}$ ▷ Feature Model
T: $\{ t_j, j = 1, \dots, m \}$ ▷ Test Suite
 t_j : $\{ f_k, k = 1, \dots, o \}$ ▷ Test case

Ensure:

- FCC_T ▷ Feature coupling complexity of T

Declaration

- $VC' = 0$ ▷ variability coverage of FM
 $CC' = 0$ ▷ cyclomatic complexity of FM
 $VC_{t_j} = 0$ ▷ variability coverage of t
 $CC_{t_j} = 0$ ▷ cyclomatic complexity of t
 $FCC_{t_j} = 0$ ▷ FCC of t
 $Max = 0$ ▷ Maximum FCC of FM

1: **procedure** FCC(T, FM)

2: **for all** f_i in FM **do** ▷ VC and CC of FM

3: **if** f_i is optional OR variation point **then**

4: $VC' = +1$

5: **end if**

6: **end for**

7: **for all** C_h in FM **do**

8: $CC' = +1$

9: **end for**

10: $Max = \sqrt{(VC')^2 + (CC')^2}$

11: **for all** t_j in T **do** ▷ VC and CC of test case

12: **for all** f_k in t_j **do**

13: **if** f_k is optional OR variation point **then**

14: $VC_{t_j} = +1$

15: **end if**

```

16:         if  $f_k$  has  $C_h$  then
17:             if  $C_h ==$  Require then
18:                 if  $f_k$  AND  $f_{k+1}$  is in  $t_j$  then
19:                      $CC_{t_j} = +1$ 
20:                 end if
21:             else if  $C_h ==$  Exclude then
22:                 if  $f_{k+1}$  is not in  $t_j$  then
23:                      $CC_{t_j} = +1$ 
24:                 end if
25:             end if
26:         end if
27:     end for
28: end for
29: for all  $t_j$  in  $T$  do ▷ FCC of test suite
30:      $FCC_{t_j} = \frac{\sqrt{(VC_{t_j})^2 + (CC_{t_j})^2}}{Max}$ 
31:      $FCC_T[t_j] = FCC_{t_j}$ 
32: end for
33: return  $FCC_T$ 
34: end procedure

```

Algorithm 3 Combining Feature Complexity and Feature Coupling Complexity

Require:

T: $\{ t_i, i = 1, \dots, n \}$ ▷ Test Suite

FC: $\{ FC_{t_i}, t_i = 1, \dots, n \}$ ▷ Feature complexity of T

FCC: $\{ FCC_{t_i}, t_i = 1, \dots, n \}$ ▷ Feature coupling complexity of T

Ensure:

PrT: ▷ Prioritized test suite

Declaration

$TcC = \phi$ ▷ Test case complexity

$\alpha = 0$ ▷ weighting factor

```

1: procedure TCC( $T, FC, FCC$ )
2:   while  $\alpha \leq 10$  do
3:     for all  $t_i$  in T do
4:        $TCC[t_i] = \frac{\alpha}{10} * FC_{t_i} + (1 - \frac{\alpha}{10}) * FCC_{t_i}$ 
5:     end for
6:      $TcC[t_i] = sortTcC[t_i]$ 
7:      $prT[\alpha] = TcC[t_i]$ 
8:      $\alpha ++$ 
9:   end while
10:  return PrT
11: end procedure

```

3.2 Proposed Criterion for Product Prioritization

The steps involved in criterion to achieve the prioritized set of test suite are given below:

1. Calculate feature complexity of test suite

2. Calculate feature coupling complexity of test suite
3. Combine feature and feature coupling complexities

The first step of proposed criterion is to measure the feature complexity by using use case metrics which is then combined with the second step of proposed approach, that is, feature coupling complexity. In the third and the last step, both complexities are combined together with the help of α factor that adjust the weights of both factors to compute the product or the test case complexity. Test cases are prioritized based on the complexities. Test case with the highest complexity assigns the top priority in ordering list. The proposed approach is illustrated in the figure 3.1 and the detailed steps of proposed criterion are discussed below:

3.2.1 Feature Complexity

As Product Lines are precise in terms of features. Products in product lines are generated by the combination of features. In product lines, features are used to model the variability whereas, outside the product line community, use cases diagram is also extensively used to model the variability. The product line community also discusses similarities between features and use cases [36]. Use cases are used to mapped into features of feature model [37–39], which means features in the feature model are derived from the use case diagrams. Consequently, we use the use case description metrics for measuring the complexity of features.

Use Case Model Use case is first introduced by Ivar Jacobson in 1990s which have become an essential part of functional requirements modeling [40]. It turns out to be the source for the initial design of a system and for requirements documentation. Use case model consists of two parts, the use case diagram, and the use case descriptions. Diagram presents an overview of actors and use cases in which list of actions or tasks describe the interaction between actor and system to accomplish a goal. Whereas the use case descriptions explain the details of requirements

in steps using the natural language which helps to understand the requirement document easily, even to non technical person [41]. There is no standard template for use case descriptions, as Ivar Jacobson did not introduce standard template with use case diagram. There exist several ways to write the descriptions of the use case in natural language, commonly used templates are given below:

1. Cockburn template, initially proposed in 1997 which is later reviewed in 2001 [42] . Most significant elements are given below:
 - (a) Name, scope, level and visibility
 - (b) Preconditions
 - (c) Trigger
 - (d) Minimal and success guarantees
 - (e) Main success scenario
 - (f) Extensions
2. Rational Unified Process (RUP) template proposed by Kruchten [43]. Most significant elements of RUP template are given below:
 - (a) Name
 - (b) Pre and post conditions
 - (c) Basic flow and Alternative flow
 - (d) Extension points
3. Durán template [44]. Most significant elements of Durán et al. template are given below:
 - (a) Name
 - (b) Description
 - (c) Pre and post conditions
 - (d) Ordinary Sequence
 - (e) Exceptions

Use cases are primarily used as textual specifications of functional requirements. Consequently, they can be considered as partial requirement of the system to be built for estimation purposes [45]. Use case metrics make possible an early estimation of the cost, development effort, implementation time and complexity of the system under development. Various software metrics have been proposed for analysis purpose but cannot be used in earlier stages of the development process. However, use case metrics is one of the metrics that can be used in the early stages of development process for analysis purpose [46].

Use Case Metrics Use case metrics defect proneness indicators. Durán et al. described some heuristics for use case metrics [47]. According to these heuristics, some use case metrics are the indicators of defect proneness. These metrics are also empirically evaluated by Genro et al. in [48]. We are using some metrics that used by these authors for measuring the complexity of feature defined below:

NOAS: A number of actor action steps of the use case.

NOUS: A number of use case action steps of the use case.(inclusion or extension).

NOSS: A number of system action steps of the use case.

NOCS: A number of conditional steps of the use case.

NOE: A number of exceptions of the use case.

NOS: A number of steps of the use case. $NOS = (NOAS + NOUS + NOSS)$

CC: Cyclomatic complexity of the use case. $CC = (NOCS + NOE + 1)$

Durán's template is used in our approach and the above metrics are used to measure the complexity of the use case that will be the feature complexity.

Use case descriptions are considered for all the features exist in the FM. Two types of features exist in FM, abstract and concrete. Concrete features can be derived from the functional requirement and non-functional requirement. There is

no use case description for abstract features, only concrete feature with functional requirement contains use case description. Abstract features are features used to construct the FM, though; these features do not have any influence on the implementation level, as selecting or removing these features from the products makes no difference [49]. Use cases are developed only for functional requirements so features with non-functional requirements and abstract are being ignored.

Feature complexity is calculated for all the concrete features and 0 value is assigned to all non-functional and abstract features. Listed below steps are involved in measuring the feature complexity.

1. Feature complexity of all features of FM
2. Feature complexity of a test case
3. Normalize the feature complexity of test case

Feature complexity of all the features present in FM will be computed with the equation 3.1.

$$FC_i(uc, FM) = \sum (NOS_i, CC_i) \quad (3.1)$$

FC is the *Feature Complexity* of feature i present in the FM which takes *use case description* uc and *feature model* FM as an input and compute the *Number of steps* NOS and *Cyclomatic Complexity* CC of feature i and returns the complexity of feature i . where,

$$NOS = NOAS + NOUS + NOSS$$

and

$$CC = NOCS + NOE + 1$$

As the product is the combination of features, thus for all features present in the product, FC is combined to calculate the FC of the product. Once complexity of all the features of FM is computed, FC of a test case is computed by simple

addition of feature complexities that are present in the test case. FC of each test case is computed with the equation 3.2 where n is the number of features present in the test case and t is the test case.

$$FC_t(FC, t) = \sum_{i=1}^n FC_i \quad (3.2)$$

Equation 3.2 computes the *feature complexity* FC of *test case* t , which takes *Feature complexities* FC of all the features of FM and *Test case* t as an input and compute the summation of all the complexities of the features present in a test case from i to n where n is the total number of features in a *test case* t .

After having FC of the test suite, third step is performed for normalization, so that in later step it can be combined with feature coupling complexity. For normalizing each test case FC is divided by the FC of FM. It will be the maximum value that any product can achieve. For this, sum of all the feature complexity of FM is taken as given in the equation 3.3, here m is the total number of features of FM.

$$FC_{FM}(FM, FC) = \sum_{i=1}^m FC_i \quad (3.3)$$

Equation 3.3 computes the *Feature Complexity* FC of *Feature Model* FM , which takes FM and FC of all the features as an input and returns the summation of all feature complexities from i to m where m is the *total number of features present in* FM .

Normalization is perform by dividing each test case value with the FC_{FM} as given in the following equation 3.4.

$$FC_{t(norm)} = \frac{FC_t}{FC_{FM}} \quad (3.4)$$

Equation 3.4 normalized the *Feature Complexity* FC of *test case* t , where FC of test case t is divided with the FC of *Feature Model* FM .

3.2.2 Feature Coupling Complexity

Feature coupling complexity FCC is calculated as defined in existing approach with the help of equation 3.5, based on FM metrics [20]. They proposed five prioritization criteria; three of them are based on the complexity of test cases. As more complex test case expected to be more error-prone, so assign the highest priority in testing order. We have selected one of the metrics which is based on the coupling complexity of features. Motivation for selecting this metric is the promising results, as it achieve the highest fault detection rate. Metric is given below:

$$VC\&CC_t(FM, t) = \sqrt{VC^2 + CC^2} \quad (3.5)$$

Equation 3.5 is used to calculate the *Variability Coverage VC* and *Cyclomatic Complexity CC* of *test case t*, which takes *test case t* and *Feature Model FM* as an input and compute the coupling complexity. Listed below steps are involved in measuring feature coupling complexity:

1. Measuring variability coverage VC
2. Measuring cyclomatic complexity CC
3. Feature coupling complexity of a test case
4. Normalize the FCC value

Variability coverage of the test case is defined as the number of variation points. Variation point of a test case is calculated from the FM, it is the feature that provides an alternative for creating the product. So the VC of a test case is the number of features that provide variants in FM and all optional features.

Cyclomatic complexity of test case is defined as the number of cycles generating in the FM. However, FM is the tree-like structure hence cycles can be calculated by the number of constraints that make the cycles in FM. So CC of the test case is the number of constraints present in the test case. Both are combined with the help of equation 3.5.

In the last step, each test case is normalized. Maximum value is calculated from the FM. Similar to the FCC of a test case, FCC of FM is calculated that gives the number of cycles and variation points which could be the maximum number of variation and cycles present in the FM. Each FCC value of a test case is then normalized with the following equation 3.6

$$FCC_{t(norm)} = \frac{VC\&CC_t}{VC\&CC_{FM}} \quad (3.6)$$

Equation 3.6 normalized the *Feature Coupling Complexity FCC* of test case t , where FCC of test case t is divided with the FCC of *Feature Model FM*.

3.2.3 Combining Feature Complexity and Feature Coupling Complexity

Final complexity of test case (TCC) is obtained by combining the FC and FCC together. Both complexities combined with the α factor that adjust the weights for each factor. An experiment is performed by assigning different weights to each factor and calculate the test suite complexity. APFD is obtained for all the test suite, from the highest APFD of test suite, we will be able to know the importance of factors. Weights of α are defined in the listed table 3.1. We used given equation 3.7 for applying weights to each factor. TCC is the test case complexity, each test case complexity is derived by multiplying the weight to its FC value and FCC value. Similarly, all test cases complexities are calculated when α is 0, 0.1, 0.2 upto 1.

$$TCC_t = \alpha * FC_t + (1 - \alpha) * FCC_t \quad (3.7)$$

When assigning 0 to α , means we assign zero importance to FC value and give all the importance to FCC of the test case. Likewise, when we assign 1 to α means giving all the importance to FC and no importance to FCC value. 0.5 value of α means giving equal importance to each factor. By assigning different values for α ,

TABLE 3.1: Weights(α) for Test Suite

Iteration	FC (α)	FCC ($1 - \alpha$)
1	0	1
2	0.1	0.9
3	0.2	0.8
4	0.3	0.7
5	0.4	0.6
6	0.5	0.5
7	0.6	0.4
8	0.7	0.3
9	0.8	0.2
10	0.9	0.1
11	1	0

we will be able to know the importance of each factor when APFD will be calculated. Overall 11 sets of test suite complexities are generated. In order to get the relative importance of each factor α is introduced. α is use to balance the weights of both factors such that when one factor is increases then second factor decreases. Our focus is on the relative weights of both factors instead of absolute values of factors that is the reason we use α for one factor and $(1 - \alpha)$ for second factor On each set APFD is calculated in order to know which weights give the better results. Experimental results will be discussed in detail in chapter 5.

3.3 Example of Proposed Approach

E-shop FM is selected as an example to explain the detail of proposed criterion given in [20]. FM is given in figure 3.2.

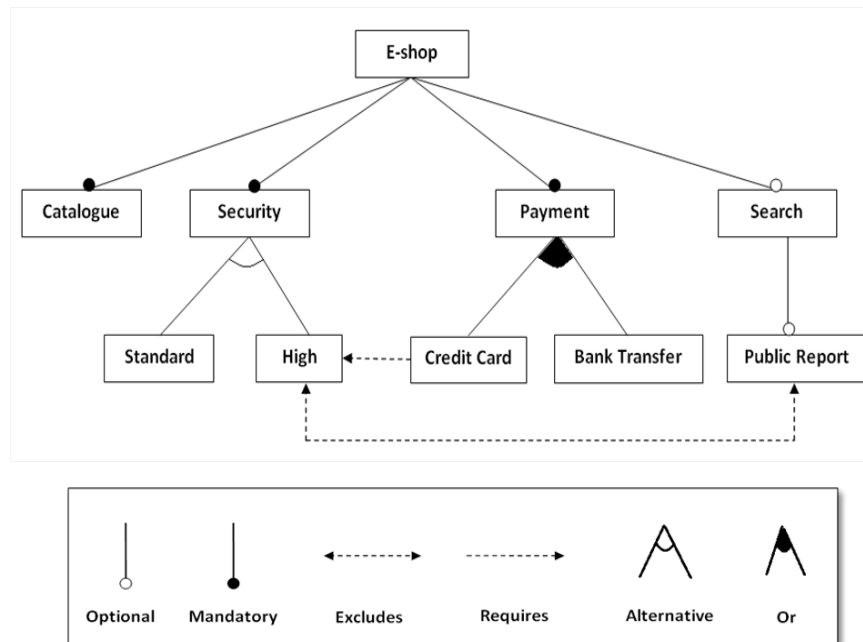


FIGURE 3.2: Feature-Model-Example.

In the example figure 3.2, FM consist of ten features from which three are abstract feature named as, 'E-shop', 'Security' and 'Payment', whereas, 'High' and 'Standard' are the non-functional requirements. These features do not contain use case descriptions. Remaining features have use case descriptions, so the first step is to calculate the FC of all the features presented in the FM with the help of use case metrics. For the simplicity, we only consider the Basic flow and Exception flow of use case. Use case description of the feature 'Catalogue' is shown in the figure 3.3.

Test Suite is generated from FM with the combination of all valid features listed below in table 3.2:

TABLE 3.2: Test Suite

Test Case	Test Cases
t1	E-Shop,Catalogue,Payment,Bank Transfer,Security,High
t2	E-Shop,Catalogue,Payment,Bank Transfer,Security,Standard
t3	E-Shop,Catalogue,Payment,Credit Card,Security,High
t4	E-Shop,Catalogue,Payment,Bank Transfer,Credit Card,Security,High
t5	E-Shop,Catalogue,Payment,Bank Transfer,Security,High,Search
t6	E-Shop,Catalogue,Payment,Bank Transfer,Security,Search,Standard
t7	E-Shop,Catalogue,Payment,Security,Standard,Public report,Search,Bank Transfer
t8	E-Shop,Catalogue,Payment,Credit Card,Security,High,Search
t9	E-Shop,Catalogue,Payment,Bank Transfer,Credit Card,High,Security,Search

Use Case Name	Catalogue	
	Step	Action
Basic Flow	1	Customer selects an item.
	2	System displays details of the item.
	3	Customer selects add to cart option for the item.
	4	System adds an item to cart.
	5	Go to step 1 if customer continue shopping.
	6	Customer selects go to cart option.
	7	System displays cart page.
Exception Flow	4a	System displays message "item is out of stock".

FIGURE 3.3: Use Case Description.

FC is computed with the equation 3.1. Actor action are the steps performed by the actor in the use case, in the 'Catalogue' use case, actor is the customer, so actor steps are 4 associated to step 1, 3, 5 and 6 of the basic flow. System action are the

steps performed by the system, here system actor action steps are 3, associated to step 2, 4, and 6 of the basic flow. This use case does not include or extends any other use case so use case step is 0. All the above actions are considered from basic flow, because these steps are designed to achieve the goal. *NOS* is the sum of all the action steps defined above that will be 7. Conditional step in the above use case is 1 associated to step 5 in the basic flow, only one exception associated to exception flow of step 4, so the *CC* will be 3. table 3.3 shows the outlined metrics of the use case example.

TABLE 3.3: Use Case Metric Values for the Use Case of Figure 3.3

Metric	Value	Explanation
NOAS	4	There are 4 actor (customer) steps associated to Basic Flow (step 1, 3, 5 and 6)
NOSS	3	There are 3 system steps (step 2, 4, 6)
NOUS	0	There is no include or extend
NOCS	1	There is 1 conditional step associated Basic Flow (step 5)
NOE	1	There is only 1 exception associated to step 4
NOS	7	$NOAS + NOSS + NOUS = 4 + 3 + 0$
CC	3	$NOCS + NOE + 1 = 1 + 1 + 1$

From the equation 3.1, we get the *FC* value:

$$FC_i(uc, FM) = \sum(NOS_i, CC_i)$$

$$FC_{catalogue} = 7 + 3 = 10$$

Similarly, all the feature's complexity is calculated from the use case descriptions which is listed in the table 3.4 below:

TABLE 3.4: Complexity of Features

Feature	Complexity
Catalouge	10
Credit Card	13
Bank Transfer	13
Search	5
Public Report	7

Next step is to calculate the FC of test suite that contains nine test cases in this example, given in the table 3.2. FC of each test case is computed from the equation 3.2.

$$FC_t(FC, t) = \sum_{i=1}^n FC_i$$

For t1 FC is computed as:

$$FC_{t1} = FC_{E-Shop} + FC_{Catalogue} + FC_{Payment} + FC_{BankTransfer} + FC_{Security} + FC_{High}$$

As we have already discussed, for the non-functional and abstract type features, we assign 0. So, for FC_{t1} we get:

$$FC_{t1} = 0 + 10 + 0 + 13 + 0 + 0 = 23$$

Similarly, for all test cases from the test suite FC is computed. table 3.5 shows the values for all the test cases.

Next step is to normalize the FC of test case by dividing by FC_{FM} .

$$FC_{FM} = FC_{E-Shop} + FC_{Catalogue} + FC_{Security} + FC_{Standard} + FC_{High} + FC_{Payment} + FC_{CreditCard} + FC_{BankTransfer} + FC_{Search} + FC_{PublicReport}$$

$$FC_{FM} = 0 + 10 + 0 + 0 + 0 + 0 + 13 + 13 + 5 + 7 = 48$$

In the above scenario, the maximum value is 48 so all the test case FC is divided by 48 to normalized the values as listed below in table 3.6:

TABLE 3.5: FC of Test Suite

Test Case No.	FC of Test Case
t1	23
t2	23
t3	23
t4	36
t5	28
t6	28
t7	35
t8	28
t9	41

TABLE 3.6: Normalized FC of Test Suite

Test Case No.	FC of Test Case
t1	0.479167
t2	0.479167
t3	0.479167
t4	0.75
t5	0.5834
t6	0.5834
t7	0.729167
t8	0.5834
t9	0.854167

The second step is to calculate the feature coupling complexity. The first test case $t1\{E-Shop, Catalogue, Payment, Bank Transfer, Security, High\}$ contains 'E-Shop', 'Payment' and 'Security' as the variant features as these are the variation points. VC of the product is the number of variation points and optional feature, hence, VC of t1 is 3. Whereas, CC is the number of constraints in the product, so in t1 only 1 constraint is present, which is in between 'High' and 'Public Report' feature.

The constraint between these two feature is exclude constraint that means both features cannot be appear in the same product and in P1 this condition is true, hence, CC of t1 is 2. In the same way VC and CC of all the test cases is computed. Following table 3.7 demonstrate the values of Variability Coverage VC , Cyclomatic Complexity CC and Feature coupling complexity FCC value of the test suite.

TABLE 3.7: VC and CC of Test Suite

Test Case	VC	CC
t1	1 + 0 + 1 + 0 + 1 + 0	1 (High excludes Public report)
t2	1 + 0 + 1 + 0 + 1 + 0	0
t3	1 + 0 + 1 + 0 + 1 + 0	1(Credit Card requires High) + 1 (High excludes Public report)
t4	1+0+1+0+0+1+0	1(Credit Card requires High) + 1 (High excludes Public report)
t5	1+0+1+0+1+0+1	1 (High excludes Public report)
t6	1+0+1+0+1+0+1	0
t7	1 + 0 + 1 + 0 + 1 + 0 + 1 + 1	1 (Public Report excludes High)
t8	1+0+1+0+1+0+1	1(Credit Card requires High) + 1 (High excludes Public report)
t9	1 + 0 + 1 + 0 + 0 + 1 + 0 + 1	1(Credit Card requires High) + 1 (High excludes Public report)

Next step is to apply the metric given in the formula 3.5. For t1 value of VC is 3 and CC is 1, both are picked in the metric as shown given below:

$$VC\&CC(FM, t) = \sqrt{VC^2 + CC^2}$$

$$VC\&CC_{t1} = \sqrt{3^2 + 1^2} = 3.16227766$$

Test suite value of $VC\&CC$ is shown in the table 3.8 that represent the FCC values of test suite.

TABLE 3.8: FCC of Test Suite

Test Case No.	VC&CC	FCC of Test Case
t1	$\sqrt{3^2 + 1^2}$	3.16227766
t2	$\sqrt{3^2 + 0^2}$	3
t3	$\sqrt{3^2 + 2^2}$	3.605551275
t4	$\sqrt{3^2 + 2^2}$	3.605551275
t5	$\sqrt{4^2 + 1^2}$	4.123105626
t6	$\sqrt{4^2 + 0^2}$	4
t7	$\sqrt{5^2 + 1^2}$	5.099019514
t8	$\sqrt{4^2 + 2^2}$	4.472135955
t9	$\sqrt{4^2 + 2^2}$	4.472135955

For normalization, maximum VC and CC value is calculated from the FM. The maximum variation points and cyclomatic complexity of FM is calculated by using the same metric given in the formula 3.5. In FM figure 3.2 'E-Shop', 'Payment', 'Security', 'Search' and 'Public Report' are the variation points whereas, one exclude constraint and one require constraint is present between 'High' and 'Public Report' and 'Credit Card' and 'High' respectively. so, VC is 5 and CC is 2. It is the maximum value of VC and CC that can be present in any product of this FM. VC&CC of FM is listed in the table 3.9 below:

TABLE 3.9: FCC of Feature Model

FM	VC	CC	VC&CC	FCC of FM
FM	5	2	$\sqrt{5^2 + 2^2}$	5.385164807

Normalized value of FCC is obtained by dividing each value of test case with the maximum value that is achieved from the FM. For t1, we obtained 3.1622 and 5.3851 for FM. So for P1 we obtained 0.5872 value. Likewise, all the FCC of test case's value is normalized as shown in the table 3.10.

TABLE 3.10: Normalized FCC of Test Suite

Test Case No.	FCC of Test Case
t1	0.58722025
t2	0.55708605
t3	0.6695341
t4	0.6695341
t5	0.7656415
t6	0.74278134
t7	0.9468642
t8	0.8304548
t9	0.8304548

The third and the last step of our approach is to combine the FC and FCC by assigning different weights for α . For this we have introduced an α factor as mentioned above. In the example 0.5 value is set to α which means we are giving the equal importance to each factor. By using equation 3.7, we get the following value for t1 test case.

$$TCC_t = \alpha * FC_t + (1 - \alpha) * FCC_t$$

$$TCC_{t1} = 0.5 * 0.5609 + (1 - 0.5) * 0.587$$

$$TCC_{t1} = 0.57395$$

Similarly all test cases TCC value is calculated as shown in the table 3.11.

Similarly complexity of test suite is calculated by assigning different values of α . After that, each test suite is prioritized according to their values. The above test suite of weight 0.5 is prioritized as $t9, t7, t8, t4, t5, t6, t3, t1, t2$ as shown in the table 3.12 with their TCC values.

TABLE 3.11: Test Suite Complexities

Test Case	FC and FCC with $\alpha = 0.5$	Complexity of Test Case
t1	$0.5 * (0.479167) + 0.5(0.5872)$	0.53318
t2	$0.5 * (0.479167) + 0.5(0.5570)$	0.51808
t3	$0.5 * (0.479167) + 0.5(0.6695)$	0.57433
t4	$0.5 * (0.75) + 0.5(0.6695)$	0.70975
t5	$0.5 * (0.5834) + 0.5(0.7656)$	0.6745
t6	$0.5 * (0.5834) + 0.5(0.7427)$	0.66305
t7	$0.5 * (0.729167) + 0.5(0.9468)$	0.8379835
t8	$0.5 * (0.5834) + 0.5(0.8304)$	0.75665
t9	$0.5 * (0.854167) + 0.5(0.8304)$	0.84228

TABLE 3.12: Prioritized Test Suite

Test Case	Test Case Complexity (TCC)
t9	0.84228
t7	0.8379835
t8	0.75665
t4	0.70975
t5	0.6745
t6	0.66305
t3	0.57433
t1	0.53318
t2	0.51808

After implementation of proposed criterion, evaluation will be performed with the different values of α to obtain the best test suite and with the existing technique. APFD for each test suite of different weights will be calculated and comparison will be performed.

Chapter 4

Implementation

This chapter comprises the implementation details of our proposed approach. For implementation, we have used FeatureIDE that is an Eclipse-based framework helps to implement the software product line (SPL) into an integrated development environment [26]. We have used this tool for modeling of feature and their configurations. Feature Models(FM) are taken from the software product lines online tool (SPLOT) repository [25] and modeled in FeatureIDE tool where all valid products are generated with the possible combinations that will be the test suite.

4.1 Implementation Details

In this section, implementation of our prioritization criterion details is described. Before implementing the proposed algorithm, we generate the test suite that consists of test cases and each test case contains several features in it. For this purpose we used FeatureIDE tool that automatically generates all the valid combinations of features, that is a test case. Example of FM modeled in FeatureIDE as shown in the following figure 4.1.

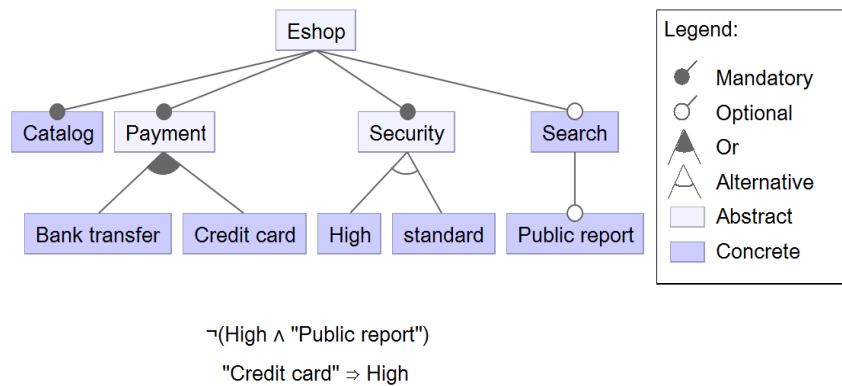


FIGURE 4.1: FM in FeatureIDE.

FM is exported from the SPLOT repository in the SXFM format and imported in the tool that automatically create the graphical view of FM as shown above in the figure. Cross tree constraints are shown by the textual form that can be created by the constraint editor in which constraint can be created or edited, constraint editor is shown in the figure 4.2

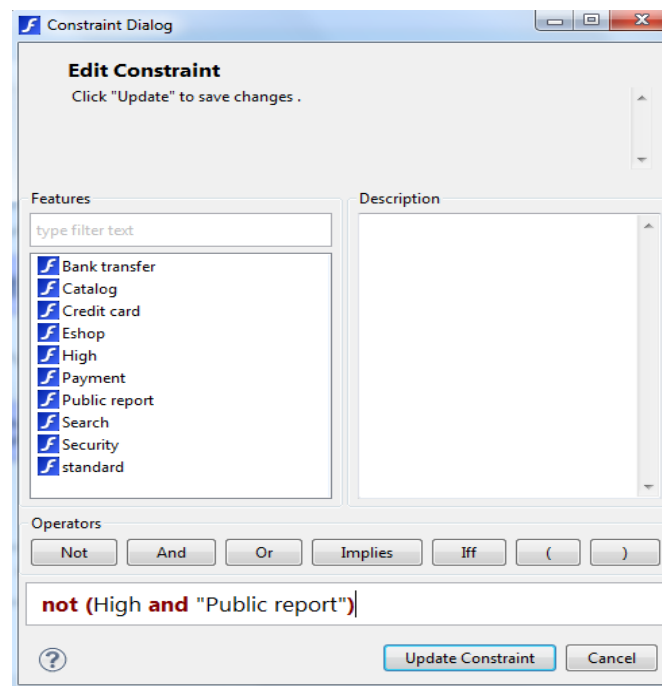


FIGURE 4.2: Constraint Editor of FeatureIDE.

Configurations from the FM is also created by the tool. It make sure that all the possible configurations are valid. Configuration window is shown in the following figure 4.3.

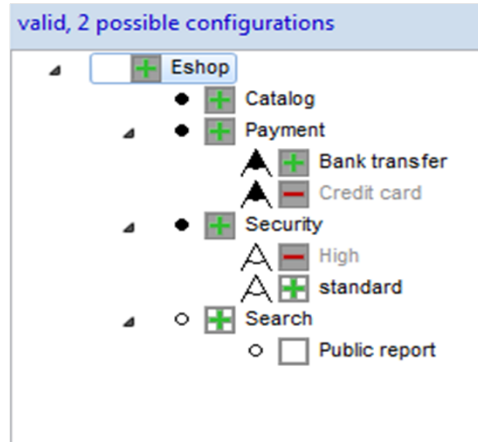


FIGURE 4.3: Configuration of FeatureIDE.

The FM created by the tool are stored in XML fromat. XML file of FM includes the information of features that are present in the FM with their types either they are abstract, concrete, mandatory or optional and types of constraint that are present between features,'require' or 'exclude'. Likewise, XML file of product or test case contains all the features with tag either they are selected in the test case or not.

Class diagram of implementation is given in figure 4.4. Whereas, details of implemented classes and methods are discussed in the next section. Each class consists of a number of methods that are used to implement the proposed algorithm.

4.1.1 Test Suite

TestSuite class is used to parse the XML files of FM and test cases in order to extract the information that is needed. This class consists of multiple methods such as: `testcaseXML()` In this method, we parse the XML file of each product and extract all the features that are part of the test cases so that we have the set of test cases with features. As already discussed above we had test cases with all

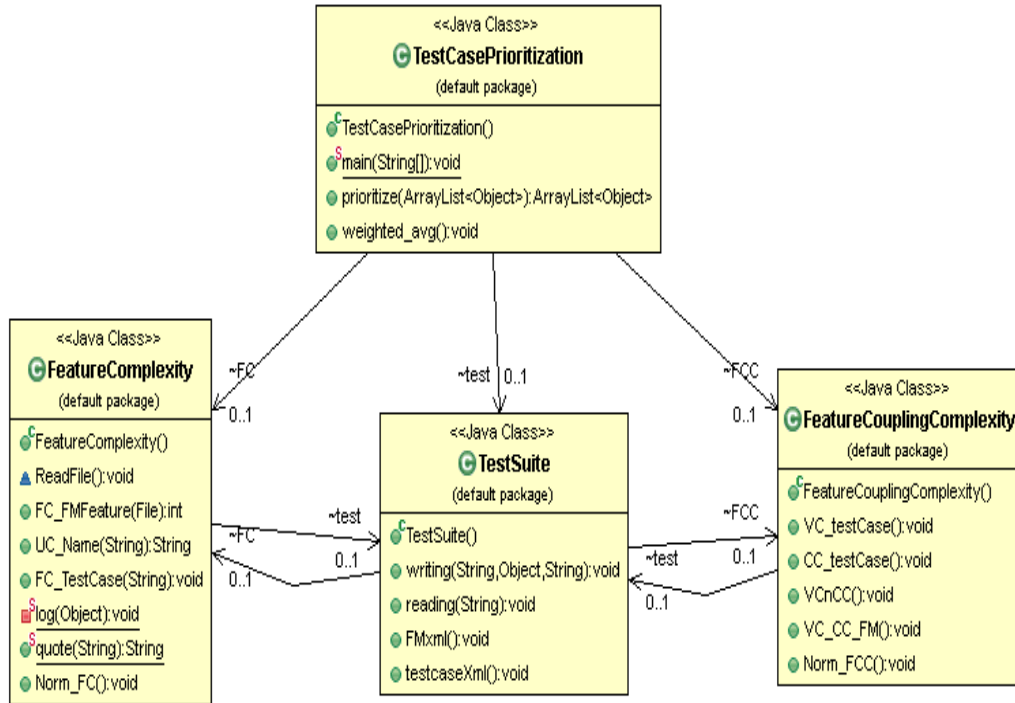


FIGURE 4.4: Class Diagram of Proposed Algorithm.

features that contain tag with the information that the feature is the part of a test case or not. So we have to extract only those features that are part of test cases. After that, all the test cases with features are saved into a file of the test suite. **FMXML()** Like *testcaseXML()*, this method also parse the XML file of FM and all the features are saved into a file with the information that features are mandatory, optional, alternative or of or type. In the same way, constraints are also extracted with the details of features that are the part of constraint and the type of constraints. **reading()** This method is created for reading an input file from the folder, as test case file is read from the folder. **writing()** This method is created for writing an output in files, like features complexities, test cases and prioritization list. Reading and writing methods in this class is used by different classes to write and read.

4.1.2 Feature Complexity

FeatureComplexity class is used to calculate the feature complexity of test suite. This class use the *.txt* file of use case description that contains the description

of features, *.csv* file of FM that contains the features of FM with the information extracted from the *FMXML()* and *.csv* file of test suite that contains the test cases obtained from the *testcaseXML()* as an input for calculating the feature complexity. It gives *.csv* file of test suite with FC as an output. **FC_FMfeature()** This method calculates the feature complexity of all the features that are the part of FM. It uses the list of features that are present in the FM and calculate complexity of all those features. If use case description is not present for any of the feature, it assigns zero value to that feature. This method returns the complexity of each feature of FM. **FC_TestCase()** This method calculates the complexity of each test case, that contains the set of features. *FC_FMfeature()* stores the complexity of each feature in the list, this method use that list and run the loop for each test case of the test suite with the inner loop that runs for features of each test case. Each feature complexity is taken from the list and combines by simple addition in order to get the test cases complexities. List of test cases with their complexities is collected from this method. **Norm_FC()** This method normalizes the complexities of test cases. Maximum value of the test case complexity is taken from the list of test cases obtained from the *FC_TestCase()* in order to normalize the set of test case. Each value of test case complexity obtained from the *FC_TestCase()* is divided with the maximum value. Feature complexity of test suite is obtained from this method which is saved in the file by using *writing()*.

4.1.3 Feature Coupling Complexity

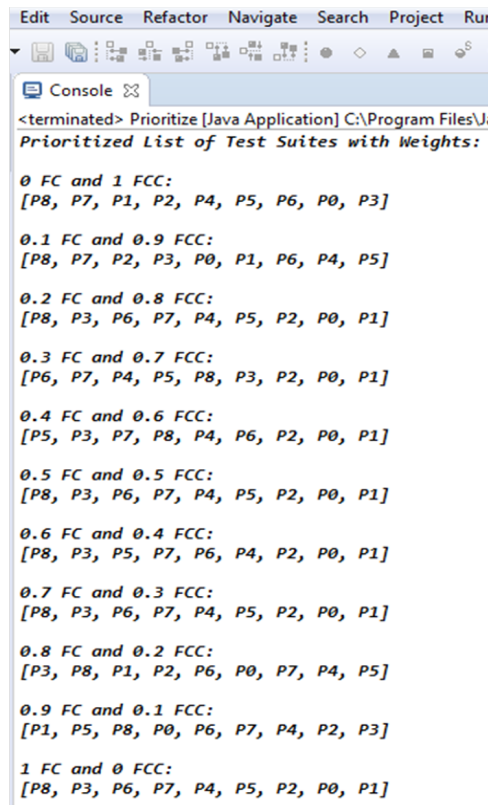
FeatureCouplingComplexity class is used to calculate the FCC of test suite. This method take *.csv* file of FM and test suite as an input. In addition, provide the *.csv* File of test suite with FCC. **VC_testCase()** This method reads the file of the test suite that contains the test cases with multiple features in it and calculates variability coverage of each test case. Variability coverage is calculated by considering each feature from the test case and checks the type of feature from the FM file. If it is of type 'optional', 'alternative' or 'or' then VC variable is incremented. **CC_testCase()** This method reads the file of the test suite

that contains the test cases with multiple features in it and calculates cyclomatic complexity of each test case. CC is calculated by checking the constraint from the FM file. As constraints present between two features so each feature is stored as start feature and end feature. Loop is used to run for all the test cases and check the constraint type as *'requires'* and *'excludes'*. For *'requires'* constraint, if start feature is present in the test case then end feature must also be present in the same test case if this condition is met than incremented in the CC value. For *'excludes'* constraint, if start feature is present in the test case then the end feature must not be present in the same test case, if this condition for *'excludes'* constraint is met then the CC value is incremented. Both VC and CC values are stored in the list. **VCnCC()** This method gets the value of VC and CC from CC and VC lists obtained from the *VC_testCase()* and *CC_testCase()* respectively. The formula of VC&CC is applied to the values of each test case and obtained the list of test case with feature coupling complexities. **VC_CC_FM()** This method calculates the VC and CC for FM. VC is calculated as defined above in *VC_testCase()* and CC is calculated by counting the number of constraints present in the FM file. In this method. type of constraints does not matter. For this, we only count the total number of constraint that is present in the FM. FCC is computed by using the same formula of VC&CC. **Norm_FCC()** This method uses the list of VC and CC obtained from the *VCnCC()* as an input to this method for the normalization. Each test case FCC value is divided by the maximum value obtained from the *VC_CC_FM()*. This method returns the values of FCC of the test suite and stored it in a file by using *writing()*.

4.1.4 Test Suite Prioritization

TestCaseProritization class is used to prioritize the test cases from the test suite. First, it combines the FC and FCC values with the help of α factor used as a weighting factor to adjust the weight of FC and FCC. FC and FCC values are taken from the *.csv* file. The weighted average is calculated and sets of test suite are obtained as an output of this method. The output is shown in the console as

well as each test suite is stored in the *.csv* file. **Weighted_avg()** This method combines the FC and FCC values. It assigns the weights to each test cases of the test suite. The first test suite is generated by assigning 0 value to FC and 1 value to FCC. Likewise, the weight of FC is incremented and FCC is decremented. Total eleven test suites are generated with different complexities and stored in a file by using *writing()*. **prioritize()** This method reads the *.csv* files of test suite complexities by using *reading()* one by one that is generated by *Weighted_avg()* and prioritize each test suite according to the complexity of test cases. Highest complexity assigns the first rank in the ordering list. All the prioritized test suite is stored in *.csv* file. The output of prioritized test suites is shown in the figure 4.5.



```

Edit Source Refactor Navigate Search Project Run
Console
<terminated> Prioritize [Java Application] C:\Program Files\J
Prioritized List of Test Suites with Weights:

0 FC and 1 FCC:
[P8, P7, P1, P2, P4, P5, P6, P0, P3]

0.1 FC and 0.9 FCC:
[P8, P7, P2, P3, P0, P1, P6, P4, P5]

0.2 FC and 0.8 FCC:
[P8, P3, P6, P7, P4, P5, P2, P0, P1]

0.3 FC and 0.7 FCC:
[P6, P7, P4, P5, P8, P3, P2, P0, P1]

0.4 FC and 0.6 FCC:
[P5, P3, P7, P8, P4, P6, P2, P0, P1]

0.5 FC and 0.5 FCC:
[P8, P3, P6, P7, P4, P5, P2, P0, P1]

0.6 FC and 0.4 FCC:
[P8, P3, P5, P7, P6, P4, P2, P0, P1]

0.7 FC and 0.3 FCC:
[P8, P3, P6, P7, P4, P5, P2, P0, P1]

0.8 FC and 0.2 FCC:
[P3, P8, P1, P2, P6, P0, P7, P4, P5]

0.9 FC and 0.1 FCC:
[P1, P5, P8, P0, P6, P7, P4, P2, P3]

1 FC and 0 FCC:
[P8, P3, P6, P7, P4, P5, P2, P0, P1]

```

FIGURE 4.5: Output of Prioritized Test Suites on Console.

Chapter 5

Results and Discussion

This chapter discusses the results of our experimentation performed on different feature models of various sizes. Feature models are collected from the SPLOT repository [25]. We generated all the valid configurations of the feature model using the FeatureIDE tool [26]. These configurations are the products or the test cases, which is the combinations of features, and it can be a large number of products so we have restricted our experiments to 100 products. The selected test suites for the prioritization include 100 test cases that are selected randomly and it makes sure that the selected products in the test suite cover all the features from the feature model. Test suites for all subject product lines are given in Appendix C. Applying the proposed criterion to the test suite, prioritized test suites are generated for each feature model by using the algorithms already discussed in chapter 3.

To measure the effectiveness of our proposed criterion, we evaluated the capability of criterion to detect the faults. For this purpose, faults are injected in the n-tuples of features and they are randomly selected to be seeded with faults where n can be 1, 2, 3 features. Different studies show that these types of faults commonly come in real tools [50] and [51]. Faults are considered to be detected by a test case if it includes the feature.

We have performed two evaluations, first; comparison among the prioritized test suites based on the different values of α to find on what values of α test suites achieves the best fault detection rate in proposed criterion and second, the comparison between the proposed criterion with the existing criterion proposed by Sánchez et al. [20]. For comparison, the existing Sánchez is implemented and prioritized test suites are generated based on the existing criterion. The average percentage of fault detection rate (APFD) metric is used to evaluate how quickly faults are detected with the criteria.

5.1 Case Studies

We have selected three case studies for experiments

1. Feature model of E-Commerce
2. Feature model of Social Network
3. Feature model of Transportation Network

5.1.1 E-Commerce SPL

is an online shopping product line. The feature model of e-commerce is composed of 24 features and 6 cross-tree constraints which comprises 704 total products. All the generated products are related to online shopping with minor changes in terms of features. FM contains 9 mandatory features, 8 optional, 5 with or relation and 2 alternative relations. There are 4 require cross-tree constraints and 2 exclude cross-tree constraints. Feature model contains 15 concrete features from which 2 features are nonfunctional and the remaining 9 features are abstract features.

5.1.2 Social Network SPL

is a social networking application. The feature model of a social network composed of 31 features and 3 requires cross-tree constraints which comprise 22222 total products. FM contains 10 mandatory features, 14 optional and 7 with or relation. 23 features are concrete features and remaining features are abstract features.

5.1.3 Transport Network SPL

is a transportation network application. Products that are generated from this FM are online transportation network. The feature model of transport network composed 24 features with 5 requires cross-tree constraints which comprise 2720 total products. FM contains 7 mandatory features, 6 optional and 11 with or relation. Feature model includes 18 concrete features and 6 abstract features.

FM of subject product line are given in Appendix A and test cases of these subject lines that are selected for the research work are given in Appendix B. Summary of case studies are illustrated in table 5.1 in which number of products, features, concrete features, abstract features and the number of faults injected in the product line are depicted.

TABLE 5.1: Summary of Subject Product Lines.

Product Line	Products	Features	Concrete Features	Abstract Features	Faults
E-Commerce	704	24	15	9	6
Social Network	22222	31	23	8	10
Transport Network	2720	24	18	6	10

5.2 Fault Injection

To compute the effectiveness of criteria faults are injected using the renowned method by Mathur et al. to assess the detection of faults of the test suite [52]. Faults are injected in the features of FM. We randomly select the features and marked as a faulty feature. Faults are marked as detected by a test case if faulted feature is present in the test case. For each fault, we make sure that at least one test case must be in the test suite detecting it so that test suite detected 100% of the faults. The technique of such fault seeding is common in [20, 32, 53] due to not having access to the case studies that have real faults and test cases.

5.3 Evaluation Metric

To evaluate the effectiveness of the proposed criterion Average Percentage of Faults Detected (APFD) metric is used [21, 23, 54, 55]. APFD is the standard metric for evaluating the prioritization techniques that helps to determine how quickly faults are detected by the prioritized test suite. Following formula is used to calculate the APFD.

$$1 - \frac{TF_1 + TF_2 + \dots + TF_m}{nm} + \frac{1}{2n}$$

Where T is the test suite containing n test cases and F is the set of m faults revealed by T . For ordering T' , TF_i is the position of the first test case that reveals the i th fault. APFD value ranges from 0 to 1. Higher APFD value of a prioritized test suite has faster fault detection rates than those with lower APFD values.

5.3.1 Example

Considering the test suite of Social Network product line with $\alpha = 0.6$ is given below:

$$T = \{ t81, t76, t83, t96, t98, t100, t70, t71, t86, t24, t30, t91, t85, t92, t66, t72, t87, t43, t88, t20, t36, t37, t78, t82, t80, t74, t50, t26, t99, t51, t55, t21, t47, t48, t61, t97, t67, t68, t69, t94, t17, t32, t77, t79, t27, t31, t75, t22, t89, t93, t15, t46, t42, t18, t44, t41, t62, t73, t29, t25, t28, t84, t56, t90, t16, t38, t64, t34, t35, t13, t65, t12, t95, t33, t39, t49, t63, t40, t23, t58, t52, t57, t59, t19, t45, t10, t14, t11, t54, t60, t53, t9, t7, t6, t4, t1, t5, t8, t2, t3 \}$$

$$F = \{ \text{message, group call, group message, audio message, comment, share, unfriend, block, search by name, create post} \}$$

$n = 100, m = 10$

$$APFD = 1 - \frac{8 + 1 + 19 + 8 + 1 + 7 + 1 + 1 + 1 + 1}{10 * 100} + \frac{1}{2 * 100}$$

$$APFD = 1 - \frac{48}{1000} + \frac{1}{200}$$

$$APFD = 0.957$$

We performed two evaluations; first is among the weighting factors of the proposed criterion in order to select the best value for α (weighting factor). APFD calculated for all the ten test suites of each subject product line and selected the weighting factor that perform best on APFD then second evaluation is performed that is a comparison among the existing criterion. Similarly, APFD value of test suite based on the existing criterion is calculated and compared it with the proposed criterion.

The following section illustrates the graph representation and the tables of the weighting factor for each subject product line. We selected the value of α (weighting factor) that gives the best APFD.

5.4 Experiment 1. Evaluation of Proposed Criterion Based on Different values of α

The first experiment is performed to obtain the test suite that attains the best APFD value. For this, we conduct the experiment by assigning different values for α as described in chapter 3.

5.4.1 Prioritized Test Suites Based on Proposed Criterion

Subject product lines are used to generate the prioritized test suites. Each FM generates ten prioritized test suites. In order to avoid the bias results, we have conducted our experiments five times on different test suites that contained different test cases with different faulted features. Five test suites are selected randomly with different faulted features. On each test suite, the experiment is performed on all values of α in order to get the best APFD. After conducting all the experiments, the average of all the APFD is taken that is given in the following section. One of the prioritized lists of e-commerce, social network, and transport network product lines are given in Appendix A. One of the prioritized lists based on the best values of α for all the subject product lines that are selected for the second experiment are given below:

Prioritized list for E-Commerce $\alpha = 0.5$ { t81, t76, t83, t98, t100, t96, t70, t71, t86, t91, t30, t24, t92, t85, t72, t87, t66, t43, t88, t82, t20, t36, t37, t78, t80, t50, t99, t74,t26, t51, t97,t21, t47, t48, t61, t55, t67, t68, t69, t94, t32, t77, t79,t17, t75, t31,t89, t27, t22,t93, t42, t41, t62, t46, t15, t18, t44, t73,t29, t84, t25,t28, t90,t56, t64, t38, t34, t35, t16, t13, t65, t95, t39,t49, t12, t33,t40, t63,

t58, t23, t57, t59, t52, t19, t45, t54, t11, t10, t14, t60, t53,t9, t7, t6, t4, t5, t1, t8, t2, t3}

APFD for E-commerce product line is given in the table 5.2 in which the column depicts different test suite that are used for experiments and the last column Final APFD depicts the average of APFD of all the test suits. The graphical representation of average APFD of E-commerce case study is illustrated in the figure 5.1.

TABLE 5.2: APFD for E-Commerce Product Line Based on Different Values of α .

α	T1 APFD	T2 APFD	T3 APFD	T4 APFD	T5 APFD	Final APFD
0	98.60	96.33	99.50	95.67	99.17	97.85
0.1	98.60	95.33	99.50	96.50	99.00	97.79
0.2	98.60	96.67	99.50	96.50	99.00	98.05
0.3	98.60	97.17	99.50	96.50	99.00	98.15
0.4	99.00	97.67	99.50	96.83	99.17	98.43
0.5	99.00	97.67	99.50	96.67	99.17	98.40
0.6	99.00	97.67	99.50	97.33	99.17	98.53
0.7	99.00	97.50	99.50	97.33	99.17	98.50
0.8	99.00	97.33	99.50	95.83	99.17	98.17
0.9	99.00	97.17	99.50	95.67	99.17	98.10
1	98.60	96.83	99.33	94.50	99.17	97.69

In the first column T1 of table 5.2, product line of e-commerce achieved 98.6% of APFD when we set the value of $\alpha = 0, 0.1, 0.2, 0.3$ and 1. Whereas, 99% of APFD is achieved when $\alpha = 0.4, 0.5, 0.6, 0.7, 0.8$ and 0.9. Difference between the values of APFD is due to the position of test case t83, it is located at 4th position when 98.6% of APFD is achieved. Whereas, t83 is located at 3rd position when 99% of APFD is achieved. From the average of all test suite, we concluded that on 0.6 value of α , the best APFD is achieved. So 0.6 value of α is selected for second evaluation.

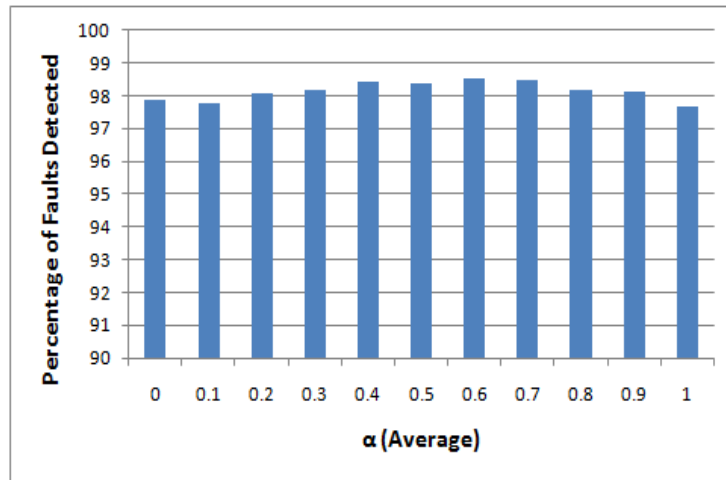


FIGURE 5.1: Graphical Representation of APFD for E-Commerce Product Line Based on Different Values of α .

Prioritized list for Social Network $\alpha = 0.6$ { t87, t88, t90, t89, t91, t92, t82, t100, t71, t73, t83, t84, t86, t85, t72, t74, t75, t48, t40, t66, t49, t50, t52, t41, t42, t99, t55, t57, t77, t67, t68, t51, t53, t54, t76, t24, t98, t97, t29, t56, t58, t59, t25, t26, t30, t31, t33, t70, t79, t13, t43, t78, t69, t32, t34, t35, t61, t80, t81, t60,t14,t15, t17,t62, t63,t94, t45,t37, t95,t96, t16,t18, t19,t44, t64,t27, t28,t36, t46,t47, t38,t39, t65,t21, t93,t20, t22,t23, t10,t11, t12,t8, t6,t7, t4,t5, t2,t3, t1,t9}

TABLE 5.3: APFD for Social Network Product Line Based on Different Values of α .

α	T1 APFD	T2 APFD	T3 APFD	T4 APFD	T5 APFD	Final APFD
0	94.80	97.90	98.50	99.30	96.83	97.47
0.1	95.60	97.70	98.50	99.30	97.67	97.75
0.2	95.30	97.30	98.30	99.30	97.67	97.57
0.3	95.10	97.30	98.10	99.10	97.67	97.45
0.4	95.30	97.70	98.30	99.10	97.67	97.61
0.5	95.20	97.70	98.50	99.10	97.83	97.67
0.6	95.70	97.70	98.50	99.10	98.33	97.87
0.7	95.30	97.50	98.30	99.10	98.00	97.64
0.8	95.50	97.50	98.50	99.10	98.17	97.75
0.9	95.70	97.50	98.50	99.10	98.17	97.79
1	94.40	97.50	99.10	99.10	97.50	97.52

Average APFD of all test suite achieved from the social network product line is given in the table 5.3 and graphical representation of average APFD is shown in the figure 5.2.

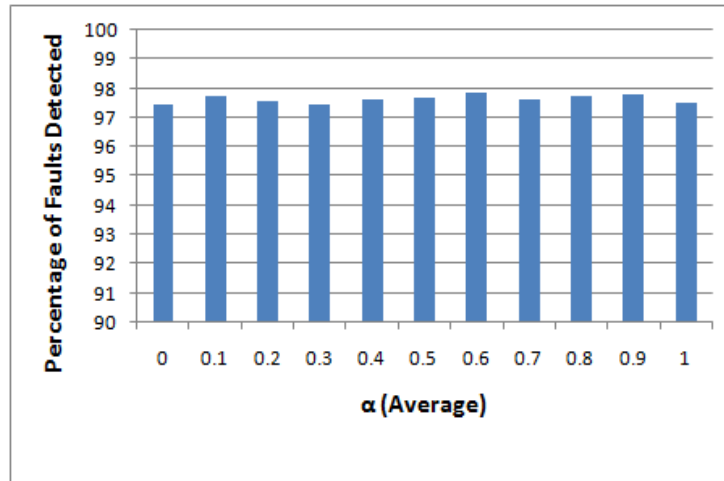


FIGURE 5.2: Graphical Representation of APFD for Social Network Product Line.

In T1 of table 5.3, the best APFD is achieved when the α value is set to 0.6 and 0.9 that is 95.7%. Whereas, all the other values of α achieved less APFD. It is due to the test cases t100, t97, t71 and t40. t40 and t97 vary from different test suites. In some test suites t97 with the combination of t100 and t71, which detected all the faults and in some test suite t40 comes with t100 and t71 that detected all the faults. The position of test cases in $\alpha = 0.6$ and $\alpha = 0.9$ is different which achieved the highest APFD. When $\alpha = 0.6$, the position of t100, t71 and t40 is 7th, 8th, and 19th whereas the position of t100, t71 and t40 in $\alpha = 0.9$ is 9th, 7th and 17th; all the faults are detected until test case t40. Lowest APFD achieved when $\alpha = 1$, the position of t100, t71 and t40 is 13th, 7th and 19th. From the average of APFD of all test suites, 0.6 value of α is performed best. So, we have 0.6 value of α for the second evaluation.

Prioritized list for Transport Network $\alpha = 0.2$ { t65, t67, t58, t66, t60, t96, t25, t68, t97, t59, t62, t27, t99, t61, t64, t95, t26, t29, t63, t98, t28, t31, t100, t30, t32, t69, t71, t77, t85, t79, t70, t73, t87, t21, t72, t75, t78, t81, t86, t89, t20, t23, t90, t74, t80, t83, t22, t92, t82, t57, t76, t88, t24, t33, t84, t91, t94, t35, t93,

t34, t36, t9, t38, t44, t52, t15, t46, t54, t37, t40, t10, t12, t39, t42, t45, t48, t53, t56, t11, t16, t47, t50, t55, t41, t17, t19, t13, t1, t49, t18, t43, t3, t14, t51, t2, t5, t4, t7, t6, t8}

TABLE 5.4: APFD for Transport Network Product Line Based on Different Values of α .

α	T1 APFD	T2 APFD	T3 APFD	T4 APFD	T5 APFD	Final APFD
0	97.80	98.90	96.30	96.83	91.67	96.30
0.1	97.80	99.10	96.67	97.67	93.50	96.95
0.2	98.20	99.10	96.20	97.67	93.50	96.93
0.3	97.70	99.10	96.20	97.67	93.17	96.77
0.4	97.20	99.30	96.30	97.67	93.50	96.79
0.5	97.00	99.30	96.30	97.83	93.33	96.75
0.6	97.00	99.30	96.10	98.33	94.17	96.98
0.7	97.00	99.30	96.10	98.00	94.17	96.91
0.8	96.10	99.30	96.10	98.17	94.50	96.83
0.9	96.10	99.20	96.11	98.33	94.17	96.78
1	93.60	99.20	96.00	97.50	91.50	95.56

Average APFD of all test suites achieved from the transport network product line is given in the table 5.4 and graphical representation of average APFD of all test suites is shown in the figure 5.3.

T1 of table 5.4 The minimum APFD is achieved when α is set to 1 that means weight is applied only to the FC and FCC is being ignored. The best APFD is achieved when the α value is set to 0.2 that is 98.2%. Whereas, all the other values of α achieved less APFD. It is due to the test cases t58 and t97. The position of test cases in $\alpha = 0.1$ are 3th and 6th, whereas in all other test suites they lie on different positions but when we take the average of APFD of all test suites, 0.6 value of α achieved the best APFD as in previous. So, from this product line, $\alpha = 0.6$ is selected as the best APFD and compared with the existing criterion in the second evaluation.

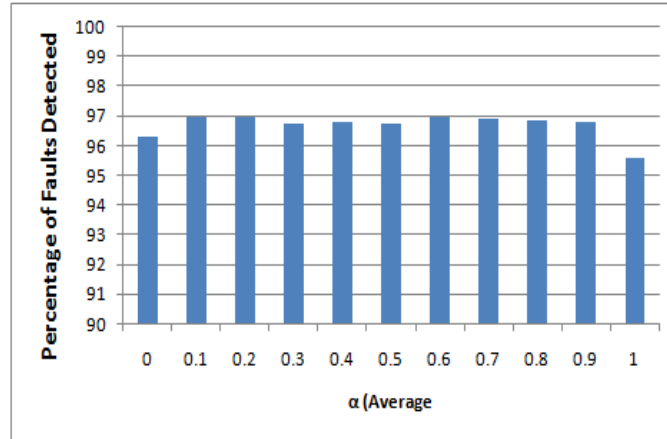


FIGURE 5.3: Graphical Representation of APFD for Transport Network Product Line Based on Different Values of α .

5.4.2 Comparison

We are able to know from experiment 1, on E-Commerce product line the best APFD (88.5%) is achieved when more weight is assigned to FC and less to FCC , that is, 0.6. whereas, when FC or FCC is taken individually, achieved APFD is not good enough. On Social Network product line, the best APFD (97.97%) is achieved when more weight is assigned to FC and less to FCC , similar to previous case study, that is, 0.6. On Transport Network product line, the best APFD (96.98%) is achieved when more weight is assigned to FC and less to FCC that is, when $\alpha = 0.6$.

We have concluded from experiment 1, when the only FC factor is applied to the test suite, it does not perform well on APFD. Similarly, when FCC factor is applied individually, achieved APFD is not good enough as well. The best APFD is obtained when approximately equal weights are assigned for both factors, that is, 0.6 value of α . Therefore, the FC factor is an important factor that can enhance the APFD of the test suite when used with FCC factor. Both factors are equally important in calculating the complexity of the product.

5.5 Experiment 2. Evaluation of Proposed and Existing Criteria

The second experiment is performed for the evaluation of proposed criterion with the existing criterion. The best APFD that obtained from the experiment 1 is compared with the APFD based on existing criterion.

5.5.1 Prioritized Test Suites Based on Existing and Proposed Criteria

The prioritized test suites for E-Commerce, Social Network and Transport Network product lines are given below in the table 5.5, 5.6 and 5.7 respectively.

TABLE 5.5: Priority List for E-Commerce Product Line

Existing Criterion based Priority List	Proposed Criterion based Priority List
t81, t98, t100, t83, t76, t91, t96, t70, t71, t72, t86, t87, t92, t30, t43, t82, t88, t99, t66, t85, t80, t97, t24, t50, t78, t89, t20, t21, t32, t36, t37, t47, t48, t51, t61, t67, t68, t69, t74, t75, t77, t79, t94, t26, t31, t41, t42, t62, t90, t55, t73, t84, t93, t17, t18, t22, t27, t29, t44, t46, t56, t64, t95, t25, t28, t34, t35, t38, t39, t49, t65, t15, t13, t40, t12, t16, t33, t58, t63, t57, t59, t60, t19, t23, t45, t54, t11, t52, t10, t14, t53, t9, t7, t6, t4, t8, t5, t1, t2, t3	t81, t76, t83, t98, t100, t96, t70, t71, t86, t91, t30, t24, t92, t85, t72, t87, t66, t43, t88, t82, t20, t36, t37, t78, t80, t50, t99, t74, t26, t51, t97, t21, t47, t48, t61, t55, t67, t68, t69, t94, t32, t77, t79, t17, t75, t31, t89, t27, t22, t93, t42, t41, t62, t46, t15, t18, t44, t73, t29, t84, t25, t28, t90, t56, t64, t38, t34, t35, t16, t13, t65, t95, t39, t49, t12, t33, t40, t63, t58, t23, t57, t59, t52, t19, t45, t54, t11, t10, t14, t60, t53, t9, t7, t6, t4, t5, t1, t8, t2, t3

TABLE 5.6: Priority List for Social Network Product Line

Existing Criterion based Priority List	Proposed Criterion based Priority List
t82, t87, t88, t90, t83, t84, t86, t89, t91, t92, t85, t55, t57, t61, t79, t97, t100, t56, t58, t59, t60, t62, t63, t77, t78, t80, t81, t94, t95, t96, t40, t41, t42, t48, t71, t73, t49, t50, t52, t72, t74, t75, t76, t99, t51, t53, t54, t93, t98, t29, t30, t31, t33, t37, t66, t70, t27, t28, t32, t34, t35, t36, t38, t39, t45, t67, t68, t69, t43, t44, t46, t47, t24, t25, t26, t64, t65, t13, t14, t15, t17, t21, t16, t18, t19, t20, t22, t23, t10, t11, t12, t8, t4, t6, t7, t2, t3, t5, t1, t9	t87, t88, t90, t89, t91, t92, t82, t100, t71, t73, t83, t84, t86, t85, t72, t74, t75, t48, t40, t66, t49, t50, t52, t41, t42, t99, t55, t57, t77, t67, t68, t51, t53, t54, t76, t24, t98, t97, t29, t56, t58, t59, t25, t26, t30, t31, t33, t70, t79, t13, t43, t78, t69, t32, t34, t35, t61, t80, t81, t60, t14, t15, t17, t62, t63, t94, t45, t37, t95, t96, t16, t18, t19, t44, t64, t27, t28, t36, t46, t47, t38, t39, t65, t21, t93, t20, t22, t23, t10, t11, t12, t8, t6, t7, t4, t5, t2, t3, t1, t9

TABLE 5.7: Priority List for Transport Network Product Line

Existing Criterion based Priority List	Proposed Criterion based Priority List
t65, t67, t66, t68, t96, t97, t99, t25, t27, t58, t60, t95, t98, t100, t26, t28, t29, t31, t59, t61, t62, t64, t30, t32, t63, t69, t71, t77, t79, t85, t20, t21, t23, t70, t72, t73, t75, t78, t80, t81, t83, t86, t87, t89, t90, t92, t22, t24, t57, t74, t76, t82, t84, t33, t35, t88, t91, t93, t94, t34, t9, t15, t36, t38, t44, t46, t52, t54, t10, t11, t12,	t65, t67, t58, t66, t60, t96, t25, t68, t97, t59, t62, t27, t99, t61, t64, t95, t26, t29, t63, t98, t28, t31, t100, t30, t32, t69, t71, t77, t85, t79, t70, t73, t87, t21, t72, t75, t78, t81, t86, t89, t20, t23, t90, t74, t80, t83, t22, t92, t82, t57, t76, t88, t24, t33, t84, t91, t94, t35, t93, t34, t36, t9, t38, t44, t52, t15, t46, t54, t37, t40, t10,

t16, t17, t19, t37, t39, t40, t42, t45, t47, t48, t50, t53, t55, t56, t13, t14, t18, t41, t43, t49, t51, t1, t3, t2, t4, t5, t7, t6, t8	t12, t39, t42, t45, t48, t53, t56, t11, t16, t47, t50, t55, t41, t17, t19, t13, t1, t49, t18, t43, t3, t14, t51, t2, t5, t4, t7, t6, t8
--	--

5.5.2 Comparison

Comparison of proposed and existing criteria for E-Commerce product line is illustrated in figure 5.4 as a graphical representation. Approximately 98% of APFD is achieved by existing criterion whereas approximately 99% is achieved by the proposed criterion, which proves that proposed criterion gives better fault detection rate than existing criterion.

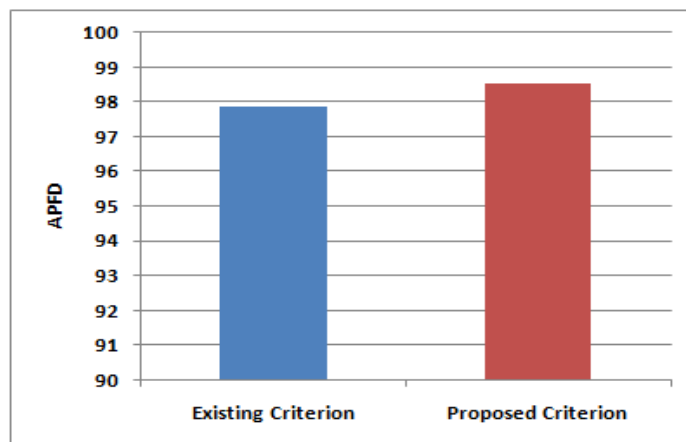


FIGURE 5.4: APFD Comparison of Existing and Proposed Criterion for E-Commerce Subject Product Line.

The difference is minor that is 0.68% between the criteria. A minor difference is due to the same prioritization list in which the only position of one test case is different. Graph representation of the execution of one of the test suite's test cases and the percentage of detected faults are depicted in figure 5.5. From which we concluded that the proposed criterion detected 100% of faults with the execution of only 3 test cases only. While the existing criterion detected 100% of faults when executing 4 test cases. It is due to the position of test case t83 and t100, which is

placed at 4th and 3rd position on existing test suite while on 3rd and 2nd position of proposed test suite respectively.

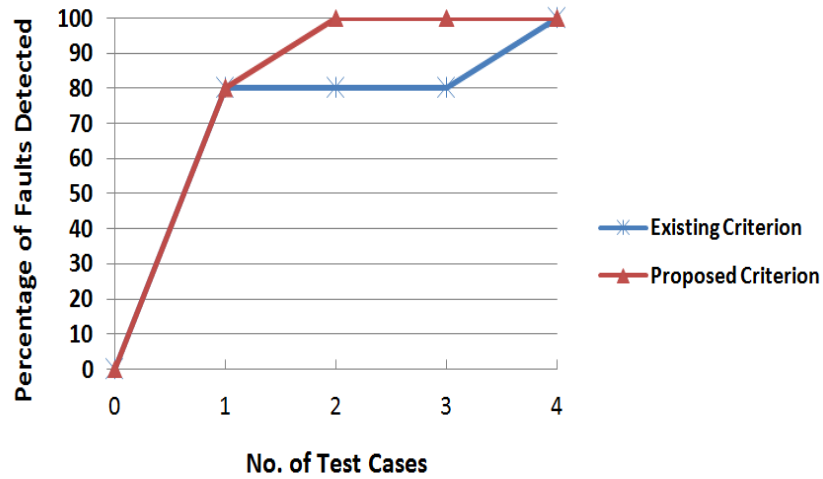


FIGURE 5.5: Graphical Representation of Fault Detection of Test Cases for E-Commerce Subject Product Line.

Social network product line comparison among existing and proposed criterion is depicted as a graph in figure 5.6. 94.8% of APFD is achieved by existing criterion whereas 95.7% is achieved by the proposed criterion, which proves that proposed criterion gives better fault detection rate than existing criterion.

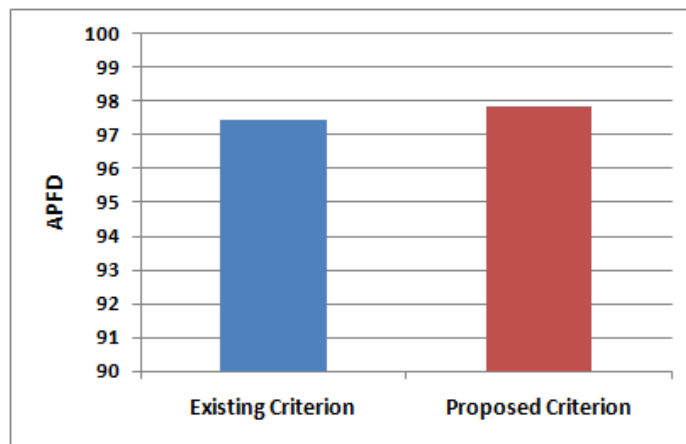


FIGURE 5.6: APFD Comparison of Existing and Proposed Criterion for Social Network Subject Product Line.

The difference is 0.4% between the criteria. Graph representation of fault detection of test cases of one test suite for social network product line is shown in figure

5.7 which concluded that the proposed criterion detected 90% of faults with the execution of 8 test cases only while existing criterion detected 90% of faults when executing 16 test cases. It is due to the position of test case t100 that is placed at 17th position in the existing criterion test suite whereas in the proposed criterion test suite t100 is at 8th position. However, both test suites detected 100% of faults with the execution of 19 test cases.

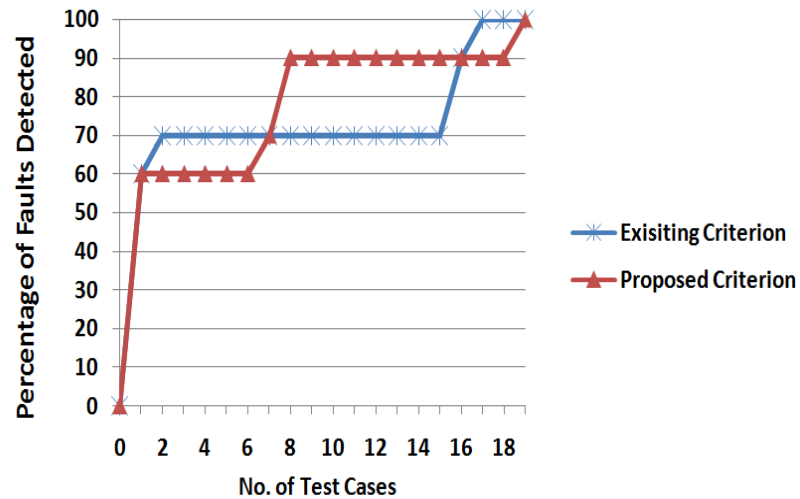


FIGURE 5.7: Graphical Representation of Fault Detection of Test Cases for Social Network Subject Product Line.

In a transport network product line, the proposed criterion performs better than the existing criterion. The difference is 0.4%. The graphical representation is shown in figure 5.8.

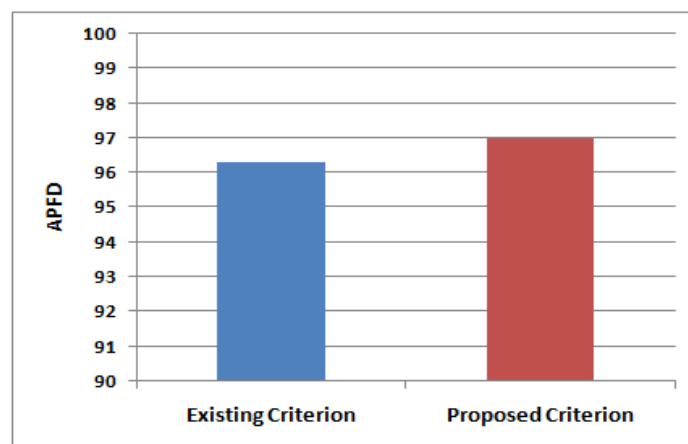


FIGURE 5.8: APFD Comparison of Existing and Proposed Criterion for Transport Network Subject Product Line.

Figure 5.8 shows that, initially both criteria performs similar and detected equal percent of faults from the test suite, but with the execution of test case 2, proposed criterion detected 100% of faults from the test suite and existing criterion detected 80% faults only till the execution of test case 4. With the execution of only two test cases proposed criterion detected all the faults whereas, existing criterion needs four test cases.

Graph representation of fault detection of test cases for transport network product line is shown in figure 5.9. The fluctuation is due to the test cases t58, t96 and t97. Test case t58 and t97 detected same faults but t58 appears on top of the proposed prioritization list and t97 appears on top in existing prioritization list at 3rd and 5th position respectively. However, t97 in proposed prioritization list is at 9th position and t58 in existing prioritization list is at 11th position. While t96 in proposed and existing criterion is at 6th and 5th position.

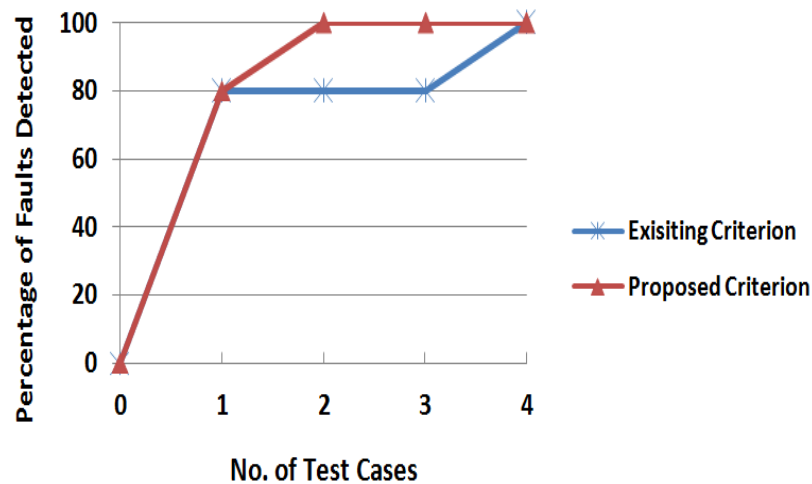


FIGURE 5.9: Graphical Representation of Fault Detection of Test Cases for Transport Network Subject Product Line.

Comparison of APFD for proposed and existing prioritization criterion concluded that the proposed criterion perform better than the other. Summary of average APFD of all test suite is shown in table 5.8 and graphical representation is illustrated in figure 5.10.

TABLE 5.8: Comparison of APFD for Subject Product Lines.

Subject Product Line	APFD for Existing Criterion	APFD for Proposed Criterion
E-commerce	97.85%	98.53%
Social network	97.47%	97.87%
Transport network	96.30%	96.98%

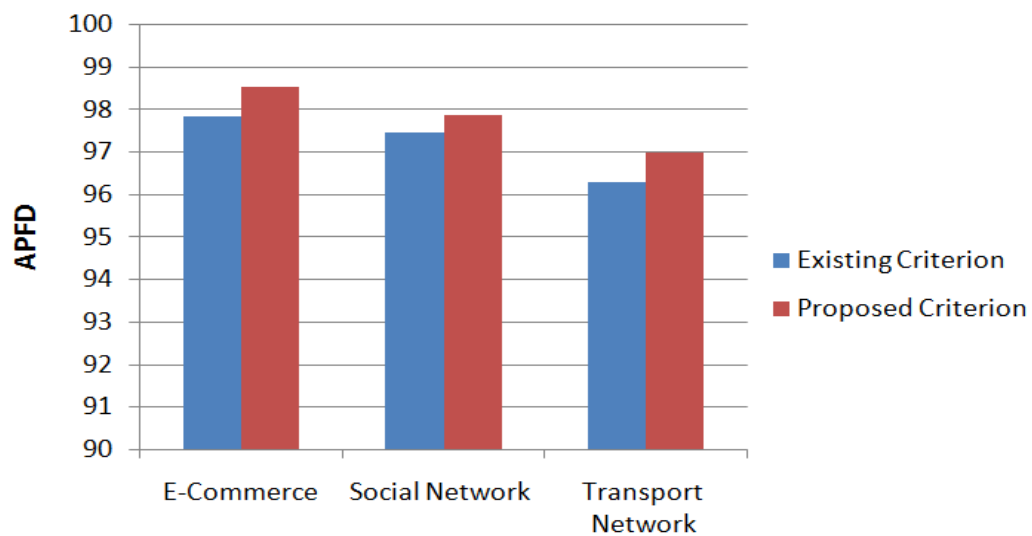


FIGURE 5.10: Graphical Representation of APFD Comparison for Subject Product Lines .

Existing criterion approach covers the coupling complexity of test cases and assumed to cover the maximum fault detection rate. It considers only those test cases that have constraints and high variability among the features. Since the proposed criterion considers the complexity of individual feature with the feature variability and constraints among the features so that test case complexity is cover entirely. From the experiment results, it is concluded that with the combination of individual and coupling complexity of feature, fault detection rate can be maximized instead of using individual factor either feature complexity or feature coupling complexity.

Chapter 6

Conclusion and Future Work

Software product line testing is one of the most laborious tasks due to a large number of products variants derived from the product line. Testing of individual product is not feasible and a tedious task. Test case prioritization is used to reduce the time and cost of testing also to fulfill the requirement of the tester to find faults as fast as possible within in the limited budget and time. We have conducted the literature survey and concluded that different criteria are used to prioritize the products of product lines. We have also proposed a new prioritization criterion that finds the complexity of the products by using individual feature complexity and feature coupling complexity and assign the highest priority to those products that attain the highest complexity. To answer the research questions depicted in chapter 1, we have performed two experiments and the results obtained to positively answer the following questions:

RQ1. Does existing approaches consider the complexity of feature for prioritization?

In existing criteria, mostly criteria depend on the coverage of features for the prioritization of products. The goal of coverage criteria is to cover the maximum part of the products but evaluation performed by Sánchez et al.[20] showed that Average Percentage of Fault Detection (APFD) based on coverage criteria is not good enough also the complexity of product is not considered in the coverage

criterion. Another criterion used for prioritizing is the individual feature criterion in which individual priority or the frequency of feature is considered, this criterion covers the individual feature but did not focus on the complexity of the feature. Evaluation results also showed that this type of criteria did not perform well on the rate of fault detection. One more criterion, feature coupling complexity is used to measure the complexity of the product and assigns the highest priority to the most complex product. Prioritization based on feature coupling complexity criterion performs well in terms of fault detection than the previous criteria but it only considers the coupling complexity of features to find the product's complexity. However, none of the existing techniques consider the complexity of the feature for prioritization.

RQ2. Can test case prioritization through feature complexity improves fault detection rate?

The criterion proposed in this research work covers the complexity of the product in order to prioritize the test suite. Product with maximum complexity has the highest priority in the list as it is the most complex product, so it listed to the top to test first. To answer this question, we perform an evaluation of our work by comparing APFD of proposed criterion with the strongest existing criterion. Three case studies of software product lines are used to calculate the complexities of the test suite based on the proposed criterion as well as on the existing criterion. The main objective of the proposed criterion is to increase the APFD of the test suite. From the evaluation, we can say that the proposed criterion performs well than the strongest existing criterion proposed by Sánchez et al.[20].

6.1 Future work

Existing criterion approach covers the coupling complexity of test cases and assumed to cover the maximum fault detection rate. It considers only those test cases that have constraints and high variability among the features. Since the proposed criterion considers the complexity of individual feature with the feature variability

and constraints among the features so that test case complexity is cover entirely. From the experiment results, it is concluded that with the combination of individual and coupling complexity of feature, fault detection rate can be maximized instead of using individual factor either feature complexity or feature coupling complexity.

Bibliography

- [1] P. Clements and L. Northrop, *Software product lines: practices and patterns*. Addison-Wesley Reading, 2002, vol. 3.
- [2] M. Kim, S. Park, V. Sugumaran, and H. Yang, “Managing requirements conflicts in software product lines: A goal and scenario based approach,” *Data & Knowledge Engineering*, vol. 61, no. 3, pp. 417–432, 2007.
- [3] J. D. McGregor, “Software product lines.” *Journal of Object Technology*, vol. 3, no. 3, pp. 65–74, 2004.
- [4] D. M. Weiss, P. C. Clements, K. Kang, and C. Krueger, “Software product line hall of fame,” in *Software Product Line Conference, 2006 10th International*. IEEE, 2006, pp. 237–237.
- [5] S. Wang, A. Gotlieb, S. Ali, and M. Liaaen, “Automated selection of test cases using feature model for product lines: An industrial case study.”
- [6] Splc2-product line hall of fame,2004. [Online]. Available: www.sei.cmu.edu/SPLC2/SPLC2_hof.html
- [7] K. Pohl, G. Böckle, and F. J. van Der Linden, *Software product line engineering: foundations, principles and techniques*. Springer Science & Business Media, 2005.
- [8] J. C. Dager, “Cumminss experience in developing a software product line architecture for real-time embedded diesel engine controls,” in *Software Product Lines*. Springer, 2000, pp. 23–45.

-
- [9] S. Apel, D. Batory, C. Kästner, and G. Saake, “A development process for feature-oriented product lines,” in *Feature-Oriented Software Product Lines*. Springer, 2013, pp. 17–44.
- [10] K. Schmid, R. Rabiser, and P. Grünbacher, “A comparison of decision modeling approaches in product lines,” in *Proceedings of the 5th Workshop on Variability Modeling of Software-Intensive Systems*. ACM, 2011, pp. 119–126.
- [11] D. Batory, D. Benavides, and A. Ruiz-Cortes, “Automated analysis of feature models: challenges ahead,” *Communications of the ACM*, vol. 49, no. 12, pp. 45–47, 2006.
- [12] K. Czarnecki, U. W. Eisenecker, and K. Czarnecki, *Generative programming: methods, tools, and applications*. Addison Wesley Reading, 2000, vol. 16.
- [13] K. C. Kang, S. G. Cohen, J. A. Hess, W. E. Novak, and A. S. Peterson, “Feature-oriented domain analysis (foda) feasibility study,” Carnegie-Mellon Univ Pittsburgh Pa Software Engineering Inst, Tech. Rep., 1990.
- [14] K. Lee, K. C. Kang, and J. Lee, “Concepts and guidelines of feature modeling for product line software engineering,” in *International Conference on Software Reuse*. Springer, 2002, pp. 62–77.
- [15] D. Benavides, S. Segura, and A. Ruiz-Cortés, “Automated analysis of feature models 20 years later: A literature review,” *Information Systems*, vol. 35, no. 6, pp. 615–636, 2010.
- [16] L. Baresi and M. Pezze, “An introduction to software testing,” *Electronic Notes in Theoretical Computer Science*, vol. 148, no. 1, pp. 89–111, 2006.
- [17] R. K. Chauhan and I. Singh, “Latest research and development on software testing techniques and tools,” *International Journal of Current Engineering and Technology*, vol. 4, no. 4, 2014.
- [18] G. Perrouin, S. Sen, J. Klein, B. Baudry, and Y. Le Traon, “Automatic and scalable t-wise test case generation strategies for software product lines,” in

- International Conference on Software Testing*. Springer Lecture Notes in Computer Science (LNCS), 2010.
- [19] C. Henard, M. Papadakis, G. Perrouin, J. Klein, and Y. L. Traon, “Multi-objective test generation for software product lines,” in *Proceedings of the 17th International Software Product Line Conference*. ACM, 2013, pp. 62–71.
- [20] A. B. Sánchez, S. Segura, and A. Ruiz-Cortés, “A comparison of test case prioritization criteria for software product lines,” in *Software Testing, Verification and Validation (ICST), 2014 IEEE Seventh International Conference on*. IEEE, 2014, pp. 41–50.
- [21] S. Elbaum, A. G. Malishevsky, and G. Rothermel, *Prioritizing test cases for regression testing*. ACM, 2000, vol. 25, no. 5.
- [22] S. Yoo and M. Harman, “Regression testing minimization, selection and prioritization: a survey,” *Software Testing, Verification and Reliability*, vol. 22, no. 2, pp. 67–120, 2012.
- [23] S. Elbaum, A. G. Malishevsky, and G. Rothermel, “Test case prioritization: A family of empirical studies,” *IEEE transactions on software engineering*, vol. 28, no. 2, pp. 159–182, 2002.
- [24] G. Rothermel, R. H. Untch, C. Chu, and M. J. Harrold, “Prioritizing test cases for regression testing,” *IEEE Transactions on software engineering*, vol. 27, no. 10, pp. 929–948, 2001.
- [25] M. Mendonca, M. Branco, and D. Cowan, “Splot: software product lines online tools,” in *Proceedings of the 24th ACM SIGPLAN conference companion on Object oriented programming systems languages and applications*. ACM, 2009, pp. 761–762.
- [26] T. Thüm, C. Kästner, F. Benduhn, J. Meinicke, G. Saake, and T. Leich, “Featureide: An extensible framework for feature-oriented software development,” *Science of Computer Programming*, vol. 79, pp. 70–85, 2014.

-
- [27] X. Devroey, G. Perrouin, M. Cordy, P.-Y. Schobbens, A. Legay, and P. Heymans, “Towards statistical prioritization for software product lines testing,” in *Proceedings of the Eighth International Workshop on Variability Modelling of Software-Intensive Systems*. ACM, 2014, p. 10.
- [28] C. Henard, M. Papadakis, G. Perrouin, J. Klein, P. Heymans, and Y. Le Traon, “Bypassing the combinatorial explosion: Using similarity to generate and prioritize t-wise test configurations for software product lines,” *IEEE Transactions on Software Engineering*, vol. 40, no. 7, pp. 650–670, 2014.
- [29] M. Al-Hajjaji, T. Thüm, J. Meinicke, M. Lochau, and G. Saake, “Similarity-based prioritization in software product-line testing,” in *Proceedings of the 18th International Software Product Line Conference-Volume 1*. ACM, 2014, pp. 197–206.
- [30] S. Wang, D. Buchmann, S. Ali, A. Gotlieb, D. Pradhan, and M. Liaaen, “Multi-objective test prioritization in software product line testing: an industrial case study,” in *Proceedings of the 18th International Software Product Line Conference-Volume 1*. ACM, 2014, pp. 32–41.
- [31] M. Al-Hajjaji, T. Thüm, M. Lochau, J. Meinicke, and G. Saake, “Effective product-line testing using similarity-based product prioritization,” *Software & Systems Modeling*, pp. 1–23, 2016.
- [32] M. Al-Hajjaji, S. Lity, R. Lachmann, T. Thüm, I. Schaefer, and G. Saake, “Delta-oriented product prioritization for similarity-based product-line testing,” in *Proceedings of the 2nd International Workshop on Variability and Complexity in Software Design*. IEEE Press, 2017, pp. 34–40.
- [33] M. Al-Hajjaji, J. Krüger, S. Schulze, T. Leich, and G. Saake, “Efficient product-line testing using cluster-based product prioritization,” in *Proceedings of the 12th International Workshop on Automation of Software Testing*. IEEE Press, 2017, pp. 16–22.

-
- [34] A. Ensan, E. Bagheri, M. Asadi, D. Gasevic, and Y. Biletskiy, “Goal-oriented test case selection and prioritization for product line feature models,” in *Information Technology: New Generations (ITNG), 2011 Eighth International Conference on*. IEEE, 2011, pp. 291–298.
- [35] M. Al-Hajjaji, J. Krüger, S. Schulze, T. Leich, and G. Saake, “Efficient product-line testing using cluster-based product prioritization,” in *Proceedings of the 12th International Workshop on Automation of Software Testing*. IEEE Press, 2017, pp. 16–22.
- [36] A. Braganca and R. J. Machado, “Automating mappings between use case diagrams and feature models for software product lines,” in *Software Product Line Conference, 2007. SPLC 2007. 11th International*. IEEE, 2007, pp. 3–12.
- [37] M. L. Griss, “Implementing product-line features with component reuse,” in *International Conference on Software Reuse*. Springer, 2000, pp. 137–152.
- [38] M. Alférez, U. Kulesza, A. Sousa, J. P. Santos, A. Moreira, J. Araújo, and V. Amaral, “A model-driven approach for software product lines requirements engineering.” in *SEKE*, 2008, pp. 779–784.
- [39] B. Wang, W. Zhang, H. Zhao, Z. Jin, and H. Mei, “A use case based approach to feature models’ construction,” in *Requirements Engineering Conference, 2009. RE’09. 17th IEEE International*. IEEE, 2009, pp. 121–130.
- [40] I. Jacobson, *Object-oriented software engineering: a use case driven approach*. Pearson Education India, 1993.
- [41] A. Bertolino and S. Gnesi, “Use case-based testing of product lines,” *ACM SIGSOFT Software Engineering Notes*, vol. 28, no. 5, pp. 355–358, 2003.
- [42] A. Cockburn, “Structuring use cases with goals,” *Journal of Object-Oriented Programming*, vol. 10, no. 5, 1997.
- [43] P. Kruchten, “The rational unified process—an introduction. addison-wesley publishing company, 298p,” 2000.

-
- [44] A. Durán, B. Bernárdez, M. Toro, R. Corchuelo, A. Ruiz, and J. Pérez, “Expressing customer requirements using natural language requirements templates and patterns,” in *IMACS/IEEE CSCC99 Proceedings*, 1999.
- [45] B. Bernárdez, A. Durán, and M. Genero, “Metrics for use cases: A survey of current proposals,” in *Metrics for software conceptual models*. World Scientific, 2005, pp. 59–98.
- [46] D. Fonte, I. V. Boas, J. Azevedo, J. J. Peixoto, P. Faria, P. Silva, T. Sá, U. Costa, D. da Cruz, and P. R. Henriques, “Modeling languages: metrics and assessing tools,” *arXiv preprint arXiv:1206.4477*, 2012.
- [47] A. Durán Toro, A. Ruiz Cortés, R. Corchuelo Gil, and M. Toro Bonilla, “Supporting requirements verification using xslt,” 2002.
- [48] M. Genero Bocco, A. Durán Toro, B. Bernárdez Jiménez *et al.*, “Empirical evaluation and review of a metrics-based approach for use case verification,” *Journal of Research and Practice in Information Technology*, vol. 36, no. 4, pp. 247–258, 2004.
- [49] T. mThum, C. Kastner, S. Erdweg, and N. Siegmund, “Abstract features in feature modeling,” in *Software Product Line Conference (SPLC), 2011 15th International*. IEEE, 2011, pp. 191–200.
- [50] D. R. Kuhn, D. R. Wallace, and A. M. Gallo, “Software fault interactions and implications for software testing,” *IEEE transactions on software engineering*, vol. 30, no. 6, pp. 418–421, 2004.
- [51] D. R. Kuhn and M. J. Reilly, “An investigation of the applicability of design of experiments to software testing,” in *Software Engineering Workshop, 2002. Proceedings. 27th Annual NASA Goddard/IEEE*. IEEE, 2002, pp. 91–95.
- [52] A. P. Mathur, *Foundations of software testing, 2/e*. Pearson Education India, 2013.
- [53] X. Devroey, G. Perrouin, A. Legay, P.-Y. Schobbens, and P. Heymans, “Search-based similarity-driven behavioural spl testing,” in *Proceedings of the*

Tenth International Workshop on Variability Modelling of Software-intensive Systems. ACM, 2016, pp. 89–96.

- [54] X. Qu, M. B. Cohen, and K. M. Woolf, “Combinatorial interaction regression testing: A study of test case generation and prioritization,” in *Software Maintenance, 2007. ICSM 2007. IEEE International Conference on.* IEEE, 2007, pp. 255–264.
- [55] H. Srikanth, M. B. Cohen, and X. Qu, “Reducing field failures in system configurable software: Cost-based prioritization,” in *Software Reliability Engineering, 2009. ISSRE’09. 20th International Symposium on.* IEEE, 2009, pp. 61–70.

Appendix A

Prioritized Test Suite for Subject Product Line

This section includes the prioritized list of test cases produced from the subject product line based on proposed criterion. Each product line produce ten prioritized list with different values of α as shown in the TABLE A.1, A.2 and A.3.

TABLE A.1: Prioritized Test Suite for E-Commerce Product Line Based on Proposed Criterion

Value for α	Prioritized Test Suite
0.1	t81, t98, t100, t83, t76, t96, t91, t70, t71, t86, t92, t72, t87, t30, t43, t88, t82, t85, t66, t99, t80, t97, t24, t78, t50, t89, t20, t36, t37, t74, t51, t21, t47, t48, t61, t67, t68, t69, t94, t32, t77, t79, t75, t26, t31, t42, t41, t62, t55, t90, t93, t73, t84, t17, t27, t22, t46, t18, t44, t29, t56, t64, t95, t25, t28, t15, t38, t34, t35, t65, t39, t49, t13, t40, t16, t12, t33, t63, t58, t57, t59, t60, t23, t19, t45, t54, t11, t52, t10, t14, t53, t9, t7, t6, t4, t5, t8, t1, t2, t3

0.2	t81, t98, t100, t83, t76, t96, t91, t70, t71, t86, t92, t72, t87, t30, t43, t85, t88, t66, t82, t99, t24, t80, t97, t78, t50, t20, t36, t37, t89, t74, t51,t21, t47, t48, t61, t67, t68,t69,t94, t32, t77, t79, t75, t26, t31, t55,t42, t41, t62, t93, t73, t90,t17,t27, t84, t22, t46, t18, t44, t29, t56, t64, t15, t25, t28, t95, t38,t34,t35, t65, t13, t39, t49,t16,t40, t12, t33, t63, t58, t57, t59, t23, t60, t19, t45, t54, t11,t52, t10,t14, t53, t9, t7, t6, t4, t5, t8,t1, t2, t3
0.3	t81, t98, t100, t83,t76, t96, t91, t70, t71,t86, t92, t72, t87, t30, t85, t66, t43, t88, t24,t82, t99, t80, t78,t50, t20, t36, t37, t97, t74, t51, t21, t47, t48, t61,t89, t26, t67,t68, t69, t94, t32, t77, t79, t75, t55, t31, t42, t41, t62,t17, t93,t27, t22, t73, t46, t90, t18, t44, t84, t29,t15, t56, t25,t28,t64, t38, t34, t35, t65, t95, t13, t16, t39, t49, t40,t12, t33, t63, t58,t57, t59, t23, t19, t45, t54, t52, t60, t11, t10, t14, t53,t9, t7,t6, t4, t5, t8, t1, t2, t3
0.4	t81, t76, t83, t98,t100, t96, t70, t71,t86, t91, t30, t92, t72, t87,t85, t24, t66, t43 t88, t82, t78, t80, t20, t36,t37, t99, t50, t97, t74,t51, t26, t21, t47,t48, t61, t67, t68, t69, t94,t32, t55, t89, t77,t79, t75, t31, t17,t27, t42, t93, t22, t41, t62, t73,t46, t18, t44, t15,t29, t84, t90, t25,t28, t56,t64, t38, t34, t35, t65, t13, t16, t95, t39,t49, t12, t40, t33,t63, t58, t23, t57,t59, t52, t19, t45, t54, t11, t10,t14, t60, t53, t9,t7, t6, t4,t5, t1, t8, t2, t3
0.5	t81, t76, t83, t98, t100, t96, t70, t71, t86, t91, t30, t24, t92, t85, t72, t87, t66, t43, t88, t82, t20, t36, t37, t78, t80, t50, t99, t74,t26, t51, t97,t21, t47, t48, t61, t55, t67, t68, t69, t94, t32, t77, t79,t17, t75, t31,t89, t27, t22,t93, t42, t41, t62, t46, t15, t18, t44, t73,t29, t84, t25,t28, t90,t56, t64, t38, t34, t35, t16, t13, t65, t95, t39,t49, t12, t33,t40, t63, t58, t23, t57, t59, t52, t19, t45, t54, t11, t10, t14, t60, t53,t9, t7, t6, t4, t5, t1, t8, t2, t3
0.6	t81, t76, t83, t96 , t98, t100, t70, t71, t86, t24, t30, t91, t85, t92, t66, t72, t87, t43, t88, t20, t36, t37,t78, t82, t80, t74, t50, t26, t99, t51,t55, t21, t47, t48, t61, t97,t67, t68, t69, t94, t17, t32, t77, t79, t27,t31, t75, t22, t89, t93, t15,t46, t42, t18, t44, t41, t62, t73, t29, t25,t28, t84, t56,t90, t16, t38,t64, t34, t35, t13, t65, t12, t95, t33, t39,t49, t63, t40,t23, t58, t52,t57, t59, t19, t45, t10, t14, t11, t54, t60,t53, t9, t7, t6, t4, t1, t5, t8, t2, t3

0.7	t81, t76, t83, t96, t70, t71, t86, t98, t100, t24, t30, t85, t91, t66, t92,t20, t36, t37, t72, t87, t43, t88, t78, t82, t26, t74, t50, t80, t55,t51, t17, t21, t47, t48, t61, t99, t67, t68, t69, t94, t27, t97, t32,t22, t31, t77, t79, t75, t15, t93,t46, t89, t18, t44, t42, t41, t62,t73, t29, t25, t28, t84, t16, t38, t56,t13, t34, t35, t64, t90, t65,t12, t33, t63, t39, t49, t95, t40, t23, t52,t58, t10, t14, t57, t59,t19, t45, t11, t54, t53, t60, t9, t7, t6, t4,t1, t5, t2, t8, t3
0.8	t81, t76, t83, t96, t70, t71,t86, t24, t98, t100, t30, t85, t66, t91, t20, t36,t37, t92, t43, t72, t87,t78, t88, t26, t74, t82, t55, t50, t17, t80, t51,t21, t47, t48, t61, t27,t67, t68, t69, t94, t99, t22, t32, t15, t31, t97,t77, t79, t75, t46, t93,t18, t44, t89, t42, t41, t62, t29, t73, t25, t28,t16, t84, t38, t56, t13,t34, t35, t64, t65, t90, t12, t33, t63, t39, t49,t95, t40, t23, t52, t10,t14, t58, t57, t59, t19, t45, t11, t54, t53, t9, t60, t7, t6, t4, t1, t5, t2, t8, t3
0.9	t81, t76, t83, t96, t24, t70, t71, t86, t98, t100, t30, t85, t20, t36, t37, t66,t91, t92, t43, t78, t72,t87, t26, t88, t74, t55, t17, t82, t50, t51, t80,t27, t21, t47, t48, t61,t22, t67, t68, t69, t94, t15, t32, t31, t99, t77,t79, t97, t46, t75, t93,t18, t44, t42, t89, t29, t25, t28, t41, t62, t73,t16,t38, t13, t84, t56, t34, t35, t64, t65, t90, t12, t33, t63, t39,t49, t40, t95, t23, t52, t10, t14, t58, t57, t59, t19, t45, t11, t54, t53,t7, t9, t60, t6, t4, t1, t5,t2, t8, t3
1	t81, t76, t24, t83, t70, t71,t86, t96, t30, t20, t36, t37, t85,t98, t100, t66, t26, t43, t78, t91, t92, t17, t55, t72, t74, t87, t88,t27, t50, t51,t82, t15, t21, t22, t47, t48, t61, t80, t67, t68, t69, t94,t31, t32, t46,t77, t79, t99, t18, t44, t75, t93, t97, t25, t28, t29, t42,t16, t41, t62,t73, t89, t13, t38, t34, t35, t56, t84, t64, t65, t12, t33,t90, t63, t23,t39, t40, t49, t10, t14, t52, t95, t11, t19, t45, t57, t58,t59, t54, t7,t9, t53, t6, t4, t60, t1, t5, t2, t8, t3

TABLE A.2: Prioritized Test Suite for Social Network Product Line Based on Proposed Criterion

Value for α	Prioritized Test Suite
0.1	t87, t88, t90, t82, t89, t91, t92, t83, t84, t86, t85, t100, t55, t57, t97, t79, t77, t61, t56, t58, t59, t78, t80, t81, t60, t62, t63, t94, t40, t95, t96, t71, t73, t48, t41, t42, t72, t74, t75, t49, t50, t52, t99, t76, t51, t53, t54, t98, t29, t93, t66, t30, t31, t33, t70, t67, t68, t37, t69, t32, t34,t35, t43,t45, t27,t28, t36,t38, t39,t44, t46,t47, t24,t25, t26,t64, t13,t65, t14,t15, t17,t21, t16,t18, t19,t20, t22,t23, t10,t11, t12,t8, t6,t7, t4,t5, t2,t3, t1,t9
0.2	t87,88, t90, t89, t91, t92, t82, t83, t84, t86, t85, t100, t55, t57, t97, t77, t79, t56, t58, t59, t71, t73, t40, t61, t78, t80, t81, t48, t60, t72, t74, t75, t62, t63, t94, t41, t42, t49, t50, t52, t95, t96, t99, t76, t51, t53, t54, t98, t29, t66, t67, t68, t30, t31, t33, t70, t93, t69, t32, t34,t35,t43, t37,t45, t27,t28, t36,t38, t39,t44, t24,t46, t47,t25, t26,t64, t13,t65, t14,t15, t17,t21, t16,t18, t19,t20, t22,t23, t10,t11, t12,t8, t6,t7, t4,t5, t2,t3, t1,t9
0.3	t87, t88, t90, t89, t91, t92, t82, t83, t84, t86, t85, t100, t71, t73, t55, t57, t97, t40, t77, t48, t72, t74, t75, t56, t58, t59, t79, t41, t42, t49, t50, t52, t61, t78, t99, t80, t81, t76, t51, t53, t54, t60, t98, t62, t63, t94, t66, t29, t95, t96, t67, t68, t30, t31, t33, t70, t69, t32, t34, t35,t43,t24, t37,t45, t93,t27, t28,t36, t44,t25, t26,t38, t39,t46, t47,t13, t64,t14, t15,t17, t65,t16, t18,t19, t21,t20, t22,t23, t10,t11, t12,t8, t6,t7, t4,t5, t2,t3, t1,t9
0.4	t87,88, t90,89, t91,92, t82,83, t84,86, t85, t100, t71, t73, t40,72, t74,75, t48, t55, t57, t77, t97, t41, t42, t49, t50, t52, t56, t58, t59, t99, t66, t76, t79, t51, t53, t54, t98, t78, t61, t29, t80, t81, t67, t68, t60, t62, t63, t94, t30, t31, t33, t70, t24, t95, t96, t69, t43, t32,t34,t35, t25,t26, t37,t45, t13,t44, t27,t28, t36,t46, t47,t38, t39,t93, t14,t15, t17,t64, t65,t16, t18,t19, t21,t20, t22,t23, t10,t11, t12,t8, t6,t7, t4,t5, t2,t3, t1,t9

0.5	t87, t88, t90, t89, t91, t92, t82, t83, t84, t86, t100, t85, t71, t73, t72, t74, t75, t40, t48, t49, t50, t52, t41, t42, t55, t57, t77, t66, t97, t99, t76, t51, t53, t54, t56, t58, t59, t98, t67, t68, t29, t79, t24, t78, t30, t31, t33, t61, t80, t81, t70, t25, t26, t60, t69, t43, t62, t63, t94, t32, t34,t35,t13, t95,t96, t45,t37, t44,t14, t15,t17, t27,t28, t36,t46, t47,t38, t39,t64, t16,t18, t19,t93, t65,t21, t20,t22, t23,t10, t11,t12, t8,t6, t7,t4, t5,t2, t3,t1, t9
0.6	t87, t88, t90, t89, t91, t92, t82, t100, t71, t73, t83, t84, t86, t85, t72, t74, t75, t48, t40, t66, t49, t50, t52, t41, t42, t99, t55, t57, t77, t67, t68, t51, t53, t54, t76, t24, t98, t97, t29, t56, t58, t59, t25, t26, t30, t31, t33, t70, t79, t13, t43, t78, t69, t32, t34, t35, t61, t80, t81, t60,t14,t15, t17,t62, t63,t94, t45,t37, t95,t96, t16,t18, t19,t44, t64,t27, t28,t36, t46,t47, t38,t39, t65,t21, t93,t20, t22,t23, t10,t11, t12,t8, t6,t7, t4,t5, t2,t3, t1,t9
0.7	t87, t88, t90, t89, t91, t92, t82, t71, t73, t100, t83, t84, t86, t72, t74, t75, t48, t85, t40, t66, t49, t50, t52, t24, t41, t42, t67, t68, t99, t51, t53, t54, t76, t98, t29, t77, t25, t26, t55, t57, t97, t13, t30, t31, t33, t56, t58, t59, t70, t43, t69, t32, t34, t35, t14, t15, t17, t79, t78, t80, t81,t16, t18,t19, t61,t60, t45,t64, t37,t62, t63,t94, t44,t27, t28,t36, t46,t47, t95,t96, t65,t38, t39,t21, t20,t22, t23,t93, t10,t11, t12,t8, t6,t7, t4,t5, t2,t3, t1,t9
0.8	t87, t88, t90, t89, t91, t92, t71, t73, t100, t82, t72, t74, t75, t83, t84, t86, t48, t66, t40, t24, t85, t49, t50, t52, t67, t68, t41, t42, t25, t26, t99, t51, t53, t54, t76, t29, t98, t13, t77, t55, t57, t30, t31, t33, t97, t14, t15, t17, t70, t43, t56, t58, t59, t69, t32, t34, t35, t16, t18, t19, t79, t78,t64, t80,t81, t45,t61, t37,t44, t21,t60, t65,t27, t28,t36, t46,t47, t62,t63, t94,t20, t38,t39, t22,t23, t95,t96, t93,t10, t11,t12, t8,t6, t7,t4, t5,t2, t3,t1, t9
0.9	t87, t88, t90, t89, t91, t92, t71, t73, t100, t72, t74, t75, t82, t66, t24, t48, t40, t83, t84, t86, t67, t68, t25, t26, t49, t50, t52, t41, t42, t13, t85, t99, t51, t53, t54, t29, t76, t98, t14, t15, t17, t30, t31, t33, t77, t70, t55, t57, t16, t18, t19, t43, t97, t69, t32, t34, t35, t56, t58, t59, t64, t79,t21, t78,t45, t65,t20, t37,t80, t81,t44, t61,t22, t23,t27, t28,t36, t46,t47, t60,t38, t39,t62, t63,t94, t95,t96, t93,t10, t11,t12, t8,t6, t7,t4, t5,t2, t3,t1, t9

1	t87, t88, t90, t89, t91, t92, t71, t73, t72, t74, t75, t24, t100, t66, t48, t82, t25, t26, t40, t67, t68, t13, t49, t50, t52, t83, t84, t86, t41, t42, t14, t15, t17, t99, t29, t51, t53, t54, t85, t76, t98, t16, t18, t19, t30, t31, t33, t70, t43, t77, t55, t57, t69, t32, t34, t35, t97, t56, t58, t59, t64,t21,t20, t65,t45, t79,t22, t23,t37, t44,t78, t27,t28, t36,t46, t47,t80, t81,t61, t38,t39, t60,t62, t63,t94, t95,t96, t10,t11, t12,t93, t8,t6, t7,t4, t5,t2, t3,t1, t9
----------	--

TABLE A.3: Prioritized Test Suite for Transport Network Product Line Based on Proposed Criterion

Value for α	Prioritized Test Suite
0.1	t65, t67, t66, t68, t96, t97, t58, t99, t60, t25, t27, t95, t59, t62, t98, t61, t64, t100, t26, t29, t28, t31, t63, t30, t32, t69, t71, t77, t85, t79, t70, t73, t21, t87, t72, t75, t78, t81, t20, t23, t86, t89, t90, t80, t83, t92, t74, t22, t82, t57, t76, t24, t84, t88, t33, t91, t94, t35, t93, t34, t36, t9, t38, t44, t52, t15, t46, t54, t37, t40, t10, t12, t39, t42, t45, t48, t53, t56, t11, t16, t47, t50, t55, t17, t19, t41, t13, t49, t1, t18, t43, t14, t3, t51, t2, t5, t4, t7, t6, t8
0.2	t65, t67, t58, t66, t60, t96, t25, t68, t97, t59, t62, t27, t99, t61, t64, t95, t26, t29, t63, t98, t28, t31, t100, t30, t32, t69, t71, t77, t85, t79, t70, t73, t87, t21, t72, t75, t78, t81, t86, t89, t20, t23, t90, t74, t80, t83, t22, t92, t82, t57, t76, t88, t24, t33, t84, t91, t94, t35, t93, t34, t36, t9, t38, t44, t52, t15, t46, t54, t37, t40, t10, t12, t39, t42, t45, t48, t53, t56, t11, t16, t47, t50, t55, t41, t17, t19, t13, t1, t49, t18, t43, t3, t14, t51, t2, t5, t4, t7, t6, t8

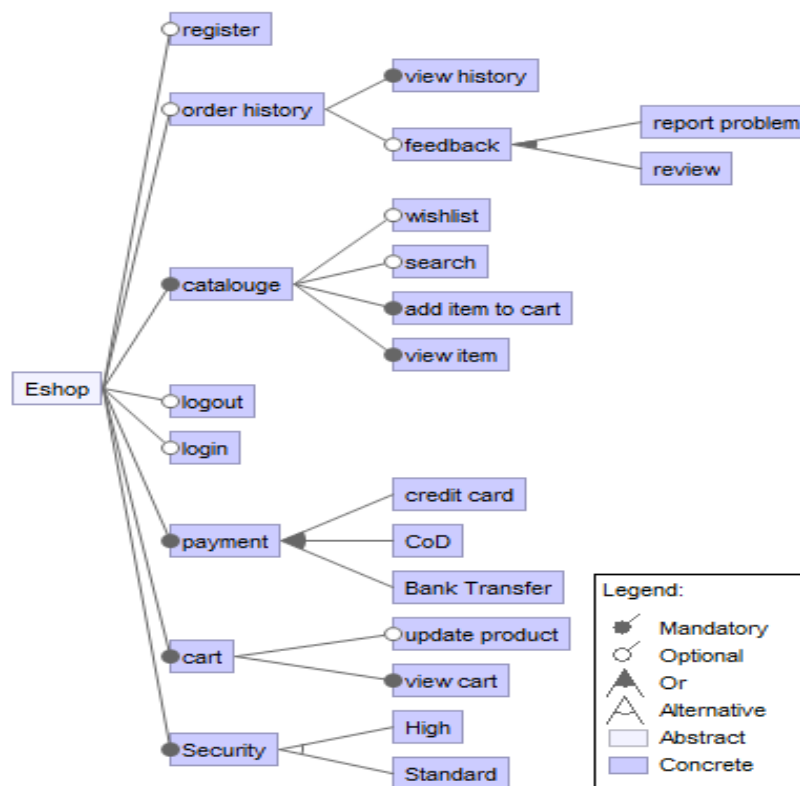
0.3	t65, t58, t67, t60, t66, t25, t59, t62, t96, t97, t27, t68, t61, t64, t63, t95, t26, t29, t99, t28, t31, t98, t30, t100, t69, t32, t71, t77, t85, t70, t73, t87, t79, t21, t72, t75, t86, t89, t78, t81, t90, t20, t23, t74, t22, t80, t83, t92, t88, t82, t33, t57, t76, t24, t91, t94, t35, t84, t93, t36, t34, t9, t38, t44, t52, t15, t46, t54, t37, t40, t10, t12, t39, t42, t45, t48, t53, t56, t11, t16, t41, t13, t47, t50, t55, t1, t17, t19, t49, t18, t43, t3, t14, t51, t2, t5, t4, t7, t6, t8
0.4	t58, t65, t60, t67, t66, t59, t62, t25, t61, t64, t27, t96, t63, t97, t68, t26, t29, t95, t99, t28, t31, t30, t98, t69, t100, t32, t71, t77, t85, t70, t73, t87, t21, t79, t72, t75, t86, t89, t78, t81, t90, t74, t20, t23, t22, t88, t80, t83, t92, t33, t82, t57, t76, t91, t94, t24, t35, t36, t9, t34, t84, t93, t38, t44, t52, t15, t37, t40, t10, t12, t46, t54, t39, t42, t45, t48, t53, t56, t11, t41, t16, t13, t1, t47, t50, t55, t17, t19, t49, t3, t18, t43, t14, t2, t5, t51, t4, t7, t6, t8
0.5	t58, t65, t60, t59, t62, t25, t67, t66, t61, t64, t63, t27, t96, t26, t29, t97, t95, t68, t28, t31, t99, t30, t98, t69, t100, t32, t71, t77, t85, t70, t73, t87, t21, t79, t72, t75, t86, t89, t78, t81, t74, t90, t20, t23, t22, t88, t33, t80, t83, t92, t82, t36, t57, t76, t91, t94, t35, t9, t24, t34, t38, t93, t44, t52, t84, t37, t40, t15, t10, t12, t46, t54, t39, t42, t45, t48, t53, t56, t41, t11, t16, t13, t1, t47, t50, t55, t49, t17, t19, t3, t18, t43, t14, t2, t5, t51, t4, t7, t6, t8
0.6	t58, t65, t60, t59, t62, t25, t67, t66, t61, t64, t63, t27, t26, t29, t96, t97, t95, t68, t28, t31, t30, t99, t69, t98, t71, t32, t100, t77, t85, t70, t73, t87, t21, t79, t72, t75, t86, t89, t74, t78, t81, t90, t20, t23, t88, t22, t33, t36, t80, t83, t92, t82, t9, t91, t94, t57, t76, t35, t38, t34, t24, t44, t52, t37, t40, t93, t15, t10, t12, t84, t46, t54, t39, t42, t45, t48, t53, t56, t41, t11, t16, t13, t1, t47, t50, t55, t49, t17, t19, t3, t18, t43, t2, t5, t14, t51, t4, t7, t6, t8

0.7	t58, t60, t65, t59, t62, t25, t67, t66, t61, t64, t63, t27, t26, t29, t96, t95, t97, t68, t28, t31, t30, t69, t99, t98, t71, t70, t73, t32, t77, t85, t87, t100, t21, t72, t75, t79, t86, t89, t74, t78, t81, t90, t88, t20, t23, t22, t33, t36, t9, t80, t83, t92, t82, t91, t94, t35, t57, t76, t38, t34, t44, t52, t37, t40, t24, t15, t10, t12, t93, t84, t46, t54, t39, t42, t41, t45, t48, t53, t56, t11, t13, t1, t16, t47, t50, t55, t49, t3, t17, t19, t18, t43, t2, t5, t14, t4, t7, t51, t6, t8
0.8	t58, t60, t59, t62, t65, t25, t61, t64, t63, t67, t66, t27, t26, t29, t96, t95, t97, t28, t31, t30, t68, t69, t99, t98, t71, t70, t73, t87, t77, t85, t32, t21, t100, t72, t75, t86, t89, t74, t79, t88, t78, t81, t90, t20, t23, t36, t22, t33, t9, t80, t83, t92, t82, t91, t94, t38, t37, t40, t35, t44, t52, t57, t76, t34, t10, t12, t15, t24, t93, t39, t42, t46, t54, t41, t45, t48, t53, t56, t84, t11, t13, t1, t16, t47, t50, t55, t49, t3, t17, t19, t2, t5, t18, t43, t14, t4, t7, t6, t51, t8
0.9	t58, t60, t59, t62, t65, t25, t61, t64, t63, t67, t66, t27, t26, t29, t96, t95, t97, t28, t31, t30, t69, t68, t99, t98, t71, t70, t73, t87, t77, t85, t32, t21, t100, t72, t75, t86, t89, t74, t88, t79, t36, t78, t81, t90, t20, t23, t22, t33, t9, t38, t80, t83, t92, t82, t37, t40, t91, t94, t44, t52, t35, t34, t57, t76, t10, t12, t15, t24, t39, t42, t93, t41, t46, t54, t45, t48, t53, t56, t11, t13, t1, t84, t16, t47, t50, t55, t49, t3, t2, t5, t17, t19, t18, t43, t14, t4, t7, t6, t51, t8
1	t58, t59, t60, t62, t65, t25, t61, t63, t64, t66, t67, t26, t27, t29, t95, t96, t28, t30, t31, t97, t69, t68, t98, t99, t70, t71, t73, t87, t77, t85, t21, t32, t100, t36, t72, t74, t75, t86, t88, t89, t78, t79, t81, t90, t9, t20, t22, t23, t33, t37, t38, t40, t44, t52, t80, t82, t83, t91, t92, t94, t10, t12, t34, t35, t15, t57, t76, t24, t39, t41, t42, t45, t46, t48, t53, t54, t56, t93, t1, t11, t13, t16, t84, t47, t49, t50, t55, t2, t3, t5, t17, t18, t19, t43, t14, t4, t6, t7, t51, t8

Appendix B

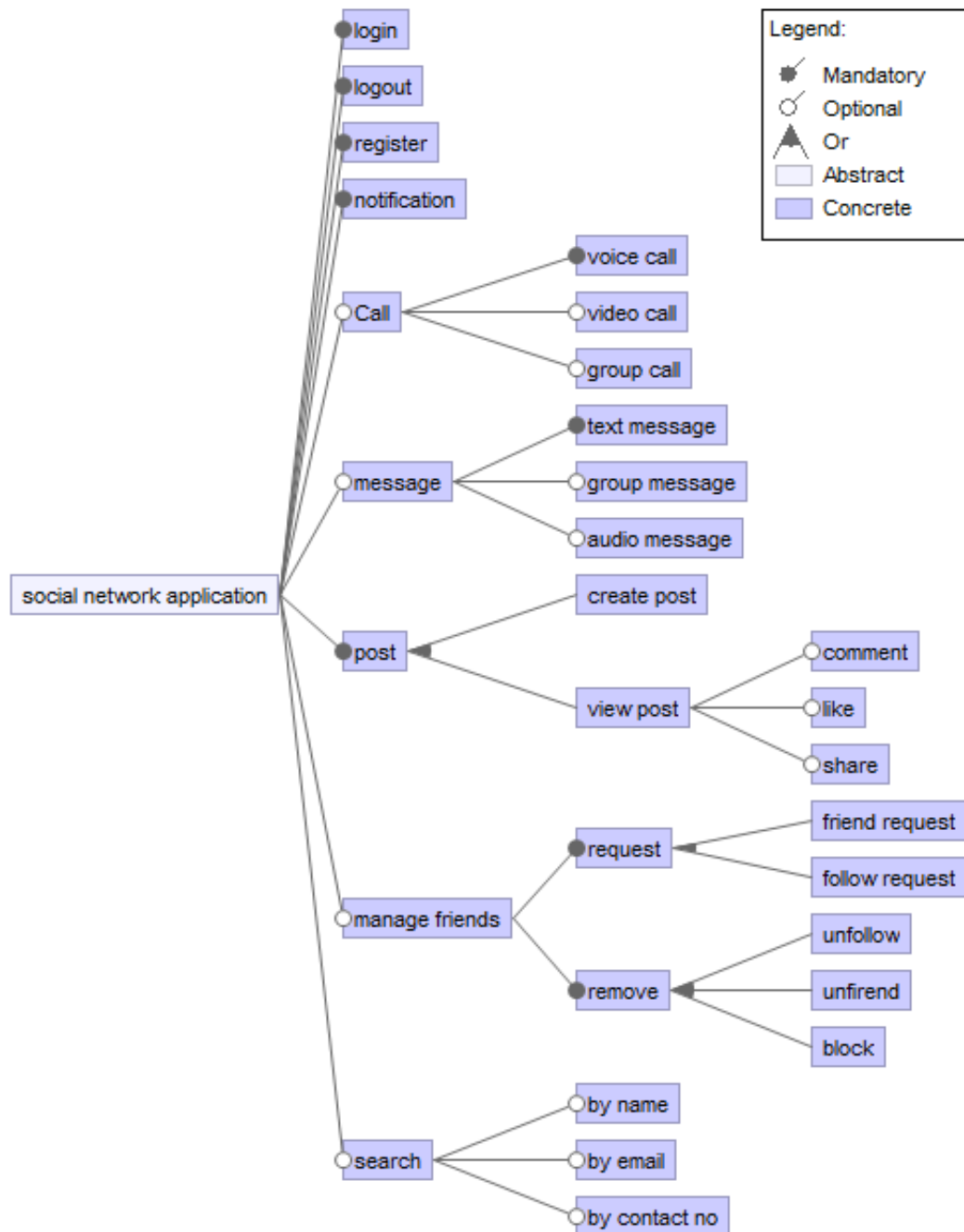
Subject Product Line

This section includes the Feature Model that are used in the research work.



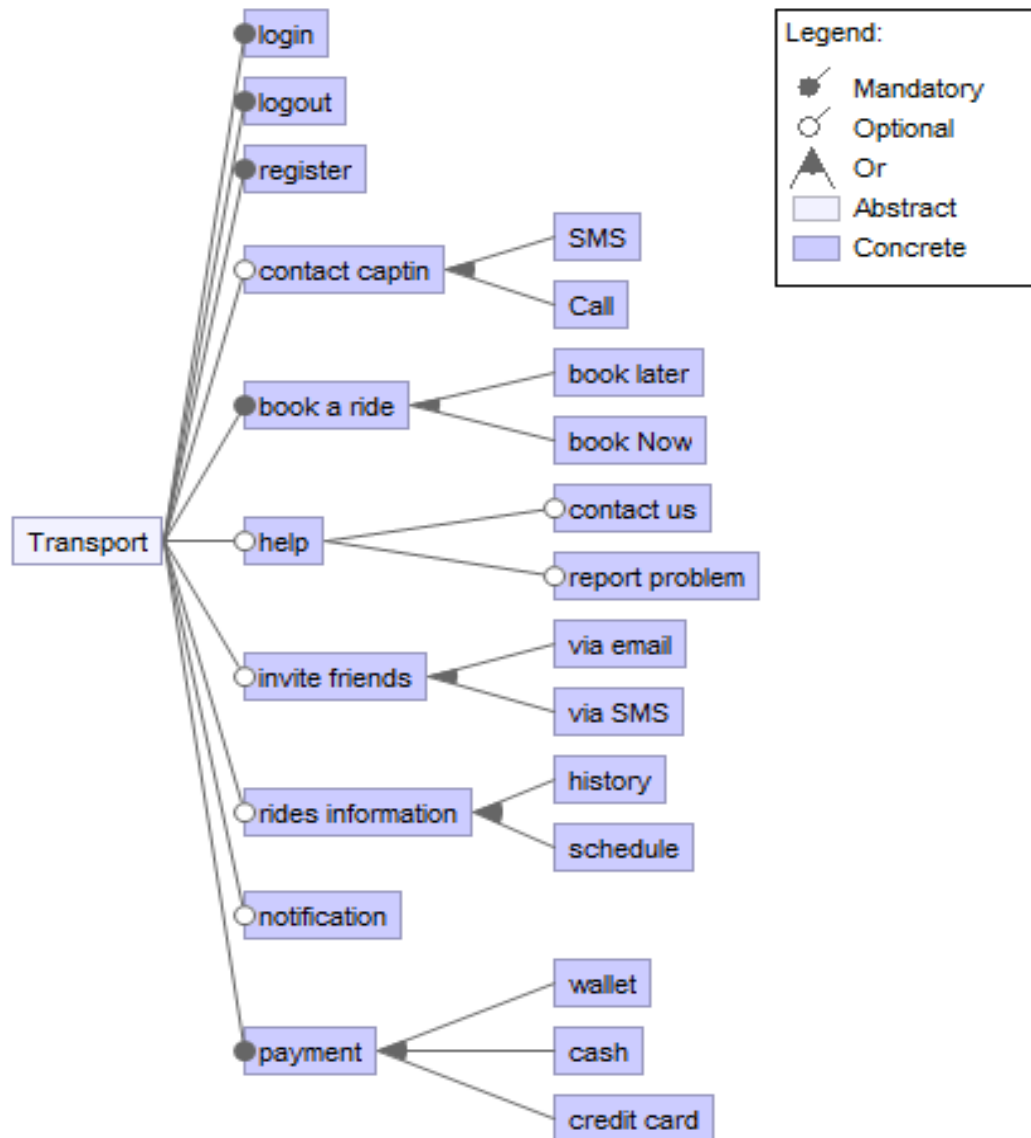
"order history" ⇒ login
wishlist ⇒ login
login ⇒ register
logout ⇒ login
¬("credit card" ∧ Standard)
¬("Bank Transfer" ∧ Standard)

FIGURE B.1: E-Commerce Feature Model.



unfollow ⇒ "follow request"
 unfriend ⇒ "friend request"
 block ⇒ "friend request"

FIGURE B.2: Social Network Feature Model.



"book Now" ⇒ "contact captin"
 "book Now" ⇒ notification
 "book later" ⇒ schedule
 "credit card" ⇒ history
 wallet ⇒ history

FIGURE B.3: Transport Network Feature Model.

Appendix C

Test Suite for Subject Product Line

This section includes the test cases that are generated from the Feature Model of subject product lines. Each test case consists of number of features in it. Each feature of E-commerce, social network and transport network product line is represented with the alphanumeric data represented in TABLE C.1, C.3 and C.5 and test suites are illustrated in TABLE C.2, C.4 and C.6 respectively.

TABLE C.1: Alphanumeric Character for Features of E-Commerce Product Line

Character	Features	Character	Features
f1	Eshop	f13	search
f2	register	f14	add item to cart
f3	order history	f15	view item
f4	catalogue	f16	credit card
f5	logout	f17	CoD
f6	login	f18	Bank Transfer
f7	payment	f19	update product

f8	cart	f20	view cart
f9	security	f21	High
f10	view history	f22	Standard
f11	feedback	f23	report problem
f12	wishlist	f24	review

TABLE C.2: E-Commerce Product Line Test Suite

Test Case	Features
t1	f4, f14, f15, f7, f16, f17, f18, f8, f20, f9,f21
t2	f4, f14, f15, f7, f16, f18, f8, f20, f9, f21
t3	f4, f14, f15, f7, f18, f8, f20, f9, f21
t4	f4, f14, f15, f7, f16, f17, f8, f19, f20, f9, f21
t5	f4, f14, f15, f7, f17, f8, f19, f20, f9, f22
t6	f4, f13, f14, f15, f7, f16, f17, f18, f8, f20, f9, f21
t7	f1,f4, f13, f14, f15, f7, f17, f18, f8, f19, f20, f9, f21
t8	f1,f4, f13, f14, f15, f7, f18, f8, f20, f9, f21
t9	f1,f2, f4, f13, f14, f15, f7, f17, f8, f19, f20, f9, f21
t10	f1,f2, f4, f14, f15, f6, f7, f16, f18, f8, f19, f20, f9, f21
t11	f1,f2, f4, f13, f14, f15, f6, f7, f17, f8, f19, f20, f9, f21
t12	f1,f2, f4, f12, f13, f14, f15, f6, f7, f16, f18, f8, f20, f9, f21
t13	f1,f2, f4, f12, f13, f14, f15, f6, f7, f17, f8, f19, f20, f9, f21
t14	f1,f2, f3, f10, f4, f14, f15, f6, f7, f16, f17, f8, f20, f9, f21,
t15	f1,f2, f3, f10, f4, f12, f14, f15, f6, f7, f16, f17, f18, f8, f20, f9, f21
t16	f1, f2, f3, f10, f4, f12, f14, f15, f6, f7, f16, f17, f8, f20, f9, f21
t17	f1,f2,f3, f10,f4,f13, f14, f15,f6, f7,f16, f17, f18, f8, f19,f20, f9, f21
t18	f1,f2, f3, f10, f4, f13, f14, f15, f6, f7, f16, f18, f8, f19, f20, f9, f21
t19	f1,f2, f3, f10, f4, f13, f14, f15, f6, f7, f17, f8, f20, f9, f22
t20	f1,f2, f3, f10, f4,f12, f13, f14, f15, f6,f7, f17, f18, f8,f19, f20,f9, f21
t21	f1,f2, f3, f10, f4, f12, f13, f14, f15, f6, f7, f16, f8, f19, f20, f9, f21

t22	f1,f2, f3, f10, f11, f23, f24, f4, f14, f15, f6, f7, f16, f18, f8, f19, f20, f9, f21
t23	f1,f2, f3, f10, f11, f23, f24, f4, f14, f15, f6, f7, f17, f8, f20, f9, f22
t24	f1,f2, f3,f10, f11,f23, f24,f4, f12,f14, f15,f6, f7,f16, f18,f8, f19,f20, f9, f21
t25	f1,f2,f3,f10, f11, f23, f24, f4,f12, f14, f15, f6, f7, f17, f8, f20, f9,f21
t26	f1,f2, f3, f10, f11, f23, f24, f4, f12, f14, f15, f6, f7, f17, f8, f19, f20, f9, f22
t27	f1,f2, f3, f10, f11, f23, f24, f4,f13, f14,f15, f6,f7, f16,f17, f18,f8,f20, f9,f21
t28	f1,f2, f3, f10, f11, f23, f24, f4, f13, f14, f15, f6, f7, f17, f18, f8, f20, f9, f21
t29	f1,f2, f3, f10, f11, f23, f24, f4, f13, f14, f15, f6, f7, f16, f18, f8, f20, f9, f21
t30	f1,f2, f3, f10, f11, f23, f24, f4,f12, f13, f14, f15, f6, f7, f18, f8,19, f20, f9,21
t31	f1, f2, f3, f10, f11, f23, f24, f4, f12, f13, f14, f15, f6, f7, f17, f8, f20, f9, f21
t32	f1, f2, f3, f10, f11, f23, f24, f4, f12, f13, f14, f15, f6, f7, f16, f8, f20, f9, f21
t33	f1, f2, f3, f10, f11, f24, f4, f14, f15, f6, f7, f16, f18, f8, f20, f9, f21
t34	f1, f2, f3, f10, f11, f24, f4, f14, f15, f6, f7, f18, f8, f19, f20, f9, f21
t35	f1, f2, f3, f10, f11, f24, f4, f14, f15, f6, f7, f16, f8, f19, f20, f9, f21
t36	f1, f2, f3, f10, f11, f24, f4, f12, f14, f15, f6, f7, f17, f18, f8, f19, f20, f9, f21
t37	f1, f2, f3, f10, f11, f24, f4, f12, f14, f15, f6, f7, f16, f17, f8, f19, f20, f9, f21
t38	f1, f2, f3, f10, f11, f24, f4, f12, f14, f15, f6, f7, f17, f8, f20, f9, f21
t39	f1, f2, f3, f10, f11, f24, f4, f13, f14, f15, f6, f7, f18, f8, f20, f9, f21
t40	f1, f2, f3, f10, f11, f24, f4, f13, f14, f15, f6, f7, f17, f8, f20, f9, f21
t41	f1, f2, f3, f10, f11, f24, f4, f13, f14, f15, f6, f7, f16, f8, f19, f20, f9, f21
t42	f1, f2, f3, f10, f11, f24, f4, f12, f13, f14, f15, f6, f7, f17, f8, f20, f9, f21
t43	f1, f2, f3, f10, f11, f24, f4, f12, f13, f14, f15, f6, f7, f16, f8, f19, f20, f9, f21
t44	f1, f2, f3, f10, f11, f23, f4, f14, f15, f6, f7, f16, f18, f8, f19, f20, f9, f21
t45	f1, f2, f3, f10, f11, f23, f4, f14, f15, f6, f7, f17, f8, f20, f9, f21
t46	f1, f2, f3, f10, f11, f23, f4, f12, f14, f15, f6, f7, f16, f17, f8, f20, f9, f21
t47	f1, f2, f3, f10, f11, f23, f4, f12, f14, f15, f6, f7, f16, f8, f19, f20, f9, f21
t48	f1, f2, f3, f10, f11, f23, f4, f13, f14, f15, f6, f7, f16, f18, f8, f19, f20, f9, f21
t49	f1, f2, f3, f10, f11, f23, f4, f13, f14, f15, f6, f7, f16, f8, f20, f9, f21
t50	f1, f2, f3, f10, f11, f23, f4, f12, f13, f14, f15, f6, f7, f16, f18, f8, f20, f9, f21
t51	f1,f2, f3, f10, f11, f23, f4, f12, f13, f14, f15, f6, f7, f16, f17, f8, f20, f9, f21

t52	f1,f2, f4, f14, f15, f5, f6, f7, f16, f17, f18, f8, f20, f9, f21
t53	f1,f2, f4, f14, f15, f5, f6, f7, f16, f17, f8, f20, f9, f21
t54	f1,f2, f4, f14, f15, f5, f6, f7, f17, f8, f19, f20, f9, f21
t55	f1,f2, f4, f12, f14, f15, f5, f6, f7, f16, f17, f18, f8, f19, f20, f9, f21
t56	f1,f2, f4, f12, f14, f15, f5, f6, f7, f18, f8, f19, f20, f9, f21
t57	f1,f2, f4, f12, f14, f15, f5, f6, f7, f17, f8, f20, f9, f22
t58	f1,f2, f4, f13, f14, f15, f5, f6, f7, f16, f18, f8, f20, f9, f21
t59	f1,f2, f4, f13, f14, f15, f5, f6, f7, f16, f17, f8, f20, f9, f21
t60	f1,f2, f4, f13, f14, f15, f5, f6, f7, f16, f8, f20, f9, f21
t61	f1,f2, f4, f12, f13, f14, f15, f5, f6, f7, f17, f18, f8, f19, f20, f9, f21
t62	f1,f2, f4, f12, f13, f14, f15, f5, f6, f7, f17, f8, f19, f20, f9, f22
t63	f1,f2, f3, f10, f4, f14, f15, f5, f6, f7, f17, f18, f8, f20, f9, f21
t64	f1,f2, f3, f10, f4, f14, f15, f5, f6, f7, f18, f8, f19, f20, f9, f21
t65	f1,f2, f3, f10, f4, f14, f15, f5, f6, f7, f17, f8, f19, f20, f9, f22
t66	f1,f2, f3, f10, f4, f12, f14, f15, f5, f6, f7, f16, f18, f8, f19, f20, f9, f21
t67	f1,f2, f3, f10, f4, f12, f14, f15, f5, f6, f7, f17, f8, f19, f20, f9, f21
t68	f1,f2, f3, f10, f4, f13, f14, f15, f5, f6, f7, f17, f18, f8, f19, f20, f9, f21
t69	f1,f2, f3, f10, f4, f13, f14, f15, f5, f6, f7, f16, f17, f8, f19, f20, f9, f21
t70	f1,f2, f3, f10, f4, f12, f13, f14, f15, f5, f6, f7, f17, f18, f8, f19, f20, f9, f21
t71	f1,f2, f3, f10, f4, f12, f13, f14, f15, f5, f6, f7, f16, f17, f8, f19, f20, f9, f21
t72	f1,f2, f3, f10, f4, f12, f13, f14, f15, f5, f6, f7, f16, f8, f19, f20, f9, f21
t73	f1,f2, f3, f10, f11, f23, f24, f4, f14, f15, f5, f6, f7, f16, f18, f8, f20, f9, f21
t74	f1,f2, f3, f10, f11, f23, f24, f4,f14, f15, f5,f6, f7,f16, f17,f8, f19,f20, f9,f21
t75	f1,f2, f3, f10, f11, f23, f24, f4, f14, f15, f5, f6, f7, f16, f8, f19, f20, f9, f21
t76	f1,f2, f3,f10, f11,f23, f24,f4, f12,f14, f15,f5, f6,f7, f16,f18, f8,f19, f20,f9, f21
t77	f1,f2, f3, f10, f11, f23, f24, f4, f12, f14, f15, f5, f6, f7, f17, f8, f20, f9, f22
t78	f1,f2, f3, f10, f11,f23, f24,f4, f13,f14, f15,f5, f6,f7, f16,f17, f18,f8, f20,f9, f21
t79	f1,f2, f3, f10, f11, f23,f24, f4,f13, f14,f15, f5,f6, f7,f16, f17,f8, f20, f9, f21
t80	f1,f2, f3, f10, f11, f23, f24, f4,f13, f14,f15, f5,f6, f7,f17, f8,f19, f20, f9,f21
t81	f1,f2, f3,f10, f11, f23, f24, f4,f12, f13,f14, f15,f5,f6,f7, f17,f18, f8, f19, f20

t82	f1,f2, f3, f10, f11, f23,f24, f4,f12, f13, f14, f15,f5,f6, f7,f17, f8,f20, f9,f21
t83	f1,f2, f3,f10, f11,f23,f24,f4, f12,f13, f14,f15, f5,f6, f7,f17, f8,f19, f20,f9, f22
t84	f1,f2, f3, f10, f11, f24, f4, f14, f15, f5, f6, f7, f16, f18, f8, f20, f9, f21
t85	f1,f2, f3, f10, f11, f24, f4, f12, f14, f15, f5,f6,f7,f16, f17, f18, f8,f20, f9, f21
t86	f1,f2, f3,f10, f11, f24, f4,f12, f14, f15, f5,f6, f7,f17, f18, f8,f19, f20, f9, f21
t87	f1,f2, f3, f10, f11, f24, f4, f12, f14, f15, f5, f6, f7, f18, f8, f19, f20, f9, f21
t88	f1,f2, f3, f10, f11, f24, f4, f12, f14, f15, f5, f6, f7, f17, f8, f19, f20, f9, f22
t89	f1,f2, f3, f10, f11, f24, f4, f13, f14, f15, f5, f6, f7, f16, f18, f8, f20, f9, f21
t90	f1,f2, f3, f10, f11, f24, f4, f13, f14, f15, f5, f6, f7, f17, f8, f20, f9, f22
t91	f1,f2, f3, f10, f11, f24, f4,f12, f13, f14, f15, f5,f6, f7,f16, f18,f8, f20,f9, f21
t92	f1,f2, f3,f10, f11, f24, f4,f12, f13, f14, f15, f5,f6,f7, f16, f17, f8,f20, f9,f21
t93	f1,f2, f3, f10, f11, f23, f4, f14, f15, f5, f6, f7, f16, f17, f18, f8, f20, f9, f21
t94	f1,f2, f3, f10, f11, f23, f4, f14, f15, f5, f6, f7, f17, f18, f8, f19, f20, f9, f21
t95	f1,f2, f3, f10, f11, f23, f4, f14, f15, f5, f6, f7, f16, f8, f20, f9, f21
t96	f1,f2, f3,f10, f11, f23, f4,f12, f14, f15, f5,f6, f7,f16, f18, f8,f19, f20, f9, f21
t97	f1,f2,f3, f10, f11, f23, f4, f13,f14, f15, f5,f6, f7,f17, f8,f19, f20, f9, f22
t98	f1,f2, f3, f10, f11, f23, f4,f12, f13, f14,f15, f5,f6, f7,f18, f8,f19, f20, f9, f21
t99	f1,f2, f3, f10, f11, f23, f4, f12, f13, f14, f15, f5, f6, f7, f17, f8, f20, f9, f22
t100	f1,f2, f3, f10, f11, f23, f4,f12, f13, f14, f15, f5,f6, f7,f16, f8, f19, f20, f9,f21

TABLE C.3: Alphanumeric Character for Features of Social Network Product Line

Character	Features	Character	Features
f1	social	f17	text message
f2	login	f18	group message
f3	logout	f19	audio message
f4	register	f20	manage friends
f5	notification	f21	request
f6	post	f22	friend request
f7	create post	f23	follow request

f8	view post	f24	remove
f9	comment	f25	unfollow
f10	like	f26	unfriend
f11	share	f27	block
f12	call	f28	search
f13	voice call	f29	by name
f14	video call	f30	by email
f15	group call	f31	by contact no
f16	message		

TABLE C.4: Social Network Product Line Test Suite

Test Case	Features
t1	f1, f2, f3, f4, f5, f6, f7, f8
t2	f1, f2, f3, f4, f5, f6, f7, f8, f9
t3	f1, f2, f3, f4, f5, f6, f7, f8, f10
t4	f1, f2, f3, f4, f5, f6, f7, f8, f9, f10
t5	f1, f2, f3, f4, f5, f6, f7, f8, f11
t6	f1, f2, f3, f4, f5, f6, f7, f8, f9, f11
t7	f1, f2, f3, f4, f5, f6, f7, f8, f10, f11
t8	f1, f2, f3, f4, f5, f6, f7, f8, f9, f10, f11
t9	f1, f2, f3, f4, f5, f6, f8
t10	f1, f2, f3, f4, f5, f12, f13, f15, f6, f7, f20, f21, f22, f24, f26, f27
t11	f1, f2, f3, f4, f5, f12, f13, f15, f6, f7, f20, f21, f22, f24, f27
t12	f1, f2, f3, f4, f5, f12, f13, f15, f6, f7, f20, f21, f22, f24, f26
t13	f1,f2,f3,f4, f5, f12, f13, f14, f15,f6,f7,f8, f20, f21, f22, f23, f24, f25, f26, f27
t14	f1, f2, f3, f4, f5, f12, f13, f14, f15, f6, f7, f8, f20, f21, f22, f23, f24, f26, f27

t15	f1, f2,f3, f4,f5,f12, f13, f14, f15,f6, f7,f8, f20, f21, f22, f23, f24, f25,f27
t16	f1,f2,f3, f4, f5, f12, f13, f14, f15, f6,f7,f8,f20, f21, f22, f23, f24, f27
t17	f1, f2, f3, f4, f5, f12, f13, f14, f15,f6,f7,f8, f20, f21, f22, f23, f24, f25, f26
t18	f1,f2,f3,f4,f5, f12, f13, f14, f15, f6,f7,f8, f20, f21, f22, f23, f24, f26
t19	f1,f2,f3,f4,f5, f12,f13,f14, f15, f6,f7, f8, f20, f21, f22, f23, f24, f25
t20	f1, f2, f3, f4, f5, f12, f13, f14, f15, f6, f7, f8, f20, f21, f23, f24, f25
t21	f1, f2, f3, f4,f5,f12,f13, f14, f15, f6,f7, f8, f20, f21, f22,f24,f26, f27
t22	f1, f2, f3, f4, f5, f12, f13, f14, f15, f6, f7, f8, f20, f21, f22, f24, f27
t23	f1, f2, f3, f4, f5, f12, f13, f14, f15, f6, f7, f8, f20, f21, f22, f24, f26
t24	f1,f2,f3,f4,f5,f12,f13,f14, f15,f6, f7,f8, f9, f20, f21, f22, f23,f24, f25,f26,f27
t25	f1, f2,f3,f4,f5,f12,f13,f14,f15, f6, f7, f8, f9, f20, f21, f22, f23, f24, f26, f27
t26	f1, f2, f3, f4, f5,f12,f13,f14,f15, f6, f7,f8,f9,f20, f21, f22, f23, f24, f25, f27
t27	f1,f2,f3,f4,f5,f12, f13, f15, f16, f17, f18, f19, f6,f8,f9, f20, f21, f22, f24, f27
t28	f1, f2,f3,f4,f5,f12,f13, f15, f16, f17, f18,f19, f6, f8, f9, f20, f21, f22, f24, f26
t29	f1, f2, f3, f4, f5, f12, f13, f15, f16, f17, f18,f19,f6,f8,f10,f20,f21,f22,f23, f24, f25, f26, f27
t30	f1, f2, f3, f4, f5, f12, f13, f15, f16, f17, f18, f19, f6, f8, f10, f20, f21, f22, f23, f24, f26, f27
t31	f1, f2, f3, f4, f5, f12, f13, f15, f16, f17, f18, f19, f6, f8, f10, f20, f21, f22, f23, f24, f25, f27
t32	f1, f2, f3, f4, f5, f12, f13, f15, f16, f17, f18, f19, f6, f8, f10, f20, f21, f22, f23, f24, f27
t33	f1,f2, f3, f4,f5,f12, f13,f15, f16, f17, f18, f19, f6, f8, f10, f20, f21, f22, f23, f24, f25,f26
t34	f1, f2,f3,f4,f5,f12,f13,f15,f16,f17,f18,f19,f6, f8,f10, f20, f21,f22,f23,f24,f26
t35	f1,f2,f3,f4,f5,f12,f13,f15,f16,f17,f18,f19,f6,f8,f10,f20,f21,f22,f23,f24,f25
t36	f1,f2,f3,f4,f5,f12,f13,f15,f16,f17,f18,f19,f6,f8,f10,f20,f21,f23,f24,f25
t37	f1,f2,f3,f4,f5,f12,f13,f15,f16,f17,f18,f19,f6,f8,f10,f20,f21,f22,f24,f26,f27
t38	f1,f2,f3,f4,f5,f12,f13,f15,f16,f17,f18,f19,f6, f8,f10,f20,f21,f22,f24,f27
t39	f1,f2,f3,f4,f5,f12,f13,f15,f16,f17,f18,f19,f6,f8,f10, f20,f21,f22,f24,f26

t40	f1,f2,f3,f4,f5,f12,f13,f15,f16,f17,f18,f19,f6,f8,f9,f10,f20,f21,f22,f23,f24, f25, f26,f27
t41	f1,f2,f3,f4,f5,f12,f13,f15,f16,f17,f18,f19,f6,f8,f9,f10,f20,f21,f22,f23,f24, f26, f27
t42	f1,f2,f3,f4,f5,f12,f13,f15,f16,f17,f18,f19,f6,f8,f9,f10,f20,f21,f22,f23,f24, f25, f27
t43	f1,f2,f3,f4,f5,f12,f13,f6,f7,f8,f10,f11,f20,f21,f22,f23,f24,f25,f28,f29,f31
t44	f1,f2,f3,f4,f5,f12,f13,f6,f7,f8,f10,f11,f20,f21,f23,f24,f25,f28,f29,f31
t45	f1,f2,f3,f4,f5,f12,f13,f6,f7,f8,f10,f11,f20,f21,f22,f24,f26,f27,f28,f29,f31
t46	f1,f2,f3,f4,f5,f12,f13,f6,f7,f8,f10,f11,f20,f21,f22,f24,f27,f28,f29,f31
t47	f1,f2,f3,f4,f5,f12,f13,f6,f7,f8,f10,f11,f20,f21,f22,f24,f26,f28,f29,f31
t48	f1,f2,f3,f4,f5,f12,f13,f6,f7,f8,f9,f10,f11,f20,f21,f22,f23,f24,f25,f26,f27, f28, f29, f31
t49	f1,f2,f3,f4,f5,f12,f13,f6,f7,f8,f9,f10,f11,f20,f21,f22,f23,f24,f26,f27,f28,f29,f31
t50	f1,f2,f3,f4,f5,f12,f13,f6,f7,f8,f9,f10,f11,f20,f21,f22,f23,f24,f25,f27,f28,f29,f31
t51	f1,f2,f3,f4,f5,f12,f13,f6,f7,f8,f9,f10,f11,f20,f21,f22,f23,f24,f27,f28,f29,f31
t52	f1,f2,f3,f4,f5,f12,f13,f6,f7,f8,f9,f10,f11,f20,f21,f22,f23,f24,f25,f26,f28,f29,f31
t53	f1,f2,f3,f4,f5,f12,f13,f6,f7,f8,f9,f10,f11,f20,f21,f22,f23,f24,f26,f28,f29,f31
t54	f1,f2,f3,f4,f5,f12,f13,f6,f7,f8,f9,f10,f11,f20,f21,f22,f23,f24,f25,f28,f29,f31
t55	f1,f2,f3,f4,f5,f12,f13,f14,f6,f8,f9,f10,f11,f20,f21,f22,f23,f24,f25,f27,f28, f30, f31
t56	f1,f2,f3,f4,f5,f12,f13,f14,f6,f8,f9,f10,f11,f20,f21,f22,f23,f24,f27, f28, f30, f31
t57	f1,f2,f3,f4,f5,f12,f13,f14,f6,f8,f9,f10,f11,f20,f21,f22,f23,f24,f25,f26,f28, f30, f31
t58	f1,f2,f3,f4,f5,f12,f13,f14,f6,f8,f9,f10,f11,f20,f21,f22,f23,f24,f26,f28,f30,f31
t59	f1,f2,f3,f4,f5,f12,f13,f14,f6,f8,f9,f10,f11,f20,f21,f22,f23,f24,f25,f28,f30,f31
t60	f1,f2,f3,f4,f5,f12,f13,f14,f6,f8,f9,f10,f11,f20,f21,f23,f24,f25,f28,f30,f31
t61	f1,f2,f3,f4,f5,f12,f13,f14,f6,f8,f9,f10,f11,f20,f21,f22,f24,f26,f27,f28,f30,f31
t62	f1,f2,f3,f4,f5,f12,f13,f14,f6,f8,f9,f10,f11,f20,f21,f22,f24,f27,f28,f30,f31

t63	f1,f2,f3,f4,f5,f12,f13,f14,f6,f8,f9,f10,f11,f20,f21,f22,f24,f26,f28,f30,f31
t64	f1,f2,f3,f4,f5,f12,f13,f14,f6,f7,f20,f21,f22,f23,f24,f25,f26,f27,f28,f30,f31
t65	f1,f2,f3,f4,f5,f12,f13,f14,f6,f7,f20,f21,f22,f23,f24,f26,f27,f28,f30,f31
t66	f1,f2,f3,f4,f5,f12,f13,f15,f6,f7,f8,f11,f20,f21,f22,f23,f24,f25,f26,f28,f29,f30
t67	f1,f2,f3,f4,f5,f12,f13,f15,f6,f7,f8,f11,f20,f21,f22,f23,f24,f26,f28,f29,f30
t68	f1,f2,f3,f4,f5,f12,f13,f15,f6,f7,f8,f11,f20,f21,f22,f23,f24,f25,f28,f29,f30
t69	f1,f2,f3,f4,f5,f12,f13,f15,f6,f7,f8,f11,f20,f21,f23,f24,f25,f28,f29,f30
t70	f1,f2,f3,f4,f5,f12,f13,f15,f6,f7,f8,f11,f20,f21,f22,f24,f26,f27,f28,f29,f30
t71	f1,f2,f3,f4,f5,f12,f13,f15,f6,f7,f8,f9,f11,f20,f21,f22,f23,f24,f25,f27,f28,f29,f30
t72	f1,f2,f3,f4,f5,f12,f13,f15,f6,f7,f8,f9,f11,f20,f21,f22,f23,f24,f27,f28,f29,f30
t73	f1,f2,f3,f4,f5,f12,f13,f15,f6,f7,f8,f9,f11,f20,f21,f22,f23,f24,f25,f26,f28,f29,f30
t74	f1,f2,f3,f4,f5,f12,f13,f15,f6,f7,f8,f9,f11,f20,f21,f22,f23,f24,f26,f28,f29,f30
t75	f1,f2,f3,f4,f5,f12,f13,f15,f6,f7,f8,f9,f11,f20,f21,f22,f23,f24,f25,f28,f29,f30
t76	f1,f2,f3,f4,f5,f12,f13,f15,f6,f7,f8,f9,f11,f20,f21,f23,f24,f25,f28,f29,f30
t77	f1,f2,f3,f4,f5,f12,f13,f15,f6,f8,f10,f11,f20,f21,f22,f23,f24,f25,f28,f29,f30,f31
t78	f1,f2,f3,f4,f5,f12,f13,f15,f6,f8,f10,f11,f20,f21,f23,f24,f25,f28,f29,f30,f31
t79	f1,f2,f3,f4,f5,f12,f13,f15,f6,f8,f10,f11,f20,f21,f22,f24,f26,f27,f28,f29,f30,f31
t80	f1,f2,f3,f4,f5,f12,f13,f15,f6,f8,f10,f11,f20,f21,f22,f24,f27,f28,f29,f30,f31
t81	f1,f2,f3,f4,f5,f12,f13,f15,f6,f8,f10,f11,f20,f21,f22,f24,f26,f28,f29,f30,f31
t82	f1,f2,f3,f4,f5,f12,f13,f15,f6,f8,f9,f10,f11,f20,f21,f22,f23,f24,f25,f26,f27,f28, f29, f30, f31
t83	f1,f2,f3,f4,f5,f12,f13,f15,f6,f8,f9,f10,f11,f20,f21,f22,f23,f24,f26,f27,f28, f29, f30, f31
t84	f1,f2,f3,f4,f5,f12,f13,f15,f6,f8,f9,f10,f11,f20,f21,f22,f23,f24,f25,f27,f28, f29, f30, f31
t85	f1,f2,f3,f4,f5,f12,f13,f15,f6,f8,f9,f10,f11,f20,f21,f22,f23,f24,f27,f28,f29, f30, f31
t86	f1,f2,f3,f4,f5,f12,f13,f15,f6,f8,f9,f10,f11,f20,f21,f22,f23,f24,f25,f26,f28,f29, f30, f31
t87	f1,f2,f3,f4,f5,f12,f13,f14,f15,f6,f7,f8,f9,f10,f20,f21,f22,f23,f24,f26,f27,f28, f29, f30, f31

t88	f1,f2,f3,f4,f5,f12,f13,f14,f15,f6,f7,f8,f9,f10,f20,f21,f22,f23,f24,f25,f27,f28, f29, f30, f31
t89	f1,f2,f3,f4,f5,f12,f13,f14,f15,f6,f7,f8,f9,f10,f20,f21,f22,f23,f24,f27,f28, f29, f30, f31
t90	f1,f2,f3,f4,f5,f12,f13,f14,f15,f6,f7,f8,f9,f10,f20,f21,f22,f23,f24,f25,f26,f28,f29, f30, f31
t91	f1,f2,f3,f4,f5,f12,f13,f14,f15,f6,f7,f8,f9,f10,f20,f21,f22,f23,f24,f26,f28,f29, f30, f31
t92	f1,f2,f3,f4,f5,f12,f13,f14,f15,f6,f7,f8,f9,f10,f20,f21,f22,f23,f24,f25,f28,f29, f30, f31
t93	f1,f2,f3,f4,f5,f16,f17,f18,f6,f8,f11,f20,f21,f22,f24,f26,f28,f29,f30,f31
t94	f1,f2,f3,f4,f5,f16,f17,f18,f6,f8,f9,f11,f20,f21,f22,f24,f26,f27,f28,f29,f30,f31
t95	f1,f2,f3,f4,f5,f16,f17,f18,f6,f8,f9,f11,f20,f21,f22,f24,f27,f28,f29,f30,f31
t96	f1,f2,f3,f4,f5,f16,f17,f18,f6,f8,f9,f11,f20,f21,f22,f24,f26,f28,f29,f30,f31
t97	f1,f2,f3,f4,f5,f16,f17,f18,f6,f8,f10,f11,f20,f21,f22,f23,f24,f25,f26,f27,f28, f29, f30, f31
t98	f1,f2,f3,f4,f5,f16,f17,f19,f6,f7,f8,f9,f11,f20,f21,f22,f23,f24,f27,f28,f29,f31
t99	f1,f2,f3,f4,f5,f16,f17,f19,f6,f7,f8,f10,f11,f20,f21,f22,f23,f24,f25,f26,f28, f29, f31
t100	f1,f2,f3, f4,f5,f16,f17,f19,f6,f7,f8,f9,f10,f11,f20,f21,f22,f23,f24,f25,f26,f27, f28, f29, f31

TABLE C.5: Alphanumeric Character for Features of Transport Network Product Line

Character	Features	Character	Features
f1	transport	f13	SMS
f2	login	f14	Call
f3	logout	f15	help
f4	register	f16	contact us
f5	book a ride	f17	report problem
f6	book later	f18	invite friends

f7	book Now	f19	via email
f8	payment	f20	via SMS
f9	wallet	f21	rides information
f10	cash	f22	history
f11	credit card	f23	schedule
f12	contact captain	f24	notification

TABLE C.6: Transport Network Product Line Test Suite

Test Case	Features
t1	f1,f2, f3, f4, f5, f6, f21, f22, f23, f8, f9, f10, f11
t2	f1,f2, f3, f4, f5, f6, f21, f22, f23, f8, f10, f11
t3	f1,f2, f3, f4, f5, f6, f21, f22, f23, f8, f9, f11
t4	f1,f2, f3, f4, f5, f6, f21, f22, f23, f8, f11
t5	f1,f2, f3, f4, f5, f6, f21, f22, f23, f8, f9, f10
t6	f1,f2, f3, f4, f5, f6, f21, f22, f23, f8, f10
t7	f1,f2, f3, f4, f5, f6, f21, f22, f23, f8, f9
t8	f1,f2, f3, f4, f5, f6, f21, f23, f8, f10
t9	f1,f2, f3, f4, f12, f13, f14, f5, f6, f21, f22, f23, f8, f9, f10, f11
t10	f1,f2, f3, f4, f12, f13, f14, f5, f6, f21, f22, f23, f8, f10, f11
t11	f1,f2, f3, f4, f12, f13, f14, f5, f6, f21, f22, f23, f8, f11
t12	f1,f2, f3, f4, f12, f13, f14, f5, f6, f21, f22, f23, f8, f9, f10
t13	f1,f2, f3, f4, f12, f13, f14, f5, f6, f21, f22, f23, f8, f10
t14	f1,f2, f3, f4, f12, f13, f14, f5, f6, f21, f23, f8, f10
t15	f1,f2, f3, f4, f12, f14, f5, f6, f21, f22, f23, f8, f9, f10, f11
t16	f1,f2, f3, f4, f12, f14, f5, f6, f21, f22, f23, f8, f10, f11
t17	f1,f2, f3, f4, f12, f14, f5, f6, f21, f22, f23, f8, f11
t18	f1,f2, f3, f4, f12, f14, f5, f6, f21, f22, f23, f8, f10
t19	f1,f2, f3, f4, f12, f14, f5, f6, f21, f22, f23, f8, f9
t20	f1,f2, f3, f4, f12, f13, f14, f5, f6, f15, f17, f21, f22, f23, f8, f11

t21	f1,f2, f3, f4, f12, f13, f14, f5, f6, f15, f17, f21, f22, f23, f8, f9, f10
t22	f1,f2, f3, f4, f12, f13, f14, f5, f6, f15, f17, f21, f22, f23, f8, f10
t23	f1,f2, f3, f4, f12, f13, f14, f5, f6, f15, f17, f21, f22, f23, f8, f9
t24	f1,f2, f3, f4, f12, f13, f14, f5, f6, f15, f17, f21, f23, f8, f10
t25	f1,f2, f3, f4, f12, f13, f14, f5, f6, f15, f16, f17, f21, f22, f23, f8, f9, f10, f11
t26	f1,f2, f3, f4, f12, f13, f14, f5, f6, f15, f16, f17, f21, f22, f23, f8, f10, f11
t27	f1,f2, f3, f4, f12, f13, f14, f5, f6, f15, f16, f17, f21, f22, f23, f8, f9, f11
t28	f1,f2, f3, f4, f12, f13, f14, f5, f6, f15, f16, f17, f21, f22, f23, f8, f11
t29	f1,f2, f3, f4, f12, f13, f14, f5, f6, f15, f16, f17, f21, f22, f23, f8, f9, f10
t30	f1,f2, f3, f4, f12, f13, f14, f5, f6, f15, f16, f17, f21, f22, f23, f8, f10
t31	f1,f2, f3, f4, f12, f13, f14, f5, f6, f15, f16, f17, f21, f22, f23, f8, f9
t32	f1,f2, f3, f4, f12, f13, f14, f5, f6, f15, f16, f17, f21, f23, f8, f10
t33	f1,f2, f3, f4, f12, f14, f5, f6, f15, f21, f22, f23, f8, f9, f10, f11
t34	f1,f2, f3, f4, f12, f14, f5, f6, f15, f21, f22, f23, f8, f10, f11
t35	f1,f2, f3, f4, f12, f14, f5, f6, f15, f21, f22, f23, f8, f9, f11
t36	f1,f2, f3, f4, f5, f6, f18, f19, f20, f21, f22, f23, f8, f9, f10, f11
t37	f1,f2, f3, f4, f5, f6, f18, f19, f20, f21, f22, f23, f8, f10, f11
t38	f1,f2, f3, f4, f5, f6, f18, f19, f20, f21, f22, f23, f8, f9, f11
t39	f1,f2, f3, f4, f5, f6, f18, f19, f20, f21, f22, f23, f8, f11
t40	f1,f2, f3, f4, f5, f6, f18, f19, f20, f21, f22, f23, f8, f9, f10
t41	f1,f2, f3, f4, f5, f6, f18, f19, f20, f21, f22, f23, f8, f10
t42	f1,f2, f3, f4, f5, f6, f18, f19, f20, f21, f22, f23, f8, f9
t43	f1,f2, f3, f4, f5, f6, f18, f19, f20, f21, f23, f8, f10
t44	f1,f2, f3, f4, f5, f6, f18, f20, f21, f22, f23, f8, f9, f10, f11
t45	f1,f2, f3, f4, f5, f6, f18, f20, f21, f22, f23, f8, f10, f11
t46	f1,f2, f3, f4, f5, f6, f18, f20, f21, f22, f23, f8, f9, f11
t47	f1,f2, f3, f4, f5, f6, f18, f20, f21, f22, f23, f8, f11
t48	f1,f2, f3, f4, f5, f6, f18, f20, f21, f22, f23, f8, f9, f10
t49	f1,f2, f3, f4, f5, f6, f18, f20, f21, f22, f23, f8, f10
t50	f1,f2, f3, f4, f5, f6, f18, f20, f21, f22, f23, f8, f9

t51	f1,f2, f3, f4, f5, f6, f18, f20, f21, f23, f8, f10
t52	f1,f2, f3, f4, f5, f6, f18, f19, f21, f22, f23, f8, f9, f10, f11
t53	f1,f2, f3, f4, f5, f6, f18, f19, f21, f22, f23, f8, f10, f11
t54	f1,f2, f3, f4, f5, f6, f18, f19, f21, f22, f23, f8, f9, f11
t55	f1,f2, f3, f4, f5, f6, f18, f19, f21, f22, f23, f8, f11
t56	f1,f2, f3, f4, f5, f6, f18, f19, f21, f22, f23, f8, f9, f10
t57	f1,f2, f3, f4, f12, f13, f14, f5, f6, f15, f18, f19, f21, f23, f8, f10
t58	f1,f2, f3, f4, f12, f13, f14, f5, f6, f15, f16, f18, f19, f20, f21, f22, f23, f8, f9, f10, f11
t59	f1,f2, f3, f4, f12, f13, f14, f5, f6, f15, f16, f18, f19, f20, f21, f22, f23, f8, f10, f11
t60	f1,f2, f3, f4, f12, f13, f14, f5, f6, f15, f16, f18, f19, f20, f21, f22, f23, f8, f9, f11
t61	f1,f2, f3,f4,f12, f13,f14, f5, f6, f15, f16, f18, f19, f20,f21,f22,f23, f8, f11
t62	f1,f2, f3, f4, f12, f13, f14, f5, f6, f15, f16, f18, f19, f20, f21, f22, f23, f8, f9, f10
t63	f1,f2, f3, f4, f12, f13, f14, f5, f6, f15, f16, f18, f19, f20, f21, f22, f23, f8, f10
t64	f1,f2, f3, f4, f12, f13, f14, f5, f6, f15, f16, f18, f19, f20, f21, f22, f23, f8, f9
t65	f1,f2, f3, f4, f12, f13, f14, f5, f6, f15, f16, f17, f18, f19, f21, f22, f23, f8, f9, f10
t66	f1,f2, f3,f4,f12,f13,f14,f5,f6,f15, f16, f17, f18, f19, f21, f22, f23, f8, f10
t67	f1,f2, f3, f4, f12, f13, f14, f5, f6, f15, f16, f17, f18, f19, f21, f22, f23, f8, f9
t68	f1,f2, f3, f4, f12, f13, f14, f5, f6, f15, f16, f17, f18, f19, f21, f23, f8, f10
t69	f1,f2, f3, f4, f12, f14, f5, f6, f15, f18, f19, f20, f21, f22, f23, f8, f9, f10, f11
t70	f1,f2, f3, f4, f12, f14, f5, f6, f15, f18, f19, f20, f21, f22, f23, f8, f10, f11
t71	f1,f2, f3, f4, f12, f14, f5, f6, f15, f18, f19, f20, f21, f22, f23, f8, f9, f11
t72	f1,f2, f3, f4, f12, f14, f5, f6, f15, f18, f19, f20, f21, f22, f23, f8, f11
t73	f1,f2, f3, f4, f12, f14, f5, f6, f15, f18, f19, f20, f21, f22, f23, f8, f9, f10
t74	f1,f2, f3, f4, f12, f14, f5, f6, f15, f18, f19, f20, f21, f22, f23, f8, f10

t75	f1,f2, f3, f4, f12, f14, f5, f6, f15, f18, f19, f20, f21, f22, f23, f8, f9
t76	f1,f2, f3, f4, f12, f14, f5, f6, f15, f18, f19, f20, f21, f23, f8, f10
t77	f1,f2, f3, f4, f12, f14, f5, f6, f15, f18, f20, f21, f22, f23, f8, f9, f10, f11
t78	f1,f2, f3, f4, f12, f14, f5, f6, f15, f18, f20, f21, f22, f23, f8, f10, f11
t79	f1,f2, f3, f4, f12, f14, f5, f6, f15, f18, f20, f21, f22, f23, f8, f9, f11
t80	f1,f2, f3, f4, f12, f14, f5, f6, f15, f18, f20, f21, f22, f23, f8, f11
t81	f1,f2, f3, f4, f12, f14, f5, f6, f15, f18, f20, f21, f22, f23, f8, f9, f10
t82	f1,f2, f3, f4, f12, f14, f5, f6, f15, f18, f20, f21, f22, f23, f8, f10
t83	f1,f2, f3, f4, f12, f14, f5, f6, f15, f18, f20, f21, f22, f23, f8, f9
t84	f1,f2, f3, f4, f12, f14, f5, f6, f15, f18, f20, f21, f23, f8, f10
t85	f1,f2, f3, f4, f12, f14, f5, f6, f15, f18, f19, f21, f22, f23, f8, f9, f10, f11
t86	f1,f2, f3, f4, f12, f13, f14, f5, f6, f7, f21, f22, f23, f24, f8, f11
t87	f1,f2, f3, f4, f12, f13, f14, f5, f6, f7, f21, f22, f23, f24, f8, f9, f10
t88	f1,f2, f3, f4, f12, f13, f14, f5, f6, f7, f21, f22, f23, f24, f8, f10
t89	f1,f2, f3, f4, f12, f13, f14, f5, f6, f7, f21, f22, f23, f24, f8, f9
t90	f1,f2, f3, f4, f12, f13, f14, f5, f7, f21, f22, f23, f24, f8, f9, f10, f11
t91	f1,f2, f3, f4, f12, f13, f14, f5, f7, f21, f22, f23, f24, f8, f10, f11
t92	f1,f2, f3, f4, f12, f13, f14, f5, f7, f21, f22, f23, f24, f8, f9, f11
t93	f1,f2, f3, f4, f12, f13, f14, f5, f7, f21, f22, f23, f24, f8, f11
t94	f1,f2, f3, f4, f12, f13, f14, f5, f7, f21, f22, f23, f24, f8, f9, f10
t95	f1,f2, f3, f4, f12, f14, f5, f6, f7, f15, f18, f19, f20, f21, f22, f23, f24, f8, f10
t96	f1,f2, f3, f4, f12, f14, f5, f6, f7, f15, f18, f19, f20, f21, f22, f23, f24, f8, f9
t97	f1,f2, f3, f4, f12, f14, f5, f7, f15, f18, f19, f20, f21, f22, f23, f24, f8, f9, f10, f11
t98	f1,f2, f3, f4, f12, f14, f5, f7, f15, f18, f19, f20, f21, f22, f23, f24, f8, f10, f11
t99	f1,f2, f3, f4, f12, f14, f5, f7, f15, f18, f19, f20, f21, f22, f23, f24, f8, f9, f11
t100	f1,f2, f3, f4, f12, f14, f5, f7, f15, f18, f19, f20, f21, f22, f23, f24, f8, f11