Master Thesis
Software Engineering
Thesis no: MSE-2007: 16
March 2007

Testing in Software Product Lines

Odia Osatretin Edwin

School of Engineering
Blekinge Institute of Technology
Box 520
SE – 372 25 Ronneby
Sweden

This thesis is submitted to the School of Engineering at Blekinge Institute of Technology in partial fulfillment of the requirements for the degree of Master of Science in Software Engineering. The thesis is equivalent to 20 weeks of full time studies.

Contact Information:
Author(s):
Osaretin Edwin Odia
Department of Software Engineering and Computer Science
Blekinge Institute of Technology
PO Box 520
SE-372 25 Ronneby
Sweden
E-mail: osod04@student.bth.se
E-mail: eddy_odia2002@yahoo.co.uk

Advisor(s):
Dr. Richard Torkar
Assistant professor
Department of System and Software Engineering
Blekinge Institute of Technology
School of Engineering
PO Box 520, SE- 372 25 Ronneby
Sweden

School of Engineering
Blekinge Institute of Technology
Box 520
SE – 372 25 Ronneby
Sweden

Internet : www.bth.se/tek
Phone: +46 457 38 50 00
Fax: +46 457 271 25

ABSTRACT

This thesis presents research aimed at investigating different activities involved in software product lines testing process and possible improvements towards achieving developing high quality software product lines at reduced cost and time.
The research was performed using systematic review procedures of Kitchenham. The reviews carried out in this research covers several areas relating to software product lines testing. The reasons for performing a systematic review in this research are to; summarize the existing evidence covering testing in software product line context, to identify gaps in current research and to suggest areas for further research.
The contribution of this thesis is research aimed at revealing the different activities, issues and challenges in software product lines testing.
The research into the different activities in software product lines lead to the proposed SPLIT Model for software product lines testing. The model helps to clarify the steps and activities involved in the software product line testing process. It provides and easy to follow map for testers and managers in software product line development organizations.
The results were mainly on how testing in software product lines can be improved upon, towards achieving software product line goals. The basic contribution is the proposed model for product line testing, investigation into, and possible improvement in, issues related to software product line testing activities.

Keywords: Testing, Test cases, Software product lines, Core assets.

ACKNOWLEDGMENTS

First of all I would like to thank my supervisor, Dr Richard Torkar for sharing his knowledge and experience and helping me with this thesis. Your guidance has really helped me in this thesis. He has given valuable comments on the thesis. I would also like to express my thanks to Dr. Robert Feldt for your guidance and helping me with new ideas.

I would also like to thank the following lecturers for sharing their knowledge and experience and helping me with my courses: Dr. Mikael Svahnberg, Dr Tony Gorsheck, Lawrence Henesey, and Anna Wikstrom.

I wish to thank my Dad for installing me the determination and drive to keep the flame burning when it matters most. I would like to express my thanks to my late Mum, who passed away couple of years ago. My past, present and future victories are for you. I thank my brothers and sisters for always being there for me and their understanding.

I also express my thanks to the Rozemas in Canada, for your love and support during our wedding. To my wife Julie, you are never forgotten, for your support and love. Thank you for always believing in me.

Finally, I thank all my friends in Nigeria and in Sweden; you all have made life interesting.

TABLE OF CONTENTS

CHAPTER 1

1 INTRODUCTION

This chapter provides a background to the thesis. The background consists of an insight into the purpose, aims and objectives of the thesis. In addition, the research questions asked and the research methodology used are presented in this chapter.

1.1 BACKGROUND

It is has been a difficult challenge for software developers and testers to develop and maintain software system for industry as a result of changes in market and customers requirements. For software developers to remain in the highly competitive software developing industry, they need to produce quality software that satisfies customers' needs.
It is expensive and time consuming to design and develop software for a single system that satisfies the need of just a single customer [1]. As a result of constraints of time and cost involved in designing and developing software for a single system and specific customer, this lead to the concept of product family development which satisfies diversity of customers needs [1]. In software product lines, several products are developed with common and variable features. Software product line approach focuses on reuse of functionality and core assets for development. This increases the quality of software product lines and specific products and decreases development cost and time.
According to Paul Clements, "Software product line is a set of software-intensive systems sharing a common, managed set of features that satisfy the specific needs of a particular market segment or mission develop from a common set of core assets in a prescribed way" [2]. Software product line increases reuse of software, increase quality and decrease maintenance cost [3]. The development of software product lines is quite tasking and challenging. It is important to subject every stage of software development to quality control as soon as development commences. If fault is discovered in product lines, the product lines or family should be reworked, else it will be very expensive to correct products with faults when rejected by customers.
In order to cut down reject rate and increase the quality of products developed from product lines, it is important to devote adequate effort to testing in software product lines.
This thesis will present research aimed at investigating testing in software product lines and possible improvements in testing steps, tools selections and application applied in software product lines testing. For software developers, to design and develop software for systems that satisfies diversity of customers needs with increase quality and decreased time and cost, there are basically certain testing steps, tools selection and application of these tools in software product lines testing. Improvements in testing steps, tools selection and application in software testing, will provide developers and testers in software organizations the ability to test software at reduced cost and time aimed at delivering quality software that is accepted by customers and market.
Chapter one of this thesis is organized in the following ways. Section 1.1 of introduction presents background to the thesis. Section 1.2 presents purpose of the thesis. Section 1.3 presents aims and objectives of the thesis. Section 1.4 of introduction presents very important notation used in the thesis with respect to research area of the thesis. Further more, section 1.5 presents related work with respect to the thesis. Section 1.6 presents the research approach as used in the thesis. This section includes, the research questions posed on the thesis and research methodology used in performing the research of this thesis. Section 1.7 presents contributions of the thesis. Section 1.8 presents outline and structure of the thesis.

1.2 PURPOSE

The purpose of the thesis is to present a clear picture of testing process in the context of software product lines, which is quite different from testing in single product. The focus of this thesis is specifically the different activities involved in software product lines testing and possible improvements in software product lines testing process towards achieving the goals of developing high quality software product lines at reduced cost and time.

1.3 AIMS AND OBJECTIVIES

This thesis will present research aimed at investigating testing in software product lines and possible improvements in testing steps, tool selection and application applied in software product lines testing. This thesis will also present research aimed at discovering and unfolding different issues regarding testing in software product lines.
The quality of software determines the acceptance of software by customers or users of the software. Software testing is a necessary process in the development of software product. Testing a product is aimed at detecting defects in the system and to validate requirements.
"Testing in a product line organization encompasses activities from the validation of the initial requirements model to verification activities carried out by customers to complete the acceptance of a product."[4]. Testing is one quality assurance techniques for software products [5].

1.4 NOTATIONS IN SOFTWARE TESTING

This section presents the different notations use in software testing context.

1.4.1 SOFTWARE QUALITY

Software quality is basically the requirements software must fulfill for the software to be accepted by the customer. Requirements are described as conditions a system must comply with as specified in documents derived from customers or users of the system [6]. Requirements can be classified into functional and non functional [7] [8]. System software is made up two types of attributes, functional attributes and non-functional attributes.
The non-functional requirements of the system are requirements that are not concerned with the functionality of the system, while the functional requirements are based on the functionality of the developed system [8]. Functional requirements show what the system must do, and non-functional requirements are basically constraints on the system design [7]. System non-functional requirements are requirements that do not state certain specific system functionality [8]. If the requirements are further specified in more detail in terms of specific system functionalities, then it could be redefined as functional requirements. Examples of non-functional requirements are: security, portability, reliability, and maintainability and safety requirements [8].
To detect defects in software during test design, an ideal quality assurance activity should be successful [5]. Product evaluation and process monitoring are software quality assurance activities that ensure software development and control processes are carried out as described in project management plan, and project's procedures and standards are followed. Software quality assurance is a systematic approach to the evaluation of software product standards, processes, and procedures [19]. Type of standards includes: documentation standard- specify the form and content for planning, control, and product documentation.

Code standard-specify the language the code is to be written. Design standard- provides the rules and methods to translate software requirements into software design [19]. Procedures are the steps to be followed to carry out a process [19]. It is practically unachievable, that all defects are eliminated during test design, since we are humans; there will always be defects in software [5]. This makes quality assurance to even fail its first goal. The first goal of quality assurance is eliminating defects from software. Defects cannot be eliminated from software; neither can defects be prevented from sneaking into software during development [5].
According to Bezier [5], quality assurance should therefore achieve its second goal, defects detection, which is accomplished my running test, which shows defects exists. It is very important to test the requirements to ensure that the requirements have fully met and fulfils the system specification. This is done by deriving test cases for requirements. Testing is an effective way of finding requirements problems such as incompleteness and ambiguity. When these problems are detected during testing, they are corrected to ensure that they do not cause system failure when the system becomes operational [8]. Defects detection, accomplished by testing leads to the third goal i.e. of determining what is responsible for the problems and providing good solutions [5].

1.4.2 SOFTWARE PRODUCT LINES

Software product line approach focuses on reuse of core assets in software development. This increases the quality of software and decreases development cost. Developing products in product lines requires two parts. Firstly, the core assets of products in product lines which shares the commonality of each product, and secondly the products specifications that varies in each of the products.
According to Clements, “Software product lines is a set of software-intensive systems sharing a common, managed set of features that satisfy the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way” [2]. Software product lines represent and increases reuse of software and quality, and decreases maintenance cost [3].

1.4.3 SOFTWARE TESTING

Software testing is a process in software development to ensure the system has reached a certain reliability level and be able to perform as defined in requirements. Testing is divided into levels for example, unit testing, integration testing and system testing. In each level of testing, the purpose of testing is different. It has specific objectives and limitations [9]. Black box and white-box testing methods are suitable for different testing levels and has different strategies to run testing [9]. Software testing basically involves the operation of a system under certain controlled conditions with strict evaluation of the results of the process. It is said to be controlled conditions, both normal and abnormal conditions in the sense that, testing should deliberately make things happen wrongly in order to determine if things happen when they are not suppose to or things do not happen when they are suppose to happen in the system.
Software testing is aimed at quality assurance of the system in order to fulfill system requirements and satisfy customers. The way organizations go about assigning the task and responsibilities for testing varies from organizations to organizations. In some organizations it is the responsibility of one group. While in some other organizations it is the responsibility of both testers and developers who work together in a project team, monitored by project managers.

1.4.4 TESTING IN SOFTWARE PRODUT LINES

Testing in software product lines is to examine software product lines core assets, product-specific examination and the interaction between them. Software testing is a necessary process in the development of software products. Testing a product is aimed at finding defects in software and to validate software requirements. Therefore, testing strategy and process for product line is different from testing in individual product development. Testing activities are run together with software development activities. The development process produces artefacts for testing process to examine. A testing process produces test results, define defects and report them in order to repair the artefacts [4].

1.4.5 TESTING IN SOFTWARE PRODUCT

As we know, products in the product lines environment are produced from core assets and common features of software product lines. The specifications of individual products are derived from the generic specifications in a product line. Testing in product level can be from the functionality test or test templates in the product line level [2].

1.5 RELATED WORK

In this section of the thesis, related work covering this thesis is presented. In order to present related work covering this thesis, few books that covered related work are presented. More so, to present related work that covered this thesis, research papers, and theses and published articles with references are presented. This thesis is aimed at profiling solutions to issues with testing in software product lines by researching into the same issues researched upon by authors of published research papers, articles and theses.

There could be many books covering the area of testing in software product lines, but in this thesis we decided to mention two books that covered the subject area of this thesis and are comprehensively well aimed at profiling solutions to the same issues with this thesis. The first book is by Paul Clements and Linda Northrop, Software Product Lines: Practices and Patterns, Addison-Wesley (2002) [4]. The second book is by Pohl, K., Böckle, G., and Van der Linden, F. Software Product Line Engineering—Foundations, Principles, and Techniques. Springer, Berlin, Heidelberg, New York, 2005 [10]

Presenting related research papers that covered this thesis; there are basically three papers that are closely related to the work of this thesis:

John D. McGregor, Testing a Software Product Line, Carnegie Mellon University (December 2001) [4].

Coverage and Adequacy in Software Product Line Testing, Myra B Cohen, Matthew B. Dwyer, Jiangfan Shi, Department of Computer Science and Engineering, University of Nebraska, Lincoln, Nebraska, USA [11].

A survey by Antti Tevanlinna and Juha Taina, Raine Kauppinen University of Helsinki Finland, product family testing ACM SIGSOFT software Engineering Notes [12].

Used case-based Testing of product lines, Antonia Bertoline IST-CNR, Stefanian Gnesi ISTI-CNR, Area della Ricerca di Pisa via Moruzzi, 1.56124, Pisa. Italy [13].

The above related works are the closest we could get relating to this research.

1.6 RESEARCH APPROACH

This section describes the approach used in this thesis. In the thesis work, qualitative research approach using systematic review method will be used to perform the research

work. This involves intensive review of journals, proceedings, projects, and Internet resources related to the research.

1.6.1 RESEARCH QUESTIONS

RQ1. What is the relationship between product lines and product?
RQ2. What steps are needed to develop and execute software testing in software product line environment?
RQ3. What approaches can be used to develop test cases from use cases that contain variability and to derive application test cases from them?
RQ4. What is the connection between software reuse and reusability of test cases?
RQ5. How do testers and developers determine the reliability of the reusable software or components?
RQ6. What are the challenges of software product lines testing?
RQ7.What tools and activities should testers and developers adopt to ensure software product lines assurance during testing process?

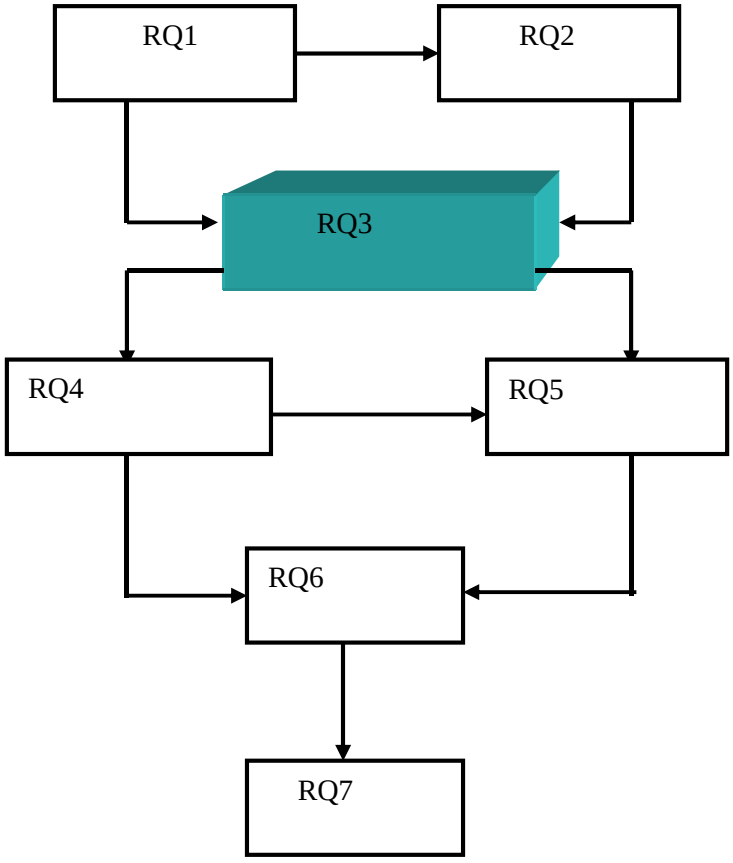

Figure 1.1: Relationship diagram between different research questions

The main idea in product lines testing is to reuse test case and other test artefacts throughout the entire product lines instead of testing every application as an independent software product.
In software product lines, software products are developed from product line core assets. It is therefore, important to perform proper testing in software product lines, using the right steps and tools, developing test case from use cases, trying to determine the reliability of reuse software in the product lines development.

The selection of appropriate test cases to test software product lines conformance to given set of requirements specification is important for the success of software product lines testing process. Considering, the importance of test cases to software testing, it is important to have well defined approaches to derive test cases from use case for product line testing. Developing test cases for product lines testing, from use cases enrich with variability should be given much attention. There are challenges facing testing in software product line context. These challenges could be both technical and non-technical.
All tools, strategies and approaches employed by developers and testers are aimed at developing software product lines and products that are reliable and acceptable by customers to decrease the time and cost of software development and testing. Tools support for product line testing can help product line achieve its goals as well.

1.6.2 RESEARCH METHOD

In the thesis work, systematic review method is used in performing the research work. This involves intensive review of journals, proceedings, projects, and Internet resources related to the research.
Kitchenham systematic review procedure is adopted in conducting our research. The steps in the process are as follow [20]:

Introduction
Work Background
Research Method

- Identification of research
- Selection of studies
- Quality assessment of primary studies
- Data extraction procedures
- Data analysis

Each of these stages in research method will be discussed in this section.

Results
Discussion
Conclusion and recommendation

The results of different articles in published journals; proceedings and literatures closely related to the research are compared and contrasted. The result of my findings is based on the compared results and findings of closely related research work. The study of this thesis is related to other large ongoing research in literatures and articles, to fill in the gaps and possibly extend prior study.
The essence of using the qualitative type of research in this paper is that there is access to already intensive and extensive review of journals, proceedings, projects and Internet resources related to the research.
Secondly, the thesis references different articles in published journals and proceedings in the course of this research and tries as much as possible to intensively and extensively review primary studies in electronic databases and literatures that have discussed the topic of the thesis.

IDENTIFICATION OF RESEARCH

The aim of systematic review is to find out as many primary studies relating to research questions as possible using unbiased search strategy [20].

GENERATING SEARCH STRATEGY FOR RESEARCH

It is important to define and follow a search strategy for the research.
The author develops the search strategy to be used. The strategies include:

Search aimed at identifying existing primary studies for our research and assessing the volume of potential studies includes.
Combination of search terms derived from thesis research questions.
Research questions will be broken down into individual small facets.
Consider headings used in journals and databases.
Constructing search strings using Booleans AND'S and OR'S

DATABASES FOR RESEARCH

Sources of primary studies to be used for the research include the following databases:

IEEE Xplore Electronic Database
ACM Digital Library
Springer Link Electronic Database
Science Direct Electronic Database
Academic Search Elite Electronic Database
Literatures (Books)
Software Engineering Institute SEI
Fraunhofer Institute for Experimental Software Engineering (IESE)

Choices of databases above are based on the following reasons:

The databases are world's leading interactive databases for high quality journals, conferences, career resources and book series.
The databases are leading authorities in areas ranging from computers and software engineering, telecommunications and among others.
They are powerful central access for researchers and scientist.
The databases provide computing and software engineering articles written by good authors. The authors of these articles are well known experts with national and international reputation. Their current practices and experiences are from a variety of information technology projects in large organizations and research institutes.

STUDY SELECTION CRITERIA

Both inclusion and exclusion criteria will follow logically from research questions pose on the thesis.
Criteria for including and excluding studies from systematic reviews are:

EXCLUSION CRITERIA

Primary studies outside testing in software product lines context are excluded from systematic reviews.

Primary studies that do not have basis of comparison with other primary studies found in databases as they relate to research questions pose on this thesis will be excluded from systematic reviews. Articles not related to software testing do not have basis of comparison. Specifically, articles that fall outside the field of software engineering are excluded.

Theses, dissertations, conferences, or presentation abstracts not published in peer reviewed journals will be excluded from our studies.

Articles will also be excluded on the ground that they are summary. It will be impossible to get information from summary of articles relating to our research questions. Comprehensive information can not obtained from summary of articles.

Primary studies that are of no importance to research questions as they aimed at providing evidence to thesis research questions are excluded from articles.

INCLUSION CRITERIA

Primary studies focusing in several areas of software product lines testing with respect to thesis research questions will be included in systematic review.

Primary studies that can be compared in order to collate results and produce quality summary as it relates to research questions will be included in systematic review.

Primary studies relating to importance of testing in software product lines context and producing relevant results relating to research questions will be included.

The authors of primary studies are taken into consideration, for primary studies to be included in the research. We ensure that primary studies included are written by known authors. The reason of coming up with top five scholars in our research is to show the reputation of scholars used in our research. They are experts in the field of software engineering.

The research institutes and institutions of scholars also determine primary studies included in research. In our five top scholars ranking, the table contains the institutes and institutions of scholars.

To include studies in our research the date of publication of studies are taken into consideration.

The origin of primary studies contributes to the quality of research. Primary studies included are from listed databases.

INVESTIGATED ARTICLES SELECTION TECHNIQUES:

Investigated articles from included articles are read to see if they provide direct data for our analyses. It is important to recognise that many of the stages in systematic reviews involve iteration.

Articles related to research question from included articles are selected. The articles are related in the sense that certain areas relating to search terms are discussed in the articles, without any relationship to research questions.

Research questions are broken into pieces, the search produces articles that discusses part of the search terms, based on that they are selected from the included articles.

Summary of articles included we not be investigated, because do they do not contain detailed information for the research.

The investigated articles from each database are read whether they provide direct evidence to research question.

The investigated articles from each database are recorded in table.

The articles that provided direct evidence to research question from investigated are selected.

The essence of investigating these articles is to find answers, direct evidence to our research question.

They do not provide direct answers, and evidence to research question.

SELECTED ARTICLES:

Selected articles provide direct evidence to our research questions.

They provide main report with respect to research questions of the thesis.

The hints reported in tables are investigated and selected articles.

Further explanation to investigated articles

Systematic reviews starts by defining research questions to be addressed.

Systematic reviews require inclusion and exclusion criteria to assess each potential primary study. There are numerous articles that are always revealed in each database search.

Study selection criteria and procedures are defined. The study selection criteria determine criteria for including, and excluding primary study from, the systematic review.
Firstly, we have to exclude primary studies from the list of numerous articles from the articles revealed. Using the criteria defined in the thesis. Then we are left with included articles.

Secondly, we used inclusion criteria defined in thesis to include primary studies.

Thirdly, from the list of included studies, we have to investigate articles relating to research question.

Fourthly, from the investigated articles, the ones that provided direct evidence to research question are selected.

From the included articles I have to investigate articles relating to research question. There after, I selected articles that provided direct evidence to research question. The total Number of articles that popped up when search terms are used in databases are not recorded and selected. Articles recorded are the ones investigated relating to research question and articles selected. Selected articles provide direct evidence to research question.

DATA EXTRACTION STRATEGY

From the search criteria used to find primary studies relating to research question, studies relating to subject area will be read meticulously. From the list of primary studies investigated, studies that are relevant and provide direct evidence to research question used for the search are selected and read comprehensively, in order to extract all relevant information with respect to the research question used for the search.
Information from these selected articles are analyzed and summarised. The results will be compared in order to draw conclusion base on findings. Information from each primary studies are extracted accurately and reported in our findings without any bias. There will not be any tendency to produce results that depart systematically from the true results. The

internal validity of our research is the unbiased results that will be produced. Unbiased results are internally valid. Also, internal validity is a prerequisite for external validity.

DATA ANALYSIS

Data analysis involves collating and summarising the results of the included primary studies. In this thesis, descriptive analysis will be used to collate and summarise the results of the included studies.

1.7 CONTRIBUTIONS OF THESIS

The research was performed using systematic reviews in software product lines testing. The main rational for undertaking systematic reviews is to produce research that is thorough and fair with scientific values relating to software product lines testing. Systematic review was performed to summarise existing evidence, the benefits and limitations in different areas of software product lines testing. To identify gaps in current research in order to suggest areas for further investigations in software product lines testing. It provides and easy to follow map for testers and managers in software product line development organizations.
From the research, the results are mainly on how testing in software product lines can be improved upon, toward achieving the goals of software product lines. Basically, the contribution of the thesis is the improvement in software product lines testing issues.

1.8 STRUCTURE AND OUTLINE OF THESIS

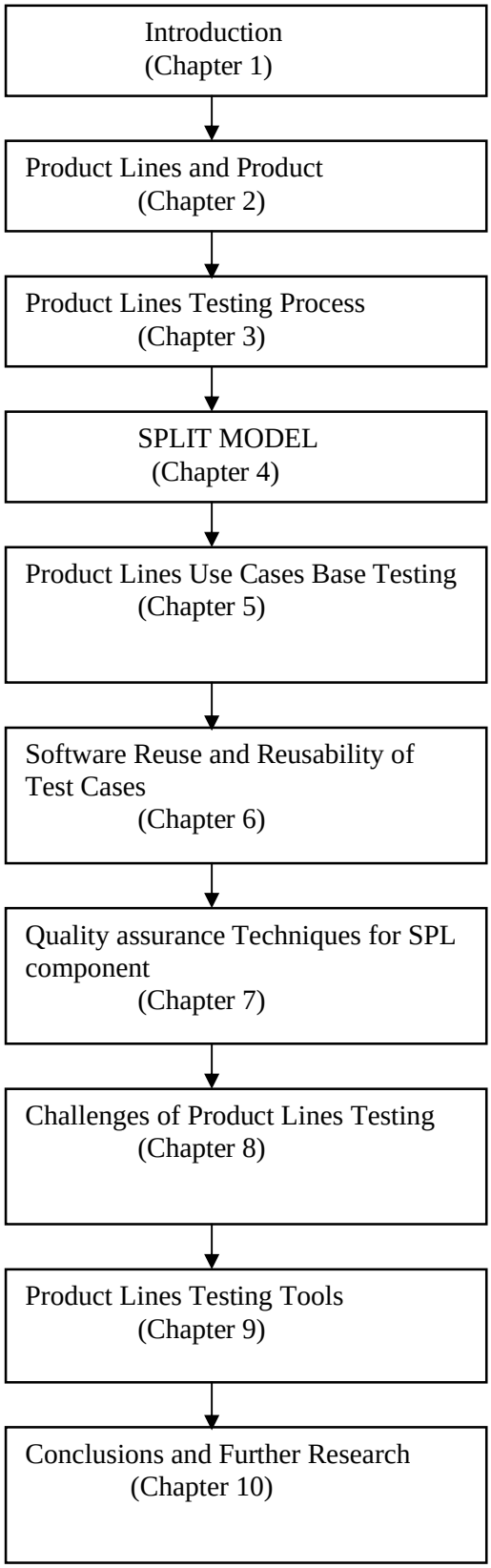

Figure 1.2: Structure of the thesis.

Chapter 1 provides an introduction to the subject of this thesis.

Chapter 2 discusses the relationship between product lines and product.

In Chapter 3 Testing process in software product lines are discussed. To be precise, the steps involved in the software product lines testing process and the activities involved in the product line testing process are discussed in chapter 2.

Chapter 4 presents software product lines testing model (SPLIT Model), proposed for product lines testing process. The model shows the steps and activities in software product lines testing process.

Chapter 5 Discusses on use case base testing in software product lines. The creation of test cases from use cases to perform testing in software product lines contest. Introduction of different methods to model use cases is described.

Chapter 6 presents software reuse and test cases reuse in software product lines development, and the implementation of test cases in software product lines specific product.

Chapter 7 presents quality assurance techniques to verify reliability of reusable software components for product lines before using them for product development. The different techniques for components assurance are discussed.

Chapter 8 introduces challenges of software product lines testing process. It describes challenges facing testing in software product lines context.

Chapter 9 presents discussion on the use of tools to support software product lines testing process. The tools available to support testing process in software product lines context to enable software product lines testers select appropriate tools to support product lines testing.

Chapter 10 presents summaries of the thesis and areas of future research.

CHAPTER 2

2 PRODUCT LINES AND PRODUCT

Software product line is a current approach to software development as is constantly gaining attention not only from the software engineering community, but also more in software organizations [2]. The goals of software product lines is aimed at producing high quality software products and at the same time significantly reduce development and maintenance cost [72]. In this chapter, the relationship between product lines and product is investigated.

2.1 RESEARH METHOD

This section presents the method use in conducting our research. How we identified articles reporting product lines and product, and how we extracted information from primary studies relevant and relating to relationship between product lines and product. The research is conducted using Kitchenham methodology for systematic review [20]. Details of the methodology can be found on page 11 of this thesis.

2.1.1 PRODUCT LINES AND PRODUCT

According to [2], a software product line is "a set of software-intensive systems sharing a common, managed set of features that satisfy the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way."
Product line development is different from single product development. Product lines development process is made up of two phases: Domain engineering development and application engineering development. Development for reuse is performed in domain engineering phase, while development with reuse is performed in application engineering phase [72].
According to [72], in domain engineering, the product lines infrastructure, or core assets base, consisting of a set of reusable assets are developed. Product lines core assets developed in domain engineering are extended to application engineering for products development. Software product lines specific products are developed in application engineering from reusable product lines core assets developed in domain engineering. Specific products development assets are functions of product lines core assets. Products development in product lines therefore depends on product lines core assets developed in domain engineering.
Product lines framework has two key engineering processes: domain engineering and application engineering [50] [104]. The framework is defined based on experiences and the results of European SPLE research projects ESAPS, CAFA. [104].
The framework show the process and the general domain engineering activities and shows the relationships and interfaces of domain engineering to traditional system development application engineering process [90].
Domain engineering process [50] main task is defining the commonality and variability of the product lines, and at the same time establish the reusable artefacts of product lines [50]. The emphasis of domain engineering is on reuse of domain artefacts.
Domain engineering is the foundation for emerging product lines software development approach and affects the maintainability, understandability and usability and reusability characteristics of systems or basically family of similar systems [107].
Domain engineering methodology is actually based on the concept of family of products, in order to achieve systematic reuse within product lines. For software product lines organizations that construct software products in the present day highly competitive software business environment benefits from adopting this approach to suite their needs.

Domain engineering process therefore supports application engineering which uses the models and architecture to build specific software products. It is the responsibility of domain engineering to make sure the variability of the product lines is right enough for building the applications that exist within the scope of the product lines. Application engineering needs to exploit the variability of reusable artefacts by binding the variability according to the specific needs of the product line in order to build product lines specific products [9].
Application engineering process builds applications from reusable artefacts of software product line that are delivered to customers. The success of specific products in the market depends on how reliable are the reusable artefacts of domain engineering.
Product lines should be able to compromise between customer requirements, existing requirements architecture constraints [75]. The key to successful product lines development is managing variability in numerous products developed. This can be archive by constructing product lines models of requirements and features. This makes it possible to derive new products by making requirements selections from product line model of requirements [75].
Product line development is a collection of related products developed by combining reused core assets with product specific assets. A product line provides architectural variation points for reuse core assets to allow specific products select variant functionality [77].
There are two inputs in product line development, core assets and product assets input. The core software assets can include requirements, architecture and design, source code, test cases, product documentation, and so forth [78]. The product assets are derived from domain engineering core assets [105]. For that, much effort goes into domain engineering compared to product development [105].
According to [106], "Software product line engineering (SPL) involves developing the requirements, architecture, and component implementations for a family of systems, from which products (family members) are derived and configured".
Individual products are constructed by selecting consistent set of corresponding artefacts required to build individual products from product lines repository [106].
According to [11], in domain engineering process, the commonalities and variability of software product lines are defined and the domain core assets are realized, while in application engineering process software product line specific applications are derived from domain core assets.
Numerous products are developed from product line, all product instances of product line share configurable, customizable components in order to save development cost, when creating similar products for different market [108]. Software product line is a set of software products that share a set of common features, that satisfies the needs of a particular market and are developed from a set of common assets [109]. Product lines is a well defined repeatable process, where new systems within a given domain are built from existing parts constructed to be reuse [110].
Software product line consists of family of products software that has common features [111]. Product line basic concept is that similar products share a common set of components and functionality called core assets, though, individual products have some functionality specific to it, called the variable part [112].

2.1.2 IDENTIFICATION OF RESEARCH

On page 13 a summary of method used in identification of research can be found.

List of combination of search term used for all databases search:

Software AND product line AND product.

IEEE Xplore Database Search

Table 2.1: Hints from IEEE explore database search.

No	Investigated Articles	Selected Articles
1	7	4
Total	7	4

List of articles selected relating to RQ1.

Articles selected in IEEE database

Brown et al. 2004 [75]
Staples et al. 2004 [77]
Krueger et al. 2006 [105]
Gomaa et al. 2007 [106]

The list of articles above, selected from IEEE database clearly provided direct evidence relevant to research question 1. The primary studies have the same point of view describing the relationship between product lines and product. The views of the articles are as follow:

Product line is made up of two development processes: Domain engineering process and application engineering process. Domain engineering consists of set of reusable core assets from which individual products from product lines can be derived during application engineering.

ACM Database Search

Table 2.2: Hints from ACM database search.

No	Investigated Articles	Selected Articles
1	15	6
Total	15	6

Articles selected from ACM database search

Pohl et al. 2006 [11]
Tevanlinna et al. 2004 [24]
Cohen et al. 1979 [46]
Verlage et al .2005 [108]
Ereno et al. 2005 [109]
Hamilton et al. 2005 [110]

15 articles were investigated from ACM database search, from which 6 primary studies were selected. The six selected articles provided direct evidence relating to the research question 1 of this thesis. The views of selected articles are same as regards relationship between software product lines and products. They are of the view that, software products are developed from product line core assets. Product line core assets are extended to application engineering for product lines specific products development.

Springer Link Database Search

Table 2.3: Hints from Springer link database search.

No	Investigated Articles	Selected Articles
1	8	3
Total	8	3

Macaulay 1996 [104]
Gomaa et al. 2006 [111]
Benarides et al. [2006]

Total of 8 articles were investigated from Springer link database search. 3 of the articles gave direct evidence as regards the relationship between software product lines and product. The relationship is, software product lines consist of family of products that have common feature. Basically, individual products features are derive from product line core assets.

Software Engineering Institute Site Search

Table 2.4: Hints from SEI site search.

No	Investigated Articles	Selected Articles
1	6	3
Total	6	3

McGregor 2001 [4]
Clements 2002 [113]
Chastek el al. 2001 [114]

The major relationship between software product lines and product is that test cases created in domain engineering for product lines are extended to specific product level. The selected articles from the 6 investigated share this view.

Literature

Table 2.5: Hints from literature search.

No	Investigated book(s)	Selected Book
1	Software Product Lines: Practices and Patterns	1

Clements et al. 2002 [2]
Software product lines is “a set of software-intensive systems sharing a common, managed set of features that satisfy the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way.” Same view as every other article selected from other listed databases.

2.1.3 REVIEW OF RESEARCH RESULTS

The research showed that all articles selected from databases have the same views as regards the relationship between software product lines and product. The general view of the articles selected from the investigated, is that specific product development assets are derived from product lines core assets for product development.

2.1.4 DATA EXTRACTION

Data was extracted from primary studies after reading the articles carefully. Primary studies were read comprehensively to comprehend the contents of primary studies in order to report result accurately.
Total of 37 articles were investigated with respect to RQ1. 17 0f these articles were selected because they provided direct evidence relating to research question 1 of the thesis. Table 2.6 presents articles reporting relationship between product lines and product.

Table 2.6: Articles and literature reporting relationship between product line and product.

Journals/Conference proceedings	Total no. of articles and literatures investigated	Total no. of articles selected. (N)	Percentage of articles selected by row (Row %)
IEEE Xplore	7	4	57.0
ACM Digital library	15	6	40.0
Springer Link	8	3	38.0
Science Direct	0	0	0.0
Software engineering Institute (SEI)	6	3	50.0
Literature (Books)	1	1	100.0
Total	37	17	46.0

Number of investigated articles = 37
Number of selected articles = 17
Ration of investigated to selected articles = 37: 17
Ratio = 2:1
For every 2 articles investigated, 1 article always provide direct evidence relating to research question.

2.2 RESULTS

46% of 37 articles selected represent relative good number of articles that have same views as regards the research question. The figure represents the percentage of articles that gave direct evidence relating to research question 1.
All primary studies have the same concept as regards the relationship between product lines and product. The major relationship is that domain engineering construct reusable core assets for product lines from which product lines specific products are derived. That is the bond holding product lines and its specific products.

2.3 CONCLUSION AND RECOMMENDATION

This chapter reports relationship between product line and product in journals and conference proceedings including a literature. Out of 37 articles investigated with respect to research question 1 of thesis, we identified 17 articles that reported the relationship between product lines and product.
Going by the figures, the percentage (46%) reporting the relationship between product lines and product is relatively high. One reason may be that large number of articles discusses software product line comprehensively for reader to understand the context before delving properly into the main subject area of discussion.
The result of the research showed that the main relationship between product lines and product is the features or facilities created at the product lines level and specialized for each product in application engineering. Further research is required in the area of methodologies to improve the development of quality features or product lines core assets created at product line level and specialize for each product in application engineering. The methodologies will help product line to achieve its goals of quality development of products at reduced time and cost.

CHAPTER 3

3 PRODUCT LINES TESTING PROCESS

3.1 INTRODUCTION

Software product line goal is to increase reuse of software, increase quality and decrease maintenance and development cost [3]. It improves the quality of product that is developed from product lines to ensure that customers are satisfied with products in terms of reliability and correctness of products. To make sure these goals are archived, software organizations should ensure quality attributes such as correctness and reliability is a continuous objective right from early phases of development [4].

"There is one approach to realizing these goals, software product line organizations should define set of comprehensive activities that validate the correctness of what has been built and verify that the correct product has been built" [4].

There are several testing activities involved in the development of software product lines. Activities from the validation of the initial requirements to verification activities carried out by customers in order to complete the acceptance of the product [4]. Testing varies in scope from the entire product lines down to focusing on the specific products and to examining the individual components that are part of the product [4].

In this chapter of the thesis, testing process in software product lines is discussed. The activities, steps involve in software product lines testing.

3.2 RESEARCH METHOD

This section describes how we identified articles reporting testing steps, activities and procedures in software product lines testing process, how we extracted data and information from the articles, and how we analysed the articles.

3.2.1 SOFTWARE PRODUCT LINES TESTING PROCESS

The aim of testing in software product lines is to uncover the presence of defects. Product lines development process is different from single product development process. Also, product lines testing process is different from single product testing process. Based on these reason, product lines testing steps, activities, and procedures differs from single product testing steps and activities. To answer RQ2 of this thesis, we search for primary studies using electronic databases, journals, and conference proceedings with respect to activities, steps in software product lines testing process.

3.2.2 IDENTIFICATION OF RESEARCH

The aim of systematic review is to find as many primary studies relating to research questions as possible using unbiased search strategy [20]. The research question pose on this chapter is RQ2; what steps are needed to develop and execute software testing in software product line environment?

SEARCH STRATEGY

The criteria to search for primary studies are as follow:

List of combination of search terms derived from RQ2
Break down research question into individual facets
Consider headings used in journals and databases
Constructing search strings using Booleans AND'S and OR'S

List of combination of search terms use for each databases:

Software AND product lines AND testing steps.
Software AND product lines AND testing process.
Product family AND testing process.
Product lines AND testing activities.
Survey AND product family AND testing process.

Testing in software product lines is performed in domain engineering and application engineering [11] [15] [17] [18]. Domain engineering process is aimed at developing the general concept of product lines together with all the assets that are common to product lines, while application engineering process is aimed at designing product lines specific products and eventually produces customers' specific applications [21].
In the domain engineering process, the commonalities and the variability are defined and the domain artefacts are realised. The second development process is application engineering. Specific products are developed in application engineering process from product lines core assets developed in domain engineering. Software product line applications are derived from domain engineering artefacts. According to [2], testing in software product lines is to examine core asset, product-specific examination and interaction between them. Therefore, testing strategy and process for product line is different from testing in individual product development.
According to John D. McGregor, "Testing in a product line organization encompasses activities from the validation of the initial requirements model to verification activities carried out by customers to complete the acceptance of a product"[4].
It is very important to have a well defined testing process in software product lines that encompasses all activities, steps, procedures that will be easy to follow by software product lines testers during testing in order to detect defects in software product line specific products to complete the acceptance of product.
According to [11], there are two development processes in SPL, domain engineering and application engineering processes: In the domain engineering process, the commonalities and the variability are defined and the domain artefacts are realised. The second development process is for application engineering. SPL applications are derived from domain engineering artefacts.
There are two development processes, domain engineering and application engineering. In domain engineering, product lines core assets are developed and extended to application engineering for product development [12]. Core assets created in domain engineering for product lines are reused in application engineering for products development.
Further more; product line development is the differentiation of two development processes (domain engineering and application engineering). Domain engineering develops product lines core assets, while application engineering uses product lines assets to develop specific products [13]. Software testing process comprise sequence of operations and involved events, activities and steps which lead to the product of some outcome. In software product line, there are two testing processes [17]: Core asset testing process, and product testing process. According to [18], testing in product lines involves two processes, domain engineering and application engineering process. Product lines testing involve automatically generating test cases from software product line requirements.

According to [29] PLSE consists of two major processes: asset development and product development. Asset developer analyses product line domain and develops architecture and reusable components in assets development process. While in product development process, product developer analyses product requirements, selects features, adapts components and generates codes for a specific product. Product lines testing process is split into product line infrastructure, core assets, and test processes for the various products in software product lines [15].
According to [14], product lines testing can be performed like traditional testing; using the V-Model. But software product lines testing process has to be integrated with higher level product lines business process. The concept is that V-Model for single product testing can also be used in product lines testing process. According to [11] testing is part of product line engineering development process with domain artefacts and application artefacts. Therefore, testing in software product lines encompasses testing domain engineering artefacts and application engineering artefacts [11].
According to John [16], testing core assets in domain testing and detect remaining defects in application testing which are mainly caused by unexpected interactions. John supports testing core assets in domain engineering and product assets in application engineering.
From Tim [16], no domain testing, but create core test assets, e.g. test models. Reuse previous test results testing new applications. Tim's point of view is that product lines core assets are created but no domain testing, and reuse test results in product lines in testing new products.
According to [18], model based testing for software product lines was introduced in the article, it explained that model based testing has to be performed in both domain engineering and application engineering for two reasons:
The domain test model and test cases are used to validate domain requirements at early stage of development. Secondly, test cases are created for reuse in application engineering. Domain test models are used in domain engineering for domain assets testing (Requirement model) to validate requirements in domain engineering. Also, test cases are created in domain engineering for product lines testing.
The domain engineering unit is responsible for building the software product lines architecture, while application engineering unit is responsible for developing the product based on the shared assets of product lines [30].

SEARCH STRATEGY FOR EACH DATABASE

Summary of search strategy can be found in page 13 of this thesis.

Search criteria for each database:

Software AND product lines AND testing steps
Software AND product lines AND testing process
Product family AND testing process
Product lines AND testing activities

IEEE Xplore Electronic Database Search

Table 3.1: Hints from IEEE database search.

No.	Investigated Articles	Selected Articles
1	0	0
2	4	1
3	2	1
4	0	0
Total	6	2

From IEEE database search, 2 articles were selected from the 6 investigated.

Linden et al. 2002 [14]
Described the activities in software product lines testing process using Esaps & Café Projects. Though, emphasis was mainly on the two development processes in software product lines testing process that is domain engineering and application engineering. The activities include: requirements analysis, design, components testing, integration and system testing.

Weyuker 1998 [119], listed all product lines artefacts that can be tested. They include requirements, architecture, components and test documents. These artefacts were described as the activities in software product lines testing process.

ACM Digital Library

ACM electronic database search

Table 3.2: Hints from ACM database search.

No.	Investigated Articles	Selected Articles
1	0	0
2	8	2
3	6	1
4	4	1
Total	18	4

From the 18 articles investigated, 4 provided direct evidence relating to research question.
Tevanlinna et al. 2004 [24]
Kolb et al. 2006 [120]
Core assets that can be tested in product line are reusable software product lines artefacts, such as components, architecture, requirements and test cases. Both articles state clearly the artefacts that can be tested in software product lines.
Product line is made up of two development processes, domain engineering and application engineering. Domain engineering develops core assets for product lines and application engineering develops specific products. [11] [12]. Therefore there are two testing processes in software product lines, domain assets testing and application assets testing.

Springer Link Electronic Database

Table 3.3: Hints from Springer link database search.

No.	Investigated Articles	Selected Articles
1	0	0
2	0	0
3	5	1
4	0	0
Total	5	1

Pohl et al. 2005 [18]
One article was selected from Springer link database. The concept presented in the article selected is basically developing models to generate test cases from product line requirements. The main focus was creating test cases from product line requirements.
Testing in product lines involves two processes, domain engineering and Application engineering processes. Product lines testing involve automatically generating test cases from software product line requirements.

Science Direct Electronic Database Search

Table 3.4: Hints from science direct electronic database search.

No.	Investigated Articles	Selected Articles
1	0	0
2	0	0
3	0	0
4	0	0
Total	0	0

No article was found from search.

Literatures (Books)

Table 3.5: Hints from literature.

No	Book investigated	Selected Book
1	Software Product Lines: Practices and Patterns, 1	1
Total	1	1

Clements et al. 2002 [2],
Testing in software product lines is to examine core asset, product specific examination and interaction between them. Therefore, testing strategy and process for product lines is different from testing in individual product development. Product lines core assets are the main testing activities in software product lines testing process. Product assets are the testing activities in product.

Software Engineering Institute (SEI)

Table 3.6: Hints from SEI database search.

No.	Investigated Articles	Selected Articles
1	0	0
2	0	0
3	4	2
4	0	0
Total	4	2

McGregor 2001 [4]
McGregor 2001 [17]

Product line testing involves the activities from the validation of requirement to verification activities carried out by customers to accept the product. Consequently, all testing artefacts are included in the verification and validation process. The artefacts to be tested include domain core assets developed for product lines and product assets developed in application engineering.

3.2.3 DATA EXTRACTION

To identify and extract primary studies, the titles of abstracts of primary studies were carefully read. 33 articles published in selected journals and conference proceedings and 1 literature were read to make a total of 34 articles and literature that were read.
From the 34 articles and literature read, 10 (29.4%) reports the steps and different testing activities in software product lines testing process. From the articles investigated, six mentioned the different testing activities in software product lines testing process, while [2]

[4] [17] mentioned and described these activities comprehensively. Below is Table 3.7, it presents list of investigated or scanned articles and selected articles. Table 3.7 presents articles and literature reporting activities in software product lines testing process.

Table 3.7: Articles report testing activities and steps in SPL testing process.

Journals/Conference	Total no. of articles and literatures investigated	Total no. of articles selected. (N)	Percentage of articles selected by row (Row %)
IEEE Xplore	6	2	33.3
ACM Digital library	18	4	22.2
Springer Link	5	1	20.0
Science Direct	0	0	0.0
Software engineering Institute (SEI)	4	2	50.0
Institute for Experimental Software Engineering IESE Fraunhofer	0	0	0.0
Literature (Books)	1	1	100.0
Total	34	10	29.4

Articles investigated = 34
Articles selected = 10
Ration of investigated articles to selected articles.
Ratio= 3:1. For every 3 articles investigated, 1 is selected.
1 from every 3 articles investigated provided direct evidence relating to research question.
The percentage value of articles that provided direct evidence to research question is small compared to the number of articles investigated. It is an indication that more research is also required in this area with respect to the research question of this chapter.

3.2.4 DATA ANALYSIS

From our search, 9 article and 1 literature, total 10 reported steps and activities in software product lines testing process of the 34 articles scanned with respect to research question 2 (RQ2), using research strategy on page 13 of this thesis.
Majority of the articles selected described testing process in software product lines mainly from the two development process perspectives.
The following articles out lined the activities in software product lines testing process: [2] [4] [17] [24] [119] [120]. The different testing activities where mentioned in the articles. Though, there was no description of what happens in the different product lines testing activities mentioned except for the following [2] [4] [17] articles that provided comprehensive description of the activities in software product lines testing process. The different steps involved in software product lines testing were described.
The following articles [11] [12] [14] [18] emphasised on the two development processes involved in software product lines testing process and the methodologies to generate test cases for software product line testing in order to handle variability in software product line.
Some primary studies used Café Project on software product line framework to describe testing process in software product lines context.
The framework gives a fundamental structure of software product line. The framework is only a partial implementation of a product line application. Using the framework in software product lines testing makes it very challenging performing testing in product line context.
While some primary studies, e.g., [12] [13] [15] [18], used it to present the argument that testing can be performed in both domain engineering and application engineering, without explaining how testing can be performed in the different phases of product lines testing

process. Explanation of the different steps involved in software product lines testing is lacking in the studies.
Some primary studies used the Café Projects on software product lines framework in supporting their arguments most especially in defining methodologies to manage and control variability in software product line.
According to [14] [23], product line testing can be performed using traditional V-Model. The view of some studies is that V-Model for single product testing can be used to perform testing in software product lines. Also, the diagram used in [4] describing testing process in software product line context. It shows the different activities in software product lines testing. But the diagram does not include the two development processes in software product line. Such a diagram is not complete and it will be difficult to use the diagram to describe and explain testing process in software product line context, because the two development processes in software product line are not represented in the diagram.

3.3 RESULTS

Product lines testing like all testing requires careful planning and a well defined process [4]. There are two development processes in software product lines, domain engineering and application engineering. In software product lines there are two testing processes; core assets testing process and product testing process [4]. Core assets are the basis for production of products in product line. Product lines core assets includes [2]:
The architecture the products in the product lines will share.
Software components developed for reuse across product lines development.
The major goal of core assets testing process is to make sure the components being produced as product lines core assets works correctly and perfectly well [17]. Activities in core assets testing include: Inspection of models, component testing, and integration testing process. While activities in product testing process include: Unit testing, integration testing and system testing [17]. McGregor [4] expands in these testing areas and describes test related activities that can be used to form the test process for a product line organization.
All articles selected have same views that software product lines comprise two development processes (domain engineering and application engineering). According to McGregor [17] there are two testing processes in product lines (core assets and product testing processes). It implies that testing in software product lines is performed in core assets and product assets.
The following articles [12] [13] [15] [18] described the two development processes in software product lines which also confirms McGregor [17] two testing processes in software product lines, core assets and product testing processes.
The number of primary studies supporting McGregor [4] concept of testing activities in software product lines are: [2] [4] [17] [24] [119] [120] and [12] [13] [15] [18]. Going by this, the activities in software product lines testing process are:
Core assets testing activities includes: Inspection of Models (requirements model, architecture model, and design model), component testing, and Integration testing process. Activities in product testing process includes: product assets (requirements, test cases), unit testing, integration testing and system testing

3.4 DISCUSSION

The result from our research showed that most of the research in software product lines testing process involves automatically generating test cases for software product line, and research into development of models to manage variability in software product lines. Primary studies in the area of testing process in software product lines, specifically in product lines testing activities are lacking massively. There are few articles that focused on procedures of testing in software product line context. There is need to have studies that presents comprehensive description on the activities that takes place in software product line

testing process. Comprehensive description of these activities in software product lines testing process gives a clear picture of what happens in the process.
The only article from search that presented a model to show testing activities in software product lines context is [24]. Though, in the article, V-Model was used to describe testing process in software product lines context. The research showed that product line testing is performed by modifying V-Modeled for single product and using Café Project on product line framework.

3.5 CONCLUSION AND RECOMMENDATION

The result from our research showed that enough is not yet done by the research community as regards presenting well defined testing process model for software product line. A well defined software product lines testing process model should be developed for software product lines organization that can aid testers in product lines organization knowing the different testing steps to follow when performing testing in product line context. Also, the current testing method in product line using the traditional V-Model for single product is quite immature; therefore, there is need for well defined testing method for product line.
Product lines testing is based on generating and reusing test assets that can be applied to several other products for product lines to achieve its goal of reducing development cost. The use of V-Model to achieve this goal definitely creates problems. Firstly, it does not support the testing of different combination of components with variability. Secondly, complete integration and system testing is not possible, because V-Model can not support the implementation of variation.
Product line framework can not be considered to be software product lines testing process model. The framework presents the structure of product line. Testing process model, specifically for product lines testing is highly recommended, rather than modeling the V-Model for product lines testing.

CHAPTER 4

4 PROPOSED SPLIT MODEL

4.1 INTRODUCTION

After a careful study of primary studies in chapter three of this thesis, we decided to propose a model called software product line test model (SPLIT Model) for software product lines testing process. A process is a sequence of operations and involves events, activities and steps which lead to the product of some outcome [123]. Therefore it is important to have a diagram that shows the structure of software product lines testing concept. The study in chapter three of this thesis motivated the design of the proposed model for software product lines testing. A model refers to the pattern, plan and representation, or description to show the structure or working of a system or concept. It is important to have a model that shows the working concept of software product lines testing.
It makes it easier to understand the activities involved in software product lines testing. The model presents a crystal clear picture of the steps and activities in software product lines testing.

4.2 REASONS FOR THE PROPOSED SPLIT MODEL

Currently, there is no model specifically for product lines testing for seeing how each steps in software product lines testing process works.
What is currently available is modification of traditional V-Model of single product testing for product lines testing by different product line organizations.
V-Model is not mature and sophisticated enough for product lines testing for the following reasons:

- V-Model cannot handle variability complexity in product lines core assets. Examples, variability in product line requirements and variable features. Variability is defined in product line use case in SPLIT Model. SPLIT Model can be used to define the variation points in software product line captured requirements, specifically use cases.

- V-Model is designed for single product testing. Product line concept is development for numerous products with common and variable features. Based on this concept, it requires a test model like SPLIT Model that can test for both common and variable features. V-Model has the capability to test for single product and not for numerous products with variable and common features.

- V-Model is best known for single product testing process. In software product lines there are two development and testing processes, Core assets or domain testing process and product testing process. V-Model does not have the capability to test for both processes in software product line development. SPLIT Model has the capability to test for cores assets or domain engineering and product assets.

- V-Model does not support the testing of different combination of components with variability. Secondly, complete integration and system testing is not possible, because V-Model can not support the implementation of variation.

Software product line is structured into domain engineering (development for reuse) and application engineering (development for reuse) [14]. SPLIT Model is made up of two development processes, domain engineering and application engineering.
In software product line development, reusable artefacts are produced during domain engineering and product line specific products are built from these artefacts during application engineering to meet customers' specific needs. Reuse is an essential concept in software product lines. Based on this, test cases derived in domain engineering for product lines are stored in the repository and extended to specific products testing. V-Model do not have repository potentials to store test cases derived in domain engineering for product lines testing which are later extended for product testing. In SPLIT Model, reusable test cases are created in domain engineering from use cases and stored in the repository and extended to specific products development to meet customers' needs.
Test cases derived in domain engineering from use cases / requirements takes into consideration variability in product lines use cases which are extended to product development. Reuse of test cases is ensured in SPLIT Model by preserving variability during test case creation in domain engineering.
For software product line to achieve its goals of reduced development cost, time to market and develop high quality specific products, it requires more efficient testing model than those used in single product development.

4.3 SPLIT MODEL IN SOFTWARE PRODUCT LINE TESTING

The goals of software product line are to reduce development, and maintenance cost, and time to market as well as to increase the quality of software product lines specific applications [2]. This can be achieved by active reuse of software [4].
Going by this concept, increased productivity requires a more efficient testing than those used in single system development. A product line testing process is influenced not only by product lines requirements and new requirements of customers, but a testing process that can be easily used and understandable by software engineers to handle variability complexity in software product line testing process.
A product line testing process following SPLIT Model move through these steps one at a time, until the current tasting task is completed.

The advantages of SPLIT Model are:

- SPLIT Model defines comprehensively the phases of product lines testing process, and defined a sequence that is logical that the defined phases showed be approached.
- It also defines a relationship between the phases.
- It provides and easy to follow map for testers and managers in software product line development organizations.
- Variation points in product assets can be defined.

The SPLIT Model is used as a tool to manage testing in software product line context. The model helps software product line testers and managers to understand the working concept of product lines testing, in order for testing to be successful in product line context.
SPLIT Model testing life cycle follows a fixed path from core assets testing (requirements, architecture, design model and components) of domain engineering to product assets (requirements, test cases) testing of application engineering.
As product lines organizations get involve in product lines development and its specific products, vast number of software product lines test are required to prove that product line specific products work correctly. SPLIT Model presents the various steps and activities in software product lines testing process and how the steps follow each other. The SPLIT

Model is useful for seeing how each steps in software product line testing works with each of the others.
A process is a sequence of operations and involves events, activities and steps which lead to the product of some outcome [123]. Every process is made up of sequence of operations which makes it easier for users of the process to know where they are and what to do next to lead to the product of some outcome. Software product line testing lack such sophisticated testing model, this motivated the design of the proposed software product line test (SPLIT) Model for product lines testing.

4.3.1 DESCRIPTION OF SPLIT MODEL WORKING PROCESS

Software product line test (SPLIT) Model, is a model to performing testing in software product line context. A model refers to the pattern, plan and representation to show the structure of a system or concept. SPLIT Model presents the pattern and description to show the working concept of testing in software product line context. It presents the different steps and phases in software product lines testing process and how the structure works specifically to product line testing.
Testing in software product lines is performed in domain engineering and application engineering [4] [11] [12]. Domain engineering process is aimed at developing the general concept of product lines together with all the assets that are common to product lines, while application engineering process is aimed at designing product line specific products and eventually produces customers' specific application [21].These two development processes are represented in SPLIT Model.
Software product line test (SPLIT) Model design is based on three concepts. The concepts are use case based testing; model based testing, and the sequence of testing steps.
SPLIT Model support proactive reuse in software product line development. Therefore proactive reuse is extended to product lines testing for product line to achieve its goals of reduce development cost, time to market and to increase quality of individual products. Following this, ScenTED technique (Scenario based TEst case Derivation) is adapted to SPLIT Model for product lines testing. ScenTED supports proactive reuse of test cases. Test cases created in domain engineering are reused for product development in application engineering.
The second concept is model based testing in software product lines testing. Though scenario based testing can be used to derive test cases for software product lines testing at the early phase of development. But, to systematically derive test cases, there is need to adapt model based testing to SPLIT Model.
The third concept is the sequence in which testing is performed in software product line. The different testing phases are represented sequentially in SPLIT Model. Software product lines testing involves core assets testing and product testing. SPLIT Model shows the order of testing in software product line context.

4.3.2 STEPS IN SPLIT MODEL

REQUIREMENTS TESTING

The first step in product lines test is requirements testing. When product line requirements are defined, which is a statement by product line developers and customers of what the system shall achieve in order to meet the need, it is important to test the requirements to ensure they meet the need of product lines organizations and customers.
As represented in SPLIT Model, there are two parts in model based testing, the test model built from requirements, using state charts or activity diagram to represent the model [76]. The second part is test cases created using coverage criteria.

Test models in domain engineering contain variability which must be considered in domain engineering. Considering the fact that there are variability and commonalities in software product lines requirements, both must be specified in domain requirements. Since requirements contain variability, test cases must also contain comprehensive variability information [18]. Test cases that contain variability are called variant test cases, while test cases without variability are called common test cases and they can be applied to all products testing in SPLIT Model.
Adapting The ScenTED technique in SPLIT Model is based on requirements specification as use cases. Activity diagrams are used as test model in SPLIT Model from which test case scenarios are derived. More so, test case scenarios are further specified in sequence diagrams. The test case scenarios basically describe the response of the systems.
Use case is chosen for requirements specification for software product line in SPLIT Model, because it is convenient and easy to document customer requirements, and also convenient for testing at the early phase of product line testing. With respect to activity and state diagrams they are the most common diagrams to represent use case scenarios both in single systems development and product lines development.

ARCHITECTURE INSPECTION - The next broad step in product line testing is architecture inspection. The technique to test the architecture of product line is inspection. Inspection of product line architecture is carried out by team of analyst and designers who will use the architecture model as input to detailed design [4]. Executable model can be used as well rather than inspecting product line architecture manually.

DESIGN INSPECTION - Detailed design can be inspected by team of architects who created the architecture model and developers who will code the interface implementation of software product line [4].

COMPONENTS TESTING - Software product line development support proactive reuse of core assets. Software component is one of product lines core assets. Therefore, it is important to test software components before reusing them in product lines development. The components testing must achieve high degree of coverage testing since the components are intended for reuse in software product lines development. Components testing are based on the specification defined in software product line architecture [17].

CORE ASSETS TRACEABLE (REPOSITORY) - All product lines core assets developed in domain engineering for product line are stored in the repository for products development and testing in application engineering. The core assets includes: common and variable features, test cases, and software components etc. They are extended to application engineering for products testing.

PRODUCT TEST ARTEFACTS

Core assets stored in the repository are extended to product development and testing in application engineering. The assets or artefacts are referred to as product assets or artefacts. The major goal of product testing is to ensure that the product being constructed is the actual product specified by the requirements [17].
In application engineering, the application engineer in collaboration with the customers defines requirements for specific product. Upon the defined product requirements, application test model is built by reusing the domain test model. Test cases for product testing are derived. This is done in two steps. The first step is to select reusable test cases from domain engineering. Some of these test cases require adaptation as a result of the selected variability in the product [17].
Secondly, new test cases must be derived when new requirements are introduced for specific products. New test cases are derived as a result of the introduction of new requirements into the test model, in order to satisfy customers' needs. Deriving product test cases in

application engineering is by binding the variability in the domain test cases according to the binding of variability in product requirements. The derivation of product test cases is made possible by the trace link between the different artefacts of the domain.

INTEGRATION TESTING - Several rounds of integration testing are performed in software product line as larger and subassemblies of software are constructed and tested [17].

SYSTEM TESTING - This testing determines whether the specific product works correctly and satisfies product requirements. The test cases used for this testing are structured in such a way that they mirror the commonality and variation points as defined in product line architecture [17].

ACCEPTANCE TESTING - This test activity is carried out in product by customers and product testers to ensure the product meet customer's requirements and quality. Product must be tested before it is delivered to customers for acceptance.

The introduction of SCenTED techniques in the requirements phase of SPLIT Model enables it to capture requirements and start testing at the early phase of software product lines development. Secondly, test cases are also derived from use cases at the early phase of software testing.

Thirdly, the combination of Model based testing and scenario based testing in SPLIT Model creates a testing environment to systematically derive test cases for software product lines.

Reuse of software product line core assets which is basically what software product lines development is centered on is supported in SPLIT Model. Test cases derived in domain engineering for software product lines are extended to product development for reuse. Most Models do not support the reuse of core assets in product lines testing.

The inclusion of core assets repository in SPLIT Model which is not present also in V-Model for single product testing is an added step that facilitates testing in software product line context.

The two development processes in SPLIT Model indicates the model support testing in domain engineering and application engineering in software product line.

The added steps in SPLIT Model in one way or the other helps:

To handle variability complexity in software product lines core assets.

Support the reuse of core assets in product line development and testing, for it to achieve its goals.

To provided a crystal clear picture of the different phases and working concept of software product lines testing process.

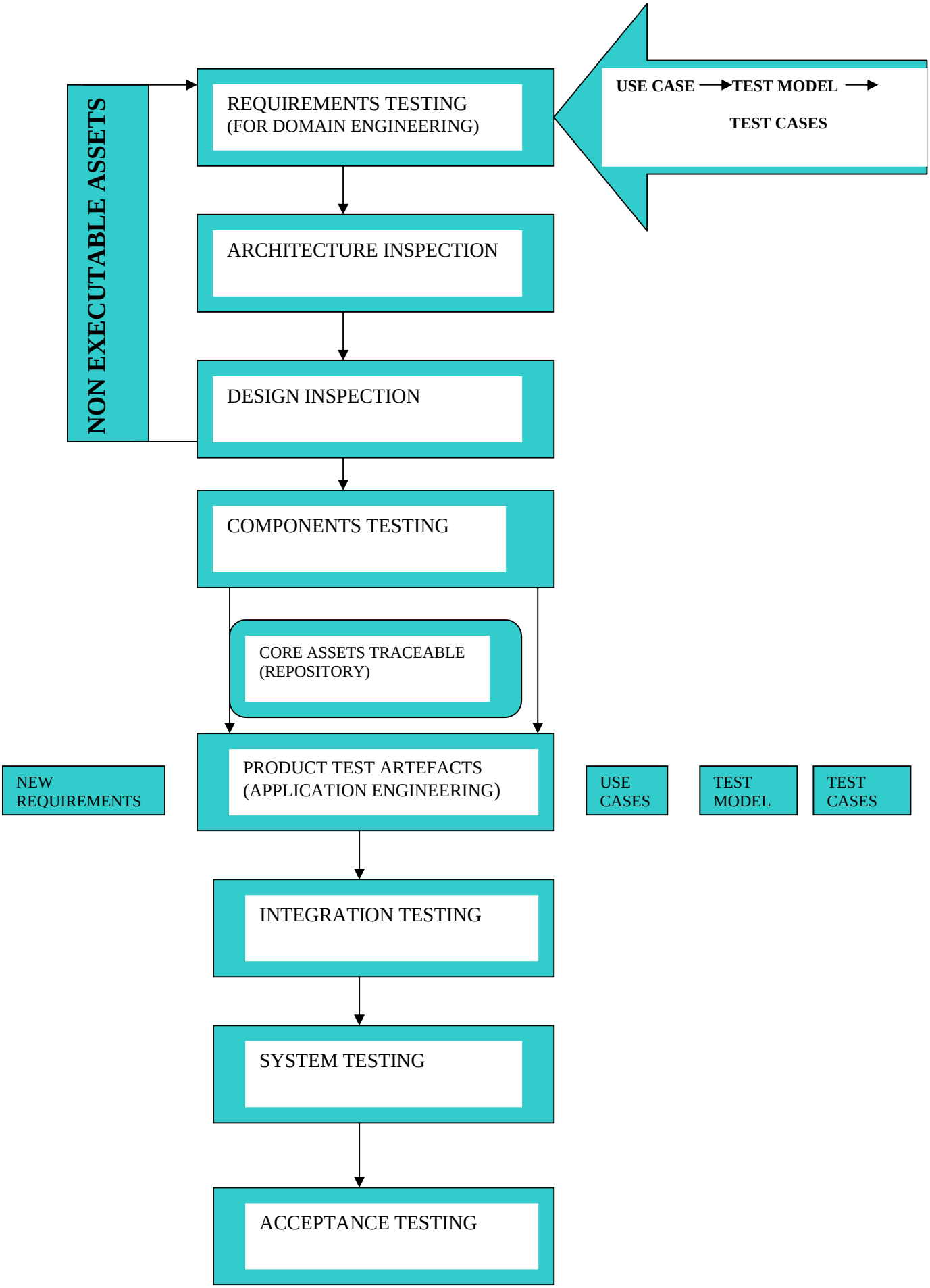

Figure 4.1: SPLIT Model

CHAPTER 5

5 PRODUCT LINES USE CASES BASED TESTING

5.1 INTRODUCTION

Software development for product line is quite challenging and tasking, right from the early phase of development which commences with requirement elicitation to the last phase of testing. Two reasons why it is so challenging developing for product lines and the major differences of software product line engineering when compared to single system development:

There are two development processes in software product line (Domain engineering and application engineering) [50].

The variability at the different levels in product lines design, which includes: Variability in product line level, the architecture level, the component level, and code level [22].

Developers of software product lines have to deal with two development processes and the variability in software product lines in order to achieve the goals of developing quality software at reduced cost and time. Capturing the variations that exist in the set of products belonging to software product lines is a key issue for test case in software product lines. Majority of the variation points are introduced into product lines during the requirements phase [118]. The major challenge and urging question in this process is how to handle and represent variability in software product lines. The delivery of quality software product lines and its specific products on time depends on some critical factors. Among these factors are requirement validation- ensuring that the requirements state the needs of the users correctly and system test- establishing that the implemented system conforms to its requirements [78]. This challenge is reflected in the large attention product lines requirement analysis, specification, and testing has drawn.
Use cases form a kind of abstract test cases for system under development. The idea to use them to create test cases for software product lines is investigated in this chapter. This chapter presents ways to capture these variations characterizing the sets of products belonging to a product lines applying use case approach to make them suitable for easy analysis, helpful in test cases generation and testing in software product lines.

5.2 BACKGROUND

Variability is introduced during project management sub-process where common and variable features of software product line applications are identified. The sum of activities concerned with the identification and documentation of variability in software product line is referred to as defining variability [10]. These variations in products represent the customisation of the specific products with respect to other members of a product line [45].
The first phase of software project is requirement analysis and specification for the system under construction. Software product lines developers are always bothered and thinking on how to handle the problems of capturing test cases common to all members of the product lines on one side, at the other side, how to specialise the general product lines test cases into

specific products [43]. To handle this problem, use case approach was introduced in software product lines.
Use cases from requirement analysis phase of product lines development are often use to create system test cases. To perform the research, primary studies relating to scenarios/use cases in software product lines testing are identified.
The goal of this chapter as it relates to research question 3 is to investigate the approaches use to develop test cases for software product lines and specific products from use cases considering the fact that software product lines core assets are entangled in variability complexity.
The research question to be addressed in this chapter is RQ3. What approaches can be use to develop test cases from use cases that contains variability and to derive application test cases from them? Like the previous chapters of this thesis, systematic review steps are used in this research to conduct the research with respect to RQ3. The research method is used to find articles that provide direct evidence relating to research question 3 of this thesis.

5.3 RESEARCH METHOD

This section presents how we identified articles reporting use cases in software product lines and developing test cases from use cases to perform testing in software product lines. How we extracted data and information from the articles, and how we analysed the articles as they relate to RQ3.

5.3.1 IDENTIFICATION OF RESEARCH

The goal of this research as it relates to RQ3, is to investigate the approaches use to develop test cases from use cases, especially considering the variability complexity in software product lines, and deriving application test cases from use cases. Summary of search strategy is on page 13 of this thesis.
On pages 13 and 14 summary of systematic review process can be found [20].

USE CASE FOR PRODUCT LINES

Use cases are powerful tool to capture functional requirements. They provide a means to specify the interaction between a system and its environment and structuring the requirements according to the users [43] [85].
Software test cases are organised around the requirements and specification under construction [4]. Software product test cases are based on the use case that comprises the product requirements model [4]. The tool helps to structure the requirements document to meet the user goal, and provide a means to specify the interactions between the system and its environment.
The process of constructing software product lines tests commences immediately the requirements process for the product lines begins. There are certain product lines artefacts that lead to test cases. Use cases are created during requirement process, and for each use case created in the requirements process captures one or more requirements that a specific product must satisfy. Use case description contains scenario that describes the usual course, alternative course, extensions, and exception course [4].
The next step towards generating test cases, is defining a domain model. The domain analysis model is concerned with creating the description of the domain from the perspective of the objects involved in software product lines system. Here, there is identification of concepts, attributes that are considered important. The result can be expressed in domain model that shows important domain concepts and attributes [76].

Further more, the interaction design model is concerned with defining software product lines objects, their responsibilities and collaboration. A Common notation to illustrate this collaboration is the sequence diagram e.g. the UML diagram. It shows the flow of messages between software objects and thus the invocation of methods. This interaction diagram is referred to as dynamic view of collaborating objects.
Requirements are captured with use cases, and test cases are derived from captured requirements which are extended by mechanism. The approach describes how to capture and derive requirements with uses cases, and derive test cases that are extended by model to express variability in the captured requirements [83][84].
In addition to dynamic view of collaborating objects, a systematic view of class definition is usually shown with design class diagram. This illustrates the attributes and the methods of the class [76].
An effective and widely used technique for specifying use cases was presented by Cockburn [44]. This technique is based on natural language specification and their extensions which make it very easy to understand and document requirements and communicate to users and non technical people [43].
Use case describes and defines a goal oriented set of interactions usually (triggered by an external actor in order to achieve a goal) between a system under consideration and its environment (external actor) [43]. The term actor is used to describe either a system or a person that has a goal against the system under consideration. There are two types of actor, primary and secondary actor. Primary actor triggers the system behavior in order to achieve a certain goal, while a secondary actor interact with the system but does not trigger the use case [43].
Cockburn use cases can be used to deal with specifications of PL requirements, called PLUCs (product lines use Cases). With PLUCs variations can be described in software product lines requirements by enclosing into the section of the use cases some tags that indicates variable parts of the product lines requirements [43]. There are quite a reasonable number of software development methods and modeling languages comprise some notion elements for use case modeling (e.g. OOSE Object-Oriented Software Engineering/Jacobson/, OMT Object Modeling Technique/Rumbaugh/, and most notably, the Unified Modeling Language /Booch, Rumbaugh, Jacobson/ as their successor) [78]. These modeling languages are used to model use cases to illustrate the flow of messages between the different software agents for the system to be considered for construction, and at the same time help to tackle variability complexity in software product lines core assets.

TEST CASE SELECTION METHODS

The selection of appropriate test cases to test software product lines conformance to given set of requirements specification is important for the success of software product lines testing process. Variability complexity associated with some core assets of software products lines makes it very challenging selecting appropriate test cases to test software product lines conformance to given requirements specifications. There are number of methods propose to derive test cases for software product lines testing that can handle variability issues in software product lines cores assets e.g. requirements. Two of these methods are investigated and presented to understand their approach for deriving test cases from use cases for software product lines and its specific product.

PLUTO METHOD

PLUTO (Product Lines Use case Test Optimization), is a methodology for early derivation of test cases from product lines requirements described as product lines use cases (PLUCs) [21][43]. The methodology is inspired by Category Partition method, but it is expanded with the ability to handle variability in software product lines and to instantiate test cases for software product lines specific product. The approach is based on structural, natural language requirements; therefore, the test derivation has to be done partially manually [21].

The goal of PLUTO is to propose a simple methodology to manage the testing process of product lines, based on use cases requirements. The methodology helps to derive a set of test specifications, and a suite of application specific test cases, when customisation application is defined within software product lines [21] [43].

SCENT-METHOD

SCENT-Method- A Method for SCENario-Based validation and Test of Software [78]. SCENT [78], as the name of the method suggest- a scenario- based method validate and verify requirements (by formalisation and natural language scenarios) and to systematically develop test cases (by path traversal in state charts) [43] [78].The terms scenario and use cases are used synonymously as a result could be used interchangeably. The methodology is widely used today in most software product lines development organizations e.g. Siemens [43].
Scenarios systems views, uses a common way of capturing requirements for software product lines and other systems, and help to uncover hidden requirements, to verify and validate and to integrate analysis of functional and non-functional requirements for product lines [78,79,]. Scenario provides the possibilities of expressing functionalities that span across software product lines. The system requirements and customers' features are represented as scenarios in the early stages of product lines development life cycle.
Scenario from another perspective is an ordered set of interactions between partners, which is usually between a system and a set of actors external to the system [78].
The goal of every software organisation is to deliver software of high quality at reduced cost and time. Though the delivering of high quality software to customer at reduced cost and time depends on several factors, which includes requirement validation-ensuring that the requirements of the products completely and sufficiently meet the needs of the users, and system test-ensuring that the implemented system conforms to its requirements [78].
Since requirements validation and testing are important activities to deliver quality software, it is also very important that these two activities are done in systematic, superficial, planned and methodological way.
To improve requirements validation, the activities should be integrated into the development method and users should be encouraged, and testing can be improved by systematic test cases development and integration of test development methods with normal system development methods [78]. The best way to develop test cases in a systematic way is by clearly defined methods of test cases development.
As mentioned above, use case is a tool to capture functional and non functional requirements, they provide an effective means to specify interaction between a certain software system and its environment and allows for well structured requirements according to the desire of the users. An effective techniques used for specifying use cases was first presented by Cockburn, which is based on natural language specifications for scenarios, which makes requirements documents easy to understand and communicate to even non technical people, including customers.
Use case can be represented in forms. There are three possible representations forms that could be used to document scenarios [78]:
Structured natural language (e.g. using a template)
A tabular representation
A graphical representation
All the above listed representation forms can be enhanced further by the use of pictures, and users interface mask.

5.3.2 STUDY SELCTION CRITERIA

Search criteria aimed at both identifying existing primary studies and assessing the volume of potential relevant studies includes:

Summary of search strategy can be found on page 13 of thesis.

List of combination of search terms use for each database:

Use case AND product lines.
Use case AND test case AND product lines.
Test case AND product lines.
Variability AND test case AND product lines
Product line AND product test case.

IEEE Xplore Electronic Database

Table 5.1: Hints from IEEE database search.

No	Investigated Articles	Selected Articles
1	5	2
2	4	1
3	0	0
4	0	0
5	0	0
Total	9	3

Scenarios systems views, uses a common way of capturing requirements for software product lines and other systems, and help to uncover hidden requirements, to verify and validate and to integrate analysis of functional and non-functional requirements for product lines.

Amyot et al. 1999 [79]
Gomaa et al. 2002 [81]
Gomaa et al. 2006 [122]

ACM Digital Library

Table 5.2: Hints from ACM digital library database search.

No	Investigated Articles	Selected Articles
1	7	0
2	4	0
3	3	1
4	4	0
5	3	0
Total	21	1

Bertolino et al. 2003 [43]
Use cases is use to capture requirement for system, and the technique is based on natural language specification and their extensions which make it very easy to understand and document requirements and communicate to users and non technical people [43].
Captured requirements are model in order to express the variability in the core assets. One methodology to model use cases requirements is PLUTO (Product Lines use case Test Optimization), is a methodology for early derivation of test cases from product lines requirements described as product lines use cases (PLUCs) [21][43].

Springer Link Electronic Database

Table 5.3: Hints from Springer link electronic database search.

No	Investigated Articles	Selected Articles
1	2	1
2	6	1
3	7	2
4	0	0
5	0	0
Total	15	4

The general ideal and approach described by the studies below, support capturing requirements of product lines with use cases and extend use cases with mechanism to express variability. The approach describes how to capture and derive requirements with uses cases, and derive test cases that are extended by model to express variability in the captured requirements.

Bertolino et al. 2004 [21]
Fantechil et al. 2004 [82]
Bertolino et al. 2007 [83]
Fantechil et al. 2004 [84]

Science Direct Electronic Database

Table 5.4: Hints from science direct electronic database search.

No	Investigated Articles	Selected Articles
1	0	0
2	0	0
3	0	0
4	0	0
5	0	0
Total	0	0

Academic Search Elite Electronic Database

Table 5.5: Hints from academic search elite electronic database search.

No	Investigated Articles	Selected Articles
1	0	0
2	0	0
3	1	0
4	0	0
5	0	0
Total	1	0

Software Engineering Institute Database Search

Table 5.6: Hints from software engineering institute database search.

No	Investigated Articles	Selected Articles
1	3	1
2	1	0
3	2	0
4	3	1
5	0	0
Total	9	2

Clements et al. 2002 [2]
McGregor 2001 [4]
The two articles found in the database above, confirm that use cases help to capture requirements of system and define requirement for software product lines. Test cases are derived from use cases, which are later used in product lines testing.

Literatures

Table 5.7: Hints from literature search.

No	Investigated book(s)	Selected Book
1	Software Product Lines: Practices and Patterns 1	1
2	Lerma [76] 1	1
3	Cockburn [44] 1	1
Total	3	3

Use cases can be use to develop test case by going through the steps below.
Capture requirements for systems.
Define requirements model for software systems including software product lines.
Use cases are modeled to define software product lines responsibilities and objectives and collaboration. The articles below share same views as regards the steps involved in deriving test cases from use cases.
Clements et al. 2002 [2]
Cockburn 2000 [44]
Larman 2004 [76]

5.3.3 REVIEW OF RESEARCH RESULTS

The articles investigated explained the importance and usefulness of use cases in defining requirements for software product lines and other software systems.
According to [78][79], Scenarios systems views, uses a common way of capturing requirements for software product lines and other systems, and help to uncover hidden requirements, to verify and validate and to integrate analysis of functional and non-functional requirements for product lines.
Use cases can be used to derive test cases for software product lines and its specific products. This helps to take care of variability in software product lines most especially at requirement analysis phase of software product lines development. Software test cases are organized around the requirements and specification under construction [4]. Software product test cases are based on the use case that comprises the product requirements model [4].
Use cases form a kind of abstract test cases for system under development, though sophisticated methods are used to model use cases to reduce test cases derivation, and they help to handle variability in software product lines. Examples of these models are SCENT method and PLUTO presented in sections 7.3.2.1 and 7.3.2.2. Other methods used to model

use cases are : (e.g. OOSE Object-Oriented Software Engineering/Jacobson/, OMT Object Modeling Technique/Rumbaugh/, and most notably, the UML Unified Modeling Language/Booch, Rumbaugh, Jacobson/ as their successor) [78].
The application of use cases in software product lines testing help to identify the variation points in software product lines requirements at the early phase of the process, and deriving test cases to test software product lines and its specific products.

5.3.4 DATA EXTRACTION

To identify and extract primary studies, the titles and abstracts of primary studies were carefully read. 55 articles published in the selected journals and conference proceedings and 3 literatures were read to make a total of 58 articles and literatures that were read.
From the total of 58 articles and literature read, 13 (22.4%) provided direct evidence relating to research question 3 of this thesis. The selected primary studies discussed use cases and test cases in software product lines and methods to model use cases to ensure that variability complexity in software product lines is handled. Table 5.8 presents list of investigated or scanned and selected articles with respect to RQ3.

Table 5.8: Use cases and test cases in software product lines.

Journals/Conference	Total no. of articles and literatures investigated	Total no. of articles selected. (N)	Percentage of articles selected by row (Row %)
IEEE Xplore	9.0	3.0	33.0
ACM Digital library	21.0	1.0	4.8
Springer Link	15.0	4.0	26.7
Science Direct	0.0	0.0	0.0
Academic Search Elite	1.0	0.0	0.0
Software engineering Institute (SEI)	9.0	2.0	0.2
Institute for Experimental Software Engineering IESE Fraunhofer	0.0	0.0	0.0
Literature (Books)	3.0	3.0	100.0
Total	58.0	13.0	22.4

5.3.5 DATA ANALYSIS

From the search, 58 primary studies were investigated. 13 primary studies provided direct evidence relevant and relating to research question 3 of this thesis. IEEE database search and literatures provided more evidence relating to research question 3.

Number of investigated primary studies = 58
Number of selected primary studies = 13
Ration of investigated to selected articles = 5:1
For every 5 primary studies investigated, 1 always provided direct evidence relating to research question.
Primary studies found from search that provided evidence to research question are quite few.

5.4 RESULTS

It is fundamental in software development to capture system requirements with use cases in order to derive test cases. But, the major difference between single system and product lines when applying test cases in testing process is the mechanism involve in deriving test cases for product lines to express variability in product lines.
From our study, as regards approaches that can be use to develop test cases from use cases that contains variability and to derive application test cases from them. The concepts of authors of some primary studies as it relates to the research question are as follow:
There are three approaches that support the concept of extending reuse to software product lines testing. McGregor [4] and Geppert et al. [87] approach support creating reusable test cases from use cases during domain engineering. Product lines test cases created during domain engineering are derived from natural language requirements and generalize from existing test cases [4]. Test cases derived from use cases during domain engineering should be extended to application engineering.
According to [88] test cases derived in domain engineering for software product lines can be extended to application engineering for reuse. Scenarios fragments are considered in domain engineering and are assembled to test case scenarios. Use case transition graph is used to specify dependencies between use cases. Though test cases are only derived from use cases for specific applications only when variability is already bounded.
Activity diagram was used by Hartmann [89], as test model, which contains variability, but test cases are only derived in application engineering. It is a model based testing approach for deriving test cases from use cases only in application engineering, and does not consider the reuse of test cases. The approaches by [4] [87] [88], support the idea of extending proactive reuse including test cases derived from use cases to product lines testing. While [89], approach support model based testing in software product lines.

5.5 DISCUSSION

Basically, test cases are derived for product lines from use cases scenarios, and there are different approaches to apply test cases created from use cases to application engineering of product lines. While some authors support reuse and extension of test cases created during domain engineering to application engineering, some prefer that test cases are created in application engineering from use cases only when variability in core assets is already bounded.

5.6 CONCLUSION AND RECOMMENDATION

In this chapter we have investigated approaches to develop test cases from use cases and few methods to model use case towards deriving test cases for software product lines testing process. Use cases help to define requirements for software product lines, and help to tackle variability in software product lines requirements which is the first phase of system development. But variability can fully be handled in software product lines core assets by introducing methods to model use cases to handle variability in software product lines.
As regards application of test cases derived from use cases, there are different approaches presented by authors. While some authors support reuse and extension of test cases created during domain engineering, some are of the view that application test cases are created in application engineering, and others propose methodologies that has the ability to handle variability in software product lines and to instantiate test cases from test specification for specific customer product.

It is important to define a standard and specific approach to apply test cases in application engineering of software product lines. Further research is required in this area as regards defined approach to apply test cases for application engineering in software product lines.

CHAPTER 6

6 SOFTWARE REUSE AND REUSABILITY OF TEST CASES

6.1 INTRODUCTION

A software product line is an emerging discipline, and the goals of software product lines are to reduce development costs and Time-to-Market as well as to increase quality of individual applications [2]. These goals can be achieved by proactive reuse [86].
Software reuse can help produce high quality software in product lines development. Software reuse is the process of using existing software artefacts in product lines development rather than building software product lines from the scratch [51].
The major reason for reusing software artefacts in product lines development is to reduce development costs and time by reducing the cost and human effort required to build the software products [53] [54]. Therefore, software reuse in product lines is a welcome development. The reuse of software artefacts across multiple projects is important strategy for improving software development efficiency and increase the quality of software product lines and its specific products.

6.2 BACKGROUND

The reuse of software has been a long-standing aim of the software engineering community. In order to reduce software crisis, at the end of the 1960s the concept of developing systems by composing components was discussed by the engineering community as a promising approach to address the issues and crisis in software [3].
Further more, during the 1970s, modules based approach was proposed to achieve reuse for system construction. Same in the direction of trying to achieve reuse in system development, during the 1980s, object oriented programming proposed the use of class as unit of reuse for constructing system. But, all approaches proposed provided reuse at individual and small-scale components that could be used as building block for new system. [3].
The approaches proposed above did not address reuse at the level of large components that suppose to be used in constructing larger system. This concept and understanding led to development of object oriented frameworks, which aim is to build larger system for a particular domain [3]. The development of object oriented framework helped in developing components that needs to fit and be incorporated into higher level structure product line defined by the software architecture [3].
In this chapter we are going to investigate and discuss the relationship between software reuse and reusability of test cases in software product lines context. The question for this research is RQ4. What is the connection between software reuse and reusability of test cases? The goal of this thesis specifically as it relates to research question 4, is to find out the connections that exist between reuse of software and test cases in software product line context.

6.3 RESEARCH METHOD

This section presents the method use in conducting our research. How we identified articles reporting software reuse and reusability of test cases, how we extracted data and information from the articles, and how we analysed the articles as they relate to RQ4.
The technique to conduct our research is Kitchenham approach of systematic reviews [20]. The basic steps involved in systematic reviews are followed in order for the research to come up with result that is valid as it relates to the research question.

6.3.1 IDENTIFICATION OF RESEARCH

The goal of this chapter is to investigate the relationship between software reuse and test cases reuse in product lines context. To carry out our investigation, we searched for primary studies relating to RQ 4 pose on this thesis, using unbiased searched strategy.

On page 13, summary of search strategy can be found.

List of combination of search terms use to for each database:

Software reuse AND reusability AND test cases.
Software AND reuse AND test cases reuse.
Product family AND software reuse AND test cases reuse.
Product lines AND software reuse AND test cases reuse.
Domain engineering AND software reuse AND test cases reuse.

SOFTWARE REUSE

Software reuse is the process of implementing or updating systems using existing software assets [60]. Software reuse is the process of using existing software artefacts rather than constructing software from the scratch [51]. Software reuse is widely used in many software organisations and industries today globally, example telecommunications firm, automobile manufacturing organisations. This is so, because majority of organisations activities are software driven, and as such they need to under go radical changes forcing many organisations and industries to update their system in order to cope with the rapid technological evolution, new challenges of global market, and increase in demand from users in terms of quality of software products and services [57]. In order for organisations to meet up with these tasks, there is need for them to establish and implement software reuse in their organisation scheme of software product development.
Reuse process involves three steps [55]:

Reusable artefacts are selected from the existing ones.
Artefacts should be adapted to the purpose of the propose application.
It should be integrated into the software product under construction.
From the statements above, existing software assets can be reused to construct new systems. These existing software assets include [4]:
Requirements
Architecture/design documents
Test Plans
Software components
Templates for asset
Design decisions
Codes
Test cases

The reuse of software artefacts across multiple projects and product is a good strategy to improve software development efficiency towards increasing the quality of software development [56] [58] [59].
The major motivation for reuse of software artefacts is that it decreases the development cost for software and reduce the human effort and time required to build improved software by reusing quality software artefacts [52][61]. Software reuse can help reduce maintenance costs due to that fact that, when reusing quality software the time and effort required to maintain the software will be reduced drastically [29] [67].
Software component reuse in development is classified into two directions and types [64]. The first one is development for reuse and the second type is development with reuse [63] [66].
Development for reuse and development with reuse can both be applied in software product lines. Software development for reuse creates reusable core assets for product lines and software development with reuse creates reusable core assets to create products [69].

TEST CASES

Testing software application requires unique test cases. Test cases are sequence of steps to test the correct behavior of a functionality/feature of an application. For testers to perform testing for product lines, they need to derive set of test case, which are conditions and variables use by testers to determine if the requirements of the product lines are satisfied. A good test case should have the capability and high probability to detect undiscovered defect in software product lines and specific products [78].
A typical test case is composed of three parts: initial conditions, input values, and expected results. The initial conditions specify the environment of application for each test case. Input values define the configuration of test cases. Expected results are the outcome of using test case [4]. The product line approach to software development is to reuse core asset in production. Test cases are part of product lines core asset. Furthermore, the development of test case takes much time and resources. Therefore, it is better to reuse test cases during testing. Test cases used for testing in the context of product lines, should be maximized and utilized, and this opportunity is provided by product lines environment [4]. Example, the mapping relationship between product assets and testing assets facilitates the reuse of these test assets [2].
When it comes to test case derivation, the architecture of product line is the basis of test cases. Test cases can be designed increasingly at each level of hierarchy [2]. Even though the abstract test cases are not applied directly in the product level, we can reuse those test cases [2]. The selection and construction of product level test cases should be based on the requirements process of product line [4].

SOFTWARE REUSE AND TEST CASES REUSE

The reuse of software artefacts across multiple projects and product is a good strategy to improve software development efficiency towards increasing the quality of software development [56] [58] [59].
“The various testing activities conducted during the development of components and products create a number of artefacts, some of which are core assets” [4].
The most important artefacts that are produced during testing process include [4]:
Test plans
Test cases and data sets
Test software and scripts
Test reports
“Domain Based Testing (DBT) is a test generation method based on two concepts from software reuse, domain analysis and domain modeling. Because DBT it is based on ideas from software reuse, DBT also provides a good structure for test case reuse” [70] [71].

During domain engineering product lines core assets consisting of sets of reusable assets from which product lines specific products can be derived during application engineering [72]. It implies software product lines core assets, including test cases which are reusable across multiple of products derived in domain engineering for product lines can be reused in application engineering for products development.
A natural core asset of a product family or product lines is a set of software assets that are reused across products derived during application engineering. The reusable assets of product lines help software product lines organization to capture commonalities between applications for the domain they operate in [24].
Software reuse is basically creating software systems from already existing software elements rather than constructing systems from the scratch. Reusable elements in software system and software development process are not limited to source code, but may also include other materials produced in the development process, e.g. guidelines for testing all communication mechanisms, and test cases [73].
Test case scenarios can be generated automatically, but test case specifications are usually developed manually, which is time consuming task. Thus, it is desirable to create both test case scenarios and specifications in domain engineering and to reuse them during application engineering [2] [4] [74]. Product lines organizations can afford to invest in the following core assets that can be reused in product development: Requirements, software architecture, test plans and test cases [2].

6.3.2 STUDY SELECTION CRITERIA

The study selection criteria are intended to identify primary studies that provide direct evidence about the research question. Inclusion and exclusion criteria are based on research question. Summary of inclusion and exclusion criteria are found on page 14 of this thesis.

IEEE Xplore Electronic Database

Software reuse AND reusability AND test cases.
Software reuses AND test cases reuse.
Product family AND software reuse AND test cases reuse.
Product lines AND software reuse AND test cases reuse.
Domain engineering AND software reuse AND test cases reuse.

Table 6.1: Hints from IEEE database search.

No	Investigated Articles	Selected Articles
1	3	1
2	4	2
3	4	1
4	6	2
5	3	1
Total	20	7

Articles selected from IEEE database search.
Poston et al. 1992 [28]
Chaver at el. 1998 [58]
Agresti et al. 1987 [61]
Basili et al 1990 [67]
Boehm et al. 1988 [68]
Mayrhauser et al. 1994 [70]
Mayrhauser et al. 1994 [71]

The general concept shared by selected articles is that; create test cases as part of reusable software artefacts in domain engineering and to reuse them during application engineering. Basically, test cases from the perspective of articles selected in IEEE database search are reusable software artefacts created in domain engineering for product lines that are extended to application engineering for product development.

ACM Digital Library

Table 6.2: Hints from ACM database search.

No	Investigated Articles	Selected Articles
1	0	0
2	5	0
3	8	0
4	7	1
5	10	2
Total	30	3

Results from ACM database search.

Kreger et al. 1992 [51]
Ravichadran et al. 2003 [55]
Denger et al. 2006 [72]

Software reuse is the process of using existing software artefacts to construct software rather than starting from the scratch [55]. Product lines reusable artefacts include requirements and test cases est. [4]. From the definition of one the articles above, it implies that test cases can be extended to application engineering to test product lines specific products. Generally, from the point of view of selected articles from ACM database, test cases are reusable artefacts for testing software.

Springer Link Electronic Database

Table 6.3: Hints from Springer link electronic database search.

No	Investigated Articles	Selected Articles
1	1	1
2	15	0
3	1	0
4	0	0
5	1	0
Total	18	1

Karinsalo et al. 2004 [73]
Reusable software in software development process are not limited to source code, but may also include other materials produced in the development process, e.g. guidelines for testing all communication mechanisms, test cases. Test cases are reusable software artefacts from product lines that can be reuse in product development. Karinsalo Supports the idea that test cases are software that can be reused in developing product lines specific products.

Science Direct Electronic Database

Table 6.4: Hints from science direct electronic database search.

No	Investigated Articles	Selected Articles
1	0	0
2	0	0
3	0	0
4	0	0
5	0	0
Total	0	0

Academic Search Elite Electronic Database

Table 6.5: Hints from academic search elite electronic database.

No	Investigated Articles	Selected Articles
1	0	0
2	0	0
3	0	0
4	0	0
5	0	0
Total	0	0

Product lines organisations can afford to invest in the following core assets that can be reused in product development: Requirements, software architecture, test plans and test cases [2]. Test case is one of the reusable core assets that are reused in product lines development.

Software Engineering Institute (SEI)

Table 6.6: Hints from software engineering institute (SEI) search.

No	Investigated Articles	Selected Articles
1	9	0
2	37	0
3	15	0
4	16	1
5	6	0
Total	83	1

Thus, it is desirable to create both test case scenarios and specifications in domain engineering and to reuse them during application engineering [4]. Test cases are reusable software artefacts developed in domain engineering for product lines, and are reusable in application engineering for products development.

Literatures

Table 6.7: Hints for literature search.

No	Book investigated	Selected Book
1	Software Product Lines: Practices and Patterns	1
Total	1	1

Clements et al. 2002 [2]
Test cases are reusable software artefacts derived in domain engineering for product lines, and reused in individual products development in application engineering process.

6.3.3 REVIEW OF RESEARCH RESULTS

The research firstly, finds out what constitute software product lines core assets, core assets and software reuse in product lines. Thereafter, went ahead to investigate the connection between software reuse and core assets reuse.
According to [4], "The various testing activities conducted during the development of components and products create a number of artefacts, some of which are core assets" [4].
The most important artefacts that are produced during testing process include:

Test plans
Test cases and data sets
Test software and scripts
Test reports
Also, some articles reported the importance of software core assets and artefacts in software product lines.
The reuse of software artefacts across multiple projects and product is a good strategy to improve software development efficiency towards increasing the quality of software development [56] [58] [59].
From our investigation and evidence provided by primary studies relating to research question 4, the selected articles presented discussions relating to research question and the views of selected articles with respect to the research question are quite clear as regards relationship between software reuse and reusability of test cases in software product lines.

6.3.4 DATA EXTRACTION

To identify and extract primary studies, the titles and abstracts of primary studies were carefully read. 151 articles published in the selected journals and conference proceedings and 1 literature were read to make a total of 152 articles and literature that were read.
From the total of 152 articles and literature read, 13 (8.6%) provided evidence relating to research question 4 of this thesis. The selected primary studies discussed software reuse and test cases reuse in software product lines. Below is a table with list of investigated or scanned and selected articles. Table 6.8 presents articles reporting software and test cases reuse in software product lines.

Table 6.8: Software and test cases reuse.

Journals/Conference	Total no. of articles and literatures investigated	Total no. of articles selected. (N)	Percentage of articles selected by row (Row %)
IEEE Xplore	20.0	7.0	35.0
ACM Digital library	30.0	3.0	10.0
Springer Link	18.0	1.0	5.6
Science Direct	0.0	0.0	0.0
Academic Search Elite	0.0	0.0	0.0
Software engineering Institute (SEI)	83.0	1.0	1.2
Institute for Experimental Software Engineering IESE Fraunhofer	0.0	0.0	0.0
Literature (Books)	1.0	1.0	100
Total	152.0	13.0	8.6

6.3.5 DATA ANALYSIS

152 primary studies were investigated from search. 13 of the studies were selected, because they provided direct evidence relating to RQ 4 of this thesis. From IEEE database search, 20 primary studies were investigated, 7 provided direct evidence relating to research question. The database search produced the highest number of articles with direct evidence.
If 13 of primary studies selected from 152 investigated, the odd in the research is
13/152 = 0.09.
The ratio of investigated articles to selected articles=12:1
Ratio 12:1
1 article was selected from every 12 articles investigated in the research.

6.4 RESULTS

Software product lines consist of sets of reusable artefacts that are reused across multiple products during application engineering. Artefacts are the general term for any work product [76]: Code, test case, test plane, architecture, model, software components, design, implementation, and so on. Further more, software components are used to develop software product lines. Software components also consist of reusable artefacts that are extended to product lines development. Test cases are also created during components construction and reused for product lines construction. Basically, the test cases created during components construction and the software components are reusable in software product lines construction.
The primary studies selected, made it clear that test cases are software product lines artefacts. Test cases are amongst software product lines core assets. These software product lines artefacts are reusable across multiple products during application engineering in order to improve software development efficiency toward increasing the quality of software development [56][58][59].
Basically, the relationship between software reuse and test cases reuse is that test cases are natural assets of product family or product lines. Therefore, test cases are set of software assets derived in domain engineering for product lines that are reusable across multiple products testing in application engineering [2] [4] [74].
Finally, just the way software components are reused in product lines construction, so also are test cases derived in domain engineering for product lines are reused in application engineering for products testing. Test cases are part of core assets developed in domain engineering for product lines and are reused in product testing in application engineering. A natural core asset of a product family or product lines is a set of software assets that are reusable across products derived during application engineering [4].

6.5 DISCUSSION

From the evidence provided by the selected primary studies, it showed that test cases are part of software product lines that are reusable during application development. The reuse of software and test cases in product lines development can certainly help software product lines achieve its goals of improve software development efficiency towards increasing the quality of software development.
The idea of proactive reuse should be extended to product lines testing to speed up testing activities in software product line. Base on this concept, reusable test cases are created in domain engineering and extended to application engineering for product testing, rather than deriving new test cases from the scratch.

6.6 CONCLUSION AND RECOMMENDATION

Software product lines development is basically the reuse of software components and reusable core assets. And product line core assets are reused across multiple of products during application engineering. Based on this, it is very important that organisations perform complete testing in reusable software components and use standard and sophisticated methods to derive test cases for testing in software product line.

CHAPTER 7

7 QUALITY ASSURANCE TECHNIQUES FOR SPL COMPONENT

7.1 INTRODUCTION

Product line approach to software development has gain attention in software organisations, as it enables them to develop software products at reduced cost and time of developing and maintaining complex systems and also help organisations to address products customisation for individual customer. For software product line to achieve this promise benefits and goals, it requires high quality reusable software artefacts [72].
Quality assurance in the context of software components reuse will be investigated in this chapter of the thesis. In this chapter the research is based on the techniques for defect detection for software components in product line context. The primary goal is to investigate quality assurance techniques to verify software component quality before reusing them in product line development.

7.2 BACKGROUND

Component based software engineering has the capability to produce reliable domain engineering and application engineering based on the quality of product line reusable software components. The products developed from these components are only reliable if the components are reliable. Therefore, the reliability of software product lines and its specific products depends on the quality of product line reusable components from which the products are constructed [91].
Component based software engineering is a new technology for software development that has been accepted by software organisations for software product line development that aims to construct applications from custom built or commercial off-the Shelf [91]. Components are used today because of the quality of applications developed from components, and also it reduces development cost and time. Therefore CDSE is a source of components reliability [91].
Productivity gains do not depend on time save from software components reuse alone, but also, the ability to omit unit testing from the product lifecycle and reliability gains depends on the degree to which component is trusted by software product lines developers [91]. Basically, there should be means for product lines developers to ensure that components they intend to use for constructing their products are reliable and complete.
In this chapter we are going to investigate how reusable software components are certified for reuse in software product lines context. The research question for this chapter is RQ5. How do testers and developers determine the reliability of the reusable software or components?

7.3 RESEARCH METHOD

This section presents the method used in conducting our research. How we identified articles reporting techniques to verify component reliability for software product line development, and how we extracted information from primary studies with respect to research question 5 of thesis.

7.3.1 VERIFYING COMPONENT RELIABILITY FOR SPL

In the context of software product line, "components means the units of software that go together to form whole systems (Products) as dictated by the entire software architecture for the products lines as a whole" [2].
Some components may be purchased with need for variability. In such cases, the component can be verified in a conventional way, through testing and technical peer reviews [2]. But in the case of developing or purchasing components for software product lines that are associated with variability, it is important the components for software product lines support variability. That means an extra dimension of testing and review is required to evaluate whether the components can be tailored in different ways and used perfectly to meet the different needs associated with software product lines [2].
Components developed for product lines and new use must be verified to ensure that proper functioning of the component is guaranty. To verify the components for reuse in software product line, the testers and most especially the purchaser must make sure that verification and acceptance criteria are clearly defined for each component before its development and the acceptance of components is based on acceptable completion of appropriate reviews and testing [2].
Software components for product lines can be verified by testers and components builders and also by purchasers for acceptance before it has to be used to build products in the following way [2]:
Unit testing must be created for each use and associated version as described in components attached process and confirms that results meet expectations. Testing reusable software components for software product lines development is a means to detect and remove defects from software components before reusing them for product lines development.
An effective complement to testing is to have group of knowledgeable developers to review the component implementation and to evaluate whether it can be tailored and provide the expected functionality in its tailored form.
Quality assurance techniques such as testing and inspection [92] [93] are proven means to detect and remove defects in software to produce high quality software. Inspection in software product line aim is to define defects in the system by strict and close examinations conducted on specification, design, codes, and other artefacts [4]. One major purpose of software inspections is to increase the quality of software artefacts use in constructing product line by detection of faults [95] [97]. Secondly, it provides useful information to decision makers in software organizations, in order for them to take well informed decision concerning the product line [95] [96].
Further more, other techniques propose to analyse the reliability of software components are [98]:
• System level reliability estimation. Reliability estimation is based on the entire system.
• Component based reliability estimation. Reliability estimation of the application is based on the reliability estimation of the individual components and their interconnection mechanisms. The interest of [98] is the second approach. The second technique analyses the reliability of application by first aggregating the reliability of individual constituent components. To perform this task according to [98], a reliability analysis models is for systems whose design is based on independent executions of scenarios. Yacoub [98] introduces the reliability analysis model, Component Dependency Graph (CDG). And Gokha

[99] discuss the flexibility offered by discrete event simulation to analyse component based applications.
Testing and certification techniques are considered by [100], to be essentially required to assess the suitability of COTS components for integration in product line architecture. Though testing and certification is considered suitable, there is need to develop evaluation criteria by which components selection is made. Going by that, The National Product lines Assets Center (NPLACE) [102], an independent software testing facility sponsored by the Air Force Electronic Systems Center, its main goal is to provide decision making information for selecting COTS product [100]. They developed criteria for certifying components to develop product lines.
In addition to the techniques to verify and certify components for reuse, Voas [101] presents black box testing methodology to determining the quality of off the shelf (OTS) components. According to [101], the methodology provides developers with information useful for choosing components for development and defending themselves against imperfect OTS components developed by someone else.

7.3.2 IDENTIFICATION OF RESEARCH

This section describes how articles are identified reporting quality assurance techniques to verify the reliability of reusable software components for software product line development, and how data is extracted from the articles. The research is aimed at investigating techniques to verify reusable software components for software product lines. Therefore, our search criteria should be detailed enough to help the research come up with valid results. Consequently, the keywords used for search must be derived from our research question by using various combinations of search terms, in order to identify existing systematic reviews relevant and relating to research question 5.

List of combination of search terms used for all databases search:

Testing AND software components AND product lines.
Evaluating AND software components AND product lines.
Certifying AND software components AND product lines.
Verifying AND software components AND product lines.
Software components AND reliability AND product lines.

IEEE Xplore Database Search

Table 7.1: Hints from IEEE explore database search.

No	Investigated Articles	Selected Articles
1	6	2
2	3	1
3	4	2
4	2	0
5	5	1
Total	20	6

List of articles selected relating to RQ5.

Articles proposing testing and inspection to verify components reliability for software product line development. The articles share same view, that testing and inspection are quality assurance techniques for reusable software components for product lines development. The primary studies below, selected from the investigated articles proposed

both techniques to verify the reliability of software components before reusing components for software product lines development.

John Morris et al. 2002 [91]
Chillarege et al. 1992 [92]
Geppert et al. 2005 [94]
Thelin et al. 2004 [95]
Voas et al. 1998 [101]
Yacoub proposed Scenario based analysis approach to verify components reliability.
Yacoub et al. 2004 [98]

ACM Database Search

Table 7.2: Hints from ACM database search.

No	Investigated Articles	Selected Articles
1	7	2
2	6	2
3	4	0
4	0	0
5	1	0
Total	18	4

Articles selected in ACM database search proposed testing and inspection for software product lines testing.

Denger et al. 2006 [72]
John Morris et al. 2002 [91]
Martin et al. 1990 [97]

The primary study below proposes use case approach to verify the reliability of software components before reuse in software product lines development. The concept is that use cases can be modeled to define the variability in software components before reusing the components for product lines development.

Blake el al. 2005 [103]

Springer Link Electronic Database Search.

Table 7.3: Hints from Springer link electronic database search.

No	Investigated Articles	Selected Articles
1	0	0
2	0	0
3	0	0
4	0	0
5	0	0
6	0	0

Literatures

Table 7.4: Hints from literature search.

No	Investigated book(s)	Selected Book
1	Software Product Lines: Practices and Patterns	1

Clements et al. 2002 [2]

The literature proposed testing and review to verify components reliability

Science Direct Electronic Database Search

Table 7.5: Hints from science direct electronic database search.

No	Investigated Articles	Selected Articles
1	0	0
2	0	0
3	0	0
4	0	0
5	0	0
6	0	0

Software Engineering Institute site search

Table 7.6: Hints from software engineering institute site search.

No	Investigated Articles	Selected Articles
1	5	1
2	3	0
3	4	0
4	0	0
5	0	0
6	12	1

McGregor 2002 [4]
Inspection in software product line aim is to detect defects in the system by strict and close examinations conducted on specification, design, components, codes, and other artifacts. Testing is mainly use for software quality assurance, tough it can be complemented by inspection. According to [4], testing is a quality assurance technique for software components that can be complemented by inspection.

7.3.3 REVIEW OF RESEARCH RESULTS

The research showed different techniques used to verify the quality of reusable software component for software product lines development. Two major techniques proposed in primary studies to verify components quality and reliability are testing and inspection/ review.
Next to testing and inspection techniques is scenarios/ use cases. Though, Denger [72] argued inspection to be the best quality assurance technique for software components compared to testing, based on the result of experiment carried out with the two different quality assurance techniques.
The other technique proposed in primary studies is certification techniques, defining standard criteria to certify components for reuse. For certification techniques, set of criteria are specified which components must fulfill before they can be reuse for software product lines development.
All quality assurance techniques mentioned above, to verify software components for reuse in software product lines development were proposed by selected primary studies. To ensure quality in software components for product lines development, the techniques proposed by the different articles from our research should be used. The techniques ensure that software components are complete and correct before reusing them in product lines development.

7.3.4 DATA EXTRACTION

Data was extracted from primary studies after careful reading of primary studies. Primary studies were read comprehensively to comprehend the contents of primary studies in order to report result accurately.
All 12 articles including a literature describing techniques to verify software components reliability were assessed of 51 articles and literature identified from research published in leading software journals and conferences. Table 7.7 presents articles reporting techniques to verify the reliability of software components for product lines development.

Table 7.7: Articles and literature reporting verifying components reliability for product lines.

Journals/Conference proceedings.	Total no. of articles and literatures investigated.	Total no. of articles selected (N).	Percentage of articles selected by row (Row %)
IEEE Xplore	20	6	30.0
ACM Digital library	18	4	22.0
Springer Link	0	0	0.0
Science Direct	0	0	0.0
Software engineering Institute (SEI)	12	1	33.0
Literature (Books)	1	1	100.0
Total	51	12	23.5

To number of articles investigated = 51
Total number of articles selected = 12
Ratio of investigated article to selected 50: 12 = 4:1
Ratio 4:1.
From the ratio, for every four articles investigated, one is selected and provides direct evidence relating to research question 5.
23.5 % represents the percentage of articles that provided direct evidence to research question 5. The value indicates that more research needs to be carried out in this area. There are not enough primary studies that provide direct evidence as regards quality assurance techniques for reusable software components in software product line context.

7.4 RESULTS

This section presents the result of our research with respect to question 5. The 12 primary studies in this review describe techniques to determine software components reliability in software product line. From the 50 articles investigated, 12 provided direct evidence with respect to RQ5 of this thesis. Different articles search from list of databases proposed different quality assurance techniques for software components, to guaranty high quality in software components before reusing software components for software product line construction. We discovered cases of duplication of two articles in two different databases.
Table 7.8, list articles relating to techniques proposed to verify software components quality for reuse in software product lines development. The search showed that more articles proposed testing and inspection / review as major techniques to verify software components reliability for software product line. While two articles support the use of scenarios/ use cases to verify software components reliability. And one article proposed using certification criteria to select components for reuse in software development.

Table 7.8: Techniques to verify software components reliability for software product lines.

Techniques	Articles
Testing and inspection	Denger el al. 2006 [72]. Morris et al. 2002 [91]. Chillarege et al. 1992 [92]. Geppert et al. 2005 [94]. Thelin et al. 2004 [95]. Martin el al. 1990 [97]. Voas et al. 1998 [101].
Use cases scenarios	Yacoub el al. 2004 [98]. Blake el al. 2005 [103].
Certification using criteria	Voas 1998 [101].
Testing and review	Clements et al. 2002 [2].

7.5 DISCUSSION

The result of our investigation as regards techniques to verify software components reliability confirmed that testing and inspection are the major techniques to verify the quality of reusable software components for software product lines development. The research did not only lead to some interesting results, but also encourages further research into quality assurance techniques in software product lines contexts, other than inspections and testing which are mostly used to verify software component reliability in product lines context. Related to this issue of improvement and further research to quality assurance techniques is the defect detection capability of techniques in earlier phases of product lines development and effect in final products.

7.6 CONCLUSION AND RECOMMENDATION

There are two development processes in software product lines, development for reuse (domain engineering) and development with reuse (application engineering). Further more, this brings about variability in software product lines core assets. The result of our research showed traditional quality assurance techniques can not be used in software product lines context.

Due to the fact that there are two development processes in software product lines, the quality assurance techniques for software product lines should be different from that of single systems. In software product lines, specific aspects should be addressed. These aspects include modeling variability in product lines artefacts and the use of common artefacts in product line specific applications. These aspects are not addressed in currently existing quality assurance techniques used for single products.

If quality assurance techniques are applied in software product lines same way they are applied in single product development, this can result in low quality of software product lines and specific products quality, and that obviously, can reduce the benefits and goals of reuse in software product line.

Finally, taking into consideration the differences between product line development and single product development, there is need to carry out further research as regards techniques to verify reusable software components reliability in software product line context that can detect variant specific defects in software product, rather than using single product quality assurance techniques.

CHAPTER 8

8 CHALLENGES OF SPL TESTING

8.1 INTRODUCTION

Software product line as it is today has become a major area of software engineering used in industrial software development, and many organizations have started considering software product line as state of practice. Considering the benefits associated with product line, it is worth adopting by software organisations and product line developers for their development. One of the benefits of software product line is delivering software product with high quality.
Testing is a sub-process in software product lines development process. It is a process to detect defects in software. But, there are some complexities associated with product line testing process. Software engineering and testing community have addressed many of these technical challenges and proposed solutions to these challenges in the last few years. In this chapter, we are going to investigate articles that have discussed this subject area in software product line context.

8.2 BACKGROUND

One of the quality assurance techniques adopted today in software development is testing. Testing is one technique for quality assurance in software product line. One topic that needs greater emphasis today in product line is testing. Product line testing is important and crucial to successfully establish product line engineering technology in product line organisations.
As important as testing is in software product development, there are challenges associated with testing process in software product line context. In this chapter, we are going to investigate into the issues by using our research question 6 of this thesis. The goal of this chapter is to highlight the challenges of product lines testing. The question to be answered in this chapter is: (RQ 6). What are the challenges of software product lines testing?
Systematic review approach is used to find answers to the question. The section below presents the strategy used by this thesis to find primary studies that provide direct evidence relating to research question.

8.3 RESEARCH METHOD

This section presents the method used to conduct research in the subject area, challenges in software product lines testing with respect to research question 6 of the thesis. The method includes: Identification of research, Selection of primary studies, quality assessment of primary studies, data extraction and data analysis. These systematic review steps are followed to conduct our research with respect to RQ 6 pose on this thesis. On pages 13 and 14 is a summary of systematic review steps.

8.3.1 CHALLENGES OF SPL TESTING

There are quite a number of challenges facing testing today in software product lines context. There are issues and challenges that seem to pop up the moment testing commence in software product line context. It is important that software product line organizations pay attention to these challenges and issues early enough during software product line development and testing process.
The goal of this chapter is to address some of the fundamental challenges of testing in software product line context. Then investigate possible ways to manage these challenges facing testing in software product line settings.
The challenges of software product lines testing includes both technical and non-technical. This chapter presents discussion on both technical and non-technical challenges facing product lines testing.

8.3.2 IDENTIFICATION OF RESEARCH

The goal of this research as it relates to RQ6, is to investigate the challenges facing testing in software product line context. To conduct our research, primary studies relating to research question 6 are searched from list of databases used for this research, using unbiased search strategy.
One area posing serious challenge for product lines testing is component reuse. A software product line is centered on reuse of software components. It is important that these components are tested and confirm reliable before reusing these software components for software product lines development. For components buyers, this poses serious challenges. COTS products pose challenges for testing at both product and system levels [2]. COTS products must be tested just like testing any piece of software. But the challenge here is that less is known about COTS product than the software you have developed yourself. Based on this, it is always difficult to have the sources code to use as a basis for testing [2].
COTS products pose new challenges for system integration testing [2]. While some COTS products provide information that is helpful to developers and testers in integration, there might be some unanticipated behaviors from COTS that might be difficult to track since the source code are not available [2].
Further more, it is important to establish and use robust testbeds that includes interface to an external systems and capacity for simulation and testing [2]. Considering the fact that organisations that purchase COTS do not have the product source code and can never know completely how the system works, it is important to use testbeds. Testbeds are essential for the development and maintenance of COTS-based systems [2].
Software development for product line is quite challenging and tasking, right from the early phase of development which commences with requirement elicitation to the last phase of testing unlike developing for single system. Two reasons why is it so challenging to develop software product lines and the principle differences of software product lines engineering when compared to single system development [50]:

Two key development processes are involved in product lines engineering (domain engineering and application engineering processes).
Representation and management of variability in software product lines core assets poses a lot of challenge.
The two development processes and variability associated with product lines core asset poses majority of challenges for software product lines testing. It is very difficult for testers to test product lines core assets that contain variability and at the same time, testing for two processes, (Domain engineering and application engineering).
Software testers are always under constant pressure to reduce the time and cost needed for software testing from management and at the same time they are battling with more and

complex software, most especially product lines software associated with variability complexity as noticed from our studies.
The lack of sophisticated and specific tools support for product line testing is responsible for product line not been able to tackle conflicting demands of time, and complexity associated with software product lines testing [2][4].
One reason why there is lack of tool support for software product lines testing is as a result of lack of test planning. If proper attention is paid to test planning and preparation, the tools required for testing will be identified [41].
Software product lines development is very complex and expensive. To achieve high quality in software product lines and its specific product, enough resources should be devoted to testing process. If enough resources are not devoted, then anticipated high level of test assets and software reuse will not be realised. When this happens, the quality of the software products lines and specific product is affected [42].
Product line testers and developers sometimes find it difficult constructing product line software as a result of variability complexity associated with product line test cases.
Capturing the variations that exist in the set of products belonging to software product lines is a key issue for test case generation in software product lines [43]. Developers and testers in software product line organizations lack adequate training to tackle the problem of variability in core assets testing. It is very challenging identifying variability and commonality for validation in software product line core assets. The key challenge of validation at product layer abstraction is the interaction between differing combination of variability binding [11] [12]. New structures are always tested in software product lines. Testing software product lines architecture with new structure is also a challenge for software product lines testing. Challenges arise in the form of a new structure that must be tested, how to deal with optional elements or with the magnitude of products that may be present [49].
Software product lines development is one of the most promising resent advances for efficient software development. Though, there are few methodologies available to develop and deploy product lines beyond existing domain engineering approach. The approach has being unsuccessful within commercial enterprises because of their deployment complexity [45].

8.3.3 STUDY SELECTION CRITERIA

This involves identification of primary studies that provide direct evidence about the research question. On page 14, a summary of inclusion and exclusion criteria used for our research can be found.

List of combination of search terms used for each database:

Challenges AND product lines AND testing.
Testing challenges AND product lines.
Product family AND testing challenges.
Product AND lines AND testing challenges.
Survey AND product family AND testing challenges.

IEEE Xplore Electronic Database

Table 8.1: Hints from IEEE database search.

No	Investigated Articles	Selected Articles
1	1	0
2	1	0
3	0	0
4	4	0
5	0	0
Total	6	0

ACM Digital Library

Table 8.2: Hints from ACM database search.

No	Investigated Articles	Selected Articles
1	1	2
2	1	0
3	7	2
4	4	1
5	7	1
Total	20	6

Result from ACM database search.
Polh et al. 2006 [11]
Metzger et al. 2003 [12]
Bertolino et al. 2006 [43]
Polh 2006 [50]
The primary studies above have same views, that the major challenges for software product lines testing are variability in software product line core assets and the two development processes involve in software product lines. These pose some challenges for software product lines testing.
Bayer et al. 1999 [45] pointed out that one of the challenges facing software product lines testing is maintenance of source code for components. It is challenging for software developers to maintain the source code of software product line components, must especially if the organizations reusing the software components are not the original developers.
Kolb et al. 2003 [120] testability is a serious issue in software product line core assets. The ease of testing is a problem in software product line as a result of variability in software product line specific products. That makes it difficult to detect variant specific defects. Further more, there are few techniques to measure and improve testability in software product lines.

Springer Link Electronic Database

Table 8.3: Hints from Springer Link database search.

No	Investigated Articles	Selected Articles
1	1	0
2	1	0
3	0	0
4	1	0
5	0	0
Total	3	0

No primary study relating to research question found in Springer link database search.

Science Direct Electronic Database

Table 8.4: Hints from science direct database search.

No	Investigated Articles	Selected Articles
1	1	0
2	1	0
3	1	0
4	1	0
5	0	0
Total	4	0

No studies found relevant to research question.

Academic Search Elite Electronic Database

Table 8.5: Hints from Academic Search Direct database search.

No	Investigated Articles	Selected Articles
1	0	0
2	0	0
3	0	0
4	0	0
5	0	0
Total	0	0

No primary studies found in database relating to research question.

Literature

Table 8.6: Hints from Literature.

No	Book investigated	Selected Book
1	Software Product Lines: Practices and Patterns	1

Clements et al. 2002
Components' testing is one of the issues facing product lines testing. It is very challenging testing software components for reuse in software product line organizations. The reasons are: Variability complexity involved in software product lines core assets makes it difficult to test software components for reuse. COTS products most be tested just like testing any piece of software. But the challenge here is that less is known about COTS product compare to the software you have developed yourself. Based on this, it is always difficult to have the sources code to use as a basis for testing.

Software Engineering Institute (SEI)

Table 8.7: Hints from Software Engineering Institute search database search.

No	Investigated Articles	Selected Articles
1	5	1
2	7	2
3	11	2
4	6	1
5	4	2
Total	33	8

McGregor 2001 [4]

Clements et al. 2002 [117]
The articles found on site focused on product lines two development processes as the main challenges in software product lines testing. It is difficult to test for software product lines, because testers have to test core assets in domain engineering and product assets in application engineering process.
Secondly, lack of tools support for product lines testing process is also challenge.

8.3.4 DATA EXTRACTION

To identify and extract primary studies, the titles and abstracts of primary studies were carefully read. 66 articles published in the selected journals and conference proceedings and 1 literature were read to make a total of 67 articles and literature that were read.
From the total of 67 articles and literature read, 15 (22.4%) reported challenges in software product lines testing. These articles presented discussions on challenges of software product lines testing. Below is Table 8.7, it presents list of investigated and selected articles from search.

Table 8.7: Articles reporting challenges of software product lines testing process.

Journals/Conference	Total number of articles and literature investigated	Total no. of articles selected. (N)	Row %
IEEE Xplore	6	0	0.0
ACM Digital library	20	6	30.0
Springer Link	3	0	0.0
Science Direct	4	0	0.0
Software engineering Institute (SEI)	33	8	24.0
Institute for Experimental Software Engineering IESE Fraunhofer	0	0	0.0
Literature (Books)	1	1	100.0
Academic search	0	0	0.0
Total	67	15	22.4

8.3.5 DATA ANALYSIS

From the research conducted, based on research question 6 of the thesis, it shows that from the 67 primary studies and literature investigated with respect to our research question, 15 (22.4%) articles were selected. The selected articles provided direct evidence relating to research question 6.
Number of investigated articles = 67
Number of articles selected = 15
Ratio of investigated articles to selected = 67:15
Ratio = 5:1
The outcome of the study showed that very few articles reported challenges in software product lines. It means for every 5 studies investigated 1 provided direct evidence relating to research question 6. More so, 22.4 % selected studies provided direct evidence relating to research question 6 from 67 primary studies investigated.

8.4 RESULTS

The study showed quite a number of challenges facing software product lines testing. Software product lines testing community has addressed many of the technical challenges and non technical challenges and proposed solutions to these challenges facing product lines testing. But still there are certain issues still remain not addressed properly.
The challenges of testing in software product lines context from our study includes:

- Techniques for strategic reduction in test cost and time
- Managing variation point in software product line
- Components Reliability in SPL Context
- Design for testability
- Test case design techniques and test cases generation for product lines
- Quality assurance techniques for software component reuse in software product lines context.
- Establishing well defined testing steps and procedures to perform testing in software product lines.

Techniques for strategic reduction in software product lines testing time and cost

Software testers are always under constant pressure to reduce the time and cost needed for software testing from management and at the same time they are battling with more and complex software, most especially product lines software associated with variability complexity as noticed from our studies.
Software product line lack sophisticated and product line testing specific tools. This creates problems for software testers and developers. Software organisations spend time modifying tools to support product line testing.
Software product lines organisations can reduce testing time, cost and pressure on testers by using sophisticated supporting tools for software product lines testing. This can only be identified with proper test planning; otherwise it will be difficult to overcome this challenge.

Managing the growth of variability in software product line

The major challenge of testing in software product line is variability in product lines core assets for specific products. The source of this complexity within product lines originates from the number of variants. Variation points are often source of defects in software product lines.
Testing all variants of core assets is usually impossible especially in large and medium scale software product lines but possibly in simple cases. Further investigation is required in this area of software product lines. The research should focus on how to test for variant specific defects using appropriate quality assurance testing techniques and sophisticated model that can handle variant specific defects rather than general and common defects.

Components Reliability in SPL context

For software product lines to be successful and achieve its goals and promised benefits, it requires high quality reusable software artefacts to construct product lines.

Software components are reused to develop software product lines and its specific products. The quality of the reusable component is very important to ensure that the product lines and its specific products are reliable. The reliability of software product lines and its specific products is a true reflection of component reliability.
Software developers need to select the right quality assurance techniques to verify component reliability. It is quite difficult to select quality assurance techniques for reusable software components. More research is required in this area, as regards to quality assurance techniques for reusable software component to detect variant specific defects in software components.

Establishing well defined testing steps and procedures to perform testing in software product lines.

Before testing begins, it is important that a documented testing procedure in product lines is in place. Organisations using procedures that are not clearly and well defined often results in time delay and cost overrun in software product lines development and testing. Time and cost of development is crucial to software organization, two things software developers and marketers cannot afford, most especially when staying in the competition is crucial.
All software product lines testing activities and steps form the procedure that should be considered by developers in order to have clear and completely defined testing procedure for software product line. These activities should be followed as prescribed in organisation documented testing procedure. Effective testing procedures must encompass all aspect of software product lines testing. The procedure should define the people that should be involved in software product lines testing, and skills of each team members. For more information on challenges of software product lines see [7] [47] [48] [49].

8.5 DISCUSSION

The challenges facing testing in software product lines context are both technical and non technical. And if proper attention is not given to these issues they will definitely affect testing process in software product line. The section above highlights some of the challenges facing product lines testing that require serious attention.
Majority of the tools used in supporting product lines are not sophisticated enough to handle variability complexity in software product lines. Product line development effort produces larger sets of more complexes, and longer artefacts than traditional or conventional development effort. Support tools are required to handle larger and more complex sets of diversely related artefacts. The tools analyses artefacts contents and traverse relationships among artefacts, and manage multidirectional change along these artefacts [30]. But the current tools support for product lines testing and development do not meet these needs.

8.6 CONCLUSION AND RECOMMENDATION

Software product line approach for product development is strictly business inclined and goal oriented type of software development process. For software product line to achieve its goals and economical success, it is relevant product line organisations understands and attach important values to testing in software product line context.
Therefore, software engineering community, researchers and practitioners need to find sophisticated models for software product lines testing. Researchers should focus on useful models that can help all product line technical personnel, testers, developers to understand and communicate what to do, who is responsible and how to do it in product lines testing. This will definitely go a long way in software product lines testing.

CHAPTER 9

9 SOFTWARE PRODUCT LINE TESTING TOOLS

9.1 INTRODUCTION

There have been some sorts of engineering tools in the market since the 1980s that were available to software analyst, designers and developers. During this period, there were no tools available to software testers to detect defects and demonstrate that the software works as specified [25].
Testing is an important technique widely used to measure and ensure software quality. It is an important part of software development project. The testing phase in software project is time consuming and takes up to 50% of the total project effort, though it contributes significantly to product cost [27]. Software is prone to changes and changes can influence the result of test. Based on this, testing has to be performed regularly [26] [27]. Software testing is a very challenging activity, time consuming and labour intensive, it is important and desirable to construct tools that facilitate software testing process [28] [29].
In this chapter we investigate the test tools available to support product lines testing process that can help to produce quality software product line and specific products at reduced cost and time.

9.2 BACKGROUND

A software product line is a large system, with large volume of source code, and such automated tools can aid developers and testers to scan through the large volume of product lines source code and reveal unreliable program features to the human program. Testing tools can support testing large size of software product lines to achieve its goals.
Most software development projects are unsuccessful in terms of software specification. The final software products delivered to customers do deviate from specifications [40]. In spite of the effort and cost invested in software testing, software systems are not free from defects, despite testing software products, faults are still detected.
To improve the reliability of software product lines and its specific products and reduce the effort, time and cost of testing in software product line, automated tools are introduced. “Automated tools in software can be defined as programs that check the presence of certain software attributes which could be program syntax correctness, proper program control structures, and testing completeness”[40].
To answer one of the research questions pose on this thesis: RQ7. What tools and activities should testers and developers adopt to ensure software product lines assurance during testing process?
Systematic review steps are used to find answer to the question. The section below presents the strategy used by this thesis to find primary studies that provide answer to research question 7.

9.3 RESEARH METHOD

This section presents how we identified articles reporting product lines testing tools, how we extracted data and information from the articles, and how we analyzed the articles with respect to the research question.

9.3.1 IDENTIFICATION OF RESEARCH

The goal of this research as it relates to research question is to investigate the tools that can be used to support testing in software product lines context to ensure high quality in software product line and its specific products. On pages 13 and 14 a summary of systematic reviews process can be found [20].

SOFTWARE PRODUCT LINE TESTING TOOLS

Automating testing tasks is important and relevant in product line organisation, considering the fact that multiple versions of the tests that will be reapplied in multiple iteration on a single product and across multiple products [4].
Tools support for software product lines testing is of great benefits and advantage to software product lines testing. Product lines and single product testing consist of different activities; consequently they require comprehensive and careful attention to most if not all of the areas in software product lines testing process. Therefore, tools support individual tasks in product lines practice areas. Tools can help software product line suppliers to develop specific products right from requirements to architecture to component, unit and system testing. Example, configuration management tool can be used to provide traceability between tests, requirements and specification they will verify and the artefacts to which they are applied [4]. To increase the benefits of automation, the following principles are followed [4]:
Each unit of automation should correspond to natural unit in software product lines development process: for example test cases and other artefacts should be grouped according to the components to which they apply. Each unit of automation should be sufficiently abstract and robust to allow for changes when necessary.
Acquirers can as well use tools to validate suppliers' product. Tools help to capture information and provide information at different levels of detail. Basically, either to the suppliers and acquirers of software products, tools provide consistent and comprehensive environment for suppliers to create products and for acquirers to validate the completeness of suppliers' development task [2].
Test case creation is time consuming in software product development. This effort can be reduced by creating test case starting at product line level, and then extend them to specific product. More so, the application of these test cases is another challenging task that is time consuming.
The automation of test cases creation and execution can be achieved in one of the following ways [4]:
Custom test harnesses, test environments such as Junit provide execution environment for test cases that have been constructed using basic programming language. These harnesses are useful at unit testing level or simulation environment for embedded systems. Object oriented programming language can be used to create flexible environment that can easily adapt to changes in specifications and implementation.
There are numerous tools available in testing, despite this, there are very few that support product lines testing. There are separate tools that provide testing environments for unit testing such as Junit framework and the Roast framework [24].
Test Script, software product lines testing requires an environment for test cases execution and language for constructing test cases. According to [4], test tools provide an environment for executing test cases and a language for constructing test cases. These languages are

becoming sophisticated that the line between custom programming and environments and commercial tools is obscure. These tools provide a means of executing scripts and few provide repository in which test results can be stored in software product lines context. Example of these tools is Junit [4].
Test-Case Generators, they use templates to generate test cases with fixed format and limited variability. Generator do not include environment for executing test cases, rather they provide guidance about the content of individual test cases. Generator can lead to complete testing, because as each new test case is generated new inputs can be selected.

9.3.2 STUDY SELECTION CRITERIA

The aim of systematic review is to find out as many primary studies relating to research questions as possible using unbiased search strategy [20].
Search criteria aimed at both identifying existing primary studies and assessing the volume of potential relevant studies includes:
Summary of search selection criteria can be found on page 14 of thesis.

List of combination of search terms for each database:

Software AND product lines AND testing tools
Testing tools AND product lines
Tool support AND product lines
CASE tools AND product lines
Evaluation AND CASE tool AND product lines
Selection AND CASE tools AND product lines

IEEE Xplore Electronic Database

Table 9.1: Hints from IEEE database search.

No	Investigated Articles	Selected Articles
1	4	0
2	4	0
3	8	0
4	6	1
5	4	0
6	1	0
Total	27	1

Kolb et al. 2006 [120]
The only tool identified from database search is DOORS and REMAP.
The tools support requirements management in software product lines.

ACM Digital Library

Table 9.2: Hints from ACM database search.

No	Investigated Articles	Selected Articles
1	6	1
2	0	0
3	5	0
4	3	0
5	2	0
6	3	0
Total	19	1

Tevanlinna et al. 2004 [24]
One primary study provided direct evidence to the research question. The tools that support product lines testing were mentioned.
There are separate tools that provide testing environments for unit testing in product lines testing environment, such as Junit framework and the Roast framework.

Springer Link Electronic Database

Table 9.3: Hints from Springer Link database search.

No	Investigated Articles	Selected Articles
1	2	0
2	3	0
3	5	0
4	9	0
5	0	0
6	0	0
Total	19	0

Science Direct Electronic Database

Table 9.4: Hints from Science Direct database search.

No	Investigated Articles	Selected Articles
1	0	0
2	0	0
3	0	0
4	0	0
5	0	0
6	0	0
Total	0	0

Academic Search Elite Electronic Database

Table 9.5: Hints from academic search elite electronic database search.

No	Investigated Articles	Selected Articles
1	0	0
2	0	0
3	2	0
4	4	0
5	0	0
6	0	0
Total	6	0

Other sources

Literatures (Books)

Table 9.6: Hints from literature.

No	Book investigated	Selected Book
1	Software Product Lines: Practices and Patterns	1

Clements et al. 2002 [2]
The above primary study proposed Junit; a tool that can support testing is software product lines. It provides a testing environment for test case execution and a language for constructing test cases.

Software Engineering Institute (SEI)

Table 9.7: Hints from Software Engineering institutes.

No	Investigated Articles	Selected Articles
1	9	0
2	5	0
3	2	0
4	2	1
5	0	0
6	0	0
Total	18	1

There are numerous tools that support testing, but very few support products lines testing.
Example of tool that supports product lines testing is Junit, others are: [4]
Software component is supported by Rational Rose, plus CORBRA-compliment framework.
Architecture is supported by Rose, alas, PowerPoint, ADL.
Documentation supported by DocExpress/Word integrated with DOORS/Rose products.
Process Supported by Rational ClearGuild
Repository supported by Rational ClearGuild

9.3.3 REVIEW OF SEARCH RESULTS

From our search, 3 articles and 1 literature, total 4 report tools in software product lines testing process of the 90 articles scanned with respect to research question 7 (RQ7), using the research search strategy and search criteria. A total of 89 papers and 1 literature were assessed, 90 papers including a literature analysed in depth.
The result gathered from primary studies analysed in depth with respect to the research question, we can deduce from primary studies:
There are few tools available specifically for product lines testing to cover the entire testing process. Though, the few available tools work in each of the phases of the life cycle to support many of the activities necessary to transform a set of identified customers need to useful product. (Requirements phase, design phase execution phase, implementation, testing phase and maintenance)[2] [25]. Requirement tool for software product lines are DOORS and REMAP [121]. They help to manage requirements in product lines.
Selecting tools for product lines testing, from the few tools available to support the business goals and technical needs of developers is very challenging [2], because of the two development processes involved. Based on that, certain criteria should be followed when choosing CASE tools to support software product lines testing process.
Examples of criteria include tool features, cost, vendor stability, training, interoperability with other tools, and process implications [30].
Tools deployed to support product lines testing should support both core assets and product testing process. The support for core assets testing must focus on commonality and variability of core assets from which products are developed. All core assets development requires tools, which includes tools for documents, coding, and test support.
Example:
Tools support for product lines testing should capture common and variant capabilities of product lines.
Product lines requirements requires tools for describing requirements for products in product lines.

Product lines architecture requires tools that describes architecture for product in product lines and takes into consideration variability and commonality in product lines core assets.
Product lines components requires tools to create and test components that are consistent with commonality and variability.
Also, tool support in product development depends on the level of automation involved in the construction of individual products. Product development may be partially or fully automated, depending on the organisations. Tools specific to product lines can be used to generate the product, since tool support is extensive.
From our studies, current tool support for product lines is based on different tools used in software engineering activities for traditional development and testing process, extended to accommodate software product lines testing process [30]. The criteria for tool selection for software product line should be evaluated to see if the selected tools worth using in software product lines testing process. In [31], evaluation process is described; it presents criteria to select tools for software product lines testing. According to [32] [33], they describe the characteristic of tools by which they can be selected and evaluated for use in software product lines testing process.

9.3.4 DATA EXTRACTION

To identify and extract primary studies, the titles and abstracts of primary studies were carefully read. 89 articles published in the selected journals and conference proceedings and 1 literature were read to make a total of 90 articles and literature that were read.
From the total of 90 articles and literature read, 4 (4.4%) reported tools in software product lines testing process. The articles described tools that can support software product lines testing process. Table 9.7 presents list of investigated and selected articles.

Table 9.7: Articles reporting testing tools for software product lines testing process.

Journals/Conference	Total no. of articles and literatures investigated	Total no. of articles selected. (N)	Percentage of articles selected by row (Row %)
IEEE Xplore	27	1	3.7
ACM Digital library	19	1	5.2
Springer Link	19	0	0.0
Science Direct	0	0	0.0
Academic search	6	0	0.0
Literature	1	1	100.0
Software engineering Institute (SEI)	18	1	5.6
Total	90	4	4.4

Number of investigated articles = 90
Number of selected articles = 4
Ratio of investigated articles to selected articles = 90:4
Ratio = 23:1
For every 23 articles investigated just 1is selected.
4 % of selected articles provided direct evidence relevant and relating to research question.
The percentage value of selected articles is very small compared to the number of articles investigated with respect to the research question. It implies that very few articles provided direct evidence relating to research question. It is an indication that more research is required in this area.

9.3.5 DATA ANALYSIS

From the study carried out the result showed from 90 articles investigated from the electronic databases and a literature, 4 selected articles including the literature provided evidence to research question 7 of this thesis. This represents 4.4% of total articles investigated from electronic databases used for our search.
The ratio of the number of articles investigated to the number of articles that provided direct evidence to the research question:

Number of articles investigated = 90
Number of selected articles that provided evidence = 4
Ratio = 23:1.
For every 23 articles investigated 1 is selected. There is little information available with respect to tools that support testing specifically in software product lines context.
Odd = 0.04.
The value indicates much still needs to be done, to provide testing tools to support product lines testing.

9.4 RESULTS

From our research, 90 articles were investigated; only 4 articles provided direct evidence relating to research question 7 as regards testing tools in software product lines testing process.

The following areas in software product lines are supported with the associated tools [4]:
Software component is supported by Rational Rose, plus CORBRA-compliment framework.
Architecture is supported by Rose, alas, PowerPoint, ADL.
Documentation supported by DocExpress/Word integrated with DOORS/Rose products.
Process Supported by Rational ClearGuild.
Repository supported by Rational ClearGuild.
Junit can support testing in software product lines. [2] [4] [24]
DOORS and REMAP tools support testing in software product lines [120].

Some of the tools presented above are commercial tools to support product lines testing process. Tools support for single product testing is quite different from product lines testing tools, as a result of variability in software product lines.
Further more; as regards tools to support the different phases of product lines development process, testing phase should be given serious attention. As regards the conformance of design or system, it should be possible to apply testing to determine the degree to which a design or system conforms to the constraints expressed in software product lines assets [31].
The following tools from other sources can be used for coverage testing in software product lines: Clover [34], Jtest [36], Dynamic [35], JCover [37], and CodeTest [38].
Tool support for software product lines testing process makes some task more convenient. Lack of tool support for software testing process will have serious effect in product lines testing process where tools should have been provided. Product line organisations should make it a practice to support product lines testing process with tools to achieve its goals.
According to [30], “current tool support for product lines is based on a variety of tools used in the software engineering activities for conventional development efforts and then "stretched" to accommodate product lines”.

"Using these tools successfully in software product lines context requires practice that emphasize the fitness of the tools purpose, the quality of the software produced using the tools, the effect of the tools on the product line development process, and the benefit associated from using the tools" [2].

9.5 DISCUSSION

The research showed that software tools can be used to support the different phases of software product lines testing process in order to reduce the testing cycle. Lack of tool support in software product lines testing can lead to frustration and longer development and product line testing cycle. Tool support for product lines testing is essential to control variability in order to create a product line that is flexible enough to accommodate customers changing needs. But, if tool support can not help represent variability early in product line testing cycle, then software engineers may not be able to express variability in architecture, trace requirements.
It is therefore important for software product lines organisations to adopt sophisticated testing tools to support testing process in software product lines for it to achieve its goals of developing quality software at reduced cost frame and time. The right tools are not available to support product lines testing process. The following tools are available for supporting product lines testing process, but they are not the appropriate tools to support product lines testing process.

Table 9.8: Tools support for product lines.

Area	What is needed	What is available
Requirement	Product lines system requirements tool. Analysis models with variability.	Presently available software system requirements tool: QSS DOORS. Limited analysis capability that includes variations.
Architecture	Tools required, product line software system architecture tool. Models including variation points.	Presently available tool support for architecture: Software system architecture tool: Rose. Limited number of views, UML not fully supported. variation points not supported
Components support tool.	Tools required: Component design, specification, implementation with variation points.	Currently available tools: rational Rose, plus CORBRA-compliment framework. Only skeleton generation with UML support tools.

9.6 CONCLUSION AND RECOMMENDATION

This chapter explores the tools available to support product line testing process using the systematic review strategy as described in Kitchenham [20].
There are differences in software product lines testing process and single product testing process. From our research, most of the tools to support software product lines testing are basically tools supporting single product testing. But software product lines testing process

not only require automated tool support, but require robust and specific kind of tool support. Taking into consideration, core assets used for building products have to express the variation and commonality among products assets.

Presently, adequate tool support is not available for software product lines testing process. What are obtainable currently are modifications of existing tools for software product lines testing process. That means organisations takes risks that can jeopardize their product lines testing process effort and spend time working out strategies to modify existing tools.

Finally, more research is required in this area of specific and sophisticated tools for software product lines testing process. This will save time spent by software product lines organisations trying to modify existing tools to support testing in software product line context.

CHAPTER 10

10 CONCLUSIONS AND FURTHER RESEARCH

10.1 VALIDITY OF RESEARCH

The internal validity of this research is based on:

Systematic reviews related to several areas of software product lines testing were conducted. Validating the findings of our research is very important.

Search strategy

Using key words in a literature-searching program, a comprehensive search was conducted on the following databases:

IEEE Xplore Electronic Database
ACM Digital Library
Springer Link Electronic Database
Science Direct Electronic Database
Academic Search Elite Electronic Database
Literatures (Books)
Software Engineering Institute SEI
Fraunhofer Institute for Experimental Software Engineering (IESE)
Details of how search strategy threat was overcome is in section 10.1.1

Data extraction

Information needed to address our research questions were extracted from primary studies by carefully reading the various articles selected from databases with respect to the research question used in searching for primary studies. Prolong time spent reading primary studies in order for the author to have a comprehensive understanding of selected primary studies. Prolong time spent in reading primary studies gave the author a better understand of primary studies content and this helped in reporting the results from primary studies accurately without attempt to deviate from true results.
The data extracted from primary studies were compared to ensure data were extracted in a consistent manner from primary studies. Numerous primary studies were selected from search, after carefully reading primary studies. The extracted results from each study were compared to in order to have summarised result that is reliable and complete.

Data analysed thoroughly

Data extracted from primary studies were analysed thoroughly. Existing primary studies relating to research questions were extracted from databases using the defined search strategy.
Hints from each databases search are tabulated in a manner that is consistent with the research question. The numbers of primary studies found in each database with respect to the research question were recorded in table. The results of primary studies selected in the databases were collated, compared and summarised

Sources of primary studies

Sources of primary studies used in conducting this research are reliable. The databases used as sources of primary studies for the research are articulate and reliable. The databases publish journals and transactions relevant and relating to software engineering in generally and specifically relevant and relating to our research, testing in software product lines. This thesis performed systematic reviews in different areas of software product lines testing. Primary studies relevant to our research question were investigated and selected from the databases to perform systematic reviews research in software product lines testing.

Authors of primary studies

The authors of primary studies are well known experts with national and international reputation. Their current practices and experience are from a variety of information technology projects in large organisations and research institutes. The articles used in our research, testing in software product lines were written by authors that are experts in software product lines specifically and generally in software engineering.

The extent to which the effects observed in this study are applicable outside of the study is based on the internal validity of the research. Internal validity is a prerequisite for external validity [20].

10.1.1 THREAT TO VALIDITY OF RESEARCH

Validating the findings of this research is very important. Based on this, special attention is required in validating the findings of this research. Since the research was conducted by selecting primary studies from databases and extracting data from selected primary studies, the threat to validity of study therefore originates from these areas.
The main threats to validity for this research are publication selection bias, inaccuracy in data extraction, and search strategy.

Publication selection bias

Systematic reviews in several areas relating to testing in software product lines testing was conducted by combining existing studies. However selection bias may occur in the research due to inability to identify and include all relevant studies relating to testing in software product lines. Publication bias refers to the problem that positive results are more likely to be reported in our research than negatives results. Such selection bias can cause false research. To overcome this threat the following were done:

- Different sources for primary studies were used for the research to ensure consistence of results with respect to the research question under investigation. Based on this, primary studies were searched for, from the different databases listed using the defined search strategy.

- Other source of evidence was search manually relating to product lines outside electronic resources. Literatures were read as well to build up research that is consistent in its result.

- Reviews of research results were performed. The results from several systematic reviews carried out with respect to software product lines testing were reviewed to ensure that accurate results are reported.

- Project supervisor is consulted as expert in the field to reconfirm the accuracy of primary studies selected for research. The selected publications for research are examined to ensure they are relevant to thesis research questions.

Inaccuracy in data extraction

There is threat in data extraction from primary studies. The question to be asked in this situation, was data extracted correctly from primary studies. If inaccurate data were extracted from primary studies, this threatens the validity of the research. The risk of incompleteness is already present. To overcome this threat the following strategies were used:

- Prolong time spent in reading primary studies selected from the list of investigated primary studies in order to have a comprehensive understanding of contents in primary studies selected with respect to research questions.

- After carefully reading primary studies, the research question to be answered is read as well, to see how the contents of primary studies relate to my research question.

- The findings from primary studies are reported in thesis with vivid description. The accuracy of extracted data is confirmed with supervisor. Confirming extracted data with experts in the field is very important, to ensure that extracted data are accurate and complete.

Search strategy threat

Search strategy threat may occur in the research due to inability to define well structured search techniques to identify primary studies. This affects the completeness of the research. To overcome this threat in our research the following were done:
It is important to define and follow a search strategy for the research.
Search strategies aimed at identifying existing primary studies for our research and assessing the volume of potential studies includes:
Combination of search terms derived from thesis research questions.
Research questions were broken down into individual small facets.
Headings used in journals and databases were considered to see how they relate to research questions.
Constructing search strings using Booleans AND'S and OR'S

- **Combination of search terms derived from thesis research questions.**

Each search term used to form a search criteria in searching for primary studies in databases were derived from research questions. The threat in this situation is that it is possible search terms were not properly derived from the research questions. To overcome this risk we ensure that the terms used were contained in the research question.

- **Breaking down research questions into small facets**

The research question pose on this thesis were broken down into small pieces in order to identify primary studies relating to specific research question to be answered. The question here is to what extent should a research questions be broken down and is it broken down in such away that relevant articles are found in databases. To overcome this risk we ensured that the questions were really broken down to the smallest facets as much as we can.

- **Constructing search strings using Booleans AND'S and OR'S.**

Search criteria are constructed by combining the broken down small facets of research questions using Booleans. The position of Booleans depends on the extent to which each research question is broken down. The question here is, where the Booleans should be positioned in the broken down question. The position of Booleans also affects primary studies found in databases as they relate to research question and the volume of primary studies produced from searched databases. To overcome this, we ensured that Booleans are positioned as far as we can break research questions to the smallest facets.

Finding primary studies relevant and relating to the different research questions and areas of research using the above search strategy determines the completeness and accuracy of the research performed in this thesis. If the above search strategy is not properly constructed as it relates to research questions, there is likelihood that primary studies relevant to research questions might not be found in databases.

10.2 ANSWERS TO RESEARH QUESTIONS

The section presents answers to research questions pose on this thesis.

RQ1. What is the relationship between product lines and product?

Answers:

- Product line is made up of two develop development processes: Domain engineering process and application engineering process. Domain engineering develops set of reusable core assets for product lines. Application engineering develops products from product lines core assets. Core assets developed in domain engineering process for product lines are extended to application engineering process for products development.
- The major relationship between software product lines and product is that test cases created in domain engineering for product lines are extended to application engineering for specific product development.
- Software product lines consist of family of products that share common features. Basically, individual product features are derived from product lines core assets.

RQ2. What steps are needed to develop and execute software testing in software product line environment?

Answers:

- In software product line there are two testing processes, domain or core assets testing process, and product testing process.
- Activities in domain or core assets testing include: Inspection of models (requirements model, detailed design, the architecture the products in the product lines will share), software components testing, for reuse across product lines development, and integration testing.
- Activities in product testing process include: product requirements, test cases, unit testing, integration testing and system testing.

RQ3. What approaches can be used to develop test cases from use cases that contain variability and to derive application test cases from them?

Answers:

- Use cases are used to capture requirements for software product lines, and the technique is based on natural language specification and their extensions which make it very easy to understand and document requirements and communicate to users and non technical people.
- Captured requirements for product lines are modeled in order to express the variability in the core assets.
- Use cases are modeled to define software product lines responsibilities and objectives and collaboration. Examples of these models are SCENT method and PLUTO.
- Other methods used to model use cases are : (e.g. OOSE Object-Oriented Software Engineering/Jacobson/, OMT Object Modeling Technique/Rumbaugh/, and most notably, the UML Unified Modeling Language/Booch, Rumbaugh, Jacobson/ as their successor).
- The application of use cases in software product lines testing help to identify the variation points in software product lines requirements at the early phase of the process, and deriving test cases to test software product lines and its specific products.

RQ4. What is the connection between software reuse and reusability of test cases?

Answers:

- Basically, the relationship between software reuse and test cases reuse is that test cases are natural assets of product family or product lines. Therefore, test cases are set of software assets derived in domain engineering for product lines that are reusable across multiple products testing in application engineering [2] [4] [74].
- Software product lines consist of sets of reusable artefacts that are reused across multiple products during application engineering. These reusable artefacts includes: test cases, software components and requirements. Test cases are also reusable software artefacts in software product lines testing just like software components that can be reused across products line development process.
- Artefacts are the general term for any work product which includes: Code, test case, test plane, architecture, model, software components, design, and implementation. Amongst these there are product lines core assets described as reusable software. Test cases are amongst these reusable software.
- Software components also consist of reusable artefacts that are extended to product line development. Test cases created during component constructions are reusable in software product lines construction.

RQ5. How do testers and developers determine the reliability of the reusable software or components?

Answers:

The research showed that the following quality assurance techniques can be used to determine the reliability of reusable software components:

- Testing
- Inspection
- Review
- Scenarios/ use case

- Certification criteria to select components for reuse in software product lines development

RQ6. What are the challenges of software product lines testing?

Answers:

- Techniques for strategic reduction in test cost and time. Software testers are always under constant pressure to reduce the time and cost needed for software testing from management.
- Managing variation point in software product lines is one of the major challenges facing software product lines testing.
- Selecting the appropriate quality assurance techniques for software components in software product lines context is very challenging because of variability complexity.
- Design for testability.
- The two development processes in software product lines makes testing complex in product lines context.
- Test case design techniques and test cases generation for product lines that takes into consideration the variability in product lines and the two development processes is challenging as well.
- Establishing well defined testing steps and procedures to perform testing in software product lines is challenging.
- Software product line lack sophisticated and product line testing specific tools. This creates problems for software testers and developers. Software organisations spend time modifying tools to support product line testing.

RQ7.What tools and activities should testers and developers adopt to ensure software product lines assurance during testing process?

Answers:

The following areas in software product lines are supported with the associated tools:

- Software component is supported by Rational Rose, plus CORBRA-compliment framework.
- Architecture is supported by Rose, alas, PowerPoint, ADL.
- Documentation supported by DocExpress/Word integrated with DOORS/Rose products.
- Process Supported by Rational ClearGuild.
- Repository supported by Rational ClearGuild.
- Junit can support testing in software product lines by providing environment for test case execution.
- DOORS and REMAP tools support testing in software product lines.
- Coverage testing tool in software product lines: Clover, Jtest, Dynamic, JCover, and CodeTest.
- Junit framework and the Roast framework provide test environment for unit testing.

10.2.1 REASONS READERS SHOULD CARE ABOUT RESEARCH ANSWERS

It will be quite interesting to read through the answers of this thesis for the following reasons:

- The research covered systematic reviews in several areas related to software product lines testing base on the research questions pose on the research.

- The answers are mainly on how testing in software product lines can be improved upon, toward achieving the goals of software product lines. The answers are very useful and relevant in the area of testing in software product lines context.

- The answers are contribution of the thesis on improvement in software product lines testing issues with respect to systematic reviews carried out in several areas related to software product lines testing. The answers help to address issues confronting testing in software product lines context.

- The answers focuses also on challenges facing testing in software product lines and how these challenges can be over come to some reasonable extent. Software product lines testing faces a lot of challenges unlike testing in single product. These challenges are addressed in the thesis.

- The answers presents the different activities in software product lines testing with respect to the different areas covered in the research.

Going through the answers provided in the thesis with respect to the research questions, will definitely be interesting considering the fact that several areas relating to software product lines testing were covered in the systematic review research. Also, issues and challenges facing software product lines testing, these answers are solutions in the context of the research questions pose on the thesis.

10.3 SUMMARY OF RESEARCH

Table 10.1 presents summary of total number of articles investigated and selected for the research.

Table 10.1: Summary of total number of articles investigated and selected for research.

Subject areas.	Number of investigated articles.	Number of selected articles.
Product lines and product (Chapter 2).	37	17
Testing activities and steps in SPL testing process (Chapter 3).	34	10
Use cases and test cases in software product lines (Chapter 5).	58	13
Software and test cases reuse Chapter 6).	152	13
Verifying components reliability for product lines (Chapter 7).	51	12
Challenges of software product lines testing process (Chapter 8).	67	15
Testing tools for software product lines testing process (Chapter 9).	90	4
Total	489	84

Table 10.1 above presents the total number of articles investigated and selected for the research.

The total number of articles investigated for the research is 489. The investigated articles evidences are related to research questions. There are related in the sense that they discussed certain areas relating to search terms used for research. These articles were selected from included research articles from databases. The articles do not provide direct evidence to research questions. After reading through the articles, it is discovered that they do not provide direct evidence to research questions.

The total number of selected articles for the research is 84. The selected articles provided direct evidence to our research questions. They provide main report with respect to research questions of the thesis. 84 articles reported testing in product lines in the context of research questions posed on the thesis.

Table 10.2: Summary of thesis

Chapters	Results	Recommendations	Databases date of check
2	Two development process: Domain engineering and application engineering. Testing activities: Inspection of models, components testing, integration tests, unit testing, integration and system testing.	Research into development of process testing model for software product lines.	27-30 January, 2007
3	SPLIT Model. Propose model for software product lines testing process.		1-5th February, 2007.
4	Tools available to support product lines testing Include: Junit, Cover, and code test. E.g. The tools are for coverage testing.	Research is required in this area to develop specific and sophisticated tools for software product lines testing process.	6-7th February, 2007.
5	The major challenges in software product lines testing are testing for two development processes and variant specific defects.	Development of sophisticated model to handle variability complexity in software product lines.	8-11th February, 2007.
6	Test cases are reusable software product lines artefacts, which can be reuse in product lines specific products development.	Considering the facts that software components and core assets are reusable in product lines, sophisticated methods should be deployed to derive test cases for software components.	12-14th February 2007.
7	The approaches to create test cases from use cases: Create usable test cases from use cases. Model use cases to derive test cases and extend use cases to application engineering for product development.	Modern models should be developed to model use cases to handle variability in product lines.	16-18th February 2007.
8	The two techniques mainly used to verify software components reliability are testing and inspection.	Presently, single product quality assurance techniques are used to verify components reliability for software product lines. There is need for software product lines to have sophisticated techniques to verify the reliability of software components.	19-20th February, 2007.
9	The main relationship between product lines and product is the feature or facilities created at the product lines level and specialized for each product.	Further research is required in the area of methodologies to improve the development of quality features or product lines core assets created at product lines level and specialize for each product.	20- 22nd February 2007.
10	The goal of the research is to investigate the different activities in software product lines testing and possible improvement in product lines testing. The research showed that there is no standard testing process model for software product lines testing. It makes it difficult for testers to know the different activities involved in product lines testing. The study specified the areas that require further research in software product lines testing.		23rd February, 2007.

10.3.1 TOP FIVE SCHOLARS IN THIS RESEARCH

Table 10.2 presents top five scholars used in this research. The ranking is based on the number of published articles of each scholar that provided direct evidence relevant and relating to research questions used in the research. For instance, the top ranked scholar, John McGregor articles provided direct evidence relating to each questions and corresponding chapters.

Table 10.3: Top five scholars in this research.

Rank	No of Papers	Scholars	Affiliation
1	10	John D. McGregor	Department of Computer Science, Clemson University Clemson, South Carolina, U.S.A.
2	9	Linda Northrop	Software Engineering Institute (SEI) Pittsburgh, U.S A.
3	6	Klaus Pohl	Software Systems Engineering, University of Duisburg-Essen, Germany
4	4	Bertolino Antonia	Researcher of National Research Council (CNR), Italy.
5	3	Alessandro Fantechi	Computer Science Foundations at the faculty of engineering, University of Florence.

John D. McGregor

08/99 to Present Visiting Scientist: Software Engineering Institute - Product Line Initiative, Carnegie Mellon University, Pittsburgh, PA.
08/89 to Present Associate Professor of Computer Science: Clemson University, Clemson, SC. [117].

Linda Northrop

Is director of the Product Line Systems Program at the Software Engineering Institute (SEI) Carnegie Mellon University Pittsburgh, and chaired the first annual International Conference on Software Product Lines. A frequent keynote speaker and highly acclaimed educator, she has more than thirty years of experience in software development, including work at Eastman Kodak and IBM [115] [116].

Prof. Dr. Klaus Pohl

Prof. Dr. Klaus Pohl holds a full professorship at the University of Duisburg-Essen and leads the Software Systems Engineering research group. He received his Ph.D. and his habilitation in computer science from RWTH Aachen, Germany. He is involved in various technology transfer projects as well as major research projects which focus on different aspects of software product line engineering. Klaus Pohl is (co-) author of over 90 refereed publications. He has served as program chair for several international and national

conferences, such as the IEEE International Requirements Engineering Conference (RE'02), the Experience Track of the 27th International Conference on Software Engineering (ICSE 2005).

Bertolino Antonia

Researcher of National Research Council (CNR), Italy

Alessandro Fantechi

A full professor of Computer Science Foundations at the faculty of engineering, University of Florence

10.4 CONCLUSIONS

The main purpose of the research as presented in this thesis is to present a clear picture of testing in the context of software product lines, which is quite different from testing in single product. The focus of this thesis is specifically the different steps and activities involved in software product lines testing and possible improvements in software product lines testing activities and issues towards achieving the goals of developing high quality software product lines at reduced cost and time.
But, for software product lines to achieve its goals, there should be a comprehensive set of testing activities in software product lines development. The development activities from performing analyses and creating designs to integrating programs in software product line context, component testing and tools support for software product lines testing should be taken into consideration.
Software product line has its goals and benefits. Testing in product lines has its benefits as well if properly implemented. There are certain actions that can be taken to maximize the benefits of software product lines testing process:

Software product lines testing process should be detailed and completely defined in test plane document. The document should include all steps and activities involved in software product lines testing process, which is quite different from testing in single product.

Presently, adequate tool support is not available for software product lines testing process. What are obtainable currently are modifications of existing tools for software product lines testing process. That means organisations takes risks that can jeopardize their product lines testing process effort and spend time working out strategies to modify existing tools.

Product lines approach to development is based on reuse of software development artefacts such as software components. For product lines to achieve the goal of quality development, the components intended for reuse must be of high quality. There must be well defined guidelines for product lines tester and developers to assure the quality of reusable software components for product lines development.

New products are derived from product lines core assets. The test assets and software must be complete. Test cases must be created earlier to build complete test cases for each of the specific products.

10.5 FURTHER RESEARCH

Software product lines approach to software development is gradually evolving and embraced by software organisations. Additional work needs to be done to improve software product lines testing process in organisations. Some examples of further work include:

Develop software product lines test model that shows all the steps and phases in software product lines testing process. Considering the variability complexity in software product lines, V-Model for single product testing is not a sophisticated model for software product lines testing.

Develop more improve techniques to verify reusable software components quality.

Develop more comprehensive techniques for identifying variants specific defects during use cases testing.

11 APPENDIX

List of Figures

12 REFERENCES:

[1] Patrick Donohoe, Software product LINES: experience and research directions, Publisher: Boston, MA: Kluwer Academic, 2000. ISBN: 0792379403, OCLC: 44613157

[2] Paul Clements and Linda Northrop, Software Product Lines: Practices and Patterns, Addison - Wesley (2002).

[3] Jan Bosch, Design and Use of Software Architectures Adopting and evolving a product-line-approach, Addison – Wesley (2000)

[4] John D. McGregor, Testing a software product line. Technical Report CMU/SEI-2001-TR-022, Software Engineering Institute, Carnegie Mellon University, December 2001.

[5] Beizer Boris, Software System Testing and Quality Assurance, International Thomson Computer Press (1996)

[6] L. K. Chung, B. A. Nixon, E. Yu., and J. Mylopoulos, Nonfunctional Requirements in software Engineering, Kluwer Publishing, 2000.

[7] Satisfying the Conflicting Software Qualities of Maintainability and performance at the Source Code Level, Bill Andreopoulos, Department O f Computer Science, York University, Toronto, Ontario, Canada, M3JIP3.
http://wer.inf.puc-rio.br/WERpapers/artigos/artigos_WER04/Bill_Andreopoulos.pdf

[8] Gerald Kotonya and lan Summerville, Requirements Engineering, John Wiley & Sons Ltd, The Atrium, Southern Gate, Chichester, West Sussex, op o19 8SQ,England, 2004.

[9] Steven R. Rakitin, Software Verification and Validation for Practitioners and Managers, 2nd edition, Artech House Publishers, Boston, London (2001)

[10] Pohl, K., Böckle, G., and van der Linden, F. Software Product Line Engineering—Foundations, Principles, and Techniques, Springer, Berlin, Heidelberg, New York, 2005.

[11] Klaus Pohl and Andreas Metzger, SOFTWARE PRODUCT LINE TESTING, Exploring principles and potential solutions, 12 COMMUNICATIONS OF THE ACM, December 2006.

[12] Andres Metzger/ University of Duisburg-Essen (D) Model Based Testing of Software product lines, Software and Systems quality Conferences, ACM, May 10-12, 2006 Congress Centre Düsseldorf, Germany.

[13] Van der Linden, F., ESAPS-CAFÉ Inputs. Proceedings of the 3rd ITEA Symposium, Amsterdam, Netherlands, October 2002, URL: http://www.itea-of_ce.org/symposium/ [March 13, 2002].

[14] Van der Linden, F., Software Product Families in Europe: The Esaps & Café Projects. IEEE Software, Volume 19, Number 4, July/August 2002, 41.49

[15] Ronny Kolb, A Risk-Driven Approach for Efficiently Testing Software Product Lines, Fraunhofer Institute for Experimental Software Engineering (IESE)

[16] Klaus Pohl, Lero, Univ. of Limerick, Ireland &Univ. of Duisburg-Essen, Germany, Frank van der Linden Philips Medical Systems, The Netherlands, Andreas Metzger, Univ. of Duisburg-Essen, Germany, Testing in a software product line

[17] John D. McGregor Building Reusable Test Assets for a Product Line7th International Conference, Springer link, ICSR-7, Austin, TX, USA, April 15-19, 2002. Proceedings

[18] Andreas Reuys, Erik Kamsties, Klaus Pohl, and Sacha Reis, Model-Based System Testing of Software Product Families, Springer Berlin / Heidelberg, May, 2005.

[19] http://satc.gsfc.nasa.gov/assure/agbsec3.txt, 09/10/2001

[20] B. Kitchenham, Procedures for Performing Systematic Reviews, Keele University Technical Report TR/SE-0401, July, 2004.

[21] Bertolino, A., and Gnesi, S., PLUTO: A Test Methodology for Product Families, 5th International Workshop on Software Product-Family Engineering, Springer Link, LNCS 3014, 2004

[22] Mikael Svahnberg and Jan Bosch, Issues Concerning Variability in Software Product Lines, Springer-Verlag Berlin Heidelberg 2000.

[23] D. E. House, W. F. Newman, Testing large software projects ACM SIGSOFT Software Engineering Notes, April 89.

[24] Antti Tevanlinna, Juha Taina, Raine Kauppinen, Product family testing: a survey ACM SIGSOFT Software Engineering Notes (March 2004)

[25] Robert M. Poston. IDE, TESTING TOOLS COMBINE THE NEW AND OLD, IEEE, MARCH 1995

[26] DWLLO, R.A., MCCRACKEN, W.M., MARTIN, R.J., and PASSAFIUME, J.F.: 'Software testing and evaluation' (Benjamin Cummings Publishing Company, Inc., California, (1987)

[27] Pieter Koopman, Artem Alimarine, Jan Tretmans, and Rinus Plasmeijer, GAST: Generic Automated Software Testing, Springer Link.

[28] POSTON, R.M., and SEXTON, M.P.: 'Evaluating and selecting testing tools', IEEE Software., 1992, 9, (3), pp. 33-42

[29] SI FAIRLEY, R.E.: 'Software testing tools'. Computer Program Testing, Proc. Summer School on Computer Program Testing, SOGESTA, Urbino, Italy, 29 June-3 July 1981. CHANDRASEKARAIY, B., and RADICCHI, S. (Eds.) (North- Holland) pp. 151-186

[30] Len Bass, Paul Clements, Patrick Donohoe, John McGregor, Linda Northrop, Fourth Product Line Practice Workshop Report, CMU/SEI-2000-TR-002 ESC-TR-2000-002, February 2000.

[31] Powell, A.; Vickers, A.; Williams, E.; & Cooke, B. Ch. 11, "A Practical Strategy for the Evaluation of Software Tools," 165-185. Method Engineering: Principles of Method Construction and Tool Support-Proceedings of the IFIP TC8, WG8.1/8.2 Working Conference on Method Engineering. Atlanta, GA, August 26-28, 1996. London, UK: Chapman & Hall, 1996

[32] International Organization for Standardization & International Electrotechnical Commission. Quality Management—Guidelines for Configuration Management [ISO 10007:1995 (E)], Geneva, Switzerland: International Organization for Standardization/ International Electrotechnical Commission, 1995.

[33] Institute of Electrical and Electronics Engineers. Information Technology-Guideline for the Evaluation and Selection of CASE Tools (IEEE Std 1462-1998), New York, NY: IEEE Computer Society Press, 1998.

[34] Clover (http://www.cenqua.com/clover/

[35] Dynamic, DMS (http://dynamic-memory.com/

[36] Parasoft Jtest (http://parasoft.com/jsp/home.jsp)

[37] JCover (http://www.mmsindia.com/JCover.html)

[38] CodeTest (http://www.metrowerks.com/MW/Develop/AMC/Code TEST/default.htm)

[39] Qian Yang, J. Jenny Li, and David Weiss, A Survey of Coverage Based Testing Tools, AST'06, Shanghai, China, May 23, 2006.

[40] C.V. Ramamoorthy and S.F, Ho, TESTING LARGE SOFTWARE WITH AUTOMATED SOFTWARE EVALUATION SYSTEMST, Computer Science Division, Department of Electrical Engineering and Computer Sciences Electronics Research Laboratory University of California, Berkeley.

[41] Georg Grütter, Challenges for testing in software product lines, http://www.biglever.com/split2006/Presentations/GruetterKeynoteAbstract.pdf, 2006-06-06

[42] Mark Ardis, Nigel Daley, Daniel Hoffman, Harvey Siy1 and David Weiss, Software product lines: a case study, Department of Computer Science, University of Victoria, Victoria, BC, Canada

[43] Antonia Bertolino, Stefania Gnesi, Use case-based Testing of product Lines, Proceedings of the 9th European software engineering conference held jointly with 11th ACM SIGSOFT international symposium on Foundations of software engineering, ACM, 2003. Pages 355-358

[44] A. Cockburn. Writing Effective Use Cases, Addison-Wesley, 2000

[45] Joachim Bayer, Oliver Flege, Peter Knauber, Roland Laqua, Dirk Muthig, Klaus Schmid, Tanya Widen, Jean-Marc DeBaud, PuLSE: a methodology to develop software product lines, ACM Press, May 1999.

[46] Myra B. Cohen, Matthew B. Dwyer, Jiangfan Shi , Coverage and adequacy in software product line testing, International Symposium on Software Testing and Analysis, Proceedings of the ISSTA 2006 workshop on Role of software architecture for testing and analysis, ACM press, February 1979.

[47] Kolb, R., and Muthig, D. Challenges in Testing Software Product Lines. In Proceedings of CONQUEST'03, Nuremberg Germany, pp. 81–95, ACM, September 2003.

[48] Kolb, R., McGregor, J. D., and Muthig, D. Proceedings of the First International Workshop on Quality Assurance in Reuse Contexts (QUARC), Boston, MA, Fraunhofer IESE Report No. 096.04/E, August 2004.

[49] H Muccini and A. van der Hoek, Towards Testing Product Line Architectures, Electronic Notes in Theoretical Computer Science, Volume 82, Issue 6, September 2003, Pages 1-11

[50] Klaus Pohl Andreas, (The Irish Software Engineering Research Centre) University of Limerick, Ireland, Software Systems Engineering University of Duisburg-Essen Schützenbahn 70 45117 Essen, Germany. Andreas Metzger, Software Systems Engineering University of Duisburg-Essen, Germany, Variability Management in Software Product Line Engineering, ICSE'06, May 20-28, 2006, Shanghai, China

[51] Krueger, C.W. Software reuse. ACM Comput. Surv. 24, 2 (1992), 131–183.

[52] Rombach, H.D. Software reuse: A key to the maintenance problem. Inf. Software Technol. J. 33, 1 (Jan./Feb. 1991), 86–92.

[53] Agresti, W.W., and McGarry, F. The Minnowbrook workshop on software reuse: A summary report. In Software Reuse: Emerging Technology, W. Tracz, Ed., IEEE Press, 1987.

[54] Thomas, W., Delis, A., and Basili, V. An evaluation of Ada source code reuse. In Proceedings of the Ada-Europe International Conference (Zandvoort, The Netherlands, June 1992).

[55] Ravichadran and Marcus RothenBerger, Software reuse strategies and Component Markets, 2003 ACM.

[56] Basili, V.R., Briand, L.C., and Melo, W.L. How reuse influences productivity in object-oriented systems. Commun. ACM 39, 10 (1996),
104–116.

[57] An Integrated Approach to Software Reuse Practice, Eliseo Manbella, Roberto Ferrari, Francesca De Carli, Angela Lo Surdo, Sodalia S.P.A., 364. Via del Brennero, Italy.

[58] Chávez, A., Tornabene, C., and Wiederhold, G. Software component licensing: A primer. IEEE Software 15, 5 (1998), 47–53.

[59] Poulin, J.S., Caruso, J.M., and Hancock, D.R. The business case for software reuse. IBM Systems Journal 32, 4 (1993), 567–594.

[60] Department of Defence. Software Reuse Executive primer fall Church, VA, April 1996.

[61] Agresti, W.W., and McGarry, F. The Minnowbrook workshop on software reuse: A summary report. In Software Reuse: Emerging Technology, W. Tracz, Ed., IEEE Press, 1987.

[62] Brooks, F.P. No silver bullet: Essence and accidents of software engineering. Computer 20, 4 (Apr. 1987

[63] An Empirical Study of Developers Views on software Reuse in Statoil ASA Odd Petter N. Slyngstad, Anita Gupta, Reidar Conradi, Parastoo Mohagheghi, Dept of Computer and

Information Science (IDI) Norwegian University of science and Technology (NTNU), Harald Ronneberg, Einar Landre Statoil KTJ/IT, Norway.

[64] Sindre, G., Conradi R, and Karlsson, E. The REBOT Approach to Software Reuse. Journal of System Software, (30, September 1995), 201-212.

[65] Software Engineering principles and practice, Wiley, 2nd edition, 2001

[66] Software Reuse Guildlines, Muthu Ramachandran, School of Computing, Leed Metropolitans University, LEEDS, UK, 2005.

[67] Basili, V. Viewing maintenance as reuse-oriented software development. IEEE Software 7, 1 (Jan. 1990), 19–25.

[68]. Boehm, B.W., and Papaccio, P.N. Understanding and controlling software costs. IEEE Trans. Software Eng. 14, 10 (Oct. 1988), 1462–1476.

[69]. Architecture-Based Problem Frames Constructing for Software Reuse, Chu Wang, Qian Depei, Liu Chuda, Dept. of Computer Science, Xian Jiaotong University Xian, P. R China.

[70] Von Mayrhauser, A.; Walls, J.; Mraz, R.; Testing applications using domain based testing and Sleuth, IEEE, Software Reliability Engineering, 1994. Proceedings, 5th International Symposium on 6-9 Nov. 1994 Page(s):206 – 215.

[71] Von Mayrhauser, A.; Mraz, R.; Walls, J.; Ocken, P.; Domain based testing: increasing test case reuse, Computer Design: VLSI in Computers and Processors, 1994. ICCD '94. Proceedings, IEEE International Conference on 10-12 Oct. 1994 Page(s):484 – 491

[72] Christian Denger and Ronny Kolb, Testing and Inspecting Reusable Product Line Components: First Empirical Results, Fraunhofer Institute for Experimental Software Engineering (IESE) Fraunhofer-Platz 1, 67663 Kaiserslautern, Germany, ISESE'06, September 21–22, 2006, Rio de Janeiro, Brazil. 2006 ACM

[73] Mikko Karinsalo and Pekka Abrahamsson, Software Reuse and the Test Development Process: A Combined Approach, Springer link.

[74] Geppert, B.; Li, J.; Roessler, F.; Weiss, D.; “Towards Generating Acceptance Tests for Product Lines”, 8th Intl. Conference on Software Reuse 2004, Madrid, Spain, Springer, New York, pp. 35-48, 2004.

[75] Boehm, B.; Brown, A.W.; Madachy, R.; Ye Yang; A software product line life cycle cost estimation model, IEEE, Empirical Software Engineering, 2004. ISESE '04. Proceedings, International Symposium on 2004.

[76] Craig Larman, Applying UML and Pathers, An introduction to Object-Oriented analysis and design and Iterative development, Prentice Hall, 2005.

[77] Mark Staples, Derrick Hill, Experiences Adopting Software Product Line Development without a Product Line Architecture, IEEE, 30 Nov.-3 Dec. 2004 Page(s):176 - 183

[78] JOHANNES RYSER, MARTIN GLINZ, SCENT: A Method Employing Scenarios to Systematically Derive Test Cases for System Test, Technical Report, Institut für Informatik, Universität Zürich.2000/2003.

[79] Amyot, D.; Logrippo, L.; Buhr, R.J.A.; Gray, T.; Use case maps for the capture and validation of distributed systems requirements, Proceedings, IEEE International Symposium on 7-11 June 1999 Page 44-53.

[80] Braganca, A.; Machado, R.J.; Extending UML 2.0 metamodel for complementary usages of the /spl Lt/extend/spl Gt/ relationship within use case variability specification, Software Product Line Conference, 2006 10th International, 21-24 Aug. 2006 Page(s):5 pp.

[81] Gomaa, H.; Shin, M.E.; Multiple-view meta-modeling of software product lines, Engineering of Complex Computer Systems, 2002. Proceedings Eighth IEEE International Conference on, 2-4 Dec. 2002 Page(s):238 – 246.

[82] A. Fantechi1, S. Gnesi, G. Lami, and E. Nesti1, A Methodology for the Derivation and Verification of Use Cases for Product Lines, Springer Berlin / Heidelberg. 2004

[83] A. Bertolino, A. Fantechi, S. Gnesi, and G. Lami, Product Line Use Cases: Scenario-Based Specification and Testing of Requirements, Springer Berlin Heidelberg, 2007.

[84] Alessandro Fantechi1, Stefania Gnesi, Isabel John, Giuseppe Lami, and Jörg Dörr, Elicitation of Use Cases for Product Lines, Springer Berlin Heidelberg. 2004.

[85] Carnegie Mellon University, Software Engineering Institute, http://www.sei.cmu.edu, 2006.

[86] McGregor, J.; Northrop, L.; Jarrad, S.; Pohl, K.; “Initiating Software Product Lines”, IEEE Software, Vol. 19, No. 4, pp. 24-27, July/August 2002

[87] Offutt, J.; Abdurazik, A.; “Generating Tests from UML Specifications”, 2nd Intl. Conference on UML’99, 1999.

[88] Nebut, C.; Fleurey, F.; Le Traon, Y.; Jézéquel, J.-M.; “A Requirement-based Approach to Test Product Families”, 5th Intl. Workshop on Product Family Engineering (PFE-5), Siena, Italy, November 2003.

[89] Hartmann, J.; Vieira, M.; Ruder, A.; “UML-based Approach for Validating Product Lines”, Intl. Workshop on Software Product Line Testing (SPLiT), Avaya Labs Technical Report, pp. 58-64, Boston, USA, August 2004.

[90] Automated Program Flaw Finding using Simulated Annealing Nigel Tracey, John Clark, Keith Mander, Department of Computer Science University of York Heslington, York England.

[91] John Morris, Peng Lam, Gareth Lee, Kris Parker, Gary A Bundell, Determining component reliability using a testing index, Proceedings of the twenty-fifth Australasian conference on Computer science, IEEE Computer Society Press Melbourne, Victoria, Australia Pages: 167 - 176 , 2002.

[92] Chillarege, R., Bhandari, I., Chaar, J., Halliday, M., Moebus, D., Ray, B., Wong, M. Orthogonal Defect Classification – A Concept for In-process Measurements, IEEE Transactions on Software Engineering, vol. 18, pp. 943–956, Nov. 1992.

[93] Geppert, B., Krueger, C., and Trew, T. (eds.). Proceedings of the Second International Workshop on Software Product Line Testing (SPLiT 2005), Rennes, France, Sep. 2005.

[94] Wiegers, K. Peer Reviews in Software – A Practical Guide. Addison-Wesley, 2002.

[95] Thomas Thelin, Team-based Fault Content Estimation in the Software Inspection Process, Dept. of Communication Systems, Lund University, Lund, Sweden, Proceedings of the 26th International Conference on Software Engineering (ICSE'04), 2004 IEEE.

[96] El Emam, K., Laitenberger, O. and Harbich, T., "The Application of Subjective Estimates of Effectiveness to Controlling Software Inspections", Journal of Systems and Software, 54(2):119- 136, 2000.

[97] Martin, J. and Tsai, W. T., "N-Fold Inspection: A Requirements Analysis Technique", Communications of ACM, 33(2):225- 232, 1990.

[98] Sherif Yacoub, Bojan Cukic, and Hany H. Ammar, A Scenario-Based Reliability Analysis Approach for Component-Based Software, IEEE TRANSACTIONS ON RELIABILITY, VOL. 53, NO. 4, DECEMBER 2004.

[99] S. Gokhale et al., "Reliability simulation of component-based software systems," in Proc. 9th Int. Symp. Software Reliability Engineering (ISSRE'98), Paderborn, Germany, Nov. 1998, pp. 192–201.

[100] Sherif Yacoub, Ali Mili, and Chakri Kaveri, Mark Dehlin, A Model for Certifying COTS Components for Product Lines. http://citeseer.ist.psu.edu/481655.html

[101] Jeffrey M.Voas, Certifying Off-the-Shelf Software Components, Volume 31 Issue 6, IEEE, June 1998 Page(s):53 – 59.

[102] The National Product Lines Assets Center (NPLACE). Certification Criteria for Database Management COTS components, http://www.nplace.wvhtf.org/criteria.html.

[103] M. Brian Blake, Kevin Cleary, Sohan R. Ranjan, Luis Ibanez, Kevin Gary, Use case-driven component specification: a medical applications perspective to product line development, Proceedings of the 2005 ACM symposium on Applied computing, ACM, 2005.

[104] Macaulay, L. A. Requirements Engineering, Springer - Verlag. 1996.

[105] Charles W. Krueger New Methods in Software Product Line Development, IEEE, Page(s):95 – 99, 21-24 Aug. 2006.

[106] Gomaa, H.; Shin, M.E.; Automated Software Product Line Engineering and Product Derivation, IEEE, Jan. 2007 Page(s):285a - 285a.

[107] Foreman, John. Product Line Based Software Development- Significant Results, Future Challenges. Software Technology Conference, Salt Lake City, UT, April 23, 1996.

[108] Martin Verlage, Thomas Kiesgen, Five Years of Product Line Engineering in a Small Company, Proceedings of the 27th international conference on Software engineering ICSE, ACM, 2005.

[109] Montse Ereño, Uxue Landa, Dra. Rebeca Cortazar, Software Product Lines structuring based upon Market Demands, Proceedings of the 2005 conference on Specification and verification of component-based systems SAVCBS '05, ACM 2005.

[110] James R. Hamilton, Harold G. Hawley, and Clinton J. Lalum, PRODUCT-LINE REUSE FOR ADA SYSTEMS, Proceedings of the conference on TRI-Ada '95: Ada's role in global markets: solutions for a changing complex world TRI-Ada '95 ,ACM 2005.

[111] Hassan Gomaa, Designing Software Product Lines with UML 2.0: From Use Cases to Pattern-Based Software Architectures, Springer-Verlag Berlin Heidelberg 2006.

[112] David Benavides, Antonio Ruiz-Cort´es, Miguel A. Serrano, and Carlos Montes de Oca V´azquez, A First Approach to Build Product Lines of Multi-organizational Web Based Systems (MOWS), Springer-Verlag Berlin Heidelberg 2006.

[113] Paul Clements, Product-Line Engineering, Software engineering Institute, Carnegie Mellon University, 2002.

[114] G. Chastek, P. Donohoe, K.C. Kang, and S. Thiel. Product Line Analysis: A Practical Introduction. Technical Report CMU/SEI-2001-TR-001, Software Engineering Institute, Carnegie Mellon University, June 2001.

[115]http://www.awprofessional.com/authors/bio.asp?a=af461a31-2133-44d7-adae-f20e9a6bd47c&rl=1

[116] http://www.sei.cmu.edu/staff/lmn/

[117] http://www.cs.clemson.edu/~johnmc/vita/McGregor12.16.04.pdf

[118] Jaring, M., and Bosch, J., Representing Variability in Software Product Lines: A Case Study. In Chastek G. J. (Ed.): Proc. Software Product Lines, 2nd Int. Conf, SPLC 2, San Diego, CA, USA, August 19-22, 2002, LNCS 2379, 15- 36.

[119] Elaine J. Weyuker. Testing component-based software: A cautionary tale. IEEE Software, 15(5):54–59, September 1998.

[120] Ronny Kolb and Dirk Muthig, Making testing product lines more efficient by improving the testability of product line architectures, ACM 2006

[121] Schmid, K.; Krennrich, K.; Eisenbarth, M. Requirements management for product lines: extending professional tools, IEEE 2006.

[122] Gomaa, H.; Designing Software Product Lines with UML 2.0: From Use Cases to Pattern-Based Software Architectures Software Product Line Conference, 2006 10th International, IEEE, 21-24 Aug. 2006 Page(s):218 – 218.

[123] http://dictionary.reference.com/